

Linux
Professional
Institute
Tutorials

LPI exam 101 prep: Hardware and architecture

Junior Level Administration (LPIC-1) topic 101

Skill Level: Introductory

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

08 Aug 2005

In this tutorial, Ian Shields begins preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 101. In this first of five tutorials, Ian introduces you to configuring your system hardware with Linux™. By the end of this tutorial, you will know how Linux configures the hardware found on a modern PC and where to look if you have problems.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 101, the five topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 101: Tutorials and topics		
LPI exam 101 topic	developerWorks tutorial	Tutorial summary
Topic 101	LPI exam 101 prep (topic 101): Hardware and architecture	(This tutorial). Learn to configure your system hardware with Linux. By the end of this tutorial, you will know how Linux configures the hardware found on a modern PC and where to look if you have problems.
Topic 102	LPI exam 101 prep: Linux installation and package management	Get an introduction to Linux installation and package management. By the end of this tutorial, you will know how Linux uses disk partitions, how Linux boots, and how to install and manage software packages.
Topic 103	LPI exam 101 prep: GNU and UNIX commands	Get an introduction to common GNU and UNIX commands. By the end of this tutorial, you will know how to use commands in the bash shell, including how to use text processing commands and filters, how to search files and directories, and how to manage processes.
Topic 104	LPI exam 104 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard.	Learn how to create filesystems on disk partitions, as well as how to make them accessible to users, manage file ownership and user quotas, and repair filesystems as needed. Also learn about hard and symbolic links, and how to locate files in your filesystem and where files should be placed. See detailed objectives below.
Topic 110	The X Window system	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot

- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Hardware and architecture," the first of five tutorials designed to prepare you for LPI exam 101. In this tutorial, you will learn about PC hardware and architecture.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.101.1 Configure fundamental BIOS settings	Weight 1	You will learn to configure fundamental system hardware by making the correct settings in the system BIOS. You will learn about configuration issues such as the use of LBA on IDE hard disks larger than 1024 cylinders, enabling or disabling integrated peripherals, and configuring systems with (or without) external peripherals such as keyboards. We also discuss correct settings for IRQ, DMA, and I/O addresses for all BIOS-administered ports and settings for error handling.
1.101.3 Configure modem and sound cards	Weight 1	You will learn how to ensure that devices meet compatibility requirements and how to set up both the modem and sound card. You will learn how to configure a modem for outbound dialup, and how to use it for outbound PPP, SLIP, or CSLIP connections.
1.101.4	Weight 1)	You will learn how to configure

Set up SCSI devices	Weight 3)	SCSI devices using the SCSI BIOS as well as the necessary Linux tools. You will review the various types of SCSI. You will learn how to set up a SCSI boot device and how to set the desired boot sequence in a mixed SCSI and IDE environment.
1.101.5 Set up different PC expansion cards	Weight 3)	You will learn about the differences between ISA and PCI cards with respect to configuration issues. You will learn how to check the settings of IRQs, DMAs, and I/O ports to avoid conflicts between devices.
1.101.6 Configure communication devices	Weight 1	You will learn how to install and configure different internal and external communication devices such as modems, ISDN adapters, and DSL switches. You will learn about compatibility requirements (especially important if that modem is a winmodem), necessary hardware settings for internal devices (IRQs, DMAs, I/O ports), and loading and configuring suitable device drivers. We will also cover interface configuration requirements.
1.101.7 Configure USB devices	Weight 1	You will learn how to activate USB support and how to use and configure different USB devices. You will learn about correct selection of your USB chipset and the corresponding module. We will also cover the basic architecture of the layer model of USB and the different modules used in the different layers.

Prerequisites

There are no formal prerequisites for this tutorial. To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. BIOS settings

This section covers material for topic 1.101.1 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

We will start with a high-level overview of a modern personal computer, and then we'll discuss the configuration issues for setting up a system. We will focus on systems using an x86 processor, such as an Intel® Pentium® or AMD Athlon processor, and a PCI bus, as these are the most common today.

Many of the topics covered here have a high level of overlap with LPI objectives for specific peripherals. Later sections of this tutorial will refer you back to this section for basic material.

System and BIOS overview

A modern personal computer (or PC) system consists of a central processing unit (CPU) for performing calculations, along with some memory for storing the data that the processor is using. To make such a device useful, we attach peripheral devices, such as keyboards, mice, displays, hard drives, CD or DVD drives, printers, scanners, and network cards, which allow us to enter, store, print, display, and transmit data.

In the computer just described, the memory used by the processor is called Random Access Memory (RAM). In a typical PC, this memory is *volatile*, meaning that it requires power to keep its data. Turn off the PC and the memory is wiped clean. Put another way, when we turn off a PC, we turn it into a collection of hardware components that will do nothing until reprogrammed. This reprogramming occurs when we turn on the machine; the process is called *bootstrapping* or *booting* the computer.

Bootstrap process and BIOS

The process of booting involves loading an operating system from an external storage device, such as a floppy disk, CD, DVD, hard drive, or memory key. The program that does this initial loading is permanently stored in the computer and is called the *Basic Input Output System (BIOS)*. The BIOS is stored in non-volatile memory, sometimes called *Read Only Memory (ROM)*. In early PCs, the ROM chip

was often soldered or socketed to the computer main board (or *motherboard*). Updating the BIOS meant replacing the ROM chip. Later, *Electrically Erasable Programmable Read Only Memories (EEPROMs)* were used. EEPROMs allowed BIOS to be upgraded in the field with a diskette instead of special tools. Today you will more often find a form of non-volatile memory known as *Flash* memory, which is also used in digital cameras and memory keys. Flash memory also permits BIOS upgrades in the field.

Besides controlling the initial bootup of a PC, today's BIOS programs usually permit a user to set or verify several configuration options on a system. These include verifying installed features such as RAM, hard drive, optical drive, keyboard, mouse, and possibly onboard display, sound and network connections. The user may enable or disable some features. For example, the onboard sound may be disabled to allow use of an installed sound card. The user may also choose which devices will be considered for booting the system and whether the system is protected by a password.

Accessing the BIOS setup screens usually requires a keyboard to be attached to the system. When a system is powered on a *Power On Self Test* or *POST* is performed. On some systems you will be briefly prompted to press a particular key to enter setup otherwise normal bootup takes over. On other systems you will need to know which key to press before the normal boot process is invoked as the prompt is either not present or may have been removed as the result of previous customization of setup options. On some systems you may have other choices besides going to the BIOS setup, such as illustrated in Figure 1. Otherwise, you should see a BIOS summary screen such as that shown in Figure 2.

Figure 1. Accessing the BIOS settings

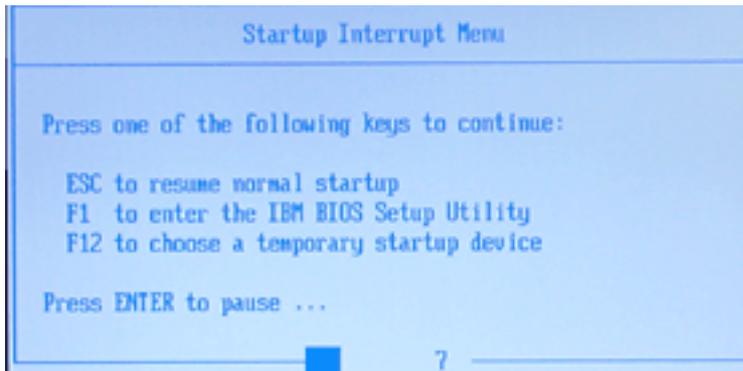
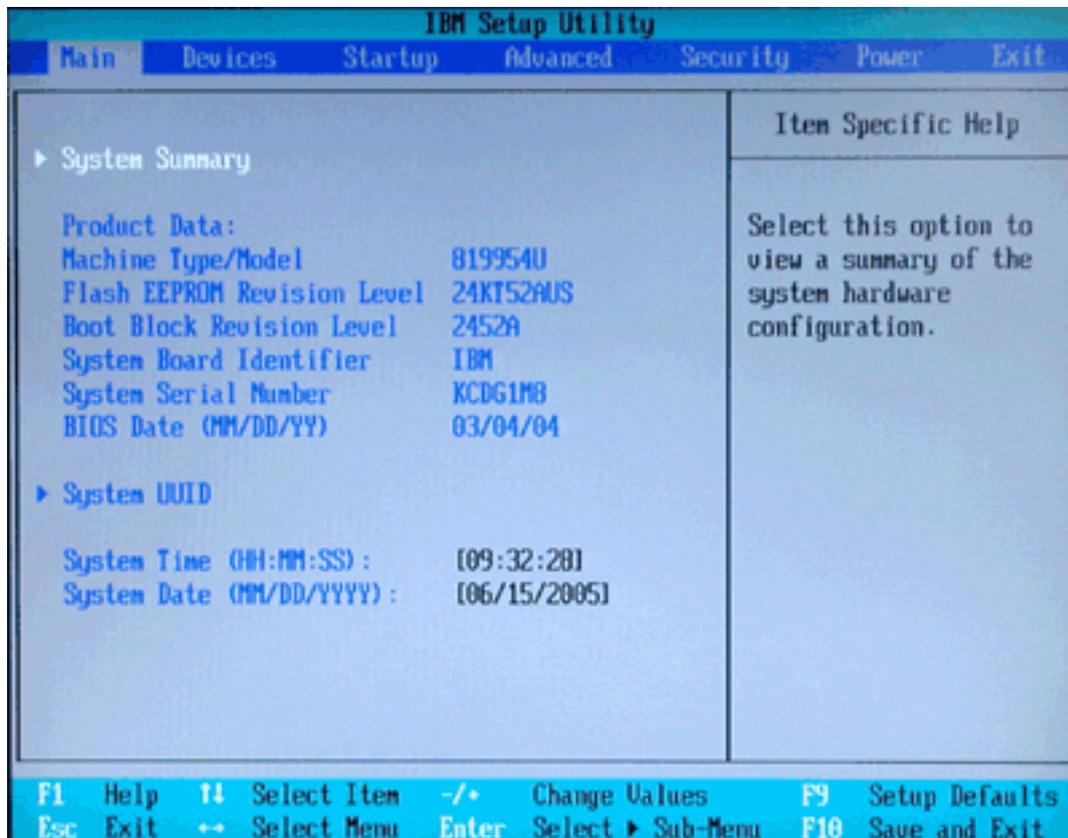


Figure 2. BIOS settings summary



The above illustrations are examples of what you may see, but BIOS setup screens vary widely, so don't be surprised if yours looks different.

Figure 2 shows us that the Flash EEPROM (or system BIOS) revision level is 24KT52AUS and it is dated March 4, 2004 while the current date on the system is June 9, 2005. A check on the manufacturer's (IBM) support site shows that several later BIOS versions are available, so it would probably be a good idea to upgrade this system's BIOS.

You will notice several other menu selections in Figure 2. We will cover these in the remaining sections of this tutorial. Before we do though, let's review a little more of the inner workings of a PC.

Buses, ports, IRQs, and DMA.

PCI and ISA buses

Peripheral devices, including those that may be built in to the system board, communicate with the CPU over a *bus*. The most common bus type in use today is the *Peripheral Component Interconnect* or *PCI* bus which has mostly superseded the earlier *Industry Standard Architecture* or *ISA* bus. The ISA bus was sometimes

called the *AT* bus after the IBM PC-AT in which it was first used in 1984. During the transition from ISA to PCI bus, many systems included both buses with slots permitting the use of either ISA or PCI peripherals.

The ISA bus supports 8-bit and 16-bit cards, while the PCI bus support 32-bit devices.

There are a couple of other bus standards that you should also know about. Many systems include an *Accelerated Graphics Port* or *AGP* slot which is a special slot based on the PCI 2.1 bus specification, but optimized for the high bandwidth and fast response required for graphics cards. This is slowly being replaced by the newer *PCI Express* or *PCI-E* bus which addresses many limitations of the original PCI design.

We'll learn more about the Linux file system in later tutorials in this series, but right now we'll introduce you the */proc* filesystem. This is not a real filesystem on disk, but a "pseudo file system" which provides information about the running system. Within this file system, the file */proc/pci* contains information about the devices on the system's PCI bus. There has been some discussion about discontinuing this particular file, as the `lspci` command gives similar information. Run the command `cat /proc/pci` to see output which will look something like Listing 1.

Listing 1. */proc/pci*

```
PCI devices found:
Bus 0, device 0, function 0:
  Host bridge: Intel Corp. 82845G/GL [Brookdale-G] Chipset Host Bridge
  (rev 1).
  Prefetchable 32 bit memory at 0xd0000000 [0xdfffffff].
Bus 0, device 2, function 0:
  VGA compatible controller: Intel Corp. 82845G/GL [Brookdale-G] Chipset
  Integrated Graphics Device (rev 1).
  IRQ 11.
  Prefetchable 32 bit memory at 0x88000000 [0x8fffffff].
  Non-prefetchable 32 bit memory at 0x80000000 [0x8007ffff].
Bus 0, device 29, function 0:
  USB Controller: Intel Corp. 82801DB USB (Hub #1) (rev 1).
  IRQ 11.
  I/O at 0x1800 [0x181f].
Bus 0, device 29, function 1:
  USB Controller: Intel Corp. 82801DB USB (Hub #2) (rev 1).
  IRQ 10.
  I/O at 0x1820 [0x183f].
Bus 0, device 29, function 2:
  USB Controller: Intel Corp. 82801DB USB (Hub #3) (rev 1).
  IRQ 5.
  I/O at 0x1840 [0x185f].
Bus 0, device 29, function 7:
  USB Controller: Intel Corp. 82801DB USB2 (rev 1).
  IRQ 9.
  Non-prefetchable 32 bit memory at 0xc0080000 [0xc00803ff].
Bus 0, device 30, function 0:
  PCI bridge: Intel Corp. 82801BA/CA/DB/EB PCI Bridge (rev 129).
  Master Capable. No bursts. Min Gnt=4.
Bus 0, device 31, function 0:
  ISA bridge: Intel Corp. 82801DB LPC Interface Controller (rev 1).
Bus 0, device 31, function 1:
  IDE interface: Intel Corp. 82801DB Ultra ATA Storage Controller
```

```

    (rev 1).
    IRQ 5.
    I/O at 0x1860 [0x186f].
    Non-prefetchable 32 bit memory at 0x60000000 [0x600003ff].
Bus 0, device 31, function 3:
    SMBus: Intel Corp. 82801DB/DBM SMBus Controller (rev 1).
    IRQ 9.
    I/O at 0x1880 [0x189f].
Bus 0, device 31, function 5:
    Multimedia audio controller: Intel Corp. 82801DB AC'97 Audio
    Controller (rev 1).
    IRQ 9.
    I/O at 0x1c00 [0x1c0f].
    I/O at 0x18c0 [0x18ff].
    Non-prefetchable 32 bit memory at 0xc0080c00 [0xc0080dff].
    Non-prefetchable 32 bit memory at 0xc0080800 [0xc00808ff].
Bus 2, device 8, function 0:
    Ethernet controller: Intel Corp. 82801BD PRO/100 VE (LOM) Ethernet
    Controller (rev 129).
    IRQ 9.
    Master Capable. Latency=66. Min Gnt=8.Max Lat=56.
    Non-prefetchable 32 bit memory at 0xc0100000 [0xc0100fff].
    I/O at 0x2000 [0x203f].

```

You might want to compare this with the output from the `lspci` command. This is usually on the path of the root user, but non-root users will probably need to give the full path `/sbin/lspci`. Try these on your own system.

IO Ports

When the CPU needs to communicate with a peripheral device it does so through an *IO port* or sometimes just simply *port*. When the CPU wants to send data or control information to the peripheral, it writes to a port. When the device has data or status ready for the CPU, the CPU reads the data or status from a port. Most devices have more than one port associated with them, typically a small power of 2, such as 8, 16 or 32. Data transfer is usually done a byte or two at a time. Devices cannot share ports, so if you have ISA cards, you must ensure that each device has its own port or ports assigned. Originally, this was done using switches or jumpers on the card. Some later ISA cards used a system called *Plug and Play* or *PnP* which will discuss later in this section. PCI cards all have PnP configuration.

Within the `/proc` file system, the file `/proc/ioports` tells us about the IO ports available on the system. Run the command `cat /proc/ioports` to see output which will look something like Listing 2.

Listing 2. /proc/ioports

```

0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2

```

```

00f0-00ff : fpu
0170-0177 : idel
01f0-01f7 : ide0
02f8-02ff : serial(auto)
0376-0376 : idel
0378-037a : parport0
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
0cf8-0cff : PCI conf1
1800-181f : Intel Corp. 82801DB USB (Hub #1)
    1800-181f : usb-uhci
1820-183f : Intel Corp. 82801DB USB (Hub #2)
    1820-183f : usb-uhci
1840-185f : Intel Corp. 82801DB USB (Hub #3)
    1840-185f : usb-uhci
1860-186f : Intel Corp. 82801DB Ultra ATA Storage Controller
    1860-1867 : ide0
    1868-186f : idel
1880-189f : Intel Corp. 82801DB/DBM SMBus Controller
18c0-18ff : Intel Corp. 82801DB AC'97 Audio Controller
    18c0-18ff : Intel ICH4
1c00-1cff : Intel Corp. 82801DB AC'97 Audio Controller
    1c00-1cff : Intel ICH4
2000-203f : Intel Corp. 82801BD PRO/100 VE (LOM) Ethernet Controller
    2000-203f : e100

```

The port numbers are in hexadecimal (base 16). You'll doubtless see several that look familiar, such as keyboard, timer, parallel (printer), serial (modem) and display (vga+). Compare these with the some of the standard IO port assignments for a PC as shown in Listing 3. Notice, for example, that the first parallel port is (parport0) has the address range 0378 to 037A allocated in the /proc/iports listing, but the standard allows it (LPT!) to use the range 378 through 37F.

Listing 3. Standard I/O Port Settings

```

1F0-1F8 - Hard Drive Controller, 16-bit ISA
200-20F - Game Control
210 - Game I/O
220 - Soundcard
278-27F - LPT2
2F8-2FF - COM2
320-32F - Hard Drive Controller, 8-bit ISA
378-37F - LPT1
3B0-3BF - Monochrome Graphics Adapter (MGA)
3D0-3DF - Colour Graphics Adapter (CGA)
3F0-3F7 - Floppy Controller
3F8-3FF - COM1

```

Interrupts

So how does the CPU know when the last output is finished or when data is waiting to be read? Usually, this information is available in a status register which may be accessed by reading one (or more) of the IO ports associated with a device. Two obvious problems arise with this scenario. Firstly, the CPU has to spend time checking the status. Secondly, if the device has data coming from somewhere, such as an attached modem, the data must be read by the CPU in a timely fashion

otherwise it might be overwritten by the next available data byte.

The dual problems of not wasting unnecessary CPU cycles and ensuring that data is read or written in a timely fashion are addressed by the concept of *interrupts*. Interrupts are also called *Interrupt Requests* or *IRQs*. When something happens in a device that the CPU needs to know about, the device raises an interrupt and the CPU temporarily stops whatever else it was doing to deal with the situation.

With our experience from the last section, it should hardly come as a surprise that information on interrupts is also kept in the /proc file system, in /proc/interrupts. Run the command `cat /proc/interrupts` to see output which will look something like Listing 4.

Listing 4. /proc/interrupts

```

          CPU0
0:   226300426      XT-PIC  timer
1:     92913       XT-PIC  keyboard
2:         0       XT-PIC  cascade
5:         0       XT-PIC  usb-uhci
8:         1       XT-PIC  rtc
9:   2641134       XT-PIC  ehci-hcd, eth0, Intel ICH4
10:        0       XT-PIC  usb-uhci
11:   213632       XT-PIC  usb-uhci
14:  1944208       XT-PIC  ide0
15:  3562845       XT-PIC  ide1
NMI:         0
ERR:         0

```

This time, the interrupt numbers are decimal in the range 0 through 15. Once again, Compare these with the standard IRQ assignments for a PC as shown in Listing 5.

Listing 5. Standard IRQ Settings

```

IRQ 0 - System Timer
IRQ 1 - Keyboard
IRQ 2(9) - Video Card
IRQ 3 - COM2, COM4
IRQ 4 - COM1, COM3
IRQ 5 - Available (LPT2 or Sound Card)
IRQ 6 - Floppy Disk Controller
IRQ 7 - LPT1
IRQ 8 - Real-Time Clock
IRQ 9 - Redirected IRQ 2
IRQ 10 - Available
IRQ 11 - Available
IRQ 12 - PS/2 Mouse
IRQ 13 - Math Co-Processor
IRQ 14 - Hard Disk Controller
IRQ 15 - Available

```

Originally, each device had its own private IRQ. In Listing 5, note, for example, that IRQ5 was often used for **either** a sound card or a second parallel (printer) port. If you wanted both, you had to find a card that could be configured (usually via

hardware jumper settings) to use another IRQ such as IRQ15.

Today, PCI devices share IRQs, so that when one interrupts the CPU, an interrupt handler checks to see if the interrupt is for it and, if not, passes it to the next handler in the chain. Listings 4 and 5 do not tell us about this sharing. We will learn about the `grep` command in a later tutorial, but for now we can use it to filter the output from the `dmesg` command to look for bootstrap messages about IRQs as shown in Listing 6. We've highlighted the shared interrupts here.

Listing 6. Interrupts found during bootstrap

```
[ian@lyrebird ian]$ dmesg | grep -i irq
PCI: Discovered primary peer bus 01 [IRQ]
PCI: Using IRQ router PIIX [8086/24c0] at 00:1f.0
PCI: Found IRQ 5 for device 00:1f.1
PCI: Sharing IRQ 5 with 00:1d.2
Serial driver version 5.05c (2001-07-08) with MANY_PORTS MULTIPORT
SHARE_IRQ SERIAL_PCI ISAPNP enabled
ttyS0 at 0x03f8 (irq = 4) is a 16550A
ttyS1 at 0x02f8 (irq = 3) is a 16550A
PCI: Found IRQ 5 for device 00:1f.1
PCI: Sharing IRQ 5 with 00:1d.2
ICH4: not 100% native mode: will probe irqs later
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0x170-0x177,0x376 on irq 15
PCI: Found IRQ 11 for device 00:1d.0
PCI: Sharing IRQ 11 with 00:02.0
usb-uhci.c: USB UHCI at I/O 0x1800, IRQ 11
PCI: Found IRQ 10 for device 00:1d.1
usb-uhci.c: USB UHCI at I/O 0x1820, IRQ 10
PCI: Found IRQ 5 for device 00:1d.2
PCI: Sharing IRQ 5 with 00:1f.1
usb-uhci.c: USB UHCI at I/O 0x1840, IRQ 5
PCI: Found IRQ 9 for device 00:1d.7
ehci-hcd 00:1d.7: irq 9, pci mem f885d000
parport0: irq 7 detected
PCI: Found IRQ 9 for device 02:08.0
PCI: Found IRQ 9 for device 02:08.0
parport0: irq 7 detected
PCI: Found IRQ 11 for device 00:02.0
PCI: Sharing IRQ 11 with 00:1d.0
PCI: Found IRQ 9 for device 00:1f.5
PCI: Sharing IRQ 9 with 00:1f.3
i810: Intel ICH4 found at IO 0x18c0 and 0x1c00, MEM 0xc0080c00 and
0xc0080800, IRQ 9
```

DMA

We mentioned earlier that communication with peripheral devices through IO ports occurs a byte or two at a time. For a fast device, servicing interrupts could use a lot of the CPU's capability. A faster method is to use *Direct Memory Access* or *DMA*, in which a few IO instructions tell the device where in RAM to read or write data and then the DMA controller provides hardware management of the actual transfer of data between RAM and the peripheral device.

Hands up anyone who can guess where we find information about the DMA channels are in use. If you said it is in `/proc/dma`, then you are right. Run the

command `cat /proc/dma` to see output which will look something like Listing 7.

Listing 7. /proc/dma

```
4: cascade
```

Is that all? It is important to remember that most devices will only request one of the limited number of DMA channels when IO is actually happening, so `/proc/dma` will frequently look nearly empty as in our example. We can also scan the bootstrap messages for evidence of DMA capable devices as we did for IRQs above. Listing 8 shows typical output.

Listing 8. /proc/dma

```
[ian@lyrebird ian]$ dmesg | grep -i dma
 ide0: BM-DMA at 0x1860-0x1867, BIOS settings: hda:DMA, hdb:pio
 ide1: BM-DMA at 0x1868-0x186f, BIOS settings: hdc:DMA, hdd:DMA
 hda: 312581808 sectors (160042 MB) w/8192KiB Cache,
     CHS=19457/255/63, UDMA(100)
 hdc: 398297088 sectors (203928 MB) w/7936KiB Cache,
     CHS=24792/255/63, UDMA(33)
 ehci-hcd 00:1d.7: enabled 64bit PCI DMA
```

Plug and play

Early PCs allocated fixed port numbers and IRQs for particular devices, such as keyboard or parallel printer port. This made it difficult to add new devices or even run two devices of the same type such as two modems or two printers. The first serial port was usually called COM1 and the second COM2. Linux systems usually refer to these as `ttyS0` and `ttyS1`. Some cards were configurable usually with hardware jumpers which allowed a modem to operate as either COM1 or COM2, for example. As devices proliferated and the original space allocated for IO port addresses and IRQs became scarce, *Plug and Play* or *PnP* was developed. The idea was to allow a device to tell the system how many and what kind of resources it needed and for the BIOS to then tell the device which particular resources it should use. This semi-automatic configuration was introduced with the IBM PS/2 which used a bus architecture called *microchannel*. Later, the idea, and the plug and play name were used for ISA cards, particularly modems and sound cards which were popular add-on cards at the time. The PCI bus advanced the idea further and all PCI devices are inherently plug and play.

If you happen to work on a system with ISA PnP devices, be aware that you must avoid port and IRQ conflicts between devices. Ports cannot be shared between two devices; each device **must** have its own ports. The same applies for DMA channels. With few exceptions, ISA devices cannot share IRQs either. If you have non-PnP

devices, you must manually configure each device so that it does not interfere with another device. The promise of PnP was that configuration could be performed automatically. However, with some ISA devices not participating in PnP, this does not always work perfectly. You may be able to resolve conflicts using the `isapnptools` that we will discuss next, or you may have to reassign some of the ports or IRQs on non-PnP devices in order to get a working system.

Prior to the 2.4 kernel, a package called *isapnptools* allows a user to configure PnP devices. The `isapnp` command interprets a configuration file (normally `/etc/isapnp.conf`) to configure PnP devices. This is usually done during the Linux boot process. The `pnpdump` command scans PnP devices and dumps a list of resources your PnP cards either need or would prefer to use. The format is suitable for use by the `isapnp` command, once you uncomment the actual commands that you wish to use. You must be sure to avoid resource conflicts. Refer to the man pages for `isapnp` and `pnpdump` for more information on using these commands.

Since the 2.4 kernel, PnP support has been integrated into the Linux kernel and the `isapnptools` package has become obsolete. For example, it was removed from Red Hat 7.3 which was released in May 2002. The support is similar to the PCI support discussed earlier. You can use the `lspnp` command (part of the `kernel-pcmcia-cs` package) to display information about PnP devices. You will also find this information in the `/proc` file system if the BIOS found PnP devices during initialization. The file `/proc/bus/pnp` will contain this information. This file will not be present on a PCI-only system.

IDE Hard drives

On modern PC systems, *Integrated Drive Electronics* or *IDE* hard drives are the most common. These are also known as *AT Attachment* or *ATA* drives after the original IBM PC-AT. Another type of drive using the *Small Computer System Interface* or *SCSI* interface is also popular, particularly on server machines. IDE drives have an advantage of low cost, while the SCSI interface permits attachment of a larger number of drives, with higher potential for overlapping operations to different drives on the same bus, and therefore higher potential performance.

A new type of drive, called *Serial ATA* or *SATA* has recently entered the market. The SATA specification seeks to address some of the limitations of the ATA specification while preserving significant compatibility with ATA.

BIOS and IDE drive sizes

IDE drives are formatted into *sectors*, data units of 512 bytes. A drive might contain multiple rotating disk platters, so the sectors are arranged in concentric circles with each circle called a *cylinder*. Data from a particular platter is read or written by a *head*. To find the data in a particular sector, the disk moves the head assembly to

the cylinder, selects the appropriate head and waits for the right sector to come under the head. This gives rise to the notion of *CHS* (for Cylinder, Head and Sector) addressing. You may also hear this called *disk geometry*.

Unfortunately for history, early BIOS implemented a limit to the size permitted for each of the C, H and S values and DOS, a popular operating system for the PC, implemented a different limitation. During the 1990s, Disk sizes quickly outstripped the artificial CHS limitations imposed by BIOS and DOS. Several intermediate strategies involved translating the real CHS values to "virtual" values that would meet the constraints, either in the BIOS itself or by means of low level software routines such as Ontrack's Disk Manager software.

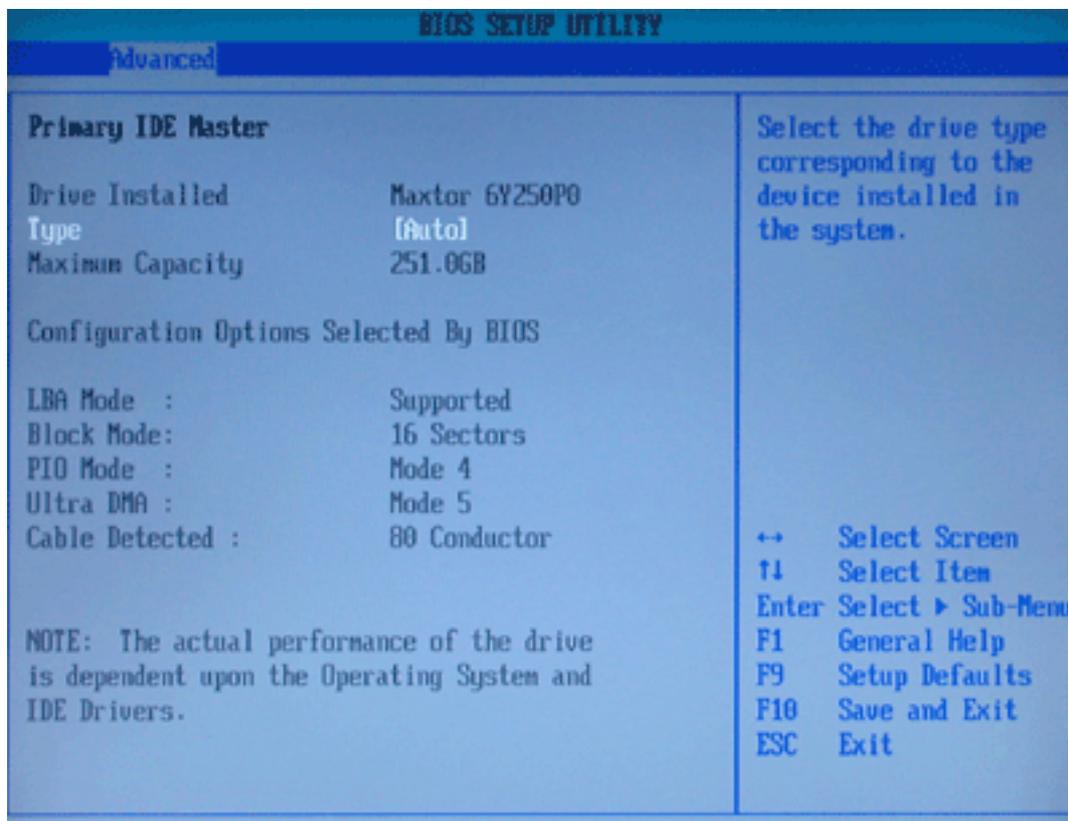
Even without the artificial limits of BIOS or DOS, the CHS design allows for up to 65536 cylinders, 16 heads, and 255 sectors/track. This limits the capacity to 267386880 sectors, or approximately 137 GB. Note that disk capacities, unlike some other PC values, are measured in powers of 10, so 1GB=1,000,000,000 bytes.

The solution was to have the system ignore the geometry and leave that to the drive to figure out. The system, instead of asking for a CHS value simply asks for a *Logical Block Address* or *LBA* and the drive electronics figure out which real sector to read or write. The process was standardized in 1996 with the adoption of the ATA-2 standard (ANSI standard X3.279-1996, *AT Attachment Interface with Extensions*).

As we discussed earlier, BIOS is needed to boot a system, so booting from a hard drive requires that the BIOS understand enough of the disk layout to locate and load the initial program that will then load the full operating system. An older BIOS that does not understand LBA disks will probably be limited to booting from within the first 1024 cylinders of a disk, or at least the first 1024 cylinders as the BIOS understands the disk geometry! Such a BIOS is probably now fairly rare, but if you do need to work with one, it may have a setting for LBA support and you may need to locate the /boot directory in a partition within the first 1024 cylinders. Even when your system will happily boot from the very end of a very large disk, many Linux partitioning tools will warn you that a partition extends beyond the 1024 cylinder limit.

Figure 3 shows information available in the BIOS of my Intel motherboard for the 250GB IDE disk on one of my Linux systems.

Figure 3. BIOS view of a large LBA disk



Listing 9 shows part of the output available on a Linux system (Fedora Core 3 in this case) using the `hdparm -I /dev/hda` command for the same disk as was used in Figure 3. Note that CHS values limit addressing to 4,128,705 sectors and the LBA value is set to 268,435,455 sectors or 137GB. These values together imply that the real capacity is in the LBA48 value. This is 490,234,752 sectors or 251GB.

Listing 9. Output from `hdparm -I /dev/hda`

```

/dev/hda:
ATA device, with non-removable media
  Model Number:      Maxtor 6Y250P0
  Serial Number:    Y638VBWE
  Firmware Revision: YAR41BW0
Standards:
  Supported: 7 6 5 4
  Likely used: 7
Configuration:
Logical          max      current
cylinders       16383  65535
heads           16       1
sectors/track   63       63
--
CHS current addressable sectors: 4128705
LBA  user addressable sectors: 268435455
LBA48 user addressable sectors: 490234752
device size with M = 1024*1024: 239372 MBytes
device size with M = 1000*1000: 251000 MBytes (251 GB)
Capabilities:

```

```
LBA, IORDY(can be disabled)
Queue depth: 1
...
```

While we are discussing booting, one other point should be noted. By default, a PC will boot from the first IDE drive in the system. Some systems have BIOS settings that will allow you to override this, but most will boot this way. The system will first load a small piece of code from the *master boot record* and that will, in turn, provide information on which partition to boot. We will cover more about boot loaders for Linux in a later tutorial.

If you'd like to know even more about the history of large disks, see [Resources](#) for a link to the *Large Disk HOWTO* which is available from the Linux Documentation Project.

Linux disk names

We will cover a lot more about how Linux uses disks in later tutorials in this series. However, right now is a good time to introduce you to another important Linux file system, the */dev* filesystem. This, like */proc*, is a pseudo file system which describes the devices that are or could be on a Linux system. Within the */dev* filesystem you will find entries such as */dev/hda*, */dev/hda5*, */dev/sda*, */dev/sdb1* and so on. You will find lots of other entries for other device types, but for now lets look at the ones that start with either */dev/hd* or */dev/sd*.

Devices that start with */dev/hd*, such as */dev/hda* or */dev/hda5* refer to IDE drives. The first drive on the first IDE controller is */dev/hda* and the second one, if present, is */dev/hdb*. Likewise, the first drive on the second IDE controller is */dev/hdc* and the second one is */dev/hdd*. As you can see from Listing 10, there are many more defined in */dev* than are likely on your system.

Listing 10. */dev/hd?* and */dev/sd?* entries

```
[ian@lyrebird ian]$ ls /dev/hd?
/dev/hda /dev/hdd /dev/hdg /dev/hdj /dev/hdm /dev/hdp /dev/hds
/dev/hdb /dev/hde /dev/hdh /dev/hdk /dev/hdn /dev/hdq /dev/hdt
/dev/hdc /dev/hdf /dev/hdi /dev/hdl /dev/hdo /dev/hdr
[ian@lyrebird ian]$ ls /dev/sd?
/dev/sda /dev/sde /dev/sdi /dev/sdm /dev/sdq /dev/sdu /dev/sdy
/dev/sdb /dev/sdf /dev/sdj /dev/sdn /dev/sdr /dev/sdv /dev/sdz
/dev/sdc /dev/sdg /dev/sdk /dev/sdo /dev/sds /dev/sdw
/dev/sdd /dev/sdh /dev/sdl /dev/sdp /dev/sdt /dev/sdx
```

As we did earlier for IRQs, we can use the *dmesg* command to find out what disk devices were found during bootstrap, Output from one of my systems is shown in Listing 11.

Listing 11. Hard drives found during bootup

```
[ian@lyrebird ian]$ dmesg | grep "[hs]d[a-z]"
Kernel command line: ro root=LABEL=RHEL3 hdd=ide-scsi
ide_setup: hdd=ide-scsi
   ide0: BM-DMA at 0x1860-0x1867, BIOS settings: hda:DMA, hdb:pio
   ide1: BM-DMA at 0x1868-0x186f, BIOS settings: hdc:DMA, hdd:DMA
hda: WDC WD1600JB-00EVA0, ATA DISK drive
hdc: Maxtor 6Y200P0, ATA DISK drive
hdd: SONY DVD RW DRU-700A, ATAPI CD/DVD-ROM drive
hda: attached ide-disk driver.
hda: host protected area => 1
hda: 312581808 sectors (160042 MB) w/8192KiB Cache,
    CHS=19457/255/63, UDMA(100)
hdc: attached ide-disk driver.
hdc: host protected area => 1
hdc: 398297088 sectors (203928 MB) w/7936KiB Cache,
    CHS=24792/255/63, UDMA(33)
   hda: hda1 hda2 hda3 hda4 < hda5 hda6 hda7 hda8 hda9 hda10 hda11 >
   hdc: hdc1 < hdc5 hdc6 hdc7 hdc8 >
hdd: attached ide-scsi driver.
```

From the highlighted lines in Listing 11, we see that the system has two IDE drives (hda and hdc) and a DVD-RW drive (hdd). Note that there is no hdb, indicating that there is no second drive on the first IDE controller on this system. An IDE drive can have up to four *primary* partitions and an unlimited number of *logical* partitions. Considering the drive hdc in Listing 11, we see that it has one primary partition (hdc1) and four logical partitions (hdc5, hdc6, hdc7, and hdc8). We will see in Topic 104 in a later tutorial in this series that hdc1 is actually a container (or *extended* partition) for the logical partitions.

Historically, devices such as sda and sdb were SCSI disks, which we will discuss further when we see how to [set up SCSI devices](#). Up to the 2.4 kernel, IDE CD and DVD devices were usually handled through SCSI emulation. Such a device often appeared in /dev as something like /dev/cdrom which was a symbolic link to the SCSI emulated device. For the above system, Listing 12 shows that /dev/cdrom is a link to /dev/scd0 rather than to /dev/hdd as might have been expected. Note the hdd=ide-scsi kernel parameter in Listing 11 as well as the indication that the ide-scsi driver was attached for hdd.

Listing 12. IDE SCSI emulation

```
[ian@lyrebird ian]$ ls -l /dev/cdrom
lrwxrwxrwx 1 root  root   9 Jan 11 17:15 /dev/cdrom -> /dev/scd0
```

Today, you will find that both USB and SATA storage devices appear as sd, rather than hd, devices.

Legacy peripherals

We have alluded above to peripherals such as serial or parallel ports that are usually

integrated into a motherboard, and we have seen some standard IO port and IRQ assignments for these devices. Serial ports, in particular, have been used for connecting a variety of devices and they have a history of being hard to configure. With the advent of *IEEE 1394*, also known as *Firewire* and *Universal Serial Bus* or *USB* devices, automatic configuration and hot plugging of devices has largely replaced the chore of ensuring correct serial or parallel port configuration. Indeed, a *legacy-free* system does not support the standard serial or parallel ports. Neither does it support a floppy drive or a PS/2 connected keyboard or mouse.

We'll now discuss some common BIOS settings that you may need to configure.

Serial ports (COMn)

The legacy serial ports are known as COM1 through COM4. If your system has a single serial port connector (originally a 25-pin DB25 connector but now more commonly a 9-pin DB9 connector) it will probably use the default base address and IRQ for COM1, namely IO port 3F8 and IRQ 4. The standard IO port addresses and IRQs for serial ports are shown in Table 3.

Name	Address	IRQ
COM1	3F8-3FF	4
COM2	2F8-2FF	3
COM3	3E8-3EF	4
COM4	2E8-2EF	3

You will notice that COM1 and COM3 share IRQ 4 and likewise COM2 and COM4 share IRQ 3. Unless the driver and the device can actually share the interrupt, or a device does not use interrupts, this means that most real systems will use only COM1 and COM2.

Occasionally, you may need to either disable an onboard serial port or configure it to use an alternate address and IRQ. The most likely reason to do this is because of conflicts with a PnP modem in an ISA slot or a desire to use the PnP modem as COM1. We recommend that you only change these if you are having problems with Linux detecting your configuration.

Parallel ports (LPTn)

The legacy parallel ports are known as LPT1 through LPT4, although usually only at most two are present. If your system has a single parallel port connector it will probably use the default base address and IRQ for LPT1, namely IO port 378 and IRQ 7. The standard IO port addresses and IRQs for parallel ports are shown in Table 4.

Table 4. Parallel port assignments

Name	Address	IRQ
LPT1	378-37F	7
LPT2	278-27F	5
LPT*	3BC-3BE	

Note that the IO ports 3BC-3BE were originally used on a Hercules graphics adapter that also had a parallel port. Many BIOS systems will assign this range to LPT1 and then the other two ranges would become LPT2 and LPT3 respectively instead of LPT1 and LPT2.

Many systems do not use interrupts for printers, so the IRQ may or may not actually be used. It is also not uncommon to share IRQs for printing and also to share IRQ 7 with a sound card (Sound Blaster compatible).

The parallel ports were originally used for printing with data flowing to the printer and a few lines reserved for reporting status. Later, the parallel port was used for attaching a variety of devices (including early CD-ROMs and tape drives), so the output-only nature of the data flow changed to a bidirectional data flow.

The current standard applicable to parallel ports is *IEEE Std. 1284-1994 Standard Signaling Method for a Bi-Directional Parallel Peripheral Interface for Personal Computers* which defines five signaling modes. Your BIOS may give you choices in setup such as *bi-directional*, *EPP*, *ECP* and *EPP and ECP*. ECP stands for *Enhanced Capabilities Port* and is designed for use with printers. EPP stands for *Enhanced Parallel Port* and is designed for devices such as CD-ROMs and Tape drives which require large amounts of data to flow in either direction. The default BIOS choice is likely to be ECP. As for serial ports, change this only when you have a device that does not work properly.

Floppy disk port

If your system has a legacy floppy disk controller, it will use ports 3F0-3F7. If you install a legacy floppy drive in a system that shipped without one, you may have to enable legacy options in your BIOS. Consult the manufacturer's information for more details.

Keyboard and mouse

The keyboard/mouse controller uses ports 0060 and 0064 for legacy keyboards and mice. That is, those connected by a round PS2 connector. Many systems will generate a Power-On-Self-Test (POST) error if a keyboard is not attached. Most machines designed to be used as servers, and many desktops, now have BIOS

options to allow clean startup without a keyboard or mouse present.

Once a system is installed, running without a keyboard (or mouse) is seldom a problem. Servers frequently run this way. Management is performed over the network using either web administration tools, or a command line interface such as telnet or (preferably) ssh.

Installation on a keyboardless system is usually accomplished using a terminal (or terminal emulator) attached through a serial port. Usually, you will need a keyboard and display to ensure that the BIOS is set up correctly with an enabled serial port. You may also need a customized boot disk or CD to perform a Linux system install.

Another approach used by systems such as the IBM JS20 blade server is to emulate a serial connection over a LAN.

Section 3. Modems and sound cards

This section covers material for topic 1.101.3 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

Modems

A *modem* (from *modulator/demodulator*) is a device for converting the digital signals used in computers to a serial stream of analog data that is transmitted over telephone lines. In the early days of PCs, modems were external devices that were attached to a serial port. Later, modems were implemented on cards that could be installed inside the computer, reducing cost for housing and power, and eliminating the need for a cable between serial port and modem. Another cost reduction occurred when some of the function normally done by a modem was transferred to software in the PC. This type of modem may be called a softmodem, HCF modem, HSP modem, HSF modem or controllerless modem, among other terms. Such modems were designed to reduce the cost of systems which generally ran Microsoft Windows. The term *winmodem* is often used for such devices, although Winmodem® is a registered trademark of U.S. Robotics, who manufactured several modems under that name.

Most external modems and full function internal modems will work under Linux without problem. Some of the modems that require software assistance from the PC operating system will also work with Linux and the list of working modems in this category is continually increasing. Software-assisted modems that work under Linux are often called *linmodems* and there is a site dedicated to these (linmodems.org). If

you have such a modem, your first step should be to check the [linmodems](#) site (see [Resources](#)) and download the latest version of the scanModem tool. This will tell you what is already known about available drivers (if any) for your modem.

If you have an ISA modem, you will need to ensure that ports, IRQs and DMA channels do not conflict with other devices. See the earlier section [BIOS settings](#) for additional information.

The modems discussed in this section are *asynchronous* modems. There is another class of modems, called *synchronous* modems used for HDLC, SDLC, BSC or ISDN. Very loosely, we can say that asynchronous transmission is concerned with transmitting individual bytes of information while synchronous communications is concerned with transmitting whole blocks of information.

Most Linux communications occurs using the *Internet Protocol* or *IP*. So a Linux system will need to run what looks like IP over an asynchronous line which was not originally designed for block protocols such as IP. The first method of doing this was called *Serial Line Interface Protocol* or *SLIP*. A variant using compressed headers is called *CSLIP*. Nowadays, most Internet Service Providers (ISPs) support dialup connections using *Point-to-Point Protocol* or *PPP*.

The *Linux Networking-HOWTO* and *The Network Administrators' Guide* available from the Linux Documentation Project (see [Resources](#)) provide information on SLIP, CSLIP and PPP configuration.

When communicating using a modem, there are a number of settings that you may need to make on your Linux system. Most importantly, you will set the speed of communications between your system and the modem. This will usually be higher than the nominal line speed and is usually set to the maximum supported by your serial port chipset and your modem. One way to set or view the modem parameters that will be used by the serial driver is with the `setserial` program. We illustrate the `setserial` command in Listing 13. Note that the `-G` option prints the output in a format suitable for use in setting parameters with `setserial`. In this case, the UART (Universal Asynchronous Receiver Transmitter) is a buffered 16550 which is a common type of UART on modern PCs. The speed is set of 115,200 bps which is also commonly used with this UART and most modern external 56kbps modems. Note that the default speed on some newer systems may be set as high as 460,800bps. If your modem does not appear to respond, this is probably the first thing you should check.

Listing 13. The `setserial` command

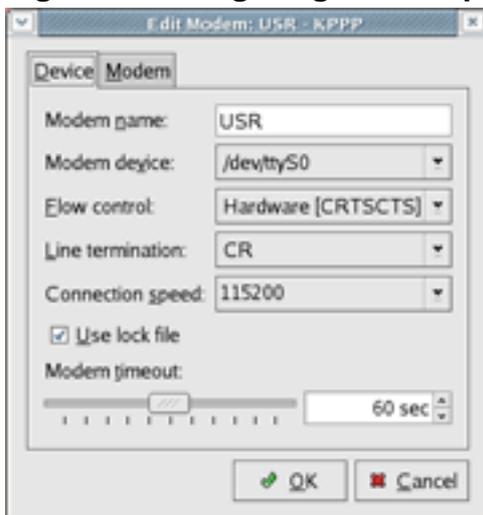
```
[root@attic4 ~]# setserial /dev/ttyS0
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
[root@attic4 ~]# setserial -G /dev/ttyS0
/dev/ttyS0 uart 16550A port 0x03f8 irq 4 baud_base 115200 spd_normal skip_test
```

One thing to note about `setserial` is that it does not probe the hardware. All it does is tell the serial driver what parameters to use, unless you use the `autoconfig` and `auto_irq` parameters. In this case, `setserial` will ask the kernel to probe the hardware. See the man pages for `setserial` for more information about these and other options of the command.

We will cover networking more in a tutorial for LPI exam 102 (See [Resources](#)). In the meantime, if you wish to set up a PPP connection, there are several excellent tools to help you do this. The `kppp` program has a nice GUI and is easy to use. The `wvdial` command provides an intelligent command line tool for setting up dial connections. In addition to these, distributions may have other tools, either specifically for PPP or dialup connections or as part of a more general network configuration tool such as `system-config-network` in Fedora Core 4.

Another aspect of modem communications that is usually under control of the communications program but may be set or have the default values set on the modem itself is *flow control*. This is a way for one end to tell the other end to wait for a moment while the receiving end clears its data buffers. This may be done in software by sending XON and XOFF characters. The preferred way, and that used for PPP connections, is called *hardware flow control* in which the state of certain modem signal lines is used to indicate readiness to receive data. The signals used are *Clear to Send* or *CTS* and *Ready to Send* or *RTS*, so you will often see this described as flow control using RTS/CTS or something similar. Figure 4 shows how the speed and hardware flow control are set using the `kppp` program.

Figure 4. Configuring modem parameters with `kppp`



Sound cards

Most personal computers sold today include audio or *sound card* capabilities.

Sound port (Sound Blaster)

The Creative Labs Sound Blaster series of sound cards have set de facto industry standards for sound cards. Even though many other brands of excellent sound cards exist, many of these provide a compatibility mode for one or more of the Sound Blaster series. The original Sound Blaster card was an 8-bit card that worked in the original IBM PC. Later 16-bit models for the PC-AT and compatibles used the 16-bit PC-AT or ISA bus. Today, most of these cards use the PCI bus. Many motherboards even provide a sound chip with Sound Blaster compatibility on board. Sound devices may also be attached through USB connections, although we will not cover those here.

The ports used by an ISA bus Sound Blaster card are 0220-022F, although base addresses of 240, 260 or 280 were often configurable. Similarly, the IRQ is usually configurable, with common choices being 2, 5, 7, or 10. The default setting is to use IRQ 5. The cards could usually be configured to use alternate DMA channels too.

As with all ISA devices, you will need to ensure that ports, IRQs and DMA channels do not conflict with other devices. See the earlier section [BIOS settings](#) for additional information.

MIDI port (MPU-401)

Many sound cards also have an interface to attach a *MIDI* (from Musical Instrument Digital Interface) device. Commonly, this interface emulates the Roland MPU-401. The standard ports used by the MPU-401 ISA interface are 0200-020F.

As with all ISA devices, you will need to ensure that ports, IRQs and DMA channels do not conflict with other devices. See the earlier section [BIOS settings](#) for additional information.

Configuring Linux sound support

Modern 2.4 and 2.6 kernels have sound support for a wide variety of sound devices built in to the kernel, usually as modules. As with other devices, we can use the `pnpdump` command for ISA devices, or the `lspci` command for PCI devices to display information about the device. Listing 14 shows the output from `lspci` for an Intel sound system on a system motherboard.

Listing 14. Using `lspci` to display sound resources

```
[root@lyrebird root]# lspci | grep aud
00:1f.5 Multimedia audio controller: Intel Corporation 82801DB/DBL/DBM
      (ICH4/ICH4-L/ICH4-M) AC'97 Audio Controller (rev 01)
```

Kernel modules are the preferred way to provide support for a variety of devices.

Modules need only be loaded for the devices actually present and they may be unloaded and reloaded without rebooting the Linux system. For 2.4 and earlier kernels, the module configuration information is stored in `/etc/modules.conf`. For 2.6 kernels, the kernel module system was redesigned and the information is now stored in `/etc/modprobe.conf`. In either case, the `lsmod` command will format the contents of `/proc/modules` and display the status of loaded modules.

Listing 15 shows the contents of `/etc/modprobe.conf` for a 2.6 kernel and Listing 16 shows the output from `lsmod` as it relates to sound devices on this system.

Listing 15. Sample `/etc/modprobe.conf` (2.6 kernel)

```
[root@attic4 ~]# cat /etc/modprobe.conf
alias eth0 e100
alias snd-card-0 snd-intel8x0
install snd-intel8x0 /sbin/modprobe --ignore-install snd-intel8x0 &&\
  /usr/sbin/alsactl restore >/dev/null 2>&1 || :
remove snd-intel8x0 { /usr/sbin/alsactl store >/dev/null 2>&1 || : ; } ; \
/sbin/modprobe -r --ignore-remove snd-intel8x0
alias usb-controller ehci-hcd
alias usb-controller1 uhci-hcd
```

Listing 16. Sound related output from `lsmod` (2.6 kernel)

```
[root@attic4 ~]# lsmod |egrep '(snd)|(Module)'
```

Module	Size	Used by
snd_intel8x0	34689	1
snd_ac97_codec	75961	1 snd_intel8x0
snd_seq_dummy	3653	0
snd_seq_oss	37057	0
snd_seq_midi_event	9153	1 snd_seq_oss
snd_seq	62289	5 snd_seq_dummy, snd_seq_oss, snd_seq_midi_event
snd_seq_device	8781	3 snd_seq_dummy, snd_seq_oss, snd_seq
snd_pcm_oss	51185	0
snd_mixer_oss	17857	1 snd_pcm_oss
snd_pcm	100169	3 snd_intel8x0, snd_ac97_codec, snd_pcm_oss
snd_timer	33605	2 snd_seq, snd_pcm
snd	57157	11 snd_intel8x0, snd_ac97_codec, snd_seq_oss,
		snd_seq, snd_seq_device, snd_pcm_oss, snd_mixer_oss, snd_pcm, snd_timer
soundcore	10913	1 snd
snd_page_alloc	9669	2 snd_intel8x0, snd_pcm

Listing 17 shows the contents of `/etc/modules.conf` for a 2.4 kernel and Listing 18 shows the output from `lsmod` as it relates to sound devices on this system. Note the similarities between the `modules.conf` and `modprobe.conf` files.

Listing 17. Sample `/etc/modules.conf` (2.4 kernel)

```
[root@lyrebird root]# cat /etc/modules.conf
alias eth0 e100
alias usb-controller usb-uhci
alias usb-controller1 ehci-hcd
alias sound-slot-0 i810_audio
post-install sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -L >/dev/null 2>&1 || :
```

```
pre-remove sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -S >/dev/null 2>&1 || :
```

Listing 18. Sound related output from lsmod (2.4 kernel)

Module	Size	Used by	Not tainted
smbfs	43568	1 (autoclean)	
i810_audio	28824	0 (autoclean)	
ac97_codec	16840	0 (autoclean)	[i810_audio]
soundcore	6436	2 (autoclean)	[i810_audio]
st	30788	0 (autoclean)	(unused)

Sound support on many 2.4 and earlier systems is provided through the *Open Sound System (OSS) Free* drivers. Many systems today use the *Advanced Linux sound architecture* or *ALSA* drivers. The `sndconfig` utility was created by Red Hat to assist in configuring ISA PnP sound cards. It also works with PCI sound cards. This utility may be present on systems that do not use the ALSA drivers, although modern module support has made it largely unnecessary. The utility will probe for sound cards, lay a test sound of Linus Torvalds speaking, and then update the `/etc/modules.conf` file. Typical operation is shown in Figures 5 and 6.

Figure 5. The `sndconfig` utility

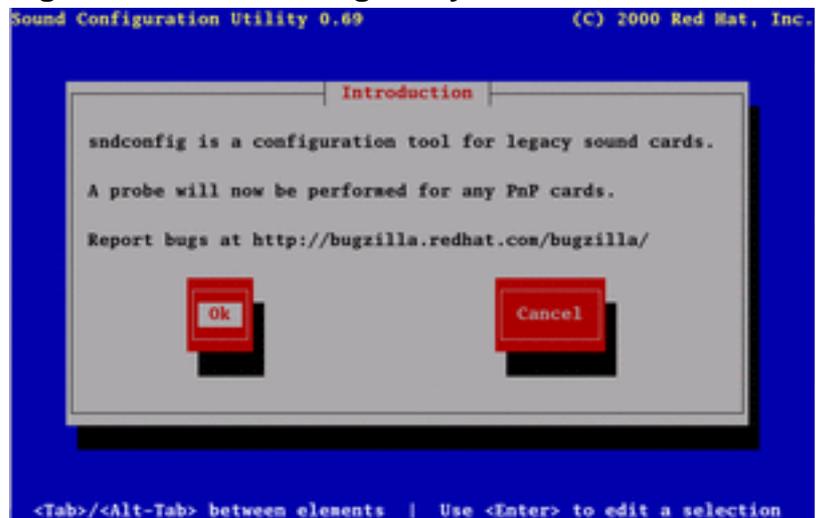
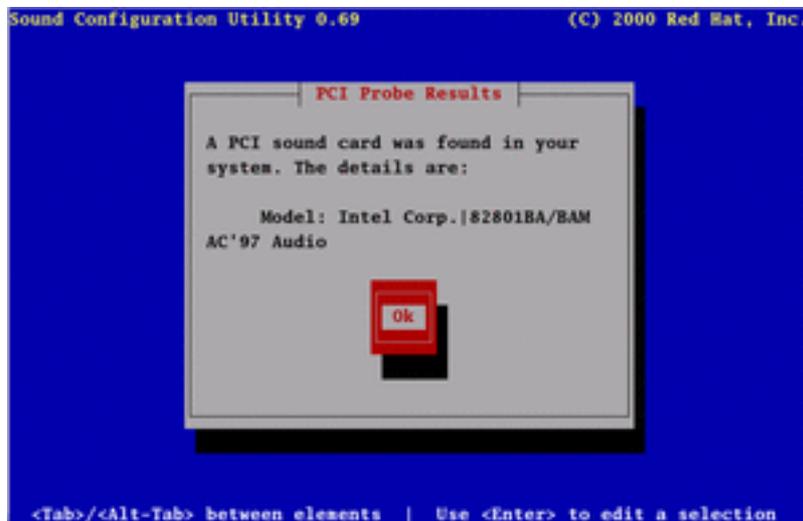


Figure 6. The `sndconfig` utility



Section 4. Set up SCSI devices

This section covers material for topic 1.101.4 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

SCSI overview

The *Small Computer System Interface*, more generally known as *SCSI*, is an interface designed for connecting streaming devices such as tapes and block storage devices such as disks, CD-ROMs, and DVDs. It has also been used for other devices, such as scanners and printers. SCSI is pronounced "scuzzy". SCSI was designed to allow multiple devices on the bus. One device, called the *controller* has responsibility for managing the bus. SCSI devices may be either internal or external.

There have been three major releases of SCSI standards from the American National Standards Institute (ANSI).

SCSI

is the original standard (X3.131-1986), now usually called *SCSI-1*. This arose from efforts by Shugart Associates to get a standard interface for disk devices. The standard supported up to 8 devices on a cable. SCSI-1 uses passive termination (more on this below). This standard has now been withdrawn, although devices may still work on current SCSI cables assuming appropriate termination. The data interface was 8 bits parallel with a maximum speed of 5 MBps (megabytes/sec). The SCSI standard was designed for disks, but is very

flexible and was used for other devices, notably scanners and slower devices such as Zip. FConnection used a 50 connector cable, originally with a Centronics connector, but later with a 50-pin D-shell connector, similar to a DB-25 RS-232 serial connector,

SCSI-2

was approved as ANSI standard X3.131-1994 in 1994. This revision doubled the speed of the bus to 10MBps as well as introducing so-called *wide* or 16-bit data transfers. A 16-bit bus running at 10MBps can transfer 20MBps of data. The 50-connector cable was used for 8-bit or *narrow* SCSI devices, while the newer wide devices used a 68-pin cable. Higher density cables were also introduced, allowing smaller and cheaper connectors. SCSI-2 also standardized the SCSI command set and introduced differential signaling to improve quality at higher speeds. This was later called *High Voltage Differential* or *HVD* signaling. HVD has active termination requirements. It is possible to mix 8-bit and 16-bit devices on a cable with appropriate care in termination. SCSI-2 supports up to 16 devices on a cable of which at most 8 may be narrow.

SCSI-3

is a set of standards rather than a single standard. This allows standards to be enhanced for technology areas that are fast-moving, while avoiding the need to revise standards for stable technology. The overall architecture is defined in ANSI standard X3.270-1996 which is also known as the *SCSI-3 Architecture Model* or *SAM*. The earlier SCSI standards are now embodied in the *SCSI Parallel Interface* or *SPI* standards. Speed was increased again and current 16-bit devices are capable of up to 320MBps data transfers at a bus speed of 160MBps.

SCSI-3 introduced Fiber Channel SCSI with support for up to 126 devices per bus allowing connection over 1GBps or 2GBps fiber channel links at distances up to several kilometers. This helps to alleviate inherent limitations involved with the use of standard SCSI cabling. Another notable introduction was *Single Connector Attachment* or *SCA* which is only used for wide (16-bit) devices. SCA is an 80-pin connector which incorporates the pins from the 68-pin connector as well as power and some additional pins. SCA is designed to allow devices to be safely hot-plugged in a running system, and is frequently used in devices implementing *Redundant Array of Independent disks* or *RAID* storage systems as well as network attached storage and server racks.

We mentioned *termination* above without saying much about it. The electrical specifications for a SCSI bus require each end of the bus to be properly terminated. You must use the appropriate type of terminator for your bus; passive, HVD or LVD. If you mix wide and narrow devices on a bus be aware that the termination for narrow devices may occur in a different place to the termination for wide devices. If the controller is controlling only an internal bus or only an external bus, it will usually

provide termination, either automatically or via BIOS configuration. Check the manuals for your particular controller. If the controller is controlling both an internal and an external segment, then it should normally not provide termination.

Some devices are capable of providing termination, either via a switch, or other means such as a jumper. Again, consult the manual for your device. Otherwise, termination is usually accomplished with a terminator block which is plugged into the cable. Whichever type of termination you use, be particularly careful if you mix wide and narrow devices on the same bus, as the narrow termination may occur at a different place on the cable than the wide termination.

SCSI Ids

By now, you may be wondering how the system manages many devices on one cable. Every device, including the controller, has an *ID*, represented by a number. For narrow (8-bit) SCSI, the ID numbers range from 0 through 7. Wide SCSI adds numbers 8 through 15. Narrow devices may only use ID numbers 0 through 7 while wide devices may use 0 through 15. The controller is generally assigned ID 7. The ID for a devices may be set via jumpers, switches or dials on the device, or through software. Devices using the Single Connector Attachment (SCA) usually have an ID assigned automatically as these devices may be hot-plugged.

Devices on a SCSI bus have a priority. Priority for narrow devices runs from 0 (lowest) through 7 (highest), so a controller at address 7 has highest priority. The extra IDs for wide SCSI have priority 8 (lowest) through 15) highest, with 15 having lower priority than 0. Thus, the overall priority sequence is 8, 9, 10, 11, 12, 13, 14, 15, 0, 1, 2, 3, 4, 5, 6, 7. Slower devices and devices that cannot tolerate delays (such as CD or DVD recorders) should be given high priority IDs to ensure they get sufficient service.

Devices such as RAID controllers may present a single ID to the bus but may incorporate several disks. In addition to the ID, the SCSI addressing allows a *Logical Unit Number* or *LUN*. Tapes and single disk drives either do not report a LUN or report an LUN of 0.

A SCSI adapter may support more than one SCSI cable or *channel*, and there may be multiple SCSI adapters in a system. The full ID of a device therefore consists of an adapter number, a channel number, a device ID and a LUN.

Devices such as CD recorders using ide-scsi emulation and USB storage devices will also appear to have their own adapter.

Linux names and files for SCSI devices

Back in the BIOS section on [IDE drives](#) we discussed the names assigned by Linux to the various IDE devices, such as `/dev/hda` and `/dev/hdc`. This is simple for an IDE controller which can support either one or two hard drives. The secondary IDE drive on the second adapter is always `/dev/hdd`, even if the only other hard drive is the primary drive on the first adapter (`/dev/hda`). With SCSI the situation becomes more complicated as we may mix hard drives, tapes, CD and DVD drives, as well as other devices on a SCSI cable.

Linux will assign device names as devices are detected during boot. Thus, the first hard drive on the first channel of the first adapter will become `/dev/sda`, the second `/dev/sdb`, and so on. The first tape drive will be `/dev/st0`, the second `/dev/st1`, and so on. The first CD device will become `/dev/sr0` or `/dev/scd0` and the second `/dev/sr1` or `/dev/scd1`. Devices using SCSI emulation, such as USB storage devices and (prior to the 2.6 kernel) IDE CD or DVD drives will also be allocated names in this name space.

While we won't cover all the intricacies of SCSI naming here, it is most important to know that this numbering is redone at each boot. If you add or remove a SCSI hard drive, then all previously higher drives will have a different device name next time you boot. The same goes for other device types. We will learn more about partitions, labels and file systems in another tutorial in this series, but for now we will warn you about one thing. Since disks can have up to 15 partitions on them, each with a name tied to the device name (for example, `/dev/sda1`, `/dev/sda2` through `/dev/sda15`), this can cause havoc when your system attempts to mount the filesystems. Plan very carefully when you add or remove SCSI devices and use disk labels rather than device names for SCSI disks whenever possible.

We introduced the `/proc` file system in the section on [BIOS settings](#). The `/proc` file system also contains information about SCSI devices. Listing 19 shows the contents of `/proc/scsi/scsi` on a system with two SCSI devices, a hard drive with ID 0 and a controller with ID 8.

Listing 19. `/proc/scsi/scsi`

```
[root@waratah root]# cat /proc/scsi/scsi
Attached devices:
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: IBM-PSG Model: DPSS-336950M F Rev: S94S
  Type: Direct-Access ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 08 Lun: 00
  Vendor: IBM Model: YGLv3 S2 Rev: 0
  Type: Processor ANSI SCSI revision: 02
```

If you want to know which real device corresponds to say `/dev/sda`, you can use the `scsi_info` command. Listing 20 confirms that our first (and only) SCSI hard drive is `/dev/sda`.

Listing 20. The `scsi_info` command

```
[root@waratah root]# scsi_info /dev/sda
SCSI_ID="0,0,0"
MODEL="IBM-PSG DPSS-336950M F"
FW_REV="S94S"
```

However, note that some systems, such as Fedora Core 2, do not include the `scsi_info` command (which is a part of the `kernel-pcmcia-cs` package).

More recent systems have switched to using the *SCSI Generic* or `sg` driver. When the `sg` driver is used, you will find additional information under the `/proc/scsi/sg` subtree in your filesystem. You will also have devices such as `/dev/sg0`, `/dev/sg1`, `/dev/sg2` and so on. These generic devices usually correspond to some other device type such as a hard disk like `/dev/sda` or a tape like `/dev/st0`.

The `sg3_utils` package contains a number of utilities for manipulating and interrogating aspects of the SCSI subsystem. In particular, the `sg_map` command will provide a map between the `sg` name and another device name if one exists. Note that scanners will not have another device name, only a generic one. Listing 21 shows the output of `sg_map` on a system with an IDE optical drive that uses SCSI emulation and two USB drives.

Listing 21. The `sg_map` command

```
[root@lyrebird root]# sg_map
/dev/sg0 /dev/scd0
/dev/sg1 /dev/sda
/dev/sg2 /dev/sdb
```

The `sg` utility corresponding to `scsi_info` is `sginfo`. You can use either the generic device name or the more familiar name with `sginfo`. Listing 22 shows the output of `sginfo` for the three devices of Listing 21. Notice that `sginfo` does not provide information for `/dev/sg1`, although as shown in the listing the `scsi_info` command does show it as a USB memory key. In this case, the device has been unplugged from the system. Information about it is retained (and can be found in `/proc/scsi/scsi`). The `sginfo` command interrogates the device for the information while the `scsi_info` will use the retained information. Thus `sginfo` must be run as root while `scsi_info` need not be run as root, although non root users may have to specify the full path of `/sbin/scsi_info`.

Listing 22. The `sginfo` command

```
[root@lyrebird root]# sginfo /dev/scd0
INQUIRY response (cmd: 0x12)
-----
Device Type                5
Vendor:                    SONY
Product:                   DVD RW DRU-700A
```

```
Revision level:          VY08

[root@lyrebird root]# sginfo /dev/sg1
INQUIRY reponse (cmd: 0x12)
-----
Device Type              0
Vendor:
Product:
Revision level:

[root@lyrebird root]# sginfo /dev/sg2
INQUIRY reponse (cmd: 0x12)
-----
Device Type              0
Vendor:                  WD
Product:                 2500JB External
Revision level:         0411

[root@lyrebird root]# scsi_info /dev/sg1
SCSI_ID="0,0,0"
MODEL=" USB DISK 12X"
FW_REV="2.00"
```

SCSI BIOS and boot sequence

While SCSI is standard on most servers, most desktop and laptop computers do not normally include SCSI support as standard. Such systems will normally boot from a floppy disk, a CD or DVD drive or the first IDE hard drive in the system. The boot order is usually configurable in BIOS setup screens such as we saw in the section [BIOS settings](#), and sometimes dynamically by pressing a key or key combination during system startup.

The BIOS Boot Specification (see [Resources](#)) defines a method for add on cards such as SCSI cards to present a message during startup and have BIOS on the card invoked for configuration purposes. SCSI cards normally use this to allow configuration of the SCSI subsystem controlled by the card. For example, an Adaptec AHA-2930U2 card will present a message

```
Press <Ctrl><A> for SCSISelect (TM) Utility!
```

allowing a user to press the ctrl and A keys together to enter the adapter BIOS. Other cards will have a similar process for entering the card BIOS to set up the card.

Once in the card BIOS, you will have screens that typically allow you to set the SCSI controller address (typically 7), the SCSI boot device (usually ID 0), the bus speed and whether the controller should provide termination or not. Some older cards may require that the boot device be ID 0, but most modern cards will allow you to choose any device. You may, and probably will, have other options, such as the ability to format a hard disk. See your card manufacturer's documentation for details. Once you have set up the SCSI view of the bus, you will usually still have to tell your PC BIOS to boot from the SCSI disk rather than an IDE drive. Consult your system reference manual to determine whether you can boot from a non-IDE drive and how

to set it if you can.

Section 5. PC expansion cards

This section covers material for topic 1.101.5 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

We covered the material that you will need to know for this section when we discussed [BIOS settings](#). You should review the discussion of DMA, IRQs, ports and the different kinds of buses and adapters in the section [Buses, ports, IRQs, and DMA](#) so you understand the contents of the `/proc/dma`, `/proc/interrupts`, and `/proc/ioports` files and how to use them to determine any conflicts. Review the material on `/proc/pci` and the `lspci` command. Also review the material in the [Plug and play](#) section for information about ISA and Plug and Play cards. There you will find information about `isapnp` and `pnpdump`.

Section 6. Communication devices

This section covers material for topic 1.101.6 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

This section covers a variety of communications devices, including modems, ISDN adapters, and DSL switches. This material for this section falls into two general categories:

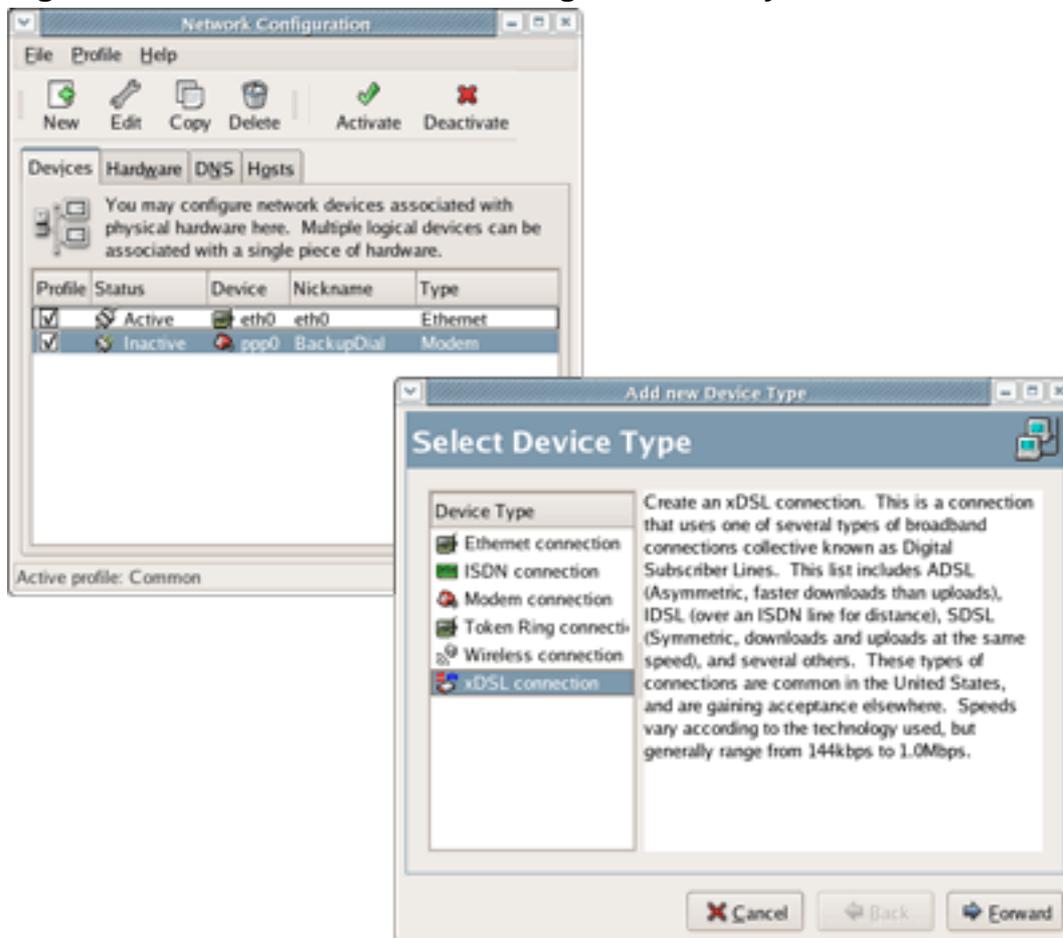
1. Selecting and installing your communications device, and
2. Communicating with your device

Selecting your communications device is like selecting any other device for your system in that it needs to match your bus type (PCI or ISA) and you need Linux support for the device. You should review the discussion of DMA, IRQs, ports and the different kinds of buses and adapters in the section [Buses, ports, IRQs, and DMA](#) so you understand the contents of the `/proc/dma`, `/proc/interrupts`, and `/proc/ioports` files and how to use them to determine any conflicts. Review the material on `/proc/pci` and the `lspci` command. Also review the material in the [Plug and play](#) section for information about ISA and Plug and Play cards. There you will

find information about isapnp and pnpdump.

The Linux kernel supports more and more devices with every release, so your first check for support should be with the distribution you are already using. If the support is already installed, your distribution may have a utility to help you configure it. Figure 7 illustrates the Fedora Core 4 network configuration tool. You can see that an ethernet connection has been configured (and is active) and a backup dial connection using PPP has also been configured. The system already supports that addition of ISDN, Token Ring, wireless and xDSL connections.

Figure 7. Fedora Core network configuration utility



If you have to install drivers for a communications device, check first to see if the required drivers are a part of your distribution that has not yet been installed and install if so. Otherwise, you should try and find a driver package that has already been built for your system. Your final choice is to build your own driver package from source. We will cover building packages in the tutorial for LPI Exam 101 Topic 102. (see [Resources](#)).

For an ISDN connection, you will also need the synchronous PPP driver, as the

normal one used with asynchronous modems is designed for character mode transmission rather than block mode. As we mentioned in the section on Modems we will discuss setting up connections more in a tutorial for LPI exam 102 (See [Resources](#)).

DSL connections may be one of several types. Some provide an ethernet port that is bridged to the ISP network. Authentication is usually done in this case using your computer's ethernet MAC address. If you attach a router (or a different computer) to the DSL modem, you may need to clone the MAC address of the computer that was originally connected in order for the connection to work. More commonly, an ISP will use *Point-to-Point Protocol over Ethernet* or *PPPoE*. In this case, you are provided with a username and password to use when establishing the connection. In this case, if you use a router, you will usually configure this address into the router and your computer will simply use a standard ethernet connection. Rarely, you may have a *PPPoA* or *PPP over ATM* connection.

Wireless connections may require you to know the name of the network you are connecting to. This is called a *Service Set Identifier* or *SSID*. If the network uses encryption such as *Wired Equivalent Privacy* or *WEP* or *WiFi Protected Access* or *WPA* you will need to configure your connection appropriately.

Section 7. USB devices

This section covers material for topic 1.101.7 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

USB overview

In this section we will look at Linux support for *Universal Serial Bus* or *USB* devices. USB was developed by a consortium of companies with the goal of providing a single, simple bus for attachment of peripherals. In the section on [BIOS settings](#), we saw the complexities of managing ports, IRQs and DMA resources in ISA bus machines. The USB design allows devices to be hot-plugged and uses standard connectors for connecting devices. USB devices include keyboards, mice, printers, scanners, hard drives, flash memory drives, cameras, modems, ethernet adapters, and speakers. The list keeps growing. Current Linux support is quite comprehensive, although some devices require special drivers and others, particularly printers, may not be supported or may be only partially supported.

A computer system may provide one or more *controllers* or *hubs*. to which either a USB device or another (external) hub may be connected. A hub can support up to 7

devices, some or all of which may be additional hubs. The hub in the system is called the *root hub*. Each such star topology can support up to 127 hubs or devices.

Note: Frequently, we speak of a *USB port* which refers to the USB capability in a computer and the connecting socket (compare with serial port or parallel port) rather than the internal port addresses used by the device.

The USB system is a layered system.

1. The *Bus Interface* layer provides physical, signaling, and packet connectivity between hosts and devices, providing data transfer between the host and devices.
2. The *Device* layer is used by the system software to do generic USB operations with a device over the bus. This allows the host to determine characteristics of the device, including device class, vendor name, device name, power requirements, and many capabilities such as device speed or USB level supported.
3. The *Function* layer provides additional capabilities that are specific to the device. Matched host and device software layers permit use device-specific functions.

The earlier USB specifications (1.0 and 1.1) support speeds up to 12Mbps (megabits per second). Devices conforming to this specification are relatively low speed devices, such as printers, mice, keyboards, scanners, and modems. The newer USB 2.0 specification supports speeds up to 480Mbps which is adequate for hard drives and external CD or DVD drives. Some USB 2.0 devices are backwards compatible to allow use on older systems, although not all faster devices are backwards compatible. If your computer does not have USB 2.0 support built in, PCI cards (or PC cards for laptops) are available to provide one or more USB 2.0 ports.

The USB cable is a thin, 4-wire cable with two signal lines plus power and ground. The end plugged into a hub has a flat rectangular connector (called an A connector) while the end plugged into a device or downstream hub has a small more square, connector (the B connector). Several different mini-B connectors exist for connecting small devices such as cameras to a computer. USB devices and hubs may draw power from the USB bus or may be self powered.

Linux USB module support

USB is now fairly well supported in Linux. Much of the development has occurred in the 2.6 kernel tree. A lot has been backported to 2.4 kernels, with some support even in 2.2 kernels. Linux supports USB 2.0 as well as the earlier specifications.

Because of the hot-pluggable nature of USB, support is usually provided through kernel modules which can be loaded or unloaded as necessary. For this tutorial we will assume that the modules you need for your distribution are either available or already installed. If you need to compile your own kernel, refer to the tutorial for Exam 201 Topic 201 (see [Resources](#)).

After you have ascertained that your computer has USB ports, you may check what your Linux system found using the `lspci` command as shown in Listing 23. We have filtered the output to show just USB related devices.

Listing 23. lspci output for USB devices

```
[root@lyrebird root]# lspci | grep -i usb
00:1d.0 USB Controller: Intel Corporation 82801DB/DBL/DBM
    (ICH4/ICH4-L/ICH4-M) USB UHCI Controller #1 (rev 01)
00:1d.1 USB Controller: Intel Corporation 82801DB/DBL/DBM
    (ICH4/ICH4-L/ICH4-M) USB UHCI Controller #2 (rev 01)
00:1d.2 USB Controller: Intel Corporation 82801DB/DBL/DBM
    (ICH4/ICH4-L/ICH4-M) USB UHCI Controller #3 (rev 01)
00:1d.7 USB Controller: Intel Corporation 82801DB/DBM
    (ICH4/ICH4-M) USB2 EHCI Controller (rev 01)
```

You will notice that there are four USB controllers in this system. The UHCI and EHCI fields indicate the driver module required to support the controller. The correct USB 1.1 driver depends on the chipset used in your controller. USB 2.0 requires the EHCI driver plus a USB 1.1 driver. See Table 5.

Table 5. Linux USB drivers

Table 5. Linux USB drivers	
Driver	Chipset
EHCI	USB 2.0 Support - requires one of UHCI, OHCI or JE
UHCI	Intel and VIA chipsets
JE	This is an alternate to UHCI for 2.4 kernels. If UHCI does not work, and you have an Intel or VIA chipset, try JE
OHCI	Compaq, most PowerMacs, iMacs, and PowerBooks, OPTi, SiS, ALi

We came across the `lsmod` command and the module configuration files `/etc/modules.conf` (2.4 kernel) and `/etc/modprobe.conf` (2.6 kernel) in our earlier discussion of sound support. Listing 24 shows some of the modules associated with

USB devices that are loaded on the same system as Listing 23. This system has a USB mouse

Listing 24. Using `lsmod` to show loaded USB modules

```
[root@lyrebird root]# lsmod | egrep 'usb|hci|hid|mouse|Module'
Module                Size  Used by      Not tainted
usbserial             23420  0  (autoclean) (unused)
mousedev              5524   1
hid                   22244  0  (unused)
input                 5888   0  [keybdev mousedev hid]
ehci-hcd              20008  0  (unused)
usb-uhci              25740  0  (unused)
usbcore               77376  1  [usbserial hid ehci-hcd usb-uhci]
```

Note particularly that the `usbcore` module is used by all the other USB modules as well as the `hid` (human interface device) module.

Displaying USB information

So now we know something of the modules that support USB, how do we find out what USB devices are attached to our system? The information is to be found in the `/proc/bus/usb` part of the file system. The file `/proc/bus/usb/devices` contains summary information for currently attached USB devices. a partial listing for our system is shown in Listing 25.

Listing 25. Partial contents of `/proc/bus/usb/devices`

```
[root@lyrebird root]# cat /proc/bus/usb/devices
T: Bus=04 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#= 1 Spd=480 MxCh= 6
B: Alloc= 0/800 us ( 0%), #Int= 0, #Iso= 0
D: Ver= 2.00 Cls=09(hub ) Sub=00 Prot=01 MxPS= 8 #Cfgs= 1
P: Vendor=0000 ProdID=0000 Rev= 2.04
S: Manufacturer=Linux 2.4.21-32.0.1.EL ehci-hcd
S: Product=Intel Corp. 82801DB USB2
S: SerialNumber=00:1d.7
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr= 0mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub
E: Ad=81(I) Atr=03(Int.) MxPS= 2 Iv1=256ms
T: Bus=03 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#= 1 Spd=12 MxCh= 2
B: Alloc= 0/900 us ( 0%), #Int= 0, #Iso= 0
D: Ver= 1.00 Cls=09(hub ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0000 ProdID=0000 Rev= 0.00
S: Product=USB UHCI Root Hub
S: SerialNumber=1840
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr= 0mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub
E: Ad=81(I) Atr=03(Int.) MxPS= 8 Iv1=255ms
```

The `Spd=480` that we've highlighted above indicates a USB 2.0 bus while the `Spd=12` indicates a USB 1.1 (or possibly USB 1.0) device. Further down this listing our mouse is shown as having `Spd=1.5`. One and a half megabits per second should be fast enough for most mice.

As with other things that we have seen in the /proc file system, you will be pleased to know that there is a `lsusb` command to help you with display of this information. In particular, you can get a tree view of your USB devices by using the `-t` option. This shows their attachment hierarchy. You can use the `-d` option for information about a specific device if your system gives an abbreviated display using the `-t` option. The `-v` option produces verbose output which interprets many of the fields that we saw in Listing 25. For Listing 26, we've plugged in an external hub, a Nikon digital camera, a USB memory key and an external USB 2.00 hard drive and shown you some of the output.

Listing 26. Using the `lsusb` command

```
[root@lyrebird root]# lsusb -t
Bus# 4
  -Dev# 1 Vendor 0x0000 Product 0x0000
    -Dev# 2 Vendor 0x0409 Product 0x0059
      -Dev# 8 Vendor 0x04b0 Product 0x0108
      -Dev# 4 Vendor 0x0d7d Product 0x1400
      -Dev# 7 Vendor 0x1058 Product 0x0401
    -Dev# 3 Vendor 0x07d0 Product 0x1202
Bus# 3
  -Dev# 1 Vendor 0x0000 Product 0x0000
Bus# 2
  -Dev# 1 Vendor 0x0000 Product 0x0000
Bus# 1
  -Dev# 1 Vendor 0x0000 Product 0x0000
  -Dev# 2 Vendor 0x1241 Product 0x1111
[root@lyrebird root]# lsusb -d 0x0409:0x0059
Bus 004 Device 002: ID 0409:0059 NEC Corp. HighSpeed Hub
[root@lyrebird root]# lsusb -d 0x04b0:0x0108
Bus 004 Device 008: ID 04b0:0108 Nikon Corp. Coolpix 2500
[root@lyrebird root]# lsusb -d 0x0d7d:0x1400
Bus 004 Device 004: ID 0d7d:1400 Phison Electronics Corp.
[root@lyrebird root]# lsusb -d 0x1058:0x0401
Bus 004 Device 007: ID 1058:0401 Western Digital Technologies, Inc.
[root@lyrebird root]# lsusb -d 0x07d0:0x1202
Bus 004 Device 003: ID 07d0:1202 Dazzle
[root@lyrebird root]# lsusb -d 0x1241:0x1111
Bus 001 Device 002: ID 1241:1111 Belkin Mouse
[root@lyrebird root]#
```

Listing 27 shows part of the verbose output available from the `lsusb` command. This is for a memory key. Note that the device has indicated its maximum power requirement (200mA). Note that this device will be treated as a SCSI device. Use either the `dmesg` command or the `fdisk -l` command to find out which SCSI device is mapped to a device. Most cameras equipped with USB ports, as well as card readers, flash devices and hard drives are treated as storage class devices and handled as SCSI devices in Linux. Many cameras come with Windows programs to help upload and pictures from the camera. In Linux you can simply mount the SCSI device representing the camera and copy the pictures to your hard drive where you can edit them with a program such as the GNU Image Manipulation Program (the GIMP). You can even erase files from the memory card or write files to it from Linux, allowing your camera to be used as an exotic replacement for a floppy disk.

Listing 27. Verbose output (partial) from lsusb command

```
[root@lyrebird root]# lsusb -vd 0x0d7d:0x1400
Bus 004 Device 004: ID 0d7d:1400 Phison Electronics Corp.
Device Descriptor:
  bLength                18
  bDescriptorType        1
  bcdUSB                  2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass        0
  bDeviceProtocol        0
  bMaxPacketSize0       64
  idVendor                0x0d7d Phison Electronics Corp.
  idProduct              0x1400
  bcdDevice              0.02
  iManufacturer          1
  iProduct               2 USB DISK 12X
  iSerial                3 0743112A0083
  bNumConfigurations     1
Configuration Descriptor:
  bLength                9
  bDescriptorType        2
  wTotalLength          32
  bNumInterfaces        1
  bConfigurationValue   1
  iConfiguration        0
  bmAttributes          0x80
  MaxPower              200mA
Interface Descriptor:
  bLength                9
  bDescriptorType        4
  bInterfaceNumber      0
  bAlternateSetting     0
  bNumEndpoints         2
  bInterfaceClass       8 Mass Storage
  bInterfaceSubClass    6 SCSI
  bInterfaceProtocol    80 Bulk (Zip)
  iInterface            0
  ...
```

One more piece of information that is available to us now that we know the bus and device ids of your USB devices from Listing 26 is a way to determine which modules are required for a particular device. We'll illustrate a couple in Listing 28.

Listing 27. Verbose output (partial) from lsusb command

```
[root@lyrebird root]# usbmodules --device /proc/bus/usb/004/003
usb-storage
[root@lyrebird root]# usbmodules --device /proc/bus/usb/004/007
usb-storage
hid
```

Hot plugging

There are two commands that your system might use to handle hot plugging of USB devices, *usbmgr* and *hotplug*. According to which you are using, you will find

configuration files in the `/etc/usbmgr` or `/etc/hotplug` directories. Newer systems are more likely to have hotplug.

Hot plugging for USB (and also PC cards) involves users plugging in devices while a system is running. The system then has to:

- Determine the device type and find a driver to run it
- Bind the driver to the device
- Notify other subsystems about the device. This allows disks to be mounted or print queues to be added for example.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- See the [Reference Guide - Hard Disk Drives](#) for a comprehensive history of hard drives. The [Hard Disk Interfaces and Configuration](#) section includes information on SCSI and a comparison of IDE/ATA and SCSI interfaces.
- [The Linux documentation project](#) is the home of lots of useful Linux documentation, including:
 - [Large Disk HOWTO](#) on disk geometry, the 1024 cylinder limit, and other limits for disks
 - [Linux 2.4 SCSI subsystem HOWTO](#), covering SCSI on Linux, including device naming.
 - [Linux SCSI Generic \(sg\) HOWTO](#) on the new generic SCSI driver and utilities on Linux
 - [The Network Administrators' Guide](#) for networking on Linux
 - [Linux Networking-HOWTO](#) on SLIP, CSLIP, and PPP
 - [Linux PPP HOWTO](#) on setting up PPP on Linux
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

LPI exam 101 prep: Linux installation and package management

Junior Level Administration (LPIC-1) topic 102

Skill Level: Introductory

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

09 Sep 2005

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 101. In this second of five tutorials, Ian introduces you to Linux™ installation and package management. By the end of this tutorial you will know how Linux uses disk partitions, how Linux boots, and how to install and manage software packages.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams. Before you take the exams, study the [LPI exam prep tutorials](#) on developerWorks to prepare for each topic covered in the exams.

The following five tutorials help you prepare for the first of the two LPI junior-level system administrator exams: LPI exam 101. A companion series of tutorials is under development for the other junior-level exam: LPI exam 102. Both exam 101 and

exam 102 are required for junior-level certification. Junior-level certification is also known as certification level 1. To pass level 1, you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

Tutorial	Topic	Summary
LPI exam 101 prep: Hardware and architecture	Topic 101	Learn to configure your system hardware with Linux. By the end of this tutorial, you will know how Linux configures the hardware found on a modern PC and where to look if you have problems.
LPI exam 101 prep: Linux installation and package management	Topic 102	(This tutorial) Get an introduction to Linux installation and package management. By the end of this tutorial, you will know how Linux uses disk partitions, how Linux boots, and how to install and manage software packages.
LPI exam 101 prep: GNU and UNIX commands	Topic 103	Coming soon!
LPI exam 104 prep: Linux, filesystems, and FHS	Topic 104	Coming soon!
LPI exam 110 prep: The X Window system	Topic 110	Coming soon!

Within each exam topic, subtopics are weighted, reflecting their importance within the topic.

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Linux installation and package management", the second of five

tutorials designed to prepare you for LPI exam 101. In this tutorial you will learn about these areas of Linux installation and package management:

Subtopic	Weight	Summary
1.102.1 Design hard disk layout	5	You will learn to design a disk partitioning scheme for a Linux system, including allocating filesystems or swap space to separate partitions or disks, and tailoring the design to the intended use of the system. You will learn how to place /boot on a partition that conforms to BIOS requirements for booting.
1.102.2 Install a boot manager	1	You will learn to select, install, and configure a boot manager. You will learn how to provide alternative boot locations and backup boot options (for example, using a boot floppy).
1.102.3 Make and install programs from source	5	You will learn to build and install an executable program from source. You will learn to unpack a file of sources and customize the Makefile, for example to change paths or add extra include directories.
1.102.4 Manage shared libraries	3	You will learn to determine the shared libraries that executable programs depend on and install them when necessary. You will also learn where system libraries are kept.
1.102.5 Use Debian package management	8	You will learn to perform package management using the Debian package manager. You will use command-line and interactive tools to install, upgrade, or uninstall packages, as well as find packages containing specific files or software. You will learn how to determine package information such as version, content, dependencies, package integrity, and installation status. You will learn how to do this for both

		installed packages and packages that are not installed.
1.102.6 Use Red Hat Package Manager (RPM)	8	You will learn to perform package management for Linux distributions that use RPMs for package distribution. You will be able to install, re-install, upgrade, and remove packages, as well as obtain status and version information on packages. You will learn how to determine package information such as version, status, dependencies, integrity, and signatures. You will know how to determine what files a package provides, as well as find which package supplies a specific file.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial. You should also understand the BIOS implications for hard drives as covered in the first tutorial in this series, ["LPI exam 101 prep \(topic 101\): Hardware and architecture."](#)

Section 2. Hard disk layout

This section covers material for topic 1.102.1 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

This section shows you how to lay out your Linux filesystems on your hard drives, and expands on what you learned about hard disks in the first tutorial in this series, ["LPI exam 101 prep \(topic 101\): Hardware and architecture."](#)

An upcoming tutorial, "LPI exam 101 prep (topic 104): Devices, Linux filesystems, and FHS" goes into more depth on filesystems and tools to create partitions of various types.

Filesystem overview

A Linux filesystem contains *files* that are arranged on a disk or other *block storage device* in *directories*. As with many other systems, directories on a Linux system may contain other directories called *subdirectories*. Unlike a system such as Microsoft® Windows® with a concept of separate file systems on different drive letters (A:, C:, etc.), a Linux filesystem is a single tree with the / directory as its *root* directory.

You might wonder why disk layout is important if the filesystem is just one big tree. Well, what really happens is that each block device, such as a hard drive partition, CD-ROM, or floppy disk, actually has a filesystem on it. You create the single tree view of the filesystem by *mounting* the filesystems on different devices at a point in the tree called a *mount point*.

Usually, you start this mount process by mounting the filesystem on some hard drive partition as /. You may mount other hard drive partitions as /boot, /tmp, or /home. You may mount the filesystem on a floppy drive as /mnt/floppy, and the filesystem on a CD-ROM as /media/cdrom1, for example. You may also mount files from other systems using a networked filesystem such as NFS. There are other types of file mounts, but this gives you an idea of the process. While the mount process actually mounts the *filesystem* on some device, it is common to simply say that you "mount the device," which is understood to mean "mount the filesystem on the device."

Now, suppose you have just mounted the root file system (/) and you want to mount an IDE CD-ROM, /dev/hdd, at the mount point /media/cdrom. The mount point must exist before you mount the CD-ROM over it. When you mount the CD-ROM, the files and subdirectories on the CD-ROM become the files and subdirectories in and below /media/cdrom. Any files or subdirectories that were already in /media/cdrom are no longer visible, although they still exist on the block device that contained the mount point /media/cdrom. If the CD-ROM is unmounted, then the original files and subdirectories become visible again. You should avoid this problem by not placing other files in a directory intended for use as a mount point.

Table 1 shows the shows the directories required in / by the Filesystem Hierarchy Standard (for more detail on FHS, see [Resources](#)).

Directory	Description
bin	Essential command binaries
boot	Static files of the boot loader
dev	Device files
etc	Host-specific system configuration

lib	Essential shared libraries and kernel modules
media	Mount point for removable media
mnt	Mount point for mounting a filesystem temporarily
opt	Add-on application software packages
sbin	Essential system binaries
srv	Data for services provided by this system
tmp	Temporary files
usr	Secondary hierarchy
var	Variable data

Partitions

The first tutorial in this series, "[LPI exam 101 prep \(topic 101\): Hardware and architecture](#)" touched on partitions on a hard disk, and now we'll go into more detail.

The first IDE hard drive on a Linux system is `/dev/hda`, and the first SCSI drive is `/dev/sda`. A hard drive is formatted into 512 byte *sectors*. All the sectors on a disk platter that can be read without moving the head constitute a *track*. Disks usually have more than one platter. The collection of tracks on the various platters that can be read without moving the head is called a *cylinder*. The *geometry* of a hard drive is expressed in cylinders, tracks (or *heads*) per cylinder and sectors/track.

Limitations on the possible sizes of each of these values used with DOS operating systems on PC systems resulted in BIOS translating geometry values so that larger hard drives could be supported. Eventually, even these methods were insufficient. More recent developments in disk drive technology have led to *logical block addressing (LBA)*, so the CHS geometry measurements are less important and the reported geometry on a disk may bear little or no relation to the actual layout of a modern disk. The larger disks in use today have forced an extension to LBA known as LBA48, which reserves up to 48 bits for sector numbers.

The space on a hard drive is divided (or partitioned) into *partitions*. Partitions cannot overlap; space that is not allocated to a partition is called *free space*. The partitions have names like `/dev/hda1`, `/dev/hda2`, `/dev/hda3`, `/dev/sda1`, and so on. IDE drives are limited to 63 partitions, while SCSI drives are limited to 15. Partitions are usually allocated as an integral number of cylinders (based on the possibly inaccurate notion of a cylinder).

If two different partitioning programs have different understandings of the nominal disk geometry, it is possible for one partitioning program to report an error or possible problem with partitions created by another partitioning program. You may also see this kind of problem if a disk is moved from one system to another, particularly if the BIOS capabilities are different. You can see the nominal geometry on a Linux system by looking at the appropriate geometry special file in the `/proc` filesystem, such as `/proc/ide/hda/geometry`. This is the geometry used by partitioning tools like `fdisk` and `parted`. Listing 1 shows the use of the `cat` command to display `/proc/ide/hda/geometry`, followed by the informational messages produced by use of the `parted` partitioning tool.

Listing 1. Hard disk geometry

```
[root@lyrebird root]# cat /proc/ide/hda/geometry
physical      19457/255/63
logical       19457/255/63
[root@lyrebird root]# parted /dev/hda
GNU Parted 1.6.3
Copyright (C) 1998, 1999, 2000, 2001, 2002 Free Software Foundation, Inc.
This program is free software, covered by the GNU General Public License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE. See the GNU General Public License for more details.

Using /dev/hda
Information: The operating system thinks the geometry on /dev/hda is
19457/255/63. Therefore, cylinder 1024 ends at 8032.499M.
(parted)
```

Note that, in Listing 1, `parted` has computed a nominal position for the end of cylinder 1024. Cylinder 1024 is important in some older systems where the BIOS is only able to boot partitions that are completely located within the first 1024 cylinders of a disk. This is most likely to occur in a BIOS that does not have LBA support. It is not usually a problem in modern machines, although you should be aware that the limit may exist.

There are three types of partition: *primary*, *logical*, and *extended*. The *partition table* is located in the *master boot record (MBR)* of a disk. The MBR is the first sector on the disk, so the partition table is not a very large part of it. This limits the number of primary partitions on a disk to four. When more than four partitions are required, as is often the case, one of the primary partitions must instead become an extended partition. A disk may contain only one extended partition.

An *extended partition* is nothing more than a container for logical partitions. This partitioning scheme was originally used with MS DOS and PC DOS and permits PC disks to be used by DOS, Windows, or Linux systems.

Linux numbers primary or extended partitions as 1 through 4, so `dev hda` may have four primary partitions, `/dev/hda1`, `/dev/hda2`, `/dev/hda3`, and `/dev/hda4`. Or it may

have a single primary partition `/dev/hda1` and an extended partition `/dev/hda2`. If logical partitions are defined, they are numbered starting at 5, so the first logical partition on `/dev/hda` will be `/dev/hda5`, even if there are no primary partitions and one extended partition (`/dev/hda1`) on the disk.

Listing 2 shows the output from the `parted` subcommand `p`, which displays the partition information for the disk of Listing 1. Note that this system has several different Windows and Linux filesystems on it.

Listing 2. Displaying the partition table with parted

```
(parted) p
Disk geometry for /dev/hda: 0.000-152627.835 megabytes
Disk label type: msdos
Minor      Start      End        Type       Filesystem  Flags
1          0.031     16300.327  primary   ntfs        boot
2          16300.327 25846.765  primary   fat32       lba
3          25846.765 26842.983  primary   ext3
4          26842.983 152625.344 extended   lba
5          26843.014 28898.173  logical   linux-swap
6          28898.205 48900.981  logical   ext3
7          48901.012 59655.432  logical   ext3
8          59655.463 75657.678  logical   ext3
9          75657.709 95001.569  logical   ext3        boot
10         95001.601 122997.656 logical   reiserfs
11        122997.687 152625.344 logical   ext3
```

Allocating disk space

As mentioned earlier, a Linux system filesystem is a single large tree rooted at `/`. It is fairly obvious why data on floppy disks or CD-ROMs must be mounted, but perhaps less obvious why you should consider separating data that is stored on hard drives. Some good reasons for separating filesystems include:

- **Boot files.** Some files must be accessible to the BIOS or boot loader at boot time.
- **Multiple hard drives.** Typically each hard drive will be divided into one or more partitions, each with a filesystem that must be mounted somewhere in the filesystem tree.
- **Shareable files.** Several system images may share static files such as executable program files. Dynamic files such as user home directories or mail spool files may also be shared, so that users can log in to any one of several machines on a network and still use the same home directory and mail system.
- **Potential overflow.** If a filesystem might fill to 100 percent of its capacity, it is usually a good idea to separate this from files that are needed to run the system.

- Quotas. Quotas limit the amount of space that users or groups can take on a filesystem.
- Read-only mounting. Before the advent of journaling filesystems, recovery of a filesystem after a system crash often took a long time. Therefore, filesystems that seldom changed (such as a directory of executable programs) could be mounted read-only so as not to waste time for checking it after a system crash.

In addition to the filesystem use covered so far, you also need to consider allocating swap space on disk. For a Linux system, this is usually one, or possibly multiple, dedicated partitions.

Making choices

Let's assume you are setting up a system that has at least one hard drive, and you want to boot from the hard drive. (This tutorial does not cover setup for a diskless workstation that is booted over a LAN or considerations for using a live CD or DVD Linux system.) Although it may be possible to change partition sizes later, this usually takes some effort, so making good choices up front is important. Let's get started.

Your first consideration is to ensure that your system will be bootable. Some older systems have a limitation that the BIOS can boot only from a partition that is wholly located within the first 1024 cylinders of disk. If you have such a system, then you **must** create a partition that will eventually be mounted as `/boot` that will hold the key files needed to boot the system. Once these have been loaded, the Linux system will take over operation of the disk and the 1024 cylinder limit will not affect further operation of the system. If you need to create a partition for `/boot`, approximately 100 megabytes (MB) is usually sufficient.

Your next consideration is likely to be the amount of required swap space. With current memory prices, swap space represents a very slow secondary memory. A once common rule of thumb was to create swap space equivalent to the amount of real RAM. Today, you might consider allocating approximately 500 MB for a workstation and perhaps 1GB for a server. If special circumstances dictate, you may need to increase these values, but if you do, your system will likely not be performing very well, and you should add real memory. It is possible to use a swap file, but a dedicated partition performs better.

Now we come to a point of divergence. Use of a personal workstation tends to be much less predictable than use of a server. My preference, particularly for new users, is to allocate most of the standard directories (`/usr`, `/opt`, `/var`, etc.) into a single large partition. This is especially useful for new users who may not have a

clear idea of what will be installed down the line. A workstation running a graphical desktop and a reasonable number of development tools will likely require 2 or 3 gigabytes of disk space plus space for user needs. Some larger development tools may require several gigabytes each. I usually allocate somewhere between 10 GB and 20 GB per operating system, and I leave the rest of my disk free to load other distributions.

Server workloads will be more stable, and running out of space in a particular filesystem is likely to be more catastrophic. So, for them, you will generally create multiple partitions, spread across multiple disks, possibly using hardware or software RAID or logical volume groups.

You will also want to consider the workload on a particular filesystem and whether the filesystem is shared among several systems or used by just one system. You may use a combination of experience, capacity planning tools, and estimated growth to determine the best allocations for your system.

Regardless of whether you are configuring a workstation or a server, you will have certain files that are unique for each system located on the local drive. Typically, these include `/etc` for system parameters, `/boot` for files needed during boot, `/sbin` for files needed for booting or system recovery, `/root` for the root user's home directory, `/var/lock` for lock files, `/var/run` for running system information, and `/var/log` for log files for this system. Other filesystems, such as `/home` for user home directories, `/usr`, `/opt`, `/var/mail`, or `/var/spool/news` may be on separate partitions, or network mounted, according to your installation needs and preferences.

Section 3. Boot managers

This section covers material for topic 1.102.2 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

This section discusses the PC boot process and the two main boot loaders used in Linux: LILO and GRUB. We will cover choosing a boot manager and recovering when things go wrong.

Boot process overview

Before we get into LILO and GRUB, let's review how a PC starts or *boots*. Code, called *BIOS* (for *Basic Input Output Service*) is stored in non-volatile memory such as a ROM, EEPROM, or flash memory. When the PC is turned on or rebooted, this code is executed. Usually it performs a power-on self test (POST) to check the

machine. Finally, it loads the first sector from the master boot record (MBR) on the boot drive.

As discussed in the previous section on [Partitions](#), the MBR also contains the partition table, so the amount of executable code in the MBR is less than 512 bytes, which is not very much code. Note that every disk, even a floppy, contains executable code in its MBR, even if the code is only enough to put out a message such as "Non-bootable disk in drive A:". This code that is loaded by BIOS from this first sector is called the *first stage boot loader* or the *stage 1 boot loader*.

The standard hard drive MBR used by MS DOS, PC DOS, and Windows operating systems checks the partition table to find a primary partition on the boot drive that is marked as *active*, loads the first sector from that partition, and passes control to the beginning of the loaded code. This new piece of code is also known as the *partition boot record*. The partition boot record is actually another stage 1 boot loader, but this one has just enough intelligence to load a set of blocks from the partition. This new code is called the *stage 2 boot loader*. As used by MS-DOS and PC-DOS, the stage 2 loader proceeds directly to load the rest of operating system. This is how your operating system pulls itself up by the bootstraps until it is up and running.

This works fine for a system with a single operating system. What happens if you want multiple operating systems, say Windows 98, Windows XP, and three different Linux distributions? You **could** use some program (such as the DOS FDISK program) to change the active partition and reboot. This is cumbersome. Furthermore, a disk can have only four primary partitions, and the standard MBR can boot only a primary partition. But our hypothetical example cited five operating systems, each of which needs a partition. Oops!

The solution lies in using some special code that allows a user to choose which operating system to boot. Examples include:

1. Loadlin, a DOS executable program that is invoked from a running DOS system to boot a Linux partition. This was popular when setting up a multi-boot system was a complex and risky process.
2. OS/2 Boot Manager, a program that is installed in a small dedicated partition. The partition was marked active and the standard MBR boot process started Boot Manager, which presented a menu allowing a user to choose which operating system to boot.
3. A smart boot loader, a program that can reside on an operating system partition and is invoked either by the partition boot record of an active partition or by the master boot record. Examples include:
 - BootMagic™, part of Norton PartitionMagic™
 - LILO, the Linux LOader

- GRUB, the GRand Unified Boot loader

Evidently, if you can get control of the system into a program that has more than 512 bytes of code to accomplish its task, then it isn't too hard to allow booting from logical partitions, or booting from partitions that are not on the boot drive. All of these solutions allow these possibilities, either because they can load a boot record from an arbitrary partition, or because they have some understanding of what file or files to load to start the boot process.

From here on, we will focus on LILO and GRUB as these are the boot loaders included with most Linux distributions. The installation process for your distribution will probably give you a choice of which one to set up. Either will work with most modern disks. Remember that disk technology has advanced rapidly, so you should always make sure that your chosen boot loader, as well as your chosen Linux distribution (or other operating system), as well as your system BIOS will work with your shiny new disk. Failure to do so may result in loss of data.

The stage 2 loaders used in LILO and GRUB allow you to choose which among several operating systems or versions to load. However, LILO and GRUB differ significantly in that a change to the system requires you to use a command to recreate the LILO boot setup whenever you upgrade a kernel or make certain other changes to your system, while GRUB can accomplish this through a configuration text file that you can edit. LILO has been around for a while. GRUB is newer. The original GRUB has now become *GRUB Legacy* and GRUB 2 is being developed under the auspices of the Free Software Foundation (see [Resources](#)).

LILO

LILO, or the LInux LOader, is one of the two most common Linux boot loaders. LILO can be installed into the MBR of your bootable hard drive, or into the partition boot record of a partition. It can also be installed on removable devices such as floppy disks, CDs or USB keys. It is a good idea to practice on a floppy disk or USB key if you are not already familiar with LILO, so that is what we will do in our examples.

During Linux installation you will usually specify either LILO or GRUB as a boot manager. If you chose GRUB, then you may not have LILO installed. If you do not, then you will need to install the package for it. We will assume that you already have the LILO package installed. See the package management sections later in this tutorial if you need help with this.

The primary function of `lilo` command, located in `/sbin/lilo`, is to write a stage 1 boot record and create a map file (`/boot/map`) using configuration information that is normally located in `/etc/lilo.conf`. There are some auxiliary uses that we will mention later. First let us look at a typical LILO configuration file that might be used on a

dual-boot system with Windows and Linux.

Listing 3. /etc/lilo.conf example

```
prompt
timeout=50
compact
default=linux
boot=/dev/fd0
map=/boot/map
install=/boot/boot.b
message=/boot/message
lba32
password=mypassword
restricted

image=/boot/vmlinuz-2.4.21-32.0.1.EL
    label=linux
    initrd=/boot/initrd-2.4.21-32.0.1.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

other=/dev/hda1
    loader=/boot/chain.b
    label=WIN-XP
```

The first set of options above are global options that control how LILO operates. The second and third give per image options for the two operating systems that we want to allow LILO to boot, Red Hat Enterprise Linux 3 or Windows XP in this example.

The global options in our example are:

prompt

forces a boot prompt to be displayed.

timeout

specifies, in tenths of a second, the timeout before the default system will be automatically loaded. Our example of `timeout=50` is equivalent to a 5 second timeout.

compact

attempts to merge read requests for adjacent sectors. This speeds up load time and keeps the map smaller.

default

specifies which operating system should be loaded by default. If not specified, the first one is used. IN our example, the Linux system will be loaded if the user does not elect otherwise within the 5 second timeout.

boot

specifies where LILO will be installed. In our example, it is the floppy disk, `/dev/fd0`. To install in the MBR of the first hard drive specify `boot=/dev/hda`. Our RHEL 3 system is actually located on `/dev/hda11`, so we would specify

`boot=/dev/hda11` is we wanted to install LILO in this partition. If this parameter is omitted, LILO will attempt to use the boot sector of the device currently mounted as root (/).

map

specifies the location of the map file that LILO uses to provide user prompts and to load the operating systems specified in the per image sections of `lilo.conf`. By default this is `/boot/map`.

install

specifies the new file to install as the boot sector. The default is `/boot/boot.b`, which is provided as part of the LILO package.

message

specifies a message that is displayed before the boot prompt. This must be less than 65535 bytes in length. If your system displays a graphical background with a LILO menu, you may find that `/boot/message` contains an image file. On some Red Hat systems this will be a 300x200 pixel file in PCX format. On SUSE systems you may find this replaced by a 16 color 640x480 pixel VGA bitmap. In this case, you would also find some additional parameters. Check the documentation that comes with your system. For example, my SUSE SLES9 system has this in `/usr/share/doc/packages/lilo/README.bitmaps`.

lba32

specifies that LILO should use LBA32 mode for the disk rather than CHS or linear sector addressing.

password

specifies a password that must be entered before booting an image. Note that this is in clear text, so the `/etc/lilo.conf` file attributes should permit the file to be viewed only by the root user. It should not be the same as your root password. Both `password` and the next option, `restricted`, are actually examples of per image options that may be specified in the global section for convenience. If so specified, the same values apply to all images unless overridden in an individual image section.

restricted

relaxes the password requirement so that a password is only required if a user tries to provide additional parameters during boot. You might use this to allow a user to boot normally without entering a password but have to provide a password to boot into single user mode.

The next section gives the per image options for our RHEL3 Linux system.

image

specifies that this section is for a Linux system that should be loaded from a

file. The parameter is the filename of a Linux kernel image.

label

is an optional label that you can enter instead of the full image name to select this image.

initrd

is the name of the *initial RAM disk* which contains modules needed by the kernel before your file systems are mounted.

read-only

specifies that the root file system should initially be mounted read-only. Later stages of boot will usually remount it read-write after it has been checked.

append

specifies options to be passed to the kernel. Our example specifies that SCSI emulation should be used for /dev/hdd (2.4, and earlier, kernels handled optical devices such as CD-ROMs this way). It also specifies that the partition with label RHEL3 should be mounted as root (/).

The final section gives the per image options for our non-Linux system.

other

specifies the name of the device containing the device (or file) that contains the boot sector of the system to be loaded..

loader

specifies the loader to be used. LILO supports chain.b which simply loads that partition boot record from a partition a bootable partition and a variant, /boot/os2_d.b which can be used to boot OS/2 from a second hard drive.

label

is an optional label that you can enter instead of the full image name to select this image.

Now, if we insert a blank floppy disk we can run the `lilo` command (/sbin/lilo) to create a bootable floppy disk as shown in Listing 4. Note that the `lilo` command has five levels of verbosity. Specify an extra `-v` for each level.

Listing 4. Creating a bootable floppy disk with lilo

```
[root@lyrebird root]# lilo -v -v
LILO version 21.4-4, Copyright (C) 1992-1998 Werner Almesberger
'lba32' extensions Copyright (C) 1999,2000 John Coffman

Reading boot sector from /dev/fd0
Merging with /boot/boot.b
```

```
Secondary loader: 11 sectors.
Mapping message file /boot/message
Compaction removed 43 BIOS calls.
Message: 74 sectors.
Boot image: /boot/vmlinuz-2.4.21-32.0.1.EL
Setup length is 10 sectors.
Compaction removed 2381 BIOS calls.
Mapped 2645 sectors.
Mapping RAM disk /boot/initrd-2.4.21-32.0.1.EL.img
Compaction removed 318 BIOS calls.
RAM disk: 354 sectors.
Added linux *
Boot other: /dev/hda1, on /dev/hda, loader /boot/chain.b
Compaction removed 0 BIOS calls.
Mapped 6 (4+1+1) sectors.
Added WIN-XP
/boot/boot.0200 exists - no backup copy made.
Map file size: 8192 bytes.
Writing boot sector.
```

We now have our bootable LILO diskette. If LILO encounters an error, you might see an error message and the boot sector will not be written. For example, if we had omitted the `lba32` option from our `/etc/lilo.conf` file we might see output such as that in Listing 5. This would be a tip to try the `linear` or `lba32` options. In this case, we use the command line to specify the `-l` option which is equivalent to specifying the `linear` option in `lilo.conf`. If we were to do this again with the `-L` option, `lilo` should be successful and the output should be as in the previous listing.

Listing 5. Incorrect `/etc/lilo.conf` example

```
[root@lyrebird root]# lilo
Warning: device 0x030b exceeds 1024 cylinder limit
Fatal: geo_comp_addr: Cylinder number is too big (16284 > 1023)
[root@lyrebird root]# lilo -l
Warning: device 0x030b exceeds 1024 cylinder limit
Fatal: sector 261613688 too large for linear mode (try 'lba32' instead)
```

When you have tested your boot diskette, change the `boot=/dev/fd0` entry in your `lilo.conf` file to install LILO on the MBR or a partition boot record. For example, `boot=/dev/hda` will install LILO in the master boot record of your first IDE hard drive.

You now have an introduction to LILO and its configuration file, including how to override some configuration options from the `lilo` command line. You will find more information in the `lilo` man page using the command `man lilo`. You will find even more extensive information in the postscript user guide that is installed with the `lilo` package. This should be installed in your documentation directory, but the exact location may vary by system. One way to locate the file is to filter the package list through `grep`. Listing 6 shows this for the rpm-based RHEL3 system that we have been using in this example.

Listing 6. Locating the LILO user guide with rpm.

```
[ian@lyrebird ian]$ rpm -ql lilo | grep ".ps$"
/usr/share/doc/lilo-21.4.4/doc/Technical_Guide.ps
/usr/share/doc/lilo-21.4.4/doc/User_Guide.ps
```

LILO auxiliary commands

LILO has several auxiliary commands.

lilo -q

will display information from the map file

lilo -R

will set lilo to automatically boot the specified system on the next reboot only. This is very convenient for automatically rebooting remote systems.

lilo -l

will display information about the path of a kernel

lilo -u

will uninstall lilo and restore the previous boot record.

When LILO boots a Linux system you may want to provide additional parameters at boot time. For example, if your graphical startup was not working, you may want to boot into mode 3 or into single user mode to recover. Any text you type after the label name will be passed to the kernel. For example, in our example, we would select the RHEL system by simply typing "linux". To boot into mode 3 or single user mode we would type one of the following strings as appropriate.

```
linux 3
linux single
```

Remember also that with LILO you **must** run the lilo command whenever you update the configuration file (/etc/lilo.conf). You should also run the lilo command if you add, move, or remove partitions or make any other changes that might invalidate the generated boot loader.

GRUB

GRUB, or the GRand Unifood Boot loader, is the other of the two most common Linux boot loaders. As with LILO, GRUB can be installed into the MBR of your bootable hard drive, or into the partition boot record of a partition. It can also be installed on removable devices such as floppy disks, CDs or USB keys. It is a good idea to practice on a floppy disk or USB key if you are not already familiar with GRUB, so that is what we will do in our examples.

GRUB, or GNU GRUB, is now developed under the auspices of the Free Software

Foundation. A new version, GRUB 2 is under development, so the original GRUB 0.9x versions are now known as Grub Legacy.

During Linux installation you will usually specify either LILO or GRUB as a boot manager. If you chose LILO, then you may not have GRUB installed. If you do not, then you will need to install the package for it. We will assume that you already have the GRUB package installed. See the package management sections later in this tutorial if you need help with this.

GRUB has a configuration file which is usually stored in `/boot/grub/grub.conf`. If your filesystem supports symbolic links, as most Linux filesystems do, you will probably have `/boot/grub/menu.lst` as a symbolic link to `/boot/grub/grub.conf`.

The `grub` command (`/sbin/grub`, or, on some systems, `/usr/sbin/grub`) is a small, but reasonably powerful shell which supports several commands for installing GRUB, booting systems, locating and displaying configuration files and similar tasks. This shell shares much code with the second stage GRUB boot loader, so it is useful to learn about GRUB without having to boot to a second stage GRUB environment. The GRUB stage 2 runs either in menu mode or command mode, to allow you to choose an operating system from a menu, or to specify individual commands to load a system. There are also several other commands, such as `grub-install` which use the grub shell and help automate tasks such as installing GRUB.

Listing 7 shows part of a GRUB configuration file. As you look through it, remember one important thing -- GRUB counting for drives, partitions and things that need to be counted starts at 0 rather than 1.

Listing 7. `/boot/grub/menu.lst` GRUB configuration example.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You do not have a /boot partition. This means that
#           all kernel and initrd paths are relative to /, eg.
#           root (hd1,5)
#           kernel /boot/vmlinuz-version ro root=/dev/hdc6
#           initrd /boot/initrd-version.img
#boot=/dev/hdc6
default=2
timeout=10
splashimage=(hd0,6)/boot/grub/splash.xpm.gz
password --md5 $1$/8Kl21$3VPIphs6REHeHccwzjQYO.

title Red Hat Linux (2.4.20-31.9)
    root (hd0,6)
    kernel /boot/vmlinuz-2.4.20-31.9 ro root=LABEL=RH9 hdd=ide-scsi
    initrd /boot/initrd-2.4.20-31.9.img

title Red Hat Linux (2.4.20-6)
    root (hd0,6)
    kernel /boot/vmlinuz-2.4.20-6 ro root=LABEL=RH9 hdd=ide-scsi
    initrd /boot/initrd-2.4.20-6.img

title Red Hat Enterprise Linux WS A (2.4.21-32.0.1.EL)
```

```

    root (hd0,10)
    kernel /boot/vmlinuz-2.4.21-32.0.1.EL ro root=LABEL=RHEL3 hdd=ide-scsi
    initrd /boot/initrd-2.4.21-32.0.1.EL.img

title      Ubuntu, kernel 2.6.10-5-386
root      (hd1,10)
kernel    /boot/vmlinuz-2.6.10-5-386 root=/dev/hdb11 ro quiet splash
initrd    /boot/initrd.img-2.6.10-5-386
savedefault
boot

title      Ubuntu, kernel 2.6.10-5-386 (recovery mode)
lock
root      (hd1,10)
kernel    /boot/vmlinuz-2.6.10-5-386 root=/dev/hdb11 ro single
initrd    /boot/initrd.img-2.6.10-5-386
boot

title Win/XP
    rootnoverify (hd0,0)
    chainloader +1

title Floppy
    root (fd0)
    chainloader +1

```

As with the LILO configuration file, the first set of options above control how GRUB operates. For GRUB, these are called *menu commands* and they must appear before other commands. The remaining sections give per image options for the operating systems that we want to allow GRUB to boot. Note that "title" is considered a menu command. Each instance of title is followed by one or more general or menu entry commands. Our LILO example was a typical basic example for a dual boot system with Windows and Linux. This example comes from the same system as we used before, but here we have added a few extra operating systems, to show you some of the power of a boot loader. You will recognize many of the same kinds of elements occurring in both LILO and GRUB configuration files. You might like to think about what would need to change if you added the extra operating systems here to the earlier LILO example.

The menu commands that apply to all other sections in our example are:

#

Any line starting with a # is a comment and is ignored by GRUB. This particular configuration file was originally generated by anaconda, the Red Hat installer. You will probably find comments added to your GRUB configuration file if you install GRUB when you install Linux. The comments will often serve as an aid to the system upgrade program so that your GRUB configuration can be kept current with upgraded kernels. Pay attention to any markers that are left for this purpose if you edit the configuration yourself.

default

specifies which system to load if the user does not make a choice within a timeout. In our example, default=2 means to load the **third** entry. Remember that GRUB counts from 0 rather than 1. If not specified, then the default is to boot the first entry, entry number 0.

timeout

specifies a timeout in seconds before booting the default entry. Note that LILO uses tenths of a second for timeouts while GRUB uses whole seconds.

splashimage

Specifies the background, or *splash*, image to be displayed with the boot menu. GRUB refers to the first hard drive as (hd0) and the first partition on that drive as (hd0,0), so the specification of `splashimage=(hd0,6)/boot/grub/splash.xpm.gz` means to use the file `/boot/grub/splash.xpm.gz` located on partition 7 of the first hard drive. Remember that bit about counting from 0. Note also, that the image is an XPM file compressed with gzip. Support for splashimage is a patch that may or may not be included in your distribution.

password

specifies a password that must be entered before a user can unlock the menu and either edit a configuration line or enter GRUB commands. As with LILO, the password may be in clear text. GRUB also permits passwords to be stored as an MD5 digest, as in our example. This is somewhat more secure and most administrators will set a password. Without a password, a user has complete access to the GRUB command line.

Our example shows five Linux distributions (three Red Hat and two Ubuntu) plus a Windows XP and a floppy boot option. The commands used in these sections are:

title

is a descriptive title that is shown as the menu item when Grub boots. You use the arrow keys to move up and down through the title list and then press the **Enter** key to select a particular entry.

root

specifies the partition that will be booted. As with splashimage, remember that counting starts at 0, so the first Red Hat system which is specified as `root (hd0,6)` is actually on partition 7 the first hard drive (`/dev/hda7` in this case), while the first Ubuntu system which is specified as `root (hd1,10)` is on the second hard drive (`/dev/hdb11`). GRUB will attempt to mount this partition to check it and provide values to the booted operating system in some cases.

kernel

specifies the kernel image to be loaded and any required kernel parameters. This is similar to a combination of the LILO image and append commands. We have two different Red Hat 9 kernels, plus a Red Hat Enterprise Linux 3 Workstation kernel, and one level of Ubuntu system with two different sets of kernel parameters in this example.

initrd

is the name of the *initial RAM disk* which contains modules needed by the kernel before your file systems are mounted.

savedefault

is shown here for illustration. If the menu command `default=saved` is specified, and the `savedefault` command is specified for an operating system, then booting that operating system will cause it to become the default until another operating system with `savedefault` specified is booted. In this example, the specification of `default=2` will override any saved default.

boot

is an optional parameter that instructs GRUB to boot the selected operating system. This is the default action when all commands for a selection have been processed.

lock

is used in this example with the second Ubuntu system. This system will boot into single user mode which permits a user to make modifications to a system that are normally restricted to root access. If this is specified, then you should also specify a password in the initial options, otherwise, a user can edit out your lock option and boot the system, or add "single" to one of the other entries. It is also possible to specify a different password for individual entries if you wish.

rootnoverify

is similar to `root`, except that GRUB does not attempt to mount the filesystem or verify its parameters. This is usually used for filesystems such as NTFS that are not supported by GRUB. You might also use this if you wanted GRUB to load the master boot record on a hard drive, for example to access a different configuration file, or to reload your previous boot loader.

chainloader

specifies that another file will be loaded as a stage 1 file. The value "+1" is equivalent to `0+1` which means to load one sector starting at sector 0, that is, load the first sector from the device specified by `root` or `rootnoverify`.

You now have some idea of what you might find in a typical `/boot/grub/grub.conf` (or `/boot/grub/menu.lst`) file. There are many many other GRUB commands to provide extensive control over the boot process as well as help with installing grub and other tasks. You can learn more about these in the GRUB manual which should be available on your system using the command `info grub`.

Now that we have a GRUB configuration file, we need to create a boot floppy to test it. The simplest way to do this is to use the `grub-install` command as shown in Listing 8. If you are installing GRUB onto a floppy or onto a partition, then you should

umount the device first. This does not apply if you are installing GRUB in the MBR of a hard drive, as you only mount partitions (`/dev/hda1`, `/dev/hda2`, etc.) and not the whole hard drive (`/dev/hda`).

Listing 8. Installing GRUB to a floppy disk.

```
[root@lyrebird root]# umount /dev/fd0
umount: /dev/fd0: not mounted
[root@lyrebird root]# grub-install /dev/fd0
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(fd0)    /dev/fd0
(hd0)    /dev/hda
(hd1)    /dev/hdc
(hd2)    /dev/sda
```

Note: You may also use the GRUB device name (`fd0`) instead of `/dev/fd0`, but if you do you must enclose it in quotes to avoid shell interpretation of the parentheses. For example:

```
grub-install '(fd0)'
```

If you started with an empty floppy, and now now mount it, you will discover that it still appears to be empty. What has happened is that GRUB wrote a customized stage 1 loader to the first sector of the disk. This does not show up in the file system. This stage 1 loader will load stage 2 and the configuration file from your hard drive. Try booting the diskette and you will see very little IO activity before your menu is displayed.

The device map tells you how GRUB will match its internal view of your disks (`fd0`, `hd0`, `hd1`) to the Linux view (`/dev/fd0`, `/dev/hda`, `/dev/hdb`). On a system with one or two IDE hard drives and perhaps a floppy drive, this will probably be correct. If a device map already exists, GRUB will reuse it without probing. If you just added a new drive and want to force a new device map to be generated add the `--resize` option to the `grub-install` command. For example

Once you have tested your boot floppy you are ready to install GRUB in the MBR of your hard drive. For the first IDE hard drive, you would use:

```
grub-install /dev/hda
or
grub-install '(hd0)'
```

To install it into the partition boot record for partition 11, use:

```
grub-install /dev/hda11
```

or
grub-install '(hd0,10)'

Remember that GRUB numbers from 0.

System updates

Most distributions provide tools for updating the system. These tools are usually aware of the boot loader in use and will often update your configuration file automatically. If you build your own custom kernel, or prefer to use a configuration file with a non-standard name or location, then you may need to update the configuration file yourself.

- If you use LILO, then you **must** run the lilo command whenever you update your configuration file or make changes such as adding a hard drive or deleting a partition.
- If you use GRUB, you can edit the /boot/grub/grub.conf file to make your changes and the GRUB stage 2 loader will read the file when you reboot. You do not normally need to reinstall GRUB just because you add a new kernel. However, if you move a partition, or add drives, you may need to reinstall GRUB. Remember the stage 1 loader is very small, so it simply has a list of block addresses for the stage 2 loader. Move the partition and the addressed change, so stage 1 can no longer locate stage 2. We'll cover some recovery strategies and also discuss GRUB's stage 1.5 loaders next.

Recovery

We will now look at some things that can go wrong with your carefully prepared boot setup, particularly when you install and boot multiple operating systems. The first thing to remember is to resist your initial temptation to panic. Recovery is usually only a few steps away. We will give you a few strategies here that should help you through many types of crisis.

These strategies and tools will show you that anyone who has physical access to a machine has a lot of power. Likewise, anyone who has access to a grub command line also has access to files on your system without the benefit of any ownership or other security provisions provided by a running system. Keep these points in mind when you select your boot loader. The choice between LILO and GRUB is largely a matter of personal preference. With what you have learned already and what we are about to show you, you should be equipped to choose the loader that best suits your particular needs and style of working.

Another install destroys your MBR.

Sometimes you will install another operating system and inadvertently overwrite your MBR. Some systems, such as DOS and Windows, always install their own MBR. It is usually very easy to recover from this situation. If you develop a habit of creating a boot floppy every time you run lilo or reinstall GRUB, you are home free. Simply boot into your Linux system from the floppy and rerun lilo or grub-install.

If you don't happen to have a boot floppy, but you still have almost any Linux distribution available, you can usually boot the Linux install media in a recovery mode. When you do so, the root filesystem on your hard drive will either be mounted at some strange recovery point, or the drive will not be mounted at all. You can use the `chroot` command to make this odd mount point become your root (`/`) directory. Then run lilo or grub-install to create a new boot floppy or to reinstall the MBR. I prefer to create a floppy and use it to boot, making sure that all is well before I go and rewrite the MBR, but you may be more courageous than I. Listing 9 shows an example using the environment we used for our earlier configuration examples. In this example, I booted a Red Hat Enterprise Linux boot disk which mounted `/dev/hda11` at `/mnt/sysimage`. Most rescue environments will dump you into a large screen with a prompt, rather than the graphical screen you might be more used to. Think of this as a terminal window with you logged in as root. In other words, be very careful what you write to your hard drive. In listing 9, user entry is shown in **bold**.

Listing 9. Using a rescue disk and chroot.

```
sh-3.00# chroot /mnt/sysimage
sh-2.05b# lilo
Added linux *
Added WIN-XP
sh-2.05b# grub-install '(fd0)'
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(fd0)    /dev/fd0
(hd0)    /dev/hda
(hd1)    /dev/hdc
(hd2)    /dev/sda
sh-2.05b#
```

Once you have your bootable floppy, press ctrl-d to exit from the chroot environment and then reboot your system, remembering to remove your installation media. If you don't happen to have an installation CD or DVD handy, there are many recovery and live LINUX CDs available online and some diskette or USB memory key ones too. See [Resources](#).

Although beyond the scope of this tutorial, you may wish to know that it is possible to have your MBR boot a Windows 2000 or Windows XP system and install Lilo or GRUB on a partition boot record. The ntldr boot program can also chain load other

boot sectors, although setup can be a little tricky. You will need to copy the boot sector to a Windows partition and modify the hidden boot.ini file to make this work.

You moved a partition.

If you moved a partition and forgot about your boot setup, you have a temporary problem. Typically, LILO or GRUB refuse to load. LILO will probably print an 'L' indicating that stage 1 was loaded and then stop. GRUB will give you an error message. What has happened here is that the stage 1 loader, which had a list of sectors to load to get to the stage 2 loader, can perhaps load the sectors from the addresses it has, but the sectors no longer have the stage 2 signature. If you built a boot diskette using the methods outlined earlier, remember that all that wither lilo or grub-install put on the diskette was a single boot sector, so your boot diskette probably won't help. As in the previous example, you will probably need to boot some kind of rescue environment and rebuild your boot floppy with LILO or GRUB. Then reboot, check your system and reinstall your boot loader in the MBR.

You may have noticed that our configuration examples used labels for partitions. for example,

```
append="hdd=ide-scsi root=LABEL=RHEL3"  
or  
kernel /boot/vmlinuz-2.4.20-31.9 ro root=LABEL=RH9 hdd=ide-scsi
```

I often use labels like this to help avoid problems when I move partitions. You still need to update the GRUB or LILO configuration file and rerun lilo, but you don't have to update /etc/fstab as well. This is particularly handy if I create a partition image on one system and restore it at a different location on another system.

Using a /boot partition.

Another approach to recovery, or perhaps avoiding it, is to use a separate partition for /boot. This partition need not be very large, perhaps 100MB or so. Put this partition somewhere where it is unlikely to be moved and where it is unlikely to have its partition number moved by the addition or removal of another partition. In a mixed Windows and Linux environment, /dev/hda2 is often a good choice for a partition for /boot.

Another reason for having a /boot partition arises when your root partition uses a file system not supported by your boot loader. For example, it is very common to have a /boot partition formatted ext2 or ext3 when the root partition (/) uses LVM.

If you have multiple distributions on your system, **do not** share the /boot partition between them. Remember to set up LILO or GRUB to boot from the partition that will later be mounted as /boot. Remember also that the update programs for a distribution will usually update the GRUB or LILO configuration for that system. In an environment with multiple systems you may want to keep one with its own /boot

partition as the main one and manually update that configuration file whenever an update of one of your systems requires it. Another approach is to have each system install a boot loader into its own partition boot record and have your main system simply chain load the partition boot records for the individual systems, giving you a two-stage menu process.

Building a self-contained boot diskette.

Finally, let's look a little more at the GRUB setup and how to make a standalone boot diskette that will get you to a GRUB prompt, no matter what has happened to your hard drive.

Remember all that stuff about cylinders on hard drives. Even though you might think of a cylinder as a fictional entity with modern drives, many aspects of your file system have not forgotten them. In particular, you will find partition use an integral number of cylinders aligned on cylinder boundaries. Within a partition, many file systems also manage space in units of cylinders. On many UNIX and Linux systems, the layout of the filesystem is stored in a *superblock* which is the first allocation unit in the file system. For systems such as ext2 or ext3 filesystems and reasonably large hard drives, the space will be broken into several sections with a copy of the superblock in the beginning of each section. This will help with recovery if you accidentally mess up partition boundaries with a program like fdisk.

One other benefit of the cylinder mentality is that there is some space at the beginning of a disk right after the MBR. GRUB takes advantage of this by embedding a stage 1.5 boot loader in this space or in similar otherwise unused space on a partition whenever possible. The stage 1.5 loader understands the file system on the partition that contains the stage 2, so it is somewhat more protected against problems associated with files being moved.

All that is well and good, but how does it relate to a bootable floppy. well, a floppy doesn't have much space or much notion of cylinders, so if you want to boot both stage 1 and stage 2 of GRUB from a floppy, you need to install stage 1 and then copy stage 2 to the sectors immediately following the boot sector. Listing 10 shows an example of how to do this. Use an empty diskette as this process will destroy data. You should copy the files that came with your grub distribution rather than the ones from your `/boot/grub` directory as `/boot/grub/stage2` has been modified to work with your hard drive partitions. You should find the original stage1 and stage2 files in a subdirectory of `/usr/share/grub`. In our example they are in `/usr/share/grub/i386-redhat`.

Listing 10. Creating a GRUB boot floppy.

```
[root@lyrebird root]# ls /usr/share/grub
i386-redhat
[root@lyrebird root]# cd /usr/share/grub/i386-redhat
[root@lyrebird i386-redhat]# ls -l st*
```

```

-rw-r--r--  1 root    root          512 Aug  3  2004 stage1
-rw-r--r--  1 root    root       104092 Aug  3  2004 stage2
[root@lyrebird i386-redhat]# dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
[root@lyrebird i386-redhat]# dd if=stage2 of=/dev/fd0 bs=512 seek=1

203+1 records in
203+1 records out

```

If you had formatted your floppy before you did this, and you now attempt to mount the floppy, the mount command will give you an error. Copying the stage2 right after the diskette's boot sector (seek=1) destroyed the filesystem on your diskette.

If you now boot this diskette, you will notice that the delay while it loads stage 2 from the diskette. You could boot this diskette in an arbitrary PC; it does not have to be one with a Linux system on it. When you boot the diskette, you will get a GRUB boot prompt. Press the tab key to see a list of commands available to you. Try `help commandname` to get help on the command called *commandname*. Listing 11, illustrates the GRUB command line.

Listing 11. The GRUB command line.

```

GRUB  version 0.93  (640K lower / 3072K upper memory)

[ Minimal BASH-like line editing is supported.  For the first word, TAB
  lists possible command completions.  Anywhere else TAB lists the possible
  completions of a device/filename.]

grub>
Possible commands are: blocklist boot cat chainloader clear cmp color configfi
le debug device displayapm displaymem dump embed find fstest geometry halt help
hide impsprobe initrd install ioprobe kernel lock makeactive map md5crypt modu
le modulenounzip pager partnew parttype password pause quit read reboot root ro
otnoverify savedefault serial setkey setup terminal terminfo testload testvbe u
nhide uppermem vbeprobe

grub> help rootnoverify
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
  Similar to `root', but don't attempt to mount the partition. This
  is useful for when an OS is outside of the area of the disk that
  GRUB can read, but setting the correct root device is still
  desired. Note that the items mentioned in `root' which derived
  from attempting the mount will NOT work correctly.

grub> find /boot/grub/grub.conf
(hd0,2)
(hd0,6)
(hd0,7)
(hd0,10)
(hd1,7)

grub>

```

In this example, we have found that there are GRUB config files on four different partitions on our first hard drive and another on the second hard drive. We could load the GRUB menu from one of these using the configfile command. For example:

configfile (hd0,2)/boot/grub/grub.conf

This would load the menu for that configuration file and we might be able to boot the system from this point. You can explore these grub commands in the GRUB manual. Try typing `info grub` in a Linux terminal window to open the manual.

One last point before we leave GRUB. We mentioned that the stage 2 GRUB file destroyed the file system on the diskette. If you want a bootable GRUB recovery diskette that loads GRUB files, including a configuration file from the diskette you can accomplish this using the following steps:

1. Use the `mkdosfs` command to create a DOS FAT filesystem on the diskette and use the `-R` option to reserve enough sectors for the stage 2 file.
2. Mount the diskette
3. Create a `/boot/grub` directory on the diskette
4. Copy the GRUB `stage1`, `stage2`, and `grub.conf` files to the `boot/grub` directory on the diskette. Copy your splash image file too if you want one.
5. Edit your `grub.conf` file on the diskette so it refers to the splash file on the diskette.
6. Unmount the diskette
7. Use the `grub` command shell to setup GRUB on the diskette using the GRUB root and setup commands.

We illustrate this in Listing 12.

Listing 12. Installing GRUB on diskette with a filesystem.

```
[root@lyrebird root]# mkdosfs -R 210 /dev/fd0
mkdosfs 2.8 (28 Feb 2001)
[root@lyrebird root]# mount /dev/fd0 /mnt/floppy
[root@lyrebird root]# mkdir /mnt/floppy/boot
[root@lyrebird root]# mkdir /mnt/floppy/boot/grub
[root@lyrebird root]# cp /boot/grub/stage1 /mnt/floppy/boot/grub
[root@lyrebird root]# cp /boot/grub/stage2 /mnt/floppy/boot/grub
[root@lyrebird root]# cp /boot/grub/splash* /mnt/floppy/boot/grub
[root@lyrebird root]# cp /boot/grub/grub.conf /mnt/floppy/boot/grub
[root@lyrebird root]# umount /dev/fd0
[root@lyrebird root]# grub
Probing devices to guess BIOS drives. This may take a long time.

GRUB version 0.93 (640K lower / 3072K upper memory)

[ Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists the possible
completions of a device/filename.]
```

```
grub> root (fd0)
Filesystem type is fat, using whole disk

grub> setup (fd0)
Checking if "/boot/grub/stage1" exists... yes
Checking if "/boot/grub/stage2" exists... yes
Checking if "/boot/grub/fat_stage1_5" exists... no
Running "install /boot/grub/stage1 (fd0) /boot/grub/stage2 p /boot/grub/grub.c
onf "... succeeded
Done.
```

With these tools you should now have enough to recover from most of the things that can go wrong with a boot loader.

Section 4. Make and install programs

This section covers material for topic 1.102.3 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you will learn how to build and install an executable program from source. You will learn how to unpack typical source bundles and customize Makefiles.

Why might you want to install a program from source? Frequent reasons include:

1. You need a program that is not part of your distribution.
2. you need a program that is only available as source.
3. You need some feature of a program that is only available by rebuilding the program from source.
4. You want to learn more about how a program works or you want to participate in its development.

These are all good reasons to consider installing a program from source.

Download and unpack

Regardless of your motivation for building from source, you will need to get the source before you can build it. You might find the package on a web site designed for hosting projects, such as the Open Source Technology Group's SourceForge.net (see [Resources](#)), or a web site dedicated to a particular package.

In this section we will look mostly at packages distributed as so-called *tarballs*. The `tar` (for *Tape ARchive*) command is used to create and manipulate *archives* from the files in a directory tree. Despite the name, the files can be stored on any media. In fact, storing them on disk allows manipulation, such as deleting part of an archive, that is not even possible on tape. The `tar` command itself does not compress the data, it merely stores it in a form from which the original files, permissions and directory structures can be restored. The `tar` command can be used in conjunction with a compression program, normally `gzip` or `bzip2` to create a compressed archive which saves storage space as well as transmission time. Such a compressed archive is a *tarball*.

Besides simple *tarballs*, source for a particular program may be packaged for your distribution in a *source package*, such as a source RPM (or SRPM). We will discuss package management later in this tutorial. For now, just remember to check for a source package for your distribution if one is available as that is usually a slightly easier way to build a program and it will already be tailored to the filesystem layout used by your distribution.

Before you download, try to learn as much as you can about the package. If there is installation or build documentation available, check it to see if you will need other packages in order to build it. Frequently, you will also need to install several libraries and perhaps development tools before you can successfully build your chosen program. This is especially likely to be true if your program uses any graphical toolkit. Sometimes you will start the build process and only then discover that you need a particular package. Don't worry, this is not uncommon. You'll just need to locate and install the missing packages and keep trying till you have all the required packages.

You will usually download using your browser or perhaps an ftp program. Your package will probably have a name that ends in one of `tar`, `tar.gz`, `tar.Z`, `tgz`, or `tar.bz2`. Sometimes you will download a package using CVS (Concurrent Version System). An example might be GNU GRUB 2 from the Free Software Foundation (see [Resources](#)). In this case, your downloaded source will be unpacked already. Occasionally you may find a `.zip` extension indicating a zip file.

Compressed tar files

Compressed tar files or *tarballs* are the most common form of source distribution for source that is not using package management such as Red Hat's RPM, or Debian's package management. These are created using the `tar` command which archives a directory tree and all its files in a single file. This will usually be compressed with some form of compression program, the usual ones being either `compress`, `gzip`, or `bzip2`. Because archiving and compressing is such a common operation, the GNU `tar` command found on most Linux systems can also handle compression and decompression using `compress`, `gzip` or `bzip2`. If your particular version of `tr` does not handle the particular compression type, UNIX and Linux systems are very good

at using pipelines to allow several commands to operate in sequence on one input source, so a two-stage process can accomplish manually what tar does for you under the covers anyway.

For illustration, suppose we download the Dr. Geo interactive geometry project (see [Resources](#)). At the time of writing, we downloaded drgeo-1.1.0.tar.gz. The gz extension tells us that this file is compressed with gzip. We will first show you how to extract the tar file from the compressed file and then how to extract the files from the tar archive. Then we will show you how to uncompress and extract with a single command or a pipeline.

To just extract the tar archive we use the `gunzip` command as shown in Listing 13.

Listing 13. Decompressing the Dr Geo source package.

```
[ian@localhost ~]$ ls drgeo*
drgeo-1.1.0.tar.gz
[ian@localhost ~]$ gunzip drgeo-1.1.0.tar.gz
[ian@localhost ~]$ ls drgeo*
drgeo-1.1.0.tar
```

Note that our `.tar.gz` file has now been replaced by a plain `.tar` file. For the other extensions mentioned, you would use one of the following commands (as appropriate) to extract the compressed tar file.

```
uncompress drgeo-1.1.0.tar.Z
gunzip drgeo-1.1.0.tar.Z
gunzip drgeo-1.1.0.tar.gz
gunzip drgeo-1.1.0.tgz
bunzip2 drgeo-1.1.0.tar.bz2
```

You'll notice that `gunzip` will handle `.Z`, `.tar.gz` and `.tgz`. In fact, your system may not even have the earlier compress and uncompress programs installed at all.

To extract the files from the tar archive, you use the `tar` command. The usual form, `tar -xvf filename.tar`, is shown in Listing 14. Optionally, you may pipe the output through the `less` filter in order to page through it.

Listing 14. Extracting files from the Dr Geo archive.

```
[ian@localhost ~]$ tar -xvf drgeo-1.1.0.tar |more
drgeo-1.1.0/
drgeo-1.1.0/po/
drgeo-1.1.0/po/ChangeLog
drgeo-1.1.0/po/Makefile.in.in
drgeo-1.1.0/po/POTFILES.in
drgeo-1.1.0/po/drgeo.pot
drgeo-1.1.0/po/az.po
```

```
drgeo-1.1.0/po/ca.po
drgeo-1.1.0/po/cs.po
...
```

The `-x` option tells tar to extract the files. The `-v` option tells tar to give verbose output. And the `-f` option, in conjunction with a file name (drgeo-1.1.0.tar in this case) tells tar what archive file to extract files from.

Well-behaved packages will create a directory in which to store the files of the package. In our example, this is the drgeo-1.1.0 directory. Occasionally, a package will not do this, so you might want to check before dumping a lot of files over your home directory. To do this, use the tar command with the `-t` option to display the table of contents, instead of using the `-x` to do the extraction. If you also drop the `-v` option, you will get enough output to see what files will be created and whether a directory will be created or whether everything will be dumped into the current directory.

Now that we have seen how to extract a tarball in two steps you are perhaps wondering about the claim that it could be done in one. It can. If you add the `-z` option to the tar command it can decompress and extract gzipped archives with a single command. For example:

```
tar -zxvf drgeo-1.1.0.tgz
or
tar -zxvf drgeo-1.1.0.tar.Z
```

To accomplish the same thing with an archive compressed with bzip2, use the `-j` option instead of the `-z` option. For example:

```
tar -jxvf drgeo-1.1.0.tar.bz2
```

You can also use the `-c` option on any of the above decompression commands to direct the decompressed file to standard output, which you then pipe as standard input the tar command. Note that this will leave your original file unchanged, rather than extracting it to a larger .tar file. Some examples are:

```
bunzip2 -c drgeo-1.1.0.tar.bz2 | tar -xvf -
uncompress -c drgeo-1.1.0.tar.Z | tar -xvf -
gunzip -c drgeo-1.1.0.tar.Z | tar -xvf -
gunzip -c drgeo-1.1.0.tar.gz | tar -xvf -
gunzip -c drgeo-1.1.0.tgz | tar -xvf -
```

Notes:

1. The `-` value for an archive file name tells tar to use standard input for the archive. Your version of tar may do this by default, in which case you do not need to specify the `-f` option at all. Just leave the trailing `f` off the

above commands.

2. The `zcat` command performs the same function as `gunzip -c`.

CVS trees

Sometimes the code for the project you need is not packaged in a tarball but is available through CVS (Concurrent Version System). At the time of writing, an example is the GRUB 2 project that we discussed in the section [Boot managers](#). Listing 15, shows an example.

Listing 15. Downloading GRUB2 with CVS.

```
[ian@attic4 ~]$ export CVS_RSH="ssh"
[ian@attic4 ~]$ cvs -z3 -d:ext:anoncvs@savannah.gnu.org:/cvsroot/grub co grub2
cvs server: Updating grub2
U grub2/.cvsignore
U grub2/AUTHORS
U grub2/COPYING
U grub2/ChangeLog
U grub2/DISTLIST
U grub2/INSTALL
U grub2/Makefile.in
U grub2/NEWS
...
```

The `export` command tells CVS how to connect to the remote server (using secure shell, or `ssh`, in this case). The `cvs` command checks out (`co` option) the `grub2` project. You will find all the project files in the `grub2` directory that the `cvs` command created for you.

Zip files

You will occasionally find source packaged as a zip file. This might be the case for a package which works on Windows as well as Linux or UNIX systems. The original PKZIP program was developed for DOS systems by PKWARE, Inc. and is now available on several other platforms. Many Linux systems include a version created by Info-ZIP.

Listing 16 shows how to use the `unzip` command to extract source for the `sphere` eversion program and screensaver.

Listing 16. Unzipping the sphere eversion source.

```
[ian@attic4 ~]$ unzip sphereEversion-0.4-src.zip
Archive:  sphereEversion-0.4-src.zip
  creating:  sphereEversion-0.4-src/
  inflating: sphereEversion-0.4-src/Camera.h
  inflating: sphereEversion-0.4-src/drawutil2D.h
  inflating: sphereEversion-0.4-src/drawutil.h
```

```
inflating: sphereEversion-0.4-src/fontdata.h
inflating: sphereEversion-0.4-src/fontDefinition.h
inflating: sphereEversion-0.4-src/generateGeometry.h
inflating: sphereEversion-0.4-src/global.h
inflating: sphereEversion-0.4-src/mathutil.h
inflating: sphereEversion-0.4-src/Camera.cpp
inflating: sphereEversion-0.4-src/drawutil2D.cpp
inflating: sphereEversion-0.4-src/drawutil.cpp
inflating: sphereEversion-0.4-src/fontdata.cpp
inflating: sphereEversion-0.4-src/generateGeometry.cpp
inflating: sphereEversion-0.4-src/main.cpp
inflating: sphereEversion-0.4-src/mathutil.cpp
inflating: sphereEversion-0.4-src/README.TXT
inflating: sphereEversion-0.4-src/Makefile
```

Build the program

Now that you have your source files unpacked into a directory tree let's look at how to build the program or programs.

Inspecting the source

Before you start building, you should look over what you unpacked. In particular, look for installation documentation. There will usually be a README or an INSTALL file, or perhaps both, located in the root directory of your new project. If the package is intended for multiple platforms, you might also find platform specific files such as README.linux or INSTALL.linux.

Configuration

You will often find a *configure* script in the main source directory. This script is designed to set up a *Makefile* that is customized to your system. It is often generated by the developers using the GNU autoconf program. The configure script will probe your system to determine its capabilities. The resulting Makefile, or Makefiles, will build the project on your particular system.

A complex configuration script may check many aspects of your system, including things such as processor type, whether it is a 32-bit or 64-bit system and so on. A simple configuration script may do little more than create Makefiles.

If you don't have a file called *configure* in your main project directory, check your documentation to see if there is some alternate method. If you do, try changing to the main project directory and running

```
./configure --help
```

This should give you some help on available configuration options. Many, such as `--prefix`, will occur in most configure scripts. Some are likely to be specific to the particular program you are building. Note the ones that you'd like to change.

Note: If your project does not include a configuration script, then it will probably have a Makefile which will work on most platforms, or some other form of installation process. For example, a package that uses only Python scripts and data files may not need to be built, so it may just have a script for installation.

In the tutorial for Topic 104, we will cover the Filesystem Hierarchy Standard (FHS). For now, we note that local programs should have executables stored in the `/usr/local` tree in `/usr/local/bin` and man pages in `/usr/local/man`. configure scripts are likely to have a `--prefix` option to specify the install location. If the program is not FHS-compliant you may need to specify this option when you run the configure script. If you are building a product to replace an installed product, you may need to install it with `/opt` or `/usr` as the prefix.

In addition to possibly specifying a prefix, you may find other options that relate to the location of specific components, such as `--mandir` or `--infodir` to specify the location of man and info pages respectively.

Once you have reviewed the possible options and identified any that you may need to change, run the `configure` script, adding any options you need. Remember to add `./` before the configure command as your project directory will probably not be in your path. For example, you might run

```
./configure
or
./configure --prefix /usr/local
```

When you run `configure`, You will usually see messages indicating what type of system you have and what required tools are present or not present. If all goes well, you should have a Makefile built at the end of the configure process.

config.cache

When the configure script completes, it will store information about the configuration in a file called `config.cache` which will be in the same directory as the configure script.

If you need to run `./configure` again be sure to remove your `config.cache` file first (using the `rm` command), as configure will use the settings from `config.cache` if it is present and it will not recheck your system.

Listing 17 shows some of the output from the configure step for the Dr Geo package that we unpacked earlier.

Listing 17. Configure Dr Geo.

```
[ian@localhost ~]$ cd drgeo-1.1.0
```

```
[ian@localhost drgeo-1.1.0]$ ./configure | less
checking for XML::Parser... ok
checking for iconv... /usr/bin/iconv
checking for msgfmt... /usr/bin/msgfmt
checking for msgmerge... /usr/bin/msgmerge
checking for xgettext... /usr/bin/xgettext
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking for g++... g++
checking for C++ compiler default output file name... a.out
checking whether the C++ compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
...
checking for guile... /usr/bin/guile
checking for guile-config... no
configure: error: guile-config required but not found
```

The configure script checks for a number of graphics conversion programs that are part of the netpbm package. There is a warning because one possible conversion program is not found on the system. There is also a warning that use of the `/usr/local` prefix will require root access (at the installation step). Since this is the first time that configure has been run, there are some error messages related to Makefiles that do not yet exist, but would exist if we ran configure again. Finally, the configure script reports success.

Make and Makefiles

Once the configuration step is complete, you should have a file named *Makefile* in your project directory. This is called a *make file* because a program called *make* is used to process it and build your program. You may also have several more make files in subdirectories.

A make file contains *rules*, which are the instructions that tell the make program how to build things. The file also contains *targets*, which tell the make program what to build. The make program analyzes the make file and determines the order in which items must be built. For example, if an executable is built from three object files, then the object files must be built before they can be linked into an executable. A make file may perform installation tasks as well as building your program. Make file targets will usually be available for several functions, such as:

make

with no options will just build the program. Technically this will build a *default target*, which usually means just build the program from sources.

make install

will install the program you built. You may need root authority for this if you are installing in `/usr/local`.

make clean

will erase files created by the make process.

make all

is sometimes used to perform the bulk of the make file's function with a single target. .

Consult your project documentation to see if there are additional targets or additional things you might need to do.

Now that your main Makefile has been created, use the `make` command, usually with no options, to build the executables, man pages and other parts of the program. Depending on the speed of your system and the complexity of the build process make may take only a minute or tow, or it may take much longer for a complex project.

Sometimes your build may not work. Common reasons include:

- Missing prerequisite packages
- Wrong level of prerequisite packages
- Wrong value for some parameter that you should have passed to `configure` or `make`.
- Missing compiler.
- Bugs in the `configure` script or generated Makefile.
- Source code bugs.

In our Dr Geo example, one of these problems was found at the configuration step, but this is not always the case. As you gain more experience with Linux, you will usually be able to identify and fix these problems. Sometimes you may need to check for a FAQ or mailing list that supports the package. Other times you may need to identify what you are missing and install it.

Installation

If all went well with your build, you are ready to install. The build step will build all the files you need, but they are not yet located in the correct places, ready for use. For example, binaries need to be copied to `/usr/local/bin`, and man pages to `/usr/local/man`, and so on.

Unless you specified a `--prefix` option, quite a few files and directories will probably be copied to your `/usr/local` tree. You will need root authority to write to the `/usr/local` tree in your filesystem. If you are not already logged in as root, use the `su` command to gain root authority. You will be prompted for the root password. Then

use `make install` to install your newly built program. Depending on the size of the program, the install may take several seconds up to a few minutes to complete. We show part of the output for installing Dr Geo in Listing 18.

Listing 18. Installing Dr Geo.

```
[ian@attic4 drgeo-1.1.0]$ su
Password:
[root@attic4 drgeo-1.1.0]# make install
Making install in po
make[1]: Entering directory `/home/ian/drgeo-1.1.0/po'
if test -n ""; then \
  /usr/local/share; \
else \
  /bin/sh ../mkinstalldirs /usr/local/share; \
fi
installing az.gmo as /usr/local/share/locale/az/LC_MESSAGES/drgeo.mo
installing ca.gmo as /usr/local/share/locale/ca/LC_MESSAGES/drgeo.mo
installing cs.gmo as /usr/local/share/locale/cs/LC_MESSAGES/drgeo.mo
installing da.gmo as /usr/local/share/locale/da/LC_MESSAGES/drgeo.mo
installing de.gmo as /usr/local/share/locale/de/LC_MESSAGES/drgeo.mo
installing el.gmo as /usr/local/share/locale/el/LC_MESSAGES/drgeo.mo
installing en_CA.gmo as /usr/local/share/locale/en_CA/LC_MESSAGES/drgeo.mo
installing en_GB.gmo as /usr/local/share/locale/en_GB/LC_MESSAGES/drgeo.mo
...
/usr/bin/install -c drgeo /usr/local/bin/drgeo
/bin/sh ./mkinstalldirs /usr/local/share/applications
/usr/bin/install -c -m 644 drgeo.desktop /usr/local/share/applications/drgeo.desktop
make[2]: Leaving directory `/home/ian/drgeo-1.1.0'
make[1]: Leaving directory `/home/ian/drgeo-1.1.0'
[root@attic4 drgeo-1.1.0]# exit
exit
[ian@attic4 drgeo-1.1.0]$
```

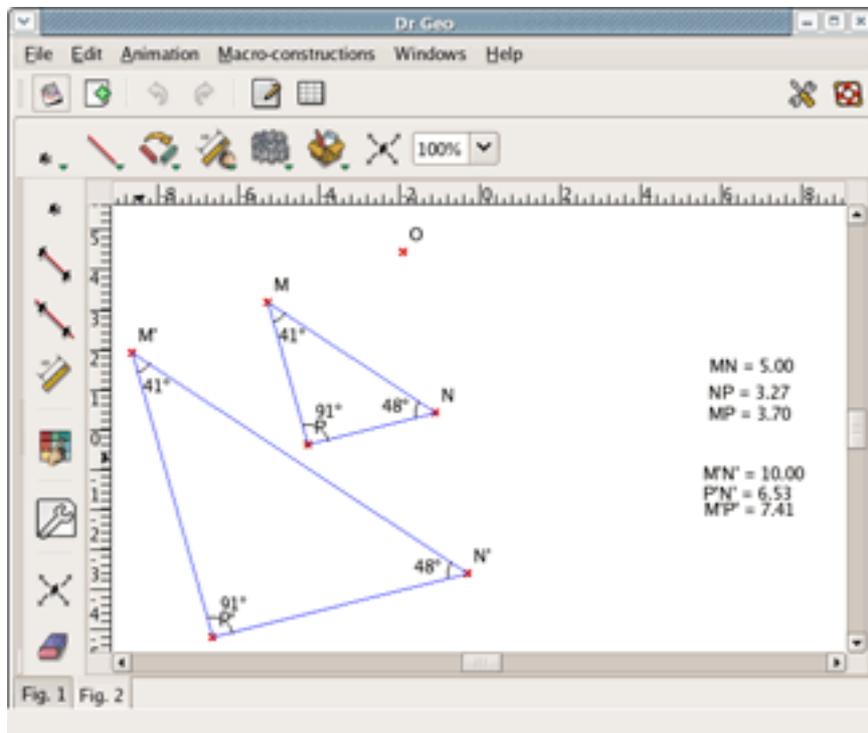
As well as copying files, `make install` should also make sure the installed files have the correct ownership and permissions. After the install finishes, the program is installed and ready for use, or possibly customize prior to using.

Note: It is very easy to make mistakes that can harm your system while you have root privileges, so remember to leave root mode by using the `exit` command or, in the bash shell, by pressing `ctrl-d`.

Run the program

If your program is now ready to run, you can try it out by typing the program name, `drgeo` in our example. Figure 1 shows a Dr Geo screen displaying one of the examples supplied with the program.

Figure 1. Running Dr Geo



Other things that you may need to do before running a program.

- Read the man page if one is part of the package. Try `man programname`.
- Customize configuration files, for example in `/etc`.
- Configure a program such as a server daemon to start automatically.

In this section we have covered relatively straight-forward installations from source. In the next few sections we'll talk more about libraries and library management as well as packages and how to install them.

Section 5. Manage shared libraries

This section covers material for topic 1.102.4 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you will learn how to determine the shared libraries that executable programs depend on. You will learn where system libraries are kept. We will cover installing packages, including shared libraries, in the next sections of this tutorial.

Static and dynamic executables

Linux systems have two types of executable program.

1. *Statically linked* executables contain all the library functions that they need to execute. All library functions are linked into the executable. They are complete programs that do not depend on external libraries to run. One advantage of statically linked programs is that they will work without installing prerequisites.
2. *Dynamically linked* executables are much smaller programs that are incomplete, in the sense that they require functions from external *shared* libraries in order to run. Besides being smaller, dynamic linking permits a package to specify prerequisite libraries without needing to include the libraries in the package. The use of dynamic linking also allows many running programs to share one copy of a library rather than occupying memory with many copies of the same code. For these reasons, most programs today use dynamic linking.

An interesting example on a typical Linux system is the `ln` command (`/bin/ln`) which creates links between files, either *hard* links, or *soft* (or *symbolic*) links. Shared libraries often involve symbolic links between a generic name for the library and a specific level of the library, so if the links aren't working, then the `ln` command might be inoperative. To protect against this possibility, Linux systems include a statically linked version of the `ln` program as the `sln` program (`/sbin/sln`). Listing 19 illustrates the great difference in size between these two programs.

Listing 19. Sizes of `sln` and `ln`.

```
[ian@lyrebird ian]$ ls -l /sbin/sln; ls -l /bin/ln
-rwxr-xr-x 1 root root 457165 Feb 23 2005 /sbin/sln
-rwxr-xr-x 1 root root 22204 Aug 12 2003 /bin/ln
```

The `ldd` command

Apart from knowing that a statically linked program is likely to be large, how do we tell whether a program is statically linked? And if it is dynamically linked, how do we know what libraries it needs? The answer to both questions is the `ldd` command which displays information about the library requirements of an executable. Listing 20 shows the output of the `ldd` command for the `ln` and `sln` executables.

Listing 20. Output of `ldd` for `sln` and `ln`.

```
[ian@lyrebird ian]$ ldd /sbin/sln /bin/ln
/sbin/sln:
not a dynamic executable
/bin/ln:
libc.so.6 => /lib/tls/libc.so.6 (0x00ebd000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00194000)
```

Since `ldd` is actually concerned with dynamic linking, it tells us that `sln` is statically linked by telling us that it is "not a dynamic executable", while it tells us the names of the two shared libraries (`libc.so.6` and `ld-linux.so.2`) that the `ln` command needs, as well as where to find these libraries. Note that `.so` indicates that these are *shared objects* or dynamic libraries. In Listing 21 we use the `ls -l` command to show that these are indeed symbolic links to specific versions of the libraries.

Listing 21. Library symbolic links.

```
[ian@lyrebird ian]$ ls -l /lib/tls/libc.so.6; ls -l /lib/ld-linux.so.2
lrwxrwxrwx 1 root root 13 May 18 16:24 /lib/tls/libc.so.6 -> libc-2.3.2.so
lrwxrwxrwx 1 root root 11 May 18 16:24 /lib/ld-linux.so.2 -> ld-2.3.2.so
```

Dynamic loading

From the preceding you might be surprised to learn that `ld-linux.so`, which looks like a shared library, is actually an executable in its own right. This is the code that is responsible for dynamically loading. It reads the header information from the executable which is in the *Executable and Linking Format* or (*ELF*) format. From this information it determines what libraries are required and which ones need to be loaded. It then performs dynamic linking to fix up all the address pointers in your executable and the loaded libraries so that the program will run.

You won't find a man page for `ld-linux.so`, but you will find it mentioned under the man entry for `ld.so`, `man ld.so`. Listing 22 illustrates using the `--list` option of `ld-linux.so` to show the same information for the `ln` command that we showed with the `ldd` command in Listing 20.

Listing 22. Using `ld-linux.so` to display library requirements.

```
[ian@lyrebird ian]$ /lib/ld-linux.so.2 --list /bin/ln
libc.so.6 => /lib/tls/libc.so.6 (0x00a83000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00f2c000)
```

Note that the hex addresses are different between the two listings. They are also likely to be different if you run `ldd` twice.

Dynamic library configuration

So how does the dynamic loader know where to look for executables? As with many things on Linux, there is a configuration file in `/etc`. In fact, there are two configuration files, `/etc/ld.so.conf` and `/etc/ld.so.cache`. Listing 23 shows the contents of `/etc/ld.so.conf` on two different systems. Note that on the `attic4` system (running fedora Core 4), `/etc/ld.so.conf` specifies that all the `.conf` files from the subdirectory `ld.so.conf.d` should be included. The actual contents of `/etc/ld.so.conf` may be different on your system.

Listing 23. Content of `/etc/ld.so.conf`.

```
[ian@lyrebird ian]$ cat /etc/ld.so.conf
/usr/kerberos/lib
/usr/X11R6/lib
/usr/lib/qt-3.1/lib
[
[ian@attic4 ~]$ cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
```

Loading of programs needs to be fast, so the `ld.so.conf` file is processed to the `ldconfig` command to process all the libraries from `ld.so.conf.d` as well as those from the trusted directories, `/lib` and `/usr/lib`. The dynamic loader uses the `ld.conf.cache` file to locate files that are to be dynamically loaded and linked. If you change `ld.conf` (or add new included files to `ld.so.conf.d`) you must run the `ldconfig` command (as root) to rebuild your `ld.conf.cache` file.

Normally, you use the `ldconfig` command without parameters to rebuild `ld.so.cache`. There are several other parameters you can specify to override this default behavior. As usual, try `man ldconfig` for more information. We illustrate the use of the `-p` parameter to display the contents of `ld.so.cache` in Listing 24.

Listing 24. Using `ldconfig` to display `ld.so.cache`.

```
[ian@lyrebird ian]$ /sbin/ldconfig -p | more
768 libs found in cache `/etc/ld.so.cache'
libzvt.so.2 (libc6) => /usr/lib/libzvt.so.2
libz.so.1 (libc6) => /usr/lib/libz.so.1
libz.so (libc6) => /usr/lib/libz.so
libx11globalcomm.so.1 (libc6) => /usr/lib/libx11globalcomm.so.1
libxsltbreakpoint.so.1 (libc6) => /usr/lib/libxsltbreakpoint.so.1
libxslt.so.1 (libc6) => /usr/lib/libxslt.so.1
libxmms.so.1 (libc6) => /usr/lib/libxmms.so.1
libxml2.so.2 (libc6) => /usr/lib/libxml2.so.2
libxml2.so (libc6) => /usr/lib/libxml2.so
libxmltok.so.0 (libc6) => /usr/lib/libxmltok.so.0
libxmlparse.so.0 (libc6) => /usr/lib/libxmlparse.so.0
libxml.so.1 (libc6) => /usr/lib/libxml.so.1
libxerces-c.so.24 (libc6) => /usr/lib/libxerces-c.so.24
...
lib-gnu-activation-20030319.so (libc6) => /usr/lib/lib-gnu-activation-20030319.so
```

```
ld-linux.so.2 (ELF) => /lib/ld-linux.so.2
```

If you're running an older application that needs a specific older version of a shared library, or if you're developing a new shared library or version of a shared library, you might want to override the default search paths used by the loader. This may also be needed by scripts that use product-specific shared libraries that are installed in the `/opt` tree.

Just as you can set the `PATH` variable to specify a search path for executables, you can set the `LD_LIBRARY_PATH` variable to a colon separated list of directories that should be searched for shared libraries before the system ones specified in `ld.so.cache`. For example, you might use a command like

```
export
LD_LIBRARY_PATH=/usr/lib/oldstuff:/opt/IBM/AgentController/lib
```

In the remaining sections of this tutorial, we will look at package management.

Section 6. Debian package management

This section covers material for topic 1.102.5 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 8.

In an earlier section we learned about installing programs from source. In this section you will learn about another alternative used by most distributions today, *package management*, in which prebuilt programs or sets of programs are distributed as a *package*, ready for installation on a particular distribution. In this section and the next we will look at package management, focusing on two widely used package management systems. These are the *Advanced Packaging Tool* or *APT* developed by Debian and the *Red Hat Package Manager* or *RPM* developed by Red Hat.

Package management overview

In the Dr Geo example of the earlier section, our configuration step failed initially because we did not have a particular prerequisite program. Package management tools formalize the notion of prerequisites and versions and standardize file locations on your system, as well as providing a tracking mechanism that helps you determine what packages are installed. The result is easier software installation, maintenance and removal.

While you may still want to install programs from source for the reasons enumerated in the previous section, you will probably do most of your system maintenance and program installation using the package manager that was set up for your distribution.

From a user perspective, the basic package management function is provided by commands. As Linux developers have striven to make Linux easier to use, the basic tools have been supplemented by other tools, including GUI tools which hide some of the complexities of the basic tools from the end user. In these two sections we focus on the basic tools, although we will mention some of the other tools so that you have a starting place to learn about them.

Installing Debian packages

Let us return to the problem we encountered earlier with the Dr Geo program source. As it happens, that problem occurred on a Fedora Core 4 system which uses RPM package management. Fortunately for this section of the tutorial, I was also missing some guile components on a Debian-based Ubuntu system where I tried installing Dr Geo. The error this time is shown in Listing 25.

Listing 25. Missing guile function.

```
ian@attic4:~$ cd drgeo-1.1.0
ian@attic4:~/drgeo-1.1.0$ ./configure
checking for perl... /usr/bin/perl
checking for XML::Parser... ok
checking for iconv... /usr/bin/iconv
checking for msgfmt... /usr/bin/msgfmt
...
checking for guile... no
configure: error: guile required but not found
i
```

The package we need is the guile package. We can install that using the `apt-get` command as shown in Listing 26. Note the use of the `sudo` command which is the usual Ubuntu method of doing work with root authority.

Listing 26. Installing guile using apt-get.

```
ian@attic4:~$ sudo apt-get install guile
Reading package lists... Done
Building dependency tree... Done
Note, selecting guile-1.6 instead of guile
Suggested packages:
  guile-1.6-doc
The following NEW packages will be installed:
  guile-1.6
0 upgraded, 1 newly installed, 0 to remove and 24 not upgraded.
Need to get 31.5kB of archives.
After unpacking 209kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com hoary/main guile-1.6 1.6.7-1ubuntu1 [31.5kB]
Fetched 31.5kB in 0s (37.4kB/s)
```

```
Preconfiguring packages ...
Selecting previously deselected package guile-1.6.
(Reading database ... 84435 files and directories currently installed.)
Unpacking guile-1.6 (from ../guile-1.6_1.6.7-1ubuntu1_i386.deb) ...
Setting up guile-1.6 (1.6.7-1ubuntu1) ...
i
```

From the output we see that apt-get has read a package list from somewhere (more on that shortly), built a dependency tree, determined that guile-doc is recommended for installation with guile, and downloaded the guile package from the Internet. The guile package has then been unpacked, installed and set up. Note that the extension used for Debian packages is .deb. The full file name of our guile package is guile-1.6_1.6.7-1ubuntu1_i386.deb.

If apt-get notices that the package you are trying to install depends on other packages, it will automatically fetch and install those as well. In our example, only guile was installed, because all dependencies were already satisfied. Based on advice in the output, we could install guile-doc (or guile-1.6.doc).

Suppose that, instead of installing guile-doc, we wanted to find out whether the installation of guile-doc depends on other packages. We can use the `-s` (for *simulate*) option on apt-get. There are several other options with equivalent function, such as `--just-print` and `--dry-run`. Check the man pages for full details. Not surprisingly, the documentation for a package we have just installed doesn't have any prerequisites, so in Listing 27 we illustrate a slightly more useful example with a simulated install of the `ssl-cert` package which requires the `openssl` package.

Listing 27. Simulated or dry-run install of `ssl-cert`.

```
ian@attic4:~$ sudo apt-get -s install ssl-cert
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
 openssl
Suggested packages:
 ca-certificates
The following NEW packages will be installed:
 openssl ssl-cert
0 upgraded, 2 newly installed, 0 to remove and 24 not upgraded.
Inst openssl (0.9.7e-3 Ubuntu:5.04/hoary)
Inst ssl-cert (1.0-11 Ubuntu:5.04/hoary)
Conf openssl (0.9.7e-3 Ubuntu:5.04/hoary)
Conf ssl-cert (1.0-11 Ubuntu:5.04/hoary)
```

We see that two new packages are required and we see the order in which they will be installed and configured.

Package resource list: apt-setup

We mentioned that apt-get read a package list from somewhere. That somewhere is `/etc/apt/sources.list`. This is a list that you can edit yourself, but you will probably

prefer to set up using the `apt-setup` command. The `apt-setup` command is an interactive tool that knows the location of the main APT repositories. You can access package sources on a CD-ROM, your local file system, or over a network using `http` or `ftp`.

If your distribution installed a `/etc/apt/sources.list` file for you, it may not have your CD-ROM as a source for packages. This can be inconvenient, particularly in the early stages of experimenting with a new system when you might want to add many packages, most of which have not yet been updated. In this case, you can use the command

```
apt-cdrom add
```

to add your CD-ROM to the list of package sources.

`Apt-get` and the other tools we will learn about, use a local database to determine what packages are installed. They can check installed levels against available levels. To do this, information on available levels is retrieved from the sources listed in `/etc/apt/sources.list` and stored on your local system. If you update your `/etc/apt/sources.list` file, then you should run

```
apt-get update
```

This will bring your stored data about available package levels up-to-date. In general, you should always do this before installing any new package.

Removing or upgrading packages.

Before leaving `apt-get`, we will mention two other useful options.

If you installed a package and later want to uninstall it, you use the `remove` option with `apt-get`. Listing 28 shows how to remove the `guile` package that we installed earlier.

Listing 28. Removing the guile package.

```
ian@attic4:~$ sudo apt-get remove guile
Reading package lists... Done
Building dependency tree... Done
Note, selecting guile-1.6 instead of guile
The following packages will be REMOVED:
  guile-1.6
0 upgraded, 0 newly installed, 1 to remove and 24 not upgraded.
Need to get 0B of archives.
After unpacking 209kB disk space will be freed.
Do you want to continue [Y/n]? Y
(Reading database ... 84455 files and directories currently installed.)
Removing guile-1.6 ...
```

The other option we will mention is the `upgrade` option. This option upgrades all the

installed packages on your system to the latest levels. Do not confuse this with the `update` option which merely refreshes the information about available packages.

For more information on other capabilities and options for `apt-get`, see the man page.

The `apt.conf` file

If you check the man page for `apt-get`, you will find that there are many options. If you use the `apt-get` command a lot and you find the default options are not to your liking, you can set new defaults in `/etc/apt/apt.conf`. A program, `apt-config` is available for scripts to interrogate the `apt.conf` file. See the man pages for `apt.conf` and `apt-config` for further information.

Debian package information

We will now look at some tools for getting information about packages. Some of these tools also do other things, but we will focus here on the informational aspects.

Package status with `dpkg`

Another tool that is part of the APT system is the `dpkg` tool. This is a medium level package management tool which can install and remove packages as well as displaying status information. Configuration of `dpkg` may be controlled by `/etc/dpkg/dpkg.cfg`. Individual users may also have a `.dpkg.cfg` file in their home directory to provide further configuration. If you do not have either of these files check `/usr/share/doc/dpkg/dpkg.cfg` for an example.

The `dpkg` tool uses many files in the `/var/lib/dpkg` tree in your filesystem. In particular, the file `/var/lib/dpkg/status` contains status information about packages on your system. Listing 29 shows the use of `dpkg -s` to display the status of the `guile` package after we installed it. Remember that we actually installed `guile-1.6`. We see in Listing 29 that we need to give the full form of the name, not just the abbreviated form.

Listing 29. Guile package status.

```
ian@attic4:~$ dpkg -s guile
Package `guile' is not installed and no info is available.

Use dpkg --info (= dpkg-deb --info) to examine archive files,
and dpkg --contents (= dpkg-deb --contents) to list their contents.
ian@attic4:~$ dpkg -s guile-1.6
Package: guile-1.6
Status: install ok installed
Priority: optional
Section: interpreters
Installed-Size: 204
Maintainer: Rob Browning <rlb@defaultvalue.org>
```

```

Architecture: i386
Version: 1.6.7-lubuntul
Provides: guile
Depends: guile-1.6-libs, libc6 (>= 2.3.2.ds1-4), libguile-ltdl-1
Suggests: guile-1.6-doc
Conflicts: libguile-dev (<= 1:1.4-24)
Description: The GNU extension language and Scheme interpreter
Guile is a Scheme implementation designed for real world programming,
providing a rich Unix interface, a module system, an interpreter, and
many extension languages. Guile can be used as a standard #! style
interpreter, via #!/usr/bin/guile, or as an extension language for
other applications via libguile.

```

Packages and the files in them

You will often want to know what is in a package or what package a particular file came from. These are both tasks for `dpkg`. Listing 30 illustrates the use of `dpkg -L` to list the files (including directories) installed by the `guile` package.

Listing 30. What is in the guile package?

```

root@attic4:~# dpkg -L guile-1.6
/.
/usr
/usr/bin
/usr/bin/guile-1.6-snarf
/usr/bin/guile-1.6-tools
/usr/bin/guile-1.6
/usr/bin/guile-1.6-config
/usr/share
/usr/share/guile
/usr/share/guile/1.6
/usr/share/guile/1.6/scripts
/usr/share/guile/1.6/scripts/autofrisk
/usr/share/guile/1.6/scripts/display-commentary
/usr/share/guile/1.6/scripts/doc-snarf
/usr/share/guile/1.6/scripts/frisk
/usr/share/guile/1.6/scripts/generate-autoload
/usr/share/guile/1.6/scripts/lint
/usr/share/guile/1.6/scripts/PROGRAM
/usr/share/guile/1.6/scripts/punify
/usr/share/guile/1.6/scripts/read-scheme-source
/usr/share/guile/1.6/scripts/snarf-check-and-output-texi
/usr/share/guile/1.6/scripts/snarf-guile-m4-docs
/usr/share/guile/1.6/scripts/use2dot
/usr/share/doc
/usr/share/doc/guile-1.6
/usr/share/doc/guile-1.6/copyright
/usr/share/doc/guile-1.6/changelog.Debian.gz
/usr/lib
/usr/lib/menu
/usr/lib/menu/guile-1.6

```

To find which package contains a specific file, use the `-S` option of `dpkg`, as shown in Listing 31. The name of the package is listed on the left.

Listing 31. What package contains a file?

```
ian@attic4:~$ dpkg -S /usr/share/guile/1.6/scripts/lint
guile-1.6: /usr/share/guile/1.6/scripts/lint
```

You may notice that the list in Figure 30 did not include `/usr/bin/guile`, yet the command `which guile` says that this is the program that will be run if you type `guile`. When this occurs you may need to do some extra sleuth work to find where a package comes from. For example, the installation set up step may perform tasks such as creating symbolic links that are not listed as part of the package contents. A recent addition to Linux systems is the *alternatives* system which is managed using the `update-alternatives` command. In Listing 32, we show how to use the `ls` command to see what the `guile` command is symbolically linked to. The link to the `/etc/alternatives` directory is a tip off that we are using the alternatives system, so we use the `update-alternatives` command to find more information and finally we can use the `dpkg -S` command to confirm that the `guile` command comes from the `guile-1.6` package. The setup for the alternatives system would have been done by a post install script that is part of the `guile-1.6` package.

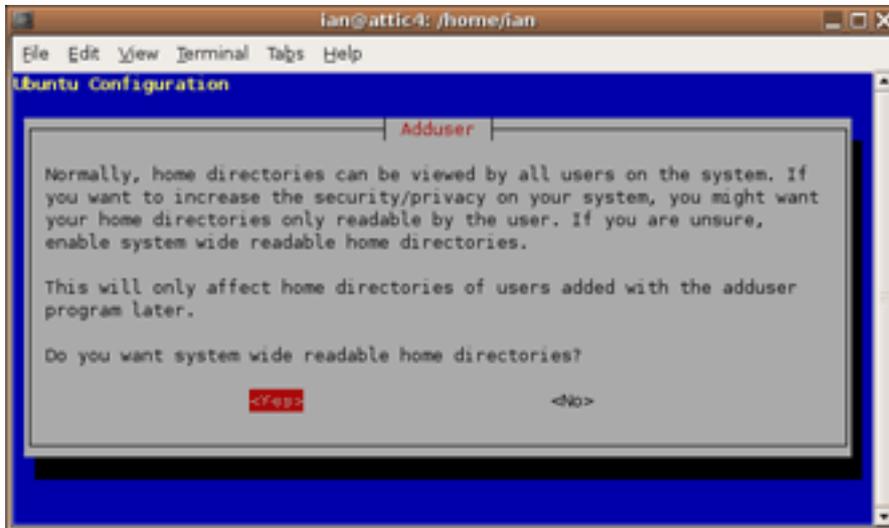
Listing 32. A more complex use of `dpkg -S`

```
ian@attic4:~$ ls -l $(which guile)
lrwxrwxrwx 1 root root 23 2005-09-06 23:38 /usr/bin/guile -> /etc/alternatives/guile
ian@attic4:~$ update-alternatives --display guile
guile - status is auto.
  link currently points to /usr/bin/guile-1.6
/usr/bin/guile-1.6 - priority 160
  slave guile-config: /usr/bin/guile-1.6-config
  slave guile-snarf: /usr/bin/guile-1.6-snarf
  slave guile-tools: /usr/bin/guile-1.6-tools
Current `best' version is /usr/bin/guile-1.6.
ian@attic4:~$ dpkg -S /usr/bin/guile-1.6
guile-1.6: /usr/bin/guile-1.6
```

Reconfiguring Debian packages.

APT includes a capability called *debconf* which is used to configure packages after they are installed. Packages that use this capability (and not all do) can be reconfigured after they are installed. The easiest way to do this is to use the `dpkg-reconfigure` command. For example, the `adduser` command may create home directories that are readable by all system users. You may not want this for privacy reasons. Figure 2 illustrates the configuration question applicable to the `adduser` package. Run `dpkg-reconfigure adduser` (as root) to generate this screen.

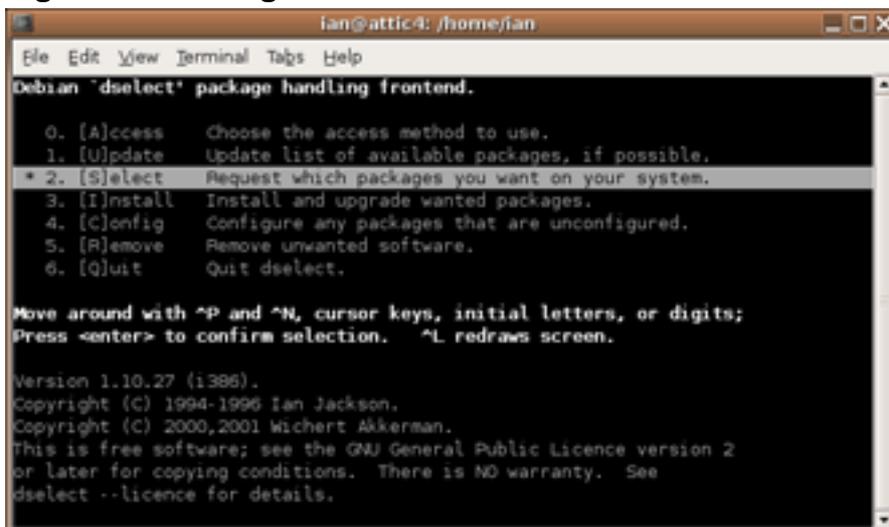
Figure 2. Using `dpkg-reconfigure`



Using dselect

Earlier, we mentioned that the status for packages is kept in `/var/lib/dpkg/status`. We also mentioned that `dpkg` could do more than just display package information. We will now take a brief look at the `dselect` command which provides a text-based full-screen interface (using `ncurses`) to the package management functions of `dpkg`. You can use `dselect` to install or remove packages as well as to control status flags that indicate whether packages should be kept up-to-date or held in their present state, for example. If you run the `dselect` command (as root) you will see a screen similar to that of Figure 3.

Figure 3. Running dselect



Using dselect Select mode

You can view and change the status of each package by choosing the Select option

that we highlighted in figure 3. You will then see a help screen. Press the space key to exit help at any time. You will then see a list of packages and package groups.

You can search for packages using / followed by a search string. Figure 4 shows the result of searching for "guile".

Figure 4. Selection screen for dselect

```

ian@attic4: /home/ian
File Edit View Terminal Tabs Help
dselect - main package listing (avail., priority)  mark:*/=/- verbose:v help:
EIOM Pri Section Package Inst.ver Avail.ver Description
-----
Up-to-date Optional packages in section interpreters
*** Opt interpre guile-1.6 1.6.7-lubun 1.6.7-lubun The GNU extension languag
*** Opt interpre liburi-perl 1.90-1 1.90-1 Manipulates and accesses
*** Opt interpre python-osd 0.2.6-1.lub 0.2.6-1.lub Python bindings for X On-
*** Opt interpre python-pyao 0.02-lubunt 0.02-lubunt A Python interface to the
*** Opt interpre python-pyogg 1.3-lubuntu 1.3-lubuntu A Python interface to the
*** Opt interpre python2.4-os 0.2.6-1.lub 0.2.6-1.lub Python bindings for X On-
*** Opt interpre tcl8.4 8.4.7-lubun 8.4.7-lubun Tcl (the Tool Command Lan
guile-1.6 installed ; install (was: install). Optional
guile-1.6 - The GNU extension language and Scheme interpreter

Guile is a Scheme implementation designed for real world programming,
providing a rich Unix interface, a module system, an interpreter, and many
extension languages. Guile can be used as a standard #! style interpreter,
via #!/usr/bin/guile, or as an extension language for other applications
via libguile.

description of guile-1.6
  
```

Package selection states

The selection state for each package can be seen under the cryptic heading *EIOM*. These letters stand for *Error*, *Installed state*, *Old mark*, and *Mark*. You may use the "v" key to toggle between the brief form of this display and a form that displays that states as words.

The fourth, or M, column is the one we look at now. This describes what will happen after we finish with the selection screen and move to the install screen. The marks have the following meanings:

*

Install or upgrade to the latest version

=

Hold this package in its present state and version

- (hyphen)

Delete the package but keep its configuration in case it is reinstalled later

_ (underscore)

Delete this package and purge the configuration.

To change the mark, press the corresponding key, except that you press the "+" key to mark a package for install or upgrade. When you are finished, press **Enter** to

confirm your package selection changes or press "X" (upper case X) to cancel without saving your changes. This will return you to the screen of Figure 3, with the Install option selected. Press **Enter** to install or upgrade your system.

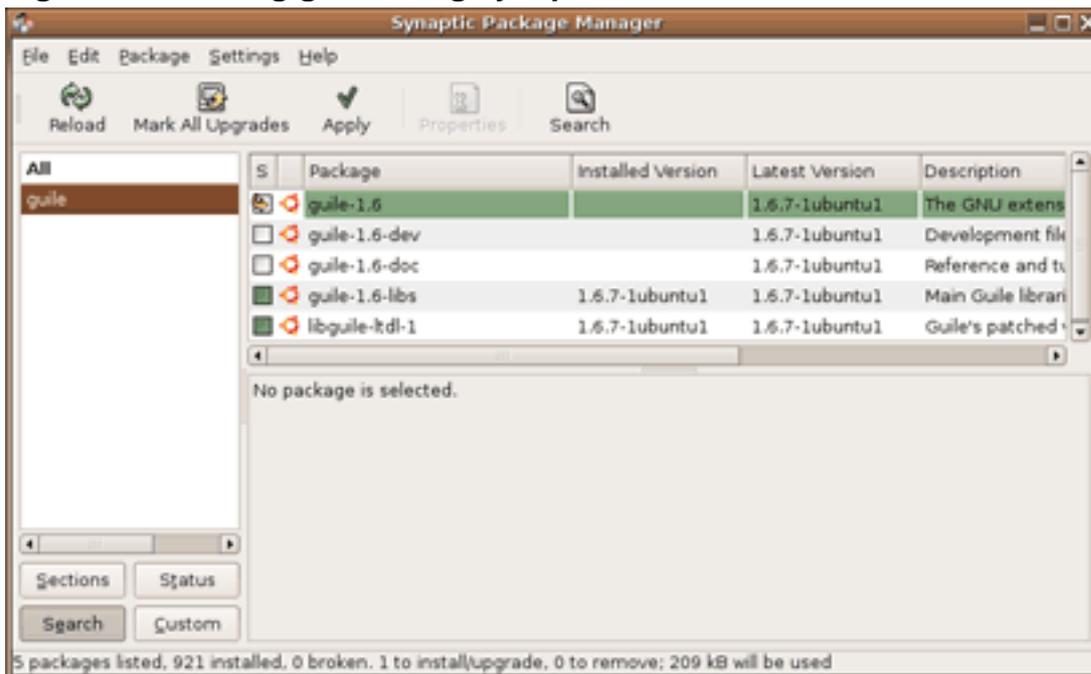
If you need help at any time, type "?" (question mark). Press the space to get back out of help.

Upgrading Debian with other tools

We have now seen that `dselect` can help you install or remove individual packages as well as upgrade all the packages on your system to the latest level. If you want to do this from the command line, you can use `apt-get dselect-upgrade`, which honors the status marks that we saw set with `dselect`.

In addition to `dselect`, there are several other interactive package management interfaces for Debian systems, including `aptitude`, `synaptic`, `gnome-apt` and `wajig`. `Synaptic` is a graphical application for use with the X Window System. Figure 5 shows the `synaptic` user interface with our old friend, the `guile` package, marked for installation.

Figure 5. Installing guile using synaptic



The **Apply** button will install `guile` and update any other packages that are scheduled for update. The **Reload** button will refresh the package lists. If you are used to GUI interfaces, you may find `synaptic` easier to use than `apt-get`, `dpkg` or `dselect`.

Finding Debian packages

In our final topic on Debian package management, we will look at ways to find packages. Usually, `apt-get` and the other tools we have looked at will already know about any Debian package you might need from the list of available packages. A command that we haven't used yet is `apt-cache`, which is useful for searching package information on your system. `apt-cache` can search using regular expressions (which we will learn more about in the tutorial for Topic 103). Suppose we wanted to find the name of the package containing the Linux loader. Listing 33 shows how we might accomplish this.

Listing 33. Searching for the Linux loader with `apt-cache`

```
ian@attic4:~$ apt-cache search "linux loader"
lilo - LInux LOader - The Classic OS loader can load Linux and others
lilo-doc - Documentation for LILO (LInux LOader)
```

We saw earlier that `dselect` and `synaptic` also offer search tools. If you use `synaptic`, note that you have options with the search menu to specify whether to search only package names or package descriptions as well.

If you still can't find the package, you may be able to find it at package among the list of packages on the Debian site (see [Resources](#)), or elsewhere on the Internet.

If you do find and download a `.deb` file, you can install it using `dpkg -i`. For example, our Dr Geo example happens to be available as a `.deb` package from the official Debian package repository.

Listing 34. Installing Dr Geo from a `.deb` package

```
ian@attic4:~$ ls drg*.deb
drgeo_1-1.0.0-1_i386.deb
ian@attic4:~$ sudo dpkg -i drgeo_1-1.0.0-1_i386.deb
Password:
Selecting previously deselected package drgeo.
(Reading database ... 84435 files and directories currently installed.)
Unpacking drgeo (from drgeo_1-1.0.0-1_i386.deb) ...
Setting up drgeo (1.0.0-1) ...
```

Note that the source tarball for this package was at a higher level (1.1.0) than the deb package (1.0.0-1). If you installed Dr Geo and, for some reason, it did not work, you might need to try installing from source.

If all else fails, there is another possible source for packages. Suppose you find a program packaged as an RPM rather than a `.deb`. You may be able to use the `alien` program which can convert between package formats. You should read the `alien` documentation carefully as not all features of all package management

systems can be converted to another format by alien.

There is a lot more to the Debian package management system than covered here. There is also a lot more to Debian than its package management system. See [Resources](#) for additional links.

Section 7. Red Hat Package Manager (RPM)

This section covers material for topic 1.102.6 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 8.

In the previous section on Debian package management, we gave you a brief [package management overview](#). In this section we will focus on the *Red Hat Package Manager* or *RPM* developed by Red Hat. RPM and APT have many similarities. Both can install and remove packages. Both keep a database of installed packages. Both have basic command line functionality as well as other tools to provide more user-friendly interfaces. Both can retrieve packages from the Internet. In general, there are not as many programs to deal with RPM packages as there are for APT packages, although the `xrpm` command has extensive capabilities. Another difference is that RPM does not maintain information about available packages on your system to the same extent that `dpkg` does.

Red Hat introduced RPM in 1995. RPM is now the package management system used for packaging in the Linux Standard Base (LSB). The `rpm` command options are grouped into three subgroups for:

- Querying and verifying packages
- Installing, upgrading and removing packages
- performing miscellaneous functions.

We will focus on the first two in this tutorial. You will find information about the miscellaneous functions in the man pages for `rpm`.

We should also note that `rpm` is the command name for the main command used with RPM, while `.rpm` is the extension used for RPM files. So "an rpm" or "the xxx rpm" will generally refer to an RPM file, while "rpm" will usually refer to the command.

Installing and removing RPM packages

As we did in the previous section, we'll look at the problems we encountered installing Dr Geo on a Fedora Core 4 system in the section [Make and install programs](#). You may recall from Listing 17 that we were missing the `guile-config` command.

Getting started with rpm

The `rpm` command can install packages from local filesystems or from the Internet using either `http` or `ftp`. Listing 35 shows installation of the `guile-devel` package using the `rpm -ivh` command and a network source for the package.

Listing 35. Installing guile-devel with rpm

```
[root@attic4 ~]# rpm -ivh http://download.fedora.redhat.com/pub/fedora\
> /linux/core/4/i386/os/Fedora/RPMS/guile-devel-1.6.7-2.i386.rpm
Retrieving http://download.fedora.redhat.com/pub/fedora/linux/core/4/i386/os/Fedora/
RPMS/guile-devel-1.6.7-2.i386.rpm
Preparing...                               ##### [100%]
 1:guile-devel                               ##### [100%]
```

Note that the `-v` option gives verbose output and the `-h` option shows hash marks (`#`) to indicate progress. If you wanted to examine the package before installing from the network, you might want to download it first and then install. We'll talk about examining packages in a moment, but for now, let us use the `wget` command to retrieve the package and then install it from our local filesystem **without** using the `-vh` options. The output is shown in Listing 36.

Listing 36. Installing guile-devel from a file

```
[root@attic4 ~]# wget http://download.fedora.redhat.com/pub/fedora/\
> linux/core/4/i386/os/Fedora/RPMS/guile-devel-1.6.7-2.i386.rpm
--22:29:58-- http://download.fedora.redhat.com/pub/fedora/linux/core/4/i386/os/Fedora/
RPMS/guile-devel-1.6.7-2.i386.rpm
=> `guile-devel-1.6.7-2.i386.rpm'
Resolving download.fedora.redhat.com... 209.132.176.221
Connecting to download.fedora.redhat.com[209.132.176.221]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 481,631 [application/x-rpm]

100%[=====] 481,631 147.12K/s ETA 00:00

22:30:02 (140.22 KB/s) - `guile-devel-1.6.7-2.i386.rpm' saved [481,631/481,631]

[root@attic4 ~]# ls guil*
guile-devel-1.6.7-2.i386.rpm
[root@attic4 ~]# rpm -i guile-devel-1.6.7-2.i386.rpm
```

No hash marks and no messages.

Re-installing an rpm

If you performed the above commands yourself, you would have seen an error on the second installation (or the first if you already had guile-devel installed) as it would have reported that guile-devel was already installed. To get around this, you should use the `-e` option to remove (or erase) the rpm before reinstalling it as shown in Listing 37. This would also apply if you needed to reinstall an rpm because you accidentally deleted some of its files.

Listing 37. Removing guile-devel

```
[root@attic4 ~]# rpm -e guile-devel
```

Forcibly installing an rpm

Sometimes removing an rpm isn't practical, particularly if there are other programs on the system that depend on it. For example, if you were trying to remove the guile package instead of the guile-devel package, you might see output like Listing 38, where many installed packages have dependencies on the guile package, so removal is not allowed.

Listing 38. Attempting to remove guile.

```
[root@attic4 ~]# rpm -q -R guile-devel
/bin/sh
/usr/bin/guile
guile = 5:1.6.7
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
[root@attic4 ~]# rpm -e guile
error: Failed dependencies:
  libguile-ltdl.so.1 is needed by (installed) g-wrap-1.3.4-8.i386
  libguile-ltdl.so.1 is needed by (installed) gnucash-1.8.11-3.i386
  libguile.so.12 is needed by (installed) g-wrap-1.3.4-8.i386
  libguile.so.12 is needed by (installed) gnucash-1.8.11-3.i386
  libqthreads.so.12 is needed by (installed) g-wrap-1.3.4-8.i386
  libqthreads.so.12 is needed by (installed) gnucash-1.8.11-3.i386
  guile is needed by (installed) g-wrap-1.3.4-8.i386
  guile = 5:1.6.7 is needed by (installed) guile-devel-1.6.7-2.i386
  /usr/bin/guile is needed by (installed) guile-devel-1.6.7-2.i386
```

Needless to say, it is impractical to remove all the packages with dependencies in such a case. The answer is to forcibly install the rpm using the `--force` option. In Listing 39 we illustrate forcibly reinstalling guile-devel from the file we downloaded for Listing 36.

Listing 39. Installing guile-devel with --force option.

```
[root@attic4 ~]# rpm -ivh --force guile-devel-1.6.7-2.i386.rpm
Preparing... ##### [100%]
 1:guile-devel ##### [100%]
```

Forcibly removing an rpm

There is an alternative to forcible install with the `--force` option that you may need on some occasions. You can remove an rpm using the `--nodeps` option which disables the internal dependency checking. You should generally do this **only** if you know what you are doing and **only** if you intend to fix the dependency problems by reinstalling the package. An example might be if you needed to revert to an earlier level of a package for some reason and wanted to be sure that all vestiges of the later version had been removed. The command you would use to remove the guile package without dependency checks is

```
rpm -e --nodeps guile
```

It is also possible to use the `--nodeps` option when installing an rpm. Again, this is not recommended, but may sometimes be necessary:

Upgrading RPM packages

Now that you know how to install and remove an RPM, let's look at upgrading an RPM to a newer level. This is similar to installing, except the we use the `-U` or the `-F` option instead of the `-i` option. The difference between these two options is that the `-U` option will upgrade an existing package **or** install the package if it is not already installed, while the `-F` option will only upgrade or *freshen* a package that is already installed. Because of this, the `-U` option is frequently used, particularly when the command line contains a list of RPMs. This way, uninstalled packages are installed while installed packages are upgraded. Listing 40 shows the effect of attempting to upgrade guile-devel to the level that it is already at and then removing it and retrying the upgrade (which now functions as an install).

Listing 40. Upgrading guile-devel.

```
[root@attic4 ~]# rpm -Uvh guile-devel-1.6.7-2.i386.rpm
Preparing...                               ##### [100%]
package guile-devel-1.6.7-2 is already installed
[root@attic4 ~]# rpm -e guile-devel
[root@attic4 ~]# rpm -Uvh guile-devel-1.6.7-2.i386.rpm
Preparing...                               ##### [100%]
 1:guile-devel                             ##### [100%]
```

Querying RPM packages

You may have noticed in our examples that installing an rpm requires the full name of the package file (or URL), such as `guile-devel-1.6.7-2.i386.rpm`. On the other hand, removing the rpm requires only the package name, such as `guile-devel`. As with APT, RPM maintains an internal database of your installed packages, allowing

you to manipulate installed packages using the package name. In this part of the tutorial, we will look at some of the information that is available to you from this database using the `-q` (for *query*) option of the `rpm` command.

The basic query simply asks if a package is installed. Add the `-i` option and you will get information about the package. Note that you need to have root authority to install, upgrade or remove packages, but non-root users can perform queries against the `rpm` database.

Listing 41. Displaying information about guile-devel.

```
[ian@attic4 ~]$ rpm -q guile-devel
guile-devel-1.6.7-2
[ian@attic4 ~]$ rpm -qi guile-devel
Name           : guile-devel                      Relocations: (not relocatable)
Version        : 1.6.7                      Vendor: Red Hat, Inc.
Release       : 2                          Build Date: Wed 02 Mar 2005 11:04:14 AM EST
Install Date: Thu 08 Sep 2005 08:35:45 AM EDT Build Host: porky.build.redhat.com
Group         : Development/Libraries      Source RPM: guile-1.6.7-2.src.rpm
Size          : 1635366                     License: GPL
Signature     : DSA/SHA1, Fri 20 May 2005 01:25:07 PM EDT, Key ID b44269d04f2a6fd2
Packager      : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary       : Libraries and header files for the GUILE extensibility library.
Description   :
The guile-devel package includes the libraries, header files, etc.,
that you will need to develop applications that are linked with the
GUILE extensibility library.

You need to install the guile-devel package if you want to develop
applications that will be linked to GUILE. You also need to install
the guile package.
```

RPM packages and files in them

You will often want to know what is in a package or what package a particular file came from. To list the files in the `guile-devel` package, use the `-ql` option as shown in Listing 42. There are many files in this package, so we've only shown part of the output.

Listing 42. Displaying information about guile-devel.

```
[ian@attic4 ~]$ rpm -ql guile-devel
/usr/bin/guile-config
/usr/bin/guile-snarf
/usr/include/guile
/usr/include/guile/gh.h
/usr/include/guile/srfi
/usr/include/guile/srfi/srfi-13.h
/usr/include/guile/srfi/srfi-14.h
/usr/include/guile/srfi/srfi-4.h
/usr/include/libguile
/usr/include/libguile.h
...
```

You can restrict the files listed to just configuration files by adding the `-c` option to

your query. Similarly, the `-d` option limits the display to just documentation files.

Querying package files

The above package query commands query the RPM database for installed packages. If you've just downloaded a package and want the same kind of information you can get this using the `-p` option (for *package file*) on your query along with specifying the package **file** name (as used for installing the package). Listing 43 repeats the queries of Listing 41 against the package file instead of the RPM database.

Listing 42. Displaying guile-devel package file information.

```
[ian@attic4 ~]$ rpm -qp guile-devel-1.6.7-2.i386.rpm
guile-devel-1.6.7-2
[ian@attic4 ~]$ rpm -qpi guile-devel-1.6.7-2.i386.rpm
Name       : guile-devel                Relocations: (not relocatable)
Version    : 1.6.7                    Vendor: Red Hat, Inc.
Release    : 2                       Build Date: Wed 02 Mar 2005 11:04:14 AM EST
Install Date: (not installed)        Build Host: porky.build.redhat.com
Group      : Development/Libraries   Source RPM: guile-1.6.7-2.src.rpm
Size       : 1635366                 License: GPL
Signature  : DSA/SHA1, Fri 20 May 2005 01:25:07 PM EDT, Key ID b44269d04f2a6fd2
Packager   : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary    : Libraries and header files for the GUILE extensibility library.
Description:
The guile-devel package includes the libraries, header files, etc.,
that you will need to develop applications that are linked with the
GUILE extensibility library.

You need to install the guile-devel package if you want to develop
applications that will be linked to GUILE. You also need to install
the guile package.
```

Querying all installed packages

The `-a` option applies your query to all installed packages. This can generate a lot of output, so you will usually use it in conjunction with one or more filters, such as `sort` to sort the listing, `more` or `less` to page through it, `wc` to obtain package or file counts, or `grep` to search for packages if you aren't sure of the name. Listing 43 shows the following queries:

1. A sorted list of all packages on the system
2. A count of all packages on the system.
3. A count of all files in all packages on the system.
4. A count of all documentation files installed with RPMs.
5. A search for all packages with "guile" (case-insensitive) as part of their name.

Listing 43. Queries against all packages.

```
[ian@attic4 ~]$ rpm -qa | sort | more
4Suite-1.0-8.bl
a2ps-4.13b-46
acl-2.2.23-8
acpid-1.0.4-1
alchemist-1.0.36-1
alsa-lib-1.0.9rf-2.FC4
alsa-utils-1.0.9rf-2.FC4
...
[ian@attic4 ~]$ rpm -qa | wc -l
874
[ian@attic4 ~]$ rpm -qa1 | wc -l
195681
[ian@attic4 ~]$ rpm -qa1d | wc -l
31881
[ian@attic4 ~]$ rpm -qa | grep -i "guile"
guile-devel-1.6.7-2
guile-1.6.7-2
```

Using `rpm -qa` can ease the administration of multiple systems. If you redirect the sorted output to a file on one machine, then do the same on the other machine, you can use the `diff` program to find differences.

Finding the owner for a file

Given that you can list all packages and all files in a package, you now have all the information you need to find which package owns a file. However, the `rpm` command provides a `-f` option to help you locate the package that owns a file. In our Dr Geo example in the [Make and install programs](#) section, we needed `guile-config`. Now that we have the `guile-devel` package installed, this is an executable file on our path. Listing 44 shows how to use the `which` command to get the full path to the `guile-config` command, and a handy tip for using this output as input to the `rpm -qf` command. Note that the tick marks surrounding `which guile-config` are back-ticks. Another way of using this in bash is to use `$(which guile-config)`

Listing 44. Which package owns guile-config.

```
[ian@attic4 ~]$ which guile-config
/usr/bin/guile-config
[ian@attic4 ~]$ rpm -qf `which guile-config`
guile-devel-1.6.7-2
[ian@attic4 ~]$ rpm -qf $(which guile-config)
guile-devel-1.6.7-2
```

RPM dependencies

We saw earlier that we could not erase the `guile` package because of *dependencies*. In addition to files, an RPM package may contain arbitrary *capabilities* which other packages may depend on. In our example many other packages required

capabilities provided by the guile package. And we could not have installed guile-devel if we had not already installed guile on our system. And once guile-devel is installed, it provides yet another reason why guile cannot be removed.

Usually, this all works out fine. If you need to install several packages at once, some of which may depend on others, simply give the whole list to your `rpm -Uvh` command and it will analyze the dependencies and perform the installs in the right order.

Besides trying to erase an installed package and getting an error message, the `rpm` command provides an option to interrogate installed packages or package files to find out what capabilities they depend on or *require*. This is the `--requires` option, which may be abbreviated to `-R`. Listing 45 shows the capabilities required by guile-config. Add the `-p` option and use the full RPM file name if you want to query the package file instead of the RPM database.

Listing 45. What does guile-config require.

```
[ian@attic4 ~]$ rpm -qR guile-devel
/bin/sh
/usr/bin/guile
guile = 5:1.6.7
rpmllib(CompressedFileNames) <= 3.0.4-1
rpmllib(PayloadFilesHavePrefix) <= 4.0-1
```

In addition, to find out what capabilities a package requires, you can find out what packages require a given capability (as is done when you attempt to remove a package). Listing 46 illustrates this for two of the capabilities required by guile-devel. Since this output may include duplicates, we've also shown you how to filter the output through `sort` and `uniq` to get each requiring package listed only once.

Listing 46. What needs /usr/bin/guile and guile.

```
[ian@attic4 ~]$ rpm -q --whatrequires /usr/bin/guile guile
guile-devel-1.6.7-2
g-wrap-1.3.4-8
guile-devel-1.6.7-2
[ian@attic4 ~]$ rpm -q --whatrequires /usr/bin/guile guile | sort|uniq
guile-devel-1.6.7-2
g-wrap-1.3.4-8
```

RPM package integrity.

To ensure the integrity of RPM packages, they will include a MD5 or SHA1 digest and they may be digitally signed. Packages that are digitally signed need a public key for verification. To check the integrity of an RPM package file use the `--checksig` (abbreviated to `-K`) option of `rpm`. You will usually find it useful to add

the `-v` option for more verbose output.

Listing 47. Checking the integrity of the guile-devel package file.

```
[ian@attic4 ~]$ rpm --checksig guile-devel-1.6.7-2.i386.rpm
guile-devel-1.6.7-2.i386.rpm: (sha1) dsa sha1 md5 gpg OK
[ian@attic4 ~]$ rpm -Kv guile-devel-1.6.7-2.i386.rpm
guile-devel-1.6.7-2.i386.rpm:
  Header V3 DSA signature: OK, key ID 4f2a6fd2
  Header SHA1 digest: OK (b2c61217cef4a72a8d2eddb8db3e140e4e7607a1)
  MD5 digest: OK (cf47354f2513ba0c2d513329c52bf72a)
  V3 DSA signature: OK, key ID 4f2a6fd2
```

You may get an output line like:

```
V3 DSA signature: NOKEY, key ID 16a61572
```

This means that you have a signed package, but you do not have the needed public key in your RPM database. Note that earlier versions of RPM may present the verification differently.

If a package is signed and you want to verify it against a signature, then you will need to locate the appropriate signature file and import it into your RPM database. You should first download the key and then check its fingerprint before importing it using the `rpm --import` command. For more information see the rpm man pages. You will also find more information on signed binaries at www.rpm.org.

Verifying an installed package

Similarly to checking the integrity of an rpm, you can also check the integrity of your installed files using `rpm -V`. This step makes sure that the files haven't been modified since they were installed from the rpm. As shown in Listing 49 there is no output from this command if the package is still good.

Listing 48. Verifying the installed guile-devel package.

```
[ian@attic4 ~]$ rpm -V guile-devel
```

Let us become root and remove `/usr/bin/guile-config` altogether and replace `/usr/bin/guile-snarf` with a copy of `/bin/bash` and try again. The results are shown in Listing 49.

Listing 48. Tampering with the guile-devel package.

```
[root@attic4 ~]# rm /usr/bin/guile-config
rm: remove regular file `/usr/bin/guile-config'? y
```

```
[root@attic4 ~]# cp /bin/bash /usr/bin/guile-snarf
cp: overwrite `/usr/bin/guile-snarf'? y
[root@attic4 ~]# rpm -V guile-devel
missing      /usr/bin/guile-config
S.5....T    /usr/bin/guile-snarf
```

This output shows us that the `/usr/bin/guile-snarf` fails MD5 sum, file size, and mtime tests. You can repair this by erasing the package and reinstalling, or forcibly installing as we saw earlier. Expect an error message if you remove the package, since one of its files is now missing.

Configuring RPM

RPM rarely needs configuring. In older versions of rpm, you could change things in `/etc/rpmrc` to control run-time operation. In recent versions, the file has been moved to `/usr/lib/rpm/rpmrc` where it is automatically replaced when the rpm package is upgraded, thus losing any changes you may have made. If any per-system configuration is required it may still be added to `/etc/rpmrc`, while per-user configuration should `.rpmrc.in` in the user's home directory. You can find documentation on the format of these files in the book *Maximum RPM* (see [Resources](#)).

If you'd like to view the rpmrc configuration, there is even an rpm command option for that. Use the command:

```
rpm rpm --showrc
```

Repositories and other tools

By now you might be wondering where all these rpm packages come from, how you find them and how you manage updates to your system. If you have an RPM-based distribution, your distributor will likely maintain a *repository* of packages. Your distributor may also provide a tool for installing packages from the repository or updating your entire system. These tools may be graphical or command line or both. Some examples include:

- YaST (SUSE)
- up2date (Red Hat)
- yum - Yellow Dog Updater Modified (Fedora and others)
- Mandrake Software Management (Mandriva)

Usually these tools will handle multiple package updates in an automatic or semi-automatic fashion. They may also provide capabilities to display contents of repositories or search for packages. Consult the documentation for your distribution

for more details.

If you can't find a particular RPM through your system tools, another good resource for locating RPMs is the Rpmfind.Net server (see [Resources](#)).

The next tutorial in this series covers Topic 103 -- GNU and Unix Commands.

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for each level of the Linux Professional Institute's Linux system administration certification.
- [The Linux documentation project](#) is the home of lots of useful Linux documentation, including the [Linux Partition HOWTO](#) on planning and creating partitions on IDE and SCSI hard drives.
- Learn more about FHS at the [Filesystem Hierarchy Standard home page](#).
- Learn about GRUB legacy and GRUB 2 at the [GNU GRUB home page](#). GNU GRUB is a multiboot boot loader derived from GRUB, GRand Unified Bootloader.
- The [Multiboot Specification](#) for an interface allows any complying boot loader to load any complying operating system.
- The [Debian home page](#) is your starting point for information about the Debian distribution.
- Refer to the "[Debian installation guide](#)" for more information on installing Debian.
- Display and search Debian package lists at [Debian packages](#).
- Learn about the Debian package management utility, APT, in the "[APT HOWTO](#)."
- Get the [Alien package converter](#).
- Check out the "[Guide to creating your own Debian packages](#)."
- Visit [Ubuntu](#), Linux for Human Beings.
- At [www.rpm.org](#) find current information on the RPM software packaging tool and pointers to more information on RPM.
- The book *Maximum RPM* offers a comprehensive and systematic treatment of all aspects of RPM. It's available in both hard and soft copy formats.
- Read the [RPM HOWTO](#) to learn about the Red Hat Package Manager, RPM.
- Search for RPMs for your distribution at [Rpmfind.Net](#) and [RPM Search](#).
- At the [LSB Home](#), learn about the Linux Standard Base (LSB), a Free Standards Group (FSG) project to develop a standard binary operating environment.

- Find certification materials in book form in [LPI Linux Certification in a Nutshell](#) (O'Reilly, 2001).
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- Download the [System rescue CD-Rom](#), one of many tools available online to help you recover a system after a crash.
- Browse [SourceForge.net](#), a large repository of open source projects for many platforms.
- [Dr. Geo, interactive geometry](#) is a source project used as an example in this tutorial.
- Go to [Info-ZIP](#) to get Zip and Unzip programs for Linux and other platforms.
- The [Sphere Eversion Program](#) shows you how to turn a sphere inside out without tearing or creasing it. The counterintuitive result behind this program is described in a video called "*Outside In*".
- Get [Debian packages](#) from the Debian home page.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

LPI exam 101 prep: GNU and UNIX commands

Junior Level Administration (LPIC-1) topic 103

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)

Senior Programmer

IBM

15 Nov 2005

GUIs are fine, but to unlock the real power of Linux, there's no substitute for the command line. Study for the LPI® 101 exam and learn about streams and filters, file and process management, regular expressions, and the vi editor. In this third of five tutorials, Ian Shields introduces you to the Linux® command line and several GNU and UNIX commands. By the end of this tutorial, you will be comfortable using commands on a Linux system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 101, the five topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 101: Tutorials and topics

LPI exam 101 topic	developerWorks tutorial	Tutorial summary
Topic 101	LPI exam 101 prep (topic 101): Hardware and architecture	Learn to configure your system hardware with Linux. By the end of this tutorial, you will know how Linux configures the hardware found on a modern PC and where to look if you have problems.
Topic 102	LPI exam 101 prep: Linux installation and package management	Get an introduction to Linux installation and package management. By the end of this tutorial, you will know how Linux uses disk partitions, how Linux boots, and how to install and manage software packages.
Topic 103	LPI exam 101 prep: GNU and UNIX commands	(This tutorial). Get an introduction to common GNU and UNIX commands. By the end of this tutorial, you will know how to use commands in the bash shell, including how to use text processing commands and filters, how to search files and directories, and how to manage processes.
Topic 104	LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard	Learn how to create filesystems on disk partitions, as well as how to make them accessible to users, manage file ownership and user quotas, and repair filesystems as needed. Also learn about hard and symbolic links, and how to locate files in your filesystem and where files should be placed.
Topic 110	LPI exam 110 prep: The X Window system	Learn how to install and maintain the X Window System, including configuring and installing an X font server. Also, learn how to set up and customize a display manager, and customize a system-wide desktop environment and window manager, including window manager menus and desktop panel menus.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "GNU and UNIX commands", the third of five tutorials designed to prepare you for LPI exam 101. In this tutorial, you learn how to use GNU and UNIX commands.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. GNU and UNIX commands: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
1.103.1 Work on the command line	Weight 5	Interact with shells and commands using the command line. This objective includes typing valid commands and command sequences, defining, referencing and exporting environment variables, using command history and editing facilities, invoking commands in the path and outside the path, using command substitution, applying commands recursively through a directory tree and using man to find out about commands.
1.103.2 Process text streams using filters	Weight 6	Apply filters to text streams. This objective includes sending text files and output streams through text utility

		filters to modify the output, using standard UNIX commands found in the GNU textutils package.
1.103.3 Perform basic file management	Weight 3	Use the basic UNIX commands to copy, move, and remove files and directories. Tasks include advanced file management operations such as copying multiple files recursively, removing directories recursively, and moving files that meet a wildcard pattern. This objective includes using simple and advanced wildcard specifications to refer to files, as well as using find to locate and act on files based on type, size, or time.
1.103.4 Use streams, pipes, and redirects	Weight 5	Redirect streams and connect them in order to efficiently process textual data. Tasks include redirecting standard input, standard output, and standard error, piping the output of one command to the input of another command, using the output of one command as arguments to another command, and sending output to both stdout and a file.
1.103.5 Create, monitor, and kill processes	Weight 5	Manage processes. This objective includes knowing how to run jobs in the foreground and background, bringing a job from the background to the foreground and vice versa, starting a process that will run without being connected to a terminal, and signaling a program to continue running after logout. Tasks also include monitoring active processes, selecting and sorting processes for display, sending signals to processes, killing processes, and identifying and killing X applications that did not terminate after the X session closed.

1.103.6 Modify process execution priorities	Weight 3	Manage process execution priorities. Tasks include running a program with higher or lower priority, determining the priority of a process, and changing the priority of a running process.
1.103.7 Search text files using regular expressions	Weight 3	Manipulate files and text data using regular expressions. This objective includes creating simple regular expressions containing several notational elements. It also includes using regular expression tools to perform searches through a filesystem or file content.
1.103.8 Perform basic file editing operations using vi	Weight 1	Edit text files using vi. This objective includes vi navigation, basic vi nodes, inserting, editing, deleting, copying, and finding text.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial. Sometimes different versions of a program will format output differently, so your results may not always look exactly like the listings and figures in this tutorial.

Section 2. Using the command line

This section covers material for topic 1.103.1 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn about the following topics:

- Interacting with shells and commands using the command line
- Valid commands and command sequences
- Defining, referencing, and exporting environment variables
- Command history and editing facilities

- Invoking commands in the path and outside the path
- Using command substitution
- Applying commands recursively through a directory tree
- Using man (manual) pages to find out about commands

This section gives you a brief introduction to some of the major features of the bash shell, with an emphasis on the features that are important for certification. But the shell is a very rich environment, and we encourage you to explore it further. Many excellent books are devoted to UNIX and Linux shells and the bash shell in particular.

The bash shell

The *bash* shell is one of several shells available for Linux. It is also called the *Bourne-again shell*, after Stephen Bourne, the creator of an earlier shell (*/bin/sh*). Bash is substantially compatible with *sh*, but provides many improvements in both function and programming capability. It incorporates features from the Korn shell (*ksh*) and C shell (*csh*), and is intended to be a POSIX-compliant shell.

Before we delve deeper into bash, recall that a *shell* is a program that accepts and executes commands. It also supports programming constructs allowing complex commands to be built from smaller parts. These complex commands, or *scripts*, can be saved as files to become new commands in their own right. Indeed, many commands on a typical Linux system **are** scripts.

Shells have some *builtin* commands, such as *cd*, *break*, and *exec*. Other commands are *external*.

Shells also use three standard I/O *streams*:

- *stdin* is the *standard input stream*, which provides input to commands.
- *stdout* is the *standard output stream*, which displays output from commands.
- *stderr* is the *standard error stream*, which displays error output from commands.

Input streams provide input to programs, usually from terminal keystrokes. Output streams print text characters, usually to the terminal. The terminal was originally an ASCII typewriter or display terminal, but it is now more often a window on a graphical desktop. More detail on how to redirect these standard I/O streams is

covered in a later section of this tutorial, [Streams, pipes, and redirects](#). The rest of this section focuses on redirection at a high level.

For the rest of this tutorial, we will assume you know how to get a shell prompt. If you don't, the developerWorks article "[Basic tasks for new Linux developers](#) shows you how to do this and more.

If you are using a Linux system without a graphical desktop, or if you open a terminal window on a graphical desktop, you will be greeted by a prompt, perhaps like one shown in Listing 1.

Listing 1. Some typical user prompts

```
[db2inst1@echidna db2inst1]$  
ian@lyrebird:~>  
$
```

If you log in as the root user (or superuser), your prompt may look like one shown in Listing 2.

Listing 2. Superuser, or root, prompt examples

```
[root@echidna root]#  
lyrebird:~ #  
#
```

The root user has considerable power, so use it with caution. When you have root privileges, most prompts include a trailing pound sign (#). Ordinary user privileges are usually delineated by a different character, commonly a dollar sign (\$). Your actual prompt may look different than the examples in this tutorial. Your prompt may include your user name, hostname, current directory, date or time that the prompt was printed, and so on.

Conventions in this tutorial

These developerWorks tutorials for the LPI 101 and 102 exams include code examples that are cut and pasted from real Linux systems using the default prompts for those systems. Our root prompts have a trailing #, so you can distinguish them from ordinary user prompts, which have a trailing \$. This convention is consistent with many books on the subject. Note the prompt carefully in any examples.

Commands and sequences

So now that you have a prompt, let's look at what you can do with it. The shell's

main function is to interpret your commands so you can interact with your Linux system. On Linux (and UNIX) systems, commands have a *command name*, and then *options* and *parameters*. Some commands have neither options nor parameters, and some have options but no parameters, while others have no options but do have parameters.

If a line contains a # character, then all remaining characters on the line are ignored. So a # character may indicate a comment as well as a root prompt. Which it is should be evident from the context.

Echo

The `echo` command prints (or echos) its arguments to the terminal as shown in Listing 3.

Listing 3. Echo examples

```
[ian@echidna ian]$ echo Word
Word
[ian@echidna ian]$ echo A phrase
A phrase
[ian@echidna ian]$ echo Where      are      my      spaces?
Where are my spaces?
[ian@echidna ian]$ echo "Here      are      my      spaces." # plus comment
Here      are      my      spaces.
```

In third example of Listing 3, all the extra spaces were compressed down to single spaces in the output. To avoid this, you need to *quote* strings, using either double quotes (") or single quotes ('). Bash uses *white space*, such as blanks, tabs, and new line characters, to separate your input line into *tokens*, which are then passed to your command. Quoting strings preserves additional white space and makes the whole string a single token. In the example above, each token after the command name is a parameter, so we have respectively 1, 2, 4, and 1 parameters.

The `echo` command has a couple of options. Normally `echo` will append a trailing new line character to the output. Use the `-n` option to suppress this. Use the `-e` option to enable certain backslash escaped characters to have special meaning. Some of these are shown in Table 3.

Table 3. Echo and escaped characters

Escape sequence	Function
\a	Alert (bell)
\b	Backspace
\c	Suppress trailing newline (same function as -n option)
\f	Form feed (clear the screen on a

	video display)
\n	New line
\r	Carriage return
\t	Horizontal tab

Escapes and line continuation

There is a small problem with using backslashes in bash. When the backslash character (\) is not quoted, it serves as an escape to signal bash itself to preserve the literal meaning of the following character. This is necessary for special shell metacharacters, which we'll cover in a moment. There is one exception to this rule: a backslash followed by a newline causes bash to swallow both characters and treat the sequence as a line continuation request. This can be handy to break long lines, particularly in shell scripts.

For the sequences described above to be properly handled by the `echo` command or one of the many other commands that use similar escaped control characters, you must include the escape sequences in quotes, or as part of a quoted string, unless you use a second backslash to have the shell preserve one for the command. Listing 4 shows some examples of the various uses of \.

Listing 4. More echo examples

```
[ian@echidna ian]$ echo -n No new line
No new line[ian@echidna ian]$ echo -e "No new line\c"
No new line[ian@echidna ian]$ echo "A line with a typed
> return"
A line with a typed
return
[ian@echidna ian]$ echo -e "A line with an escaped\nreturn"
A line with an escaped
return
[ian@echidna ian]$ echo "A line with an escaped\nreturn but no -e option"
A line with an escaped\nreturn but no -e option
[ian@echidna ian]$ echo -e Doubly escaped\\n\\tmetacharacters
Doubly escaped
    metacharacters
[ian@echidna ian]$ echo Backslash \
> followed by newline \
> serves as line continuation.
Backslash followed by newline serves as line continuation.
```

Note that bash displays a special prompt (>) when you type a line with unmatched quotes. Your input string continues onto a second line and includes the new line character.

Bash shell metacharacters and control operators

Bash has several *metacharacters*, which, when not quoted, also serve to separate input into words. Besides a blank, these are '|', '&', ';', '(', ')', '<', and '>'. We will

discuss some of these in more detail in other sections of this tutorial. For now, note that if you want to include a metacharacter as part of your text, it must be either quoted or escaped using a backslash (\) as shown in Listing 4.

The new line and certain metacharacters or pairs of metacharacters also serve as *control operators*. These are '|', '&&', '&', ';', '::', '|' '(', and ')'. Some of these control operators allow you to create *sequences* or *lists* of commands.

The simplest command sequence is just two commands separated by a semicolon (;). Each command is executed in sequence. In any programmable environment, commands return an indication of success or failure; Linux commands usually return a zero value for success and a non-zero in the event of failure. You can introduce some conditional processing into your list using the && and || control operators. If you separate two commands with the control operator && then the second is executed if and only if the first returns an exit value of zero. If you separate the commands with ||, then the second one is executed only if the first one returns a non-zero exit code. Listing 5 shows some command sequences using the echo command. These aren't very exciting since echo returns 0, but you will see more examples later when we have a few more commands to use.

Listing 5. Command sequences

```
[ian@echidna ian]$ echo line 1;echo line 2; echo line 3
line 1
line 2
line 3
[ian@echidna ian]$ echo line 1&&echo line 2&&echo line 3
line 1
line 2
line 3
[ian@echidna ian]$ echo line 1||echo line 2; echo line 3
line 1
line 3
```

Exit

You can terminate a shell using the `exit` command. You may optionally give an exit code as a parameter. If you are running your shell in a terminal window on a graphical desktop, your window will close. Similarly, if you have connected to a remote system using `ssh` or `telnet` (for example), your connection will end. In the bash shell, you can also hold the **Ctrl** key and press the **d** key to exit.

Let's look at another control operator. If you enclose a command or a command list in parentheses, then the command or sequence is executed in a subshell, so the `exit` command exits the subshell rather than exiting the shell you are working in. Listing 6 shows a simple example in conjunction with && and ||.

Listing 6. Subshells and sequences

```
[ian@echidna ian]$ (echo In subshell; exit 0) && echo OK || echo Bad exit
In subshell
OK
[ian@echidna ian]$ (echo In subshell; exit 4) && echo OK || echo Bad exit
In subshell
Bad exit
```

Stay tuned for more command sequences to come in this tutorial.

Environment variables

When you are running in a bash shell, many things constitute your *environment*, such as the form of your prompt, your home directory, your working directory, the name of your shell, files that you have opened, functions that you have defined, and so on. Your environment includes many *variables* that may have been set by bash or by you. The bash shell also allows you to have *shell variables*, which you may *export* to your environment for use by other processes running in the shell or by other shells that you may spawn from the current shell.

Both environment variables and shell variables have a *name*. You reference the value of a variable by prefixing its name with '\$'. Some of the common bash environment variables that you will encounter are shown in Table 4.

Table 4. Some common bash environment variables	
Name	Function
USER	The name of the logged-in user
UID	The numeric user id of the logged-in user
HOME	The user's home directory
PWD	The current working directory
SHELL	The name of the shell
\$	The process id (or <i>PID</i>) of the running bash shell (or other) process
PPID	The process id of the process that started this process (that is, the id of the parent process)
?	The exit code of the last command

Listing 7 shows what you might see in some of these common bash variables.

Listing 7. Environment and shell variables

```
[ian@echidna ian]$ echo $USER $UID
ian 500
[ian@echidna ian]$ echo $SHELL $HOME $PWD
/bin/bash /home/ian /home/ian
[ian@echidna ian]$ (exit 0);echo $?;(exit 4);echo $?
0
4
[ian@echidna ian]$ echo $$ $PPID
30576 30575
```

Not using bash?

The bash shell is the default shell on most Linux distributions. If you are not running under the bash shell, you may want to consider one of the following ways to practice with the bash shell.

- Use the `chsh -s /bin/bash` command to change your default shell. The default will take over next time you log in.
- Use the `su - $USER -s /bin/bash` command to create another process as a child of your current shell. The new process will be a login shell using bash.
- Create an id with a default of a bash shell to use for LPI exam preparation.

You may create or *set* a shell variable by typing a name followed immediately by an equal sign (=). Variables are case sensitive, so `var1` and `VAR1` are different variables. By convention, variables, particularly exported variables, are upper case, but this is not a requirement. Technically, `$$` and `$?` are shell *parameters* rather than variables. They may only be referenced; you cannot assign a value to them.

When you create a shell variable, you will often want to *export* it to the environment so it will be available to other processes that you start from this shell. Variables that you export are **not** available to a parent shell. You use the `export` command to export a variable name. As a shortcut in bash, you can assign a value and export a variable in one step.

To illustrate assignment and exporting, let's run the bash command while in the bash shell and then run the Korn shell (ksh) from the new bash shell. We will use the `ps` command to display information about the command that is running. We'll learn more about `ps` when we learn about [process status](#) later in this tutorial.

Listing 8. More environment and shell variables

```
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
PID  PPID  CMD
```

```

30576 30575 -bash
[ian@echidna ian]$ bash
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16353 30576 bash
[ian@echidna ian]$ VAR1=var1
[ian@echidna ian]$ VAR2=var2
[ian@echidna ian]$ export VAR2
[ian@echidna ian]$ export VAR3=var3
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3
var1 var2 var3
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $SHELL
var1 var2 var3 /bin/bash
[ian@echidna ian]$ ksh
$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16448 16353 ksh
$ export VAR4=var4
$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
var2 var3 var4 /bin/bash
$ exit
$ [ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
var1 var2 var3 /bin/bash
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16353 30576 bash
[ian@echidna ian]$ exit
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
30576 30575 -bash
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
/bin/bash

```

Notes:

1. At the start of this sequence, the bash shell had PID 30576 .
2. The second bash shell has PID 16353, and its parent is PID 30576, the original bash shell.
3. We created VAR1, VAR2, and VAR3 in the second bash shell, but only exported VAR2 and VAR3.
4. In the Korn shell, we created VAR4. The `echo` command displayed values only for VAR2, VAR3, and VAR4, confirming that VAR1 was not exported. Were you surprised to see that the value of the SHELL variable had not changed, even though the prompt had changed? You cannot always rely on SHELL to tell you what shell you are running under, but the `ps` command does tell you the actual command. Note that `ps` puts a hyphen (-) in front of the first bash shell to indicate that this is the *login shell*.
5. Back in the second bash shell, we can see VAR1, VAR2, and VAR3.
6. And finally, when we return to the original shell, none of our new variables still exist.

The earlier discussion of quoting mentioned that you could use either single or double quotes. There is an important difference between them. The shell expands shell variables that are between double quotes (`"`), but expansion is not done when single quotes (`'`) are used. In the previous example, we started another shell within our shell and we got a new process id. Using the `-c` option, you can pass a command to the other shell, which will execute the command and return. If you pass a quoted string as a command, your outer shell will strip the quotes and pass the string. If double quotes are used, variable expansion occurs **before** the string is passed, so the results may not be as you expect. The shell and command will run in another process so they will have another PID. Listing 9 illustrates these concepts. The PID of the top-level bash shell is highlighted.

Listing 9. Quoting and shell variables

```
[ian@echidna ian]$ echo "$SHELL" '$SHELL' "$$" '$$'
/bin/bash $SHELL 19244 $$
[ian@echidna ian]$ bash -c "echo Expand in parent $$ $PPID"
Expand in parent 19244 19243
[ian@echidna ian]$ bash -c 'echo Expand in child $$ $PPID'
Expand in child 19297 19244
```

So far, all our variable references have terminated with white space, so it has been clear where the variable name ends. In fact, variable names may be composed only of letters, numbers or the underscore character. The shell knows that a variable name ends where another character is found. Sometimes you need to use variables in expressions where the meaning is ambiguous. In such cases, you can use curly braces to delineate a variable name as shown in Listing 10.

Listing 10. Using curly braces with variable names

```
[ian@echidna ian]$ echo "-$HOME/abc-"
-/home/ian/abc-
[ian@echidna ian]$ echo "-$HOME_abc-"
--
[ian@echidna ian]$ echo "-${HOME}_abc-"
-/home/ian_abc-
```

Env

The `env` command without any options or parameters displays the current environment variables. You can also use it to execute a command in a custom environment. The `-i` (or just `-`) option clears the current environment before running the command, while the `-u` option unsets environment variables that you do not wish to pass.

Listing 11 shows partial output of the `env` command without any parameters and

then three examples of invoking different shells without the parent environment. Look carefully at these before we discuss them.

Listing 11. The env command

```
[ian@echidna ian]$ env
HOSTNAME=echidna
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=9.27.89.137 4339 22
SSH_TTY=/dev/pts/2
USER=ian
...
_=/bin/env
OLDPWD=/usr/src
[ian@echidna ian]$ env -i bash -c 'echo $SHELL; env'
/bin/bash
PWD=/home/ian
SHLVL=1
_=/bin/env
[ian@echidna ian]$ env -i ksh -c 'echo $SHELL; env'

_=/bin/env
PATH=/bin:/usr/bin
[ian@echidna ian]$ env -i tcsh -c 'echo $SHELL; env'
SHELL: Undefined variable.
```

Notice that bash has set the SHELL variable, but not exported it to the environment, although there are three other variables that bash has created in the environment. In the ksh example, we have two environment variables, but our attempt to echo the value of the SHELL variable gives a blank line. Finally, tcsh has not created any environment variables and produces an error at our attempt to reference the value of SHELL.

Unset and set

Listing 11 showed different behavior in how shells handle variables and environments. While this tutorial focuses on bash, it is good to know that not all shells behave the same way. Furthermore, shells behave differently according to whether they are a *login shell* or not. For now, we will just say that a login shell is the shell you get when you log in to a system; you can start other shells to behave as login shells if you wish. The three shells started above using `env -i` were not login shells. Try passing the `-l` option to the shell command itself to see what differences you would get with a login shell.

So let's consider our attempt to display the SHELL variable's value in these non-login shells:

1. When bash started, it set the SHELL variable, but it did not automatically export it to the environment.
2. When ksh started, it did not set the SHELL variable. However, referencing

an undefined environment variable is equivalent to referencing one with an empty value.

3. When `tcsh` started, it did not set the `SHELL` variable. In this case, the default behavior is different than `ksh` (and `bash`) in that an error is reported when we attempt to use a variable.

You can use the `unset` command to unset a variable and remove it from the shell variable list. If the variable was exported to the environment, this will also remove it from the environment. You can use the `set` command to control many facets of the way `bash` (or other shells) work. `Set` is a shell builtin, so the various options are shell specific. In `bash`, the `-u` option causes `bash` to report an error with undefined variables rather than treat them as defined but empty. You can turn on the various options to `set` with a `-` and turn them off with a `+`. You can display currently set options using `echo $-`.

Listing 12. Unset and set

```
[ian@echidna ian]$ echo $-
himBH
[ian@echidna ian]$ echo $VAR1

[ian@echidna ian]$ set -u;echo $-
himuBH
[ian@echidna ian]$ echo $VAR1
bash: VAR1: unbound variable
[ian@echidna ian]$ VAR1=v1
[ian@echidna ian]$ VAR1=v1;echo $VAR1
v1
[ian@echidna ian]$ unset VAR1;echo $VAR1
bash: VAR1: unbound variable
[ian@echidna ian]$ set +u;echo $VAR1;echo $-
himBH
```

If you use the `set` command without any options, it displays all your shell variables and their values (if any). There is also another command, `declare`, which you can use to create, export, and display values of shell variables. You can explore the many remaining `set` options and the `declare` command using the man pages. We will discuss [man pages](#) later in this section.

Exec

One final command to cover in this section is `exec`. You can use the `exec` command to run another program that **replaces** the current shell. Listing 13 starts a child `bash` shell and then uses `exec` to replace it with a Korn shell. Upon exit from the Korn shell, you are back at the original `bash` shell (PID 22985, in this example).

Listing 13. Using exec

```
[ian@echidna ian]$ echo $$
22985
[ian@echidna ian]$ bash
[ian@echidna ian]$ echo $$
25063
[ian@echidna ian]$ exec ksh
$ echo $$
25063
$ exit
[ian@echidna ian]$ echo $$
22985
```

Command history

If you are typing in commands as you read, you may notice that you often use a command many times, either exactly the same, or with slight changes. The good news is that the bash shell can maintain a *history* of your commands. By default, history is on. You can turn it off using the command `set +o history` and turn it back on using `set -o history`. An environment variable called `HISTSIZE` tells bash how many history lines to keep. A number of other settings control how history works and is managed. See the bash man pages for full details.

Some of the commands that you can use with the history facility are:

history

Displays the entire history

history *N*

Displays the last *N* lines of your history

history -d *N*

Deletes line *N* from your history; you might do this if the line contains a password, for example

!!

Your most recent command

!*N*

The *N*th history command

!*-N*

The command that is *N* commands back in the history (!-1 is equivalent to !!)

!#

The current command you are typing

!*string*

The most recent command that starts with *string*

!*string*?

The most recent command that contains *string*

You can also use a colon (:) followed by certain values to access or modify part or a command from your history. Listing 14 illustrates some of the history capabilities.

Listing 14. Manipulating history

```
[ian@echidna ian]$ echo $$
22985
[ian@echidna ian]$ env -i bash -c 'echo $$'
1542
[ian@echidna ian]$ !!
env -i bash -c 'echo $$'
1555
[ian@echidna ian]$ !ec
echo $$
22985
[ian@echidna ian]$ !en:s/$$/ $PPID/
env -i bash -c 'echo $PPID'
22985
[ian@echidna ian]$ history 6
1097 echo $$
1098 env -i bash -c 'echo $$'
1099 env -i bash -c 'echo $$'
1100 echo $$
1101 env -i bash -c 'echo $PPID'
1102 history 6
[ian@echidna ian]$ history -d1100
```

The commands in Listing 14 do the following:

1. Echo the current shell's PID
2. Run an echo command in a new shell and echo that shell's PID
3. Rerun the last command
4. Rerun the last command starting with 'ec'; this reruns the first command in this example
5. Rerun the last command starting with 'en', but substitute '\$PPID' for '\$\$', so the parent PID is displayed instead
6. Display the last 6 commands of the history
7. Delete history entry 1100, the last echo command

You can also edit the history interactively. The bash shell uses the readline library to manage command editing and history. By default, the keys and key combinations used to move through the history or edit lines are similar to those used in the GNU

Emacs editor. Emacs keystroke combinations are usually expressed as **C-x** or **M-x**, where **x** is a regular key and **C** and **M** are the *Control* and *Meta* keys, respectively. On a typical PC system, the **Ctrl** key serves as the Emacs Control key, and the **Alt** key serves as the Meta key. Table 5 summarizes some of the history editing functions available. Besides the key combinations shown in Table 5, cursor movement keys such as the right, left, up, and down arrows, and the Home and End keys are usually set to work in a logical way. Additional functions as well as how to customize these options using a readline init file (usually `inputrc` in your home directory) can be found in the man pages.

Command	Common PC key	Description
C-f	Right arrow	Move one space to the right
C-b	Left arrow	Move one space to the left
M-f	Alt-f	Move to beginning of next word; GUI environments usually take this key combination to open the File menu of the window
M-b	Alt-b	Move to beginning of previous word
C-a	Home	Move to beginning of line
C-e	End	Move to end of line
Backspace	Backspace	Delete the character preceding the cursor
C-d	Del	Delete the character under the cursor (Del and Backspace functions may be configured with opposite meanings)
C-k	Ctrl-k	Delete (kill) to end of line and save removed text for later use
M-d	Alt-d	Delete (kill) to end of word and save removed text for later use
C-y	Ctrl-y	Yank back text removed with a kill command

If you prefer to manipulate the history using a vi-like editing mode, use the command `set -o vi` to switch to vi mode. Switch back to emacs mode using `set -o emacs`. When you retrieve a command in vi mode, you are initially in vi's insert mode. More details on the vi editor are in the section [File editing with vi](#).

Paths

Some bash commands are builtin, while others are external. Let's now look at external commands and how to run them, and how to tell if a command is internal.

Where does the shell find commands?

External commands are just files in your file system. The later section [Basic file management](#) of this tutorial and the tutorial for Topic 104 cover more details. On Linux and UNIX systems, all files are accessed as part of a single large tree that is rooted at /. In our examples so far, our current directory has been the user's home directory. Non-root users usually have a home directory within the /home directory, such as /home/ian, in my case. Root's home is usually /root. If you type a command name, then bash looks for that command on your *path*, which is a colon-separated list of directories in the PATH environment variable.

If you want to know what command will be executed if you type a particular string, use the `which` or `type` command. Listing 15 shows my default path along with the locations of several commands.

Listing 15. Finding command locations

```
[ian@echidna ian]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/ian/bin
[ian@echidna ian]$ which bash env zip xclock echo set ls
alias ls='ls --color=tty'
/bin/ls
/bin/bash
/bin/env
/usr/bin/zip
/usr/X11R6/bin/xclock
/bin/echo
/usr/bin/which: no set in (/usr/local/bin:/bin:/usr/bin:/usr/X11R6/b
in:/home/ian/bin)
[ian@echidna ian]$ type bash env zip xclock echo set ls
bash is /bin/bash
env is /bin/env
zip is /usr/bin/zip
xclock is /usr/X11R6/bin/xclock
echo is a shell builtin
set is a shell builtin
ls is aliased to `ls --color=tty`
```

Note that the directories in the path all end in /bin. This is a common convention, but not a requirement. The `which` command reported that the `ls` command is an *alias* and that the `set` command could not be found. In this case, we interpret that to mean that it does not exist or that it is a builtin. The `type` command reports that the `ls` command is an *alias*, but it identifies the `set` command as a shell builtin. It also reports that there is a builtin `echo` command as well as the one in /bin that was found by `which`. The two commands also produce output in different orders.

We saw that the `ls` command, used for listing directory contents, is an *alias*. Aliases are a handy way to configure some commands to use different sets of defaults or to provide an alternate name for a command. In our example, the `--color=ttty` option causes directory listings to be color coded according to the type of file or directory. Try running `dircolors --print-database` to see how the color codings are controlled and which colors are used for what kind of file.

Each of these commands has additional options. Depending on your need, you may use either command. I tend to use `which` when I am reasonably sure I'll find an executable and I just need its full path specification. I find that `type` gives me more precise information, which I sometimes need in a shell script.

Running other commands

We saw in Listing 15 that executable files have a full path starting with `/`, the root directory. For example, the `xclock` program is really `/usr/X11R6/bin/xclock`, a file located in the `/usr/X11R6/bin` directory. If a command is **not** in your `PATH` specification, you may still run it by specifying a path as well as a command name. There are two types of paths that you can use:

- *Absolute* paths are those starting with `/`, such as those we saw in Listing 15 (`/bin/bash`, `/bin/env`, etc).
- *Relative* paths are relative to your *current working directory*, as reported by the `pwd` command. These commands do not start with `/`, but do contain at least one `.`.

You may use absolute paths regardless of your current working directory, but you will probably use relative paths only when a command is close to your current directory. Suppose you are developing a new version of the classic "Hello World!" program in a subdirectory of your home directory called `mytestbin`. You might use the relative path to run your command as `mytestbin/hello`. There are two special names you can use in a path; a single dot (`.`) refers to the current directory, and a pair of dots (`..`) refers to the parent of the current directory. Because your home directory is usually not on your `PATH` (and generally should not be), you will need to explicitly provide a path for any executable that you want to run from your home directory. For example, if you had a copy of your `hello` program in your home directory, you could run it using the command `./hello`. You can use both `.` and `..` as part of an absolute path, although a single `.` is not very useful in such a case. You can also use a tilde (`~`) to mean your own home directory and `~username` to mean the home directory of the user named `username`. Some examples are shown in Listing 16.

Listing 16. Absolute and relative paths

```
[ian@echidna ian]$ /bin/echo Use echo command rather than builtin
```

```

Use echo command rather than builtin
[ian@echidna ian]$ /usr/./bin/echo Include parent dir in path
Include parent dir in path
[ian@echidna ian]$ /bin/./echo Add a couple of useless path components
Add a couple of useless path components
[ian@echidna ian]$ pwd # See where we are
/home/ian
[ian@echidna ian]$ ../../bin/echo Use a relative path to echo
Use a relative path to echo
[ian@echidna ian]$ myprogs/hello # Use a relative path with no dots
-bash: myprogs/hello: No such file or directory
[ian@echidna ian]$ mytestbin/hello # Use a relative path with no dots
Hello World!
[ian@echidna ian]$ ./hello # Run program in current directory
Hello World!
[ian@echidna mytestbin]$ ~/mytestbin/hello # run hello using ~
Hello World!
[ian@echidna ian]$ ../hello # Try running hello from parent
-bash: ../hello: No such file or directory

```

Changing your working directory

Just as you can execute programs from various directories in the system, so too can you change your current working directory using the `cd` command. The argument to `cd` must be the absolute or relative path to a directory. As for commands, you can use `.`, `..`, `~`, and `~username` in paths. If you use `cd` with no parameters, the change will be to your home directory. A single hyphen (`-`) as a parameter means to change to the previous working directory. Your home directory is stored in the `HOME` environment variable, and the previous directory is stored in the `OLDPWD` variable, so `cd` alone is equivalent to `cd $HOME` and `cd -` is equivalent to `cd $OLDPWD`. Usually we say *change directory* instead of the full *change current working directory*.

As for commands, there is also an environment variable, `CDPATH`, which contains a colon-separated set of directories that should be searched (in addition to the current working directory) when resolving relative paths. If resolution used a path from `CDPATH`, then `cd` will print the full path of the resulting directory as output. Normally, a successful directory change results in no output other than a `new`, and possibly changed, prompt. Some examples are shown in Listing 17.

Listing 17. Changing directories

```

[ian@echidna home]$ cd /;pwd
/
[ian@echidna /]$ cd /usr/X11R6;pwd
/usr/X11R6
[ian@echidna X11R6]$ cd ;pwd
/home/ian
[ian@echidna ian]$ cd -;pwd
/usr/X11R6
/usr/X11R6
[ian@echidna X11R6]$ cd ~ian/..;pwd
/home
[ian@echidna home]$ cd ~;pwd
/home/ian
[ian@echidna ian]$ export CDPATH=~
[ian@echidna mytestbin]$ cd /;pwd

```

```
/
[ian@echidna ~]$ cd mytestbin
/home/ian/mytestbin
```

Applying commands recursively

Many Linux commands can be applied recursively to all files in a directory tree. For example, the `ls` command has a `-R` option to list directories recursively, and the `cp`, `mv`, `rm`, and `diff` commands all have a `-r` option to apply them recursively. The section [Basic file management](#) covers recursive application of commands in detail.

Command substitution

The bash shell has an extremely powerful capability that allows the results of one command to be used as input to another; this is called *command substitution*. This can be done by enclosing the command whose results you wish to use in backticks (```). This is still common, but a way that is easier to use with multiply nested commands is to enclose the command between `$(` and `)`.

In the previous tutorial, "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)," we saw that the `rpm` command can tell you which package a command comes from; we used the command substitution capability as a handy technique. Now you know that's what we were really doing.

Command substitution is an invaluable tool in shell scripts and is also useful on the command line. Listing 18 shows an example of how to get the absolute path of a directory from a relative path, how to find which RPM provides the `/bin/echo` command, and how (as root) to list the labels of three partitions on a hard drive. The last one uses the `seq` command to generate a sequence of integers.

Listing 18. Command substitution

```
[ian@echidna ian]$ echo '../usr/bin' dir is $(cd ../usr/bin;pwd)
../usr/bin dir is /usr/bin
[ian@echidna ian]$ which echo
/bin/echo
[ian@echidna ian]$ rpm -qf `which echo`
sh-utils-2.0.12-3
[ian@echidna ian]$ su -
Password:
[root@echidna root]# for n in $(seq 7 9); do echo p$n `e2label /dev/hda$n`;done
p7 RH73
p8 SUSE81
p9 IMAGES
```

Man pages

Our final topic in this section of the tutorial tells you how to get documentation for Linux commands through manual pages and other sources of documentation.

Manual pages and sections

The primary (and traditional) source of documentation is the *manual pages*, which you can access using the `man` command. Figure 1 illustrates the manual page for the `man` command itself. Use the command `man man` to display this information.

Figure 1. Man page for the man command

```
lan@echidna:~  
File Edit View Terminal Go Help  
1 man(1) man(1)  
2 NAME  
   man - format and display the on-line manual pages  
   manpath - determine user's search path for man pages  
3 SYNOPSIS  
   man [-acdfFhkKtW] [--path] [-m system] [-p string] [-C config_file]  
   [-M pathlist] [-P pager] [-S section_list] [section] name ...  
4 DESCRIPTION  
   man formats and displays the on-line manual pages.  If you specify sec-  
tion, man only looks in that section of the manual.  name is normally  
   the name of the manual page, which is typically the name of a command,  
   function, or file.  However, if name contains a slash (/) then man  
   interprets it as a file specification, so that you can do man ./foo.5  
   or even man /cd/foo/bar.1.gz.  
  
   See below for a description of where man looks for the manual page  
   files.  
5 OPTIONS  
   -C config_file  
       Specify the configuration file to use; the default is  
       /etc/man.config. (See man.conf(5).)  
  
   -M path  
       Specify the list of directories to search for man pages.  Sepa-  
       rate the directories with colons.  An empty list is the same as  
       not specifying -M at all.  See SEARCH PATH FOR MANUAL PAGES.  
  
   -P pager  
       Specify which pager to use.  This option overrides the MANPAGER  
       environment variable, which in turn overrides the PAGER vari-  
       able.  By default, man uses /usr/bin/less -isr.  
i
```

Figure 1 shows some typical items in man pages:

- A heading with the name of the command followed by its section number in parentheses
- The name of the command and any related commands that are described on the same man page
- A synopsis of the options and parameters applicable to the command

- A short description of the command
- Detailed information on each of the options

You might find other sections on usage, how to report bugs, author information, and a list of any related commands. For example, the man page for `man` tells us that related commands (and their manual sections) are: `apropos(1)`, `whatis(1)`, `less(1)`, `groff(1)`, and `man.conf(5)`.

There are eight common manual page sections. Manual pages are usually installed when you install a package, so if you do not have a package installed, you probably won't have a manual page for it. Similarly, some of your manual sections may be empty or nearly empty. The common manual sections, with some example contents are:

1. User commands (`env`, `ls`, `echo`, `mkdir`, `tty`)
2. System calls or kernel functions (`link`, `sethostname`, `mkdir`)
3. Library routines (`acosh`, `asctime`, `btree`, `locale`, `XML::Parser`)
4. Device related information (`isdn_audio`, `mouse`, `tty`, `zero`)
5. File format descriptions (`keymaps`, `motd`, `wvdial.conf`)
6. Games (note that many games are now graphical and have graphical help outside the man page system)
7. Miscellaneous (`arp`, `boot`, `regex`, `unix utf8`)
8. System administration (`debugfs`, `fdisk`, `fsck`, `mount`, `renice`, `rpm`)

Other sections that you might find include `9` for Linux kernel documentation, `n` for new documentation, `o` for old documentation, and `l` for local documentation.

Some entries appear in multiple sections. Our examples show `mkdir` in sections 1 and 2, and `tty` in sections 1 and 4. You can specify a particular section, for example, `man 4 tty` or `man 2 mkdir`, or you can specify the `-a` option to list all applicable manual sections.

You may have noticed in the figure that `man` has many options for you to explore on your own. For now, let's take a quick look at some of the "See also" commands related to `man`.

See also

Two important commands related to `man` are `whatis` and `apropos`. The `whatis` command searches `man` pages for the name you give and displays the name information from the appropriate manual pages. The `apropos` command does a keyword search of manual pages and lists ones containing your keyword. Listing 19 illustrates these commands.

Listing 19. `Whatis` and `apropos` examples

```
[ian@lyrebird ian]$ whatis man
man                (1) - format and display the on-line manual pages
man                (7) - macros to format man pages
man [manpath]     (1) - format and display the on-line manual pages
man.conf [man]    (5) - configuration data for man
[ian@lyrebird ian]$ whatis mkdir
mkdir              (1) - make directories
mkdir              (2) - create a directory
[ian@lyrebird ian]$ apropos mkdir
mkdir              (1) - make directories
mkdir              (2) - create a directory
mkdirhier          (1x) - makes a directory hierarchy
```

By the way, if you cannot find the manual page for `man.conf`, try running `man man.config` instead.

The `man` command pages output onto your display using a paging program. On most Linux systems, this is likely to be the `less` program. Another choice might be the older `more` program. If you wish to print the page, specify the `-t` option to format the page for printing using the `groff` or `troff` program.

The `less` pager has several commands that help you search for strings within the displayed output. Use `man less` to find out more about `/` (search forwards), `?` (search backwards), and `n` (repeat last search), among many other commands.

Other documentation sources

In addition to manual pages accessible from a command line, the Free Software Foundation has created a number of *info* files that are processed with the *info* program. These provide extensive navigation facilities including the ability to jump to other sections. Try `man info` or `info info` for more information. Not all commands are documented with *info*, so you will find yourself using both `man` and `info` if you become an *info* user.

There are also some graphical interfaces to `man` pages, such as `xman` (from the XFree86 Project) and `yelp` (the Gnome 2.0 help browser).

If you can't find help for a command, try running the command with the `--help` option. This may provide the command's help, or it may tell you how to get the help you need.

The next section looks at processing text streams with filters.

Section 3. Text streams and filters

This section covers material for topic 1.103.2 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 6.

In this section, you learn about the following topics:

- Sending text files and output streams through text utility filters to modify the output
- Using standard UNIX commands found in the GNU textutils package

Text filtering

Text *filtering* is the process of taking an input stream of text and performing some conversion on the text before sending it to an output stream. Although either the input or the output can come from a file, in the Linux and UNIX environments, filtering is most often done by constructing a *pipeline* of commands where the output from one command is *piped* or *redirected* to be used as input to the next. Pipes and redirection are covered more fully in the section on [Streams, pipes, and redirects](#), but for now, let's look at pipes and basic output redirection using the | and > operators.

Piping with |

Recall from the previous section that shells use three standard I/O *streams*:

- *stdin* is the *standard input stream*, which provides input to commands.
- *stdout* is the *standard output stream*, which displays output from commands.
- *stderr* is the *standard error stream*, which displays error output from commands.

So far in this tutorial, input has come from parameters we supply to commands, and output has been displayed on our terminal. Many text processing commands (filters) can take input either from the standard input stream or from a file. In order to use the output of a command, `command1`, as input to a filter, `command2`, you connect the commands using the pipe operator (|) as shown in Listing 20.

Listing 20. Piping output from command 1 to input of command2

```
command1 | command2
```

Either command may have options or arguments, as you will see later in this section. You can also use `|` to redirect the output of `command2` in this pipeline to yet another command, `command3`. Constructing long pipelines of commands that each have limited capability is a common Linux and UNIX way of accomplishing tasks. You will also sometimes see a hyphen (`-`) used in place of a filename as an argument to a command, meaning the input should come from `stdin` rather than a file.

Output redirection with `>`

While it is nice to be able to create a pipeline of several commands and see the output on your terminal, there are times when you want to save the output in a file. You do this with the output redirection operator (`>`).

For the rest of this section, we will be using some small files, so let's create a directory called `lpi103` and then `cd` into that directory. We will then use `>` to redirect the output of the `echo` command into a file called `text1`. This is all shown in Listing 21. Notice that the output does not display on the terminal because it has been redirected to the file.

Listing 21. Piping output from command 1 to a file

```
[ian@echidna ian]$ mkdir lpi103
[ian@echidna ian]$ cd lpi103
[ian@echidna lpi103]$ echo -e "1 apple\n2 pear\n3 banana">text1
```

Now that we have a couple of basic tools for pipelining and redirection, let's look at some of the common UNIX and Linux text processing commands and filters. This section shows you some of the basic capabilities; check the appropriate man pages to find out more about these commands.

Cat, tac, od, and split

Now that you have created the `test1` file, you might want to check what is in it. Use the `cat` (short for *catenate*) command to display the contents of a file on `stdout`. Listing 22 verifies the contents of the file created above.

Listing 22. Displaying file contents with `cat`

```
[ian@echidna lpi103]$ cat text1
1 apple
2 pear
```

```
3 banana
```

The `cat` command takes input from `stdin` if you do not specify a file name (or if you specify - as the filename). Let's use this along with output redirection to create another text file as shown in Listing 23.

Listing 23. Creating a text file with `cat`

```
[ian@echidna lpil03]$ cat>text2
9      plum
3      banana
10     apple
```

In Listing 23, `cat` keeps reading from `stdin` until end of file. Use the **Ctrl-d** (hold **Ctrl** and press **d**) combination to signal end of file. This is the same key combination to exit from the bash shell. Note also that the tab key helps line up the fruit names in a column.

Occasionally, you might want to display a file in reverse order. Naturally, there is a text filter for that too, called `tac` (reverse of `cat`). Listing 24 shows the new `text2` file as well as the old `text1` file in reverse order. Notice how the display simply concatenates the two files.

Listing 24. Reverse display with `tac`

```
[ian@echidna lpil03]$ tac text2 text1
10     apple
3      banana
9      plum
3 banana
2 pear
1 apple
```

Now, suppose you display the two text files using `cat` or `tac` and notice alignment differences. To learn what causes this, you need to look at the control characters that are in the file. Since these are acted upon in text display output rather than having some representation of the control character itself displayed, we need to *dump* the file in a format that allows you to find and interpret these special characters. The GNU text utilities include an `od` (or *Octal Dump*) command for this purpose.

There are several options to `od`, such as the `-A` option to control the radix of the file offsets and `-t` to control the form of the displayed file contents. The radix may be specified as `o`, (octal - the default), `d` (decimal), `x` (hexadecimal), or `n` (no offsets displayed). You can display output as octal, hex, decimal, floating point, ASCII with backslash escapes or named characters (`nl` for newline, `ht` for horizontal tab, etc.). Listing 25 shows some of the formats available for dumping the `text2` example file.

Listing 25. Dumping files with od

```
[ian@echidna lpil03]$ od text2
0000000 004471 066160 066565 031412 061011 067141 067141 005141
0000020 030061 060411 070160 062554 000012
0000031
[ian@echidna lpil03]$ od -A d -t c text2
0000000 9 \t p l u m \n 3 \t b a n a n a \n
0000016 1 0 \t a p p l e \n
0000025
[ian@echidna lpil03]$ od -A n -t a text2
 9 ht p l u m nl 3 ht b a n a n a nl
 1 0 ht a p p l e nl
```

Notes:

1. The `-A` option of `cat` provides an alternate way of seeing where your tabs and line endings are. See the man page for more information.
2. If you have a mainframe background, you may be interested in the hexdump utility, which is part of a different utility set. It's not covered here, so check the man pages.

Our sample files are very small, but sometimes you will have large files that you need to split into smaller pieces. For example, you might want to break a large file into CD-sized chunks so you can write it to CD for sending through the mail to someone who could create a DVD for you. The `split` command will do this in such a way that the `cat` command can be used to recreate the file easily. By default, the files resulting from the `split` command have a prefix in their name of 'x' followed by a suffix of 'aa', 'ab', 'ac', ..., 'ba', 'bb', etc. Options permit you to change these defaults. You can also control the size of the output files and whether the resulting files contain whole lines or just byte counts. Listing 26 illustrates splitting our two text files with different prefixes for the output files. We split `text1` into files containing at most two lines, and `text2` into files containing at most 18 bytes. We then use `cat` to display some of the pieces individually as well as to display a complete file using *globbing*, which is covered in the section on [wildcards and globbing](#) later in this tutorial.

Listing 26. Splitting and recombining with split and cat

```
[ian@echidna lpil03]$ split -l 2 text1
[ian@echidna lpil03]$ split -b 18 text2 y
[ian@echidna lpil03]$ cat yaa
9 plum
3 banana
10[ian@echidna lpil03]$ cat yab
apple
[ian@echidna lpil03]$ cat y*
9 plum
3 banana
10 apple
```

Note that the split file named `yab` did not finish with a newline character, so our prompt was offset after we used `cat` to display it.

Wc, head, and tail

`cat` and `tac` display the whole file. That's fine for small files like our examples, but suppose you have a large file. Well, first you might want to use the `wc` (*Word Count*) command to see how big the file is. The `wc` command displays the number of lines, words, and bytes in a file. You can also find the number of bytes by using `ls -l`. Listing 27 shows the long format directory listing for our two text files, as well as the output from `wc`.

Listing 27. Using `wc` with text files

```
[ian@echidna lpi103]$ ls -l text*
-rw-rw-r--  1 ian      ian           24 Sep 23 12:27 text1
-rw-rw-r--  1 ian      ian           25 Sep 23 13:39 text2
[ian@echidna lpi103]$ wc text*
  3   6   24 text1
  3   6   25 text2
  6  12   49 total
```

Options allow you to control the output from `wc` or to display other information such as maximum line length. See the man page for details.

Two commands allow you to display either the first part (*head*) or last part (*tail*). These commands are the `head` and `tail` commands. They can be used as filters, or they can take a file name as an argument. By default they display the first (or last) 10 lines of the file or stream. Listing 28 uses the `dmesg` command to display bootup messages, in conjunction with `wc`, `tail`, and `head` to discover that there are 177 messages, then to display the last 10 of these, and finally to display the six messages starting 15 from the end. Some lines have been truncated in this output (indicated by ...).

Listing 28. Using `wc`, `head`, and `tail` to display boot messages

```
[ian@echidna lpi103]$ dmesg | wc
 177   1164   8366
[ian@echidna lpi103]$ dmesg | tail
i810: Intel ICH2 found at IO 0x1880 and 0x1c00, MEM 0x0000 and ...
i810_audio: Audio Controller supports 6 channels.
i810_audio: Defaulting to base 2 channel mode.
i810_audio: Resetting connection 0
ac97_codec: AC97 Audio codec, id: ADS98 (Unknown)
i810_audio: AC'97 codec 0 Unable to map surround DAC's (or ...
i810_audio: setting clocking to 41319
Attached scsi CD-ROM sr0 at scsi0, channel 0, id 0, lun 0
sr0: scsi3-mmc drive: 0x/32x writer cd/rw xa/form2 cdda tray
```

```
Uniform CD-ROM driver Revision: 3.12
[ian@echidna lpil03]$ dmesg | tail -n15 | head -n 6
agpgart: Maximum main memory to use for agp memory: 941M
agpgart: Detected Intel i845 chipset
agpgart: AGP aperture is 64M @ 0xf4000000
Intel 810 + AC97 Audio, version 0.24, 13:01:43 Dec 18 2003
PCI: Setting latency timer of device 00:1f.5 to 64
i810: Intel ICH2 found at IO 0x1880 and 0x1c00, MEM 0x0000 and ...
```

Another common use of `tail` is to *follow* a file using the `-f` option, usually with a line count of 1. You might use this when you have a background process that is generating output in a file and you want to check in and see how it is doing. In this mode, `tail` will run until you cancel it (using **Ctrl-c**), displaying lines as they are written to the file.

Expand, unexpand, and tr

When we created our `text1` and `text2` files, we created `text2` with tab characters. Sometimes you may want to swap tabs for spaces or vice versa. The `expand` and `unexpand` commands do this. The `-t` option to both commands allows you to set the tab stops. A single value sets repeated tabs at that interval. Listing 29 shows how to expand the tabs in `text2` to single spaces and another whimsical sequence of `expand` and `unexpand` that unaligns the text in `text2`.

Listing 29. Using `expand` and `unexpand`

```
[ian@echidna lpil03]$ expand -t 1 text2
9 plum
3 banana
10 apple
[ian@echidna lpil03]$ expand -t8 text2|unexpand -a -t2|expand -t3
9      plum
3      banana
10     apple
```

Unfortunately, you cannot use `unexpand` to replace the spaces in `text1` with tabs as `unexpand` requires at least two spaces to convert to tabs. However, you can use the `tr` command, which translates characters in one set (*set1*) to corresponding characters in another set (*set2*). Listing 30 shows how to use `tr` to translate spaces to tabs. Since `tr` is purely a filter, you generate input for it using the `cat` command. This example also illustrates the use of `-` to signify standard input to `cat`.

Listing 30. Using `expand` and `unexpand`

```
[ian@echidna lpil03]$ cat text1 |tr ' ' '\t'|cat - text2
1      apple
2      pear
3      banana
9      plum
3      banana
```

```
10      apple
```

If you are not sure what is happening in the last two examples, try using `od` to terminate each stage of the pipeline in turn; for example,

```
cat text1 | tr ' ' '\t' | od -tc
```

Pr, nl, and fmt

The `pr` command is used to format files for printing. The default header includes the file name and file creation date and time, along with a page number and two lines of blank footer. When output is created from multiple files or the standard input stream, the current date and time are used instead of the file name and creation date. You can print files side-by-side in columns and control many aspects of formatting through options. As usual, refer to the man page for details.

The `nl` command numbers lines, which can be convenient when printing files. You can also number lines with the `-n` option of the `cat` command. Listing 31 shows how to print our `text1` file, and then how to number `text2` and print it side-by-side with `text1`.

Listing 31. Numbering and formatting for print

```
[ian@echidna lpi103]$ pr text1 | head

2005-09-23 12:27                                text1                                Page 1

1 apple
2 pear
3 banana

[ian@echidna lpi103]$ nl text2 | pr -m - text1 | head

2005-09-26 11:48                                Page 1

      1  9      plum                1 apple
      2  3      banana              2 pear
      3 10      apple                3 banana
```

Another useful command for formatting text is the `fmt` command, which formats text so it fits within margins. You can join several short lines as well as split long ones. In Listing 32, we create `text3` with a single long line of text using variants of the `!#:*` history feature to save typing our sentence four times. We also create `text4` with one word per line. Then we use `cat` to display them unformatted including a displayed '\$' character to show line endings. Finally, we use `fmt` to format them to a maximum

width of 60 characters. Again, consult the man page for details on additional options.

Listing 32. Formatting to a maximum line length

```
[ian@echidna lpil03]$ echo "This is a sentence. " !#:* !#:1-$>text3
echo "This is a sentence. " "This is a sentence. " "This is a sentenc
e. " "This is a sentence. ">text3
[ian@echidna lpil03]$ echo -e "This\nis\nanother\nsentence.">text4
[ian@echidna lpil03]$ cat -et text3 text4
This is a sentence. This is a sentence. This is a sentence. This i
s a sentence. $
This$
is$
another$
sentence.$
[ian@echidna lpil03]$ fmt -w 60 text3 text4
This is a sentence. This is a sentence. This is a
sentence. This is a sentence.
This is another sentence.
```

Sort and uniq

The `sort` command sorts the input using the collating sequence for the locale (`LC_COLLATE`) of the system. The `sort` command can also merge already sorted files and check whether a file is sorted or not.

Listing 33 illustrates using the `sort` command to sort our two text files after translating blanks to tabs in `text1`. Since the sort order is by character, you might be surprised at the results. Fortunately, the `sort` command can sort by numeric values or by character values. You can specify this choice for the whole record or for each *field*. Unless you specify a different field separator, fields are delimited by blanks or tabs. The second example in Listing 33 shows sorting the first field numerically and the second by collating sequence (alphabetically). It also illustrates the use of the `-u` option to eliminate any duplicate lines and keep only lines that are unique.

Listing 33. Character and numeric sorting

```
[ian@echidna lpil03]$ cat text1 | tr ' ' '\t' | sort - text2
10  apple
1   apple
2   pear
3   banana
3   banana
9   plum
[ian@echidna lpil03]$ cat text1|tr ' ' '\t'|sort -u -k1n -k2 - text2
1   apple
2   pear
3   banana
9   plum
10  apple
```

Notice that we still have two lines containing the fruit "apple". Another command

called `uniq` gives us additional control over the elimination of duplicate lines. The `uniq` command normally operates on sorted files, but remove **consecutive** identical lines from any file, whether sorted or not. The `uniq` command can also ignore some fields. Listing 34 sorts our two text files using the second field (fruit name) and then eliminates lines that are identical, starting at the second field (that is, we skip the first field when testing with `uniq`).

Listing 34. Using `uniq`

```
[ian@echidna lpil03]$ cat text1|tr ' ' '\t'|sort -k2 - text2|uniq -f1
10      apple
3       banana
2       pear
9       plum
```

Our sort was by collating sequence, so `uniq` gives us the "10 apple" line instead of the "1 apple". Try adding a numeric sort on key field 1 to see how to change this.

Cut, paste, and join

Now let's look at three more commands that deal with fields in textual data. These commands are particularly useful for dealing with tabular data. The first is the `cut` command, which extracts fields from text files. The default field delimiter is the tab character. Listing 35 uses `cut` to separate the two columns of `text2` and then uses a space as an output delimiter, which is an exotic way of converting the tab in each line to a space.

Listing 35. Using `cut`

```
[ian@echidna lpil03]$ cut -f1-2 --output-delimiter=' ' text2
9 plum
3 banana
10 apple
```

The `paste` command pastes lines from two or more files side-by-side, similar to the way that the `pr` command merges files using its `-m` option. Listing 36 shows the result of pasting our two text files.

Listing 36. Pasting files

```
[ian@echidna lpil03]$ paste text1 text2
1 apple 9      plum
2 pear  3      banana
3 banana 10     apple
```

These examples show simple pasting, but `paste` can paste data from one or more

files in several other ways. Consult the man page for details.

Our final field-manipulating command is `join`, which joins files based on a matching field. The files should be sorted on the join field. Since `text2` is not sorted in numeric order, we could sort it and then `join` would join the two lines that have a matching join field (3 in this case). Let's also create a new file, `text5`, by sorting `text 1` on the second field (the fruit name) and then replacing spaces with tabs. If we then sort `text2` and join that with `text 5` using the second field, we should have two matches (apple and banana). Listing 37 illustrates both these joins.

Listing 37. Joining files with join fields

```
[ian@echidna lpil03]$ sort -n text2|join -1 1 -2 1 text1 -
3 banana banana
[ian@echidna lpil03]$ sort -k2 text1|tr ' ' '\t'>text5
[ian@echidna lpil03]$ sort -k2 text2 | join -1 2 -2 2 text5 -
apple 1 10
banana 3 3
```

The field to use for the join is specified separately for each file. You could, for example, join based on field 3 from one file and field 10 from another.

Sed

Sed is the stream editor. Several developerWorks articles, as well as many books and book chapters, are available on sed (see [Resources](#)). Sed is extremely powerful, and the tasks it can accomplish are limited only by your imagination. This small introduction should whet your appetite for sed, but is not intended to be complete or extensive.

As with many of the text commands we have looked at so far, sed can work as a filter or take its input from a file. Output is to the standard output stream. Sed loads lines from the input into the *pattern space*, applies sed editing commands to the contents of the pattern space, and then writes the pattern space to standard output. Sed may combine several lines in the pattern space, and it may write to a file, write only selected output, or not write at all.

Sed uses regular expression syntax (see [Searching with regular expressions](#) later in this tutorial) to search for and replace text selectively in the pattern space as well as to control which lines of text should be operated on by sets of editing commands. A *hold buffer* provides temporary storage for text. The hold buffer may replace the pattern space, be added to the pattern space, or be exchanged with the pattern space. Sed has a limited set of commands, but these combined with regular expression syntax and the hold buffer make for some amazing capabilities. A set of sed commands is usually called a *sed script*.

Listing 38 shows three simple sed scripts. In the first one, we use the `s` (substitute) command to substitute an upper case for a lower case 'a' on each line. This example replaces only the first 'a', so in the second example, we add the 'g' (for global) flag to cause sed to change all occurrences. In the third script, we introduce the `d` (delete) command to delete a line. In our example, we use an address of 2 to indicate that only line 2 should be deleted. We separate commands using a semi-colon (;) and use the same global substitution that we used in the second script to replace 'a' with 'A'.

Listing 38. Beginning sed scripts

```
[ian@echidna lpil03]$ sed 's/a/A/' text1
1 Apple
2 peAr
3 bAnana
[ian@echidna lpil03]$ sed 's/a/A/g' text1
1 Apple
2 peAr
3 bAnAnA
[ian@echidna lpil03]$ sed '2d;$s/a/A/g' text1
1 apple
3 bAnAnA
```

In addition to operating on individual lines, sed can operate on a range of lines. The beginning and end of the range are separated by a comma (,) and may be specified as a line number, a caret (^) for beginning of file, or a dollar sign (\$) for end of file. Given an address or a range of addresses, you can group several commands between curly braces ({ and }) to have these commands operate only on lines selected by the range. Listing 39 illustrates two ways of having our global substitution applied to only the last two lines of our file. It also illustrates the use of the `-e` option to add commands to the pattern space. When using braces, we must separate commands in this way.

Listing 39. Sed addresses

```
[ian@echidna lpil03]$ sed -e '2,${' -e 's/a/A/g' -e '}' text1
1 apple
2 peAr
3 bAnAnA
[ian@echidna lpil03]$ sed -e '/pear/,/bana/{' -e 's/a/A/g' -e '}' text1
1 apple
2 peAr
3 bAnAnA
```

Sed scripts may also be stored in files. In fact, you will probably want to do this for frequently used scripts. Remember earlier we used the `tr` command to change blanks in text1 to tabs. Let's now do that with a sed script stored in a file. We will use the `echo` command to create the file. The results are shown in Listing 40.

Listing 40. A sed one-liner

```
[ian@echidna lpil03]$ echo -e "s/ /\t/g">sedtab
[ian@echidna lpil03]$ cat sedtab
s/ / /g
[ian@echidna lpil03]$ sed -f sedtab text1
1      apple
2      pear
3      banana
```

There are many handy sed one-liners such as Listing 40. See [Resources](#) for links to some.

Our final sed example uses the = command to print line numbers and then filter the resulting output through sed again to mimic the effect of the n1 command to number lines. Listing 41 uses = to print line numbers, then uses the N command to read a second input line into the pattern space, and finally removes the newline character (\n) between the two lines in the pattern space.

Listing 41. Numbering lines with sed

```
[ian@echidna lpil03]$ sed '=' text2
1
9      plum
2
3      banana
3
10     apple
[ian@echidna lpil03]$ sed '=' text2|sed 'N;s/\n//'
19     plum
23     banana
310    apple
```

Not quite what we wanted! What we would really like is to have our numbers aligned in a column with some space before the lines from the file. In Listing 42, we enter several lines of command (note the > secondary prompt). Study the example and refer to the explanation below.

Listing 42. Numbering lines with sed - round two

```
[ian@echidna lpil03]$ cat text1 text2 text1 text2>text6
[ian@echidna lpil03]$ ht=$(echo -en "\t")
[ian@echidna lpil03]$ sed '=' text6|sed "N
> s/^/ /
> s/^\.*\(\.....\)\n/\1$ht/"
1 1 apple
2 2 pear
3 3 banana
4 9 plum
5 3 banana
6 10 apple
7 1 apple
8 2 pear
9 3 banana
10 9 plum
11 3 banana
```

Here are the steps we took:

1. We first used `cat` to create a twelve-line file from two copies each of our `text1` and `text2` files. There's no fun in formatting numbers in columns if we don't have differing numbers of digits.
2. The bash shell uses the tab key for command completion, so it can be handy to have a captive tab character that you can use when you want a real tab. We use the `echo` command to accomplish this and save the character in the shell variable `'ht'`.
3. We create a stream containing line numbers followed by data lines as we did before and filter it through a second copy of `sed`.
4. We read a second line into the pattern space
5. We prefix our line number at the start of the pattern space (denoted by `^`) with six blanks.
6. We then substitute all of the line up to the newline with the last six characters before the newline plus a tab character. Note that the left part of the `'s'` command uses `\(` and `\)` to mark the characters that we want to use in the right part. In the right part, we reference the first such marked set (and only such set in this example) as `\1`. Note that our command is contained between double quotes (`"`) so that substitution will occur for `$ht`.

Recent (version 4) versions of `sed` contain documentation in `info` format and include many excellent examples. These are not included in the older version 3.02. GNU `sed` will accept `sed --version` to display the version.

Section 4. Basic file management

This section covers material for topic 1.103.3 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn about the following topics:

- Listing directory contents
- Copying, moving, and removing files and directories

- Manipulating multiple files and directories recursively
- Using wildcard patterns for manipulating files
- Using the `find` command to locate and act on files based on type, size, or time

Listing directories

As we said earlier in our discussion of paths in the section on [using the command line](#), all files on Linux and UNIX® systems are accessed as part of a single large tree-structured filesystem that is rooted at `/`.

Listing directory entries

If you worked through the previous section with us, you will have created a directory, `lpi103`, in your home directory. File and directory names are either *absolute* which means they begin with a `/` or they are *relative* to the *current working directory*, which means they do not begin with a `/`. The absolute path to a file or directory consists of a `/` followed by series of 0 or more directory names, each followed by another `/` and then a final name. Given a file or directory name that is relative to the current working directory, simply concatenate the absolute name of the working directory, a `/`, and the relative name. For example, the directory, `lpi103`, that we created in the last section was created in my home directory, `/home/ian`, so its full, or absolute, path is `/home/ian/lpi103`. Listing 43 shows three different ways to use the `ls` command to list the files in this directory.

Listing 43. Listing directory entries

```
[ian@echidna lpi103]$ pwd
/home/ian/lpi103
[ian@echidna lpi103]$ echo $PWD
/home/ian/lpi103
[ian@echidna lpi103]$ ls
sedtab  text2  text4  text6  xab  yab
text1   text3  text5  xaa    yaa
[ian@echidna lpi103]$ ls "$PWD"
sedtab  text2  text4  text6  xab  yab
text1   text3  text5  xaa    yaa
[ian@echidna lpi103]$ ls /home/ian/lpi103
sedtab  text2  text4  text6  xab  yab
text1   text3  text5  xaa    yaa
```

As you can see, you can give a directory name as a parameter to the `ls` command and it will list the contents of that directory.

Listing details

On a storage device, a file or directory is contained in a collection of *blocks*. Information about a file is contained in an *inode* which records information such as the owner, when the file was last accessed, how large it is, whether it is a directory or not, and who can read or write it. The inode number is also known as the *file serial number* and is unique within a particular filesystem. We can use the `-l` (or `--format=long`) option to display some of the information stored in the inode.

By default, the `ls` command does not list special files, those whose names start with a dot (`.`). Every directory other than the root directory has two special entries at least, the directory itself (`.`) and the parent directory (`..`). The root directory does not have a parent directory.

Listing 44 uses the `-l` and `-a` options to display a long format listing of all files including the `.` and `..` directory entries.

Listing 44. Long directory listing

```
[ian@echidna lpil03]$ ls -al
total 56
drwxrwxr-x   2 ian   ian   4096 Sep 30 15:01 .
drwxr-xr-x  94 ian   ian  8192 Sep 27 12:57 ..
-rw-rw-r--   1 ian   ian    8 Sep 26 15:24 sedtab
-rw-rw-r--   1 ian   ian   24 Sep 23 12:27 text1
-rw-rw-r--   1 ian   ian   25 Sep 23 13:39 text2
-rw-rw-r--   1 ian   ian   84 Sep 25 17:47 text3
-rw-rw-r--   1 ian   ian   26 Sep 25 22:28 text4
-rw-rw-r--   1 ian   ian   24 Sep 26 12:46 text5
-rw-rw-r--   1 ian   ian   98 Sep 26 16:09 text6
-rw-rw-r--   1 ian   ian   15 Sep 23 14:11 xaa
-rw-rw-r--   1 ian   ian    9 Sep 23 14:11 xab
-rw-rw-r--   1 ian   ian   18 Sep 23 14:11 yaa
-rw-rw-r--   1 ian   ian    7 Sep 23 14:11 yab
```

In Listing 44, the first line shows the total number of disk blocks (56) used by the listed files. The remaining fields tell you about the file.

- The first field (`drwxrwxr-x` or `-rw-rw-r--` in this case) tells us whether the file is a directory (`d`) or a regular file (`-`). You may also see symbolic links (`l`) which we shall learn about later or other values for some special files (such as files in the `/dev` filesystem). The type is followed by three sets of permissions (such as `rw` or `r--`) for the owner, the members of the owner's group and everyone. The three values respectively indicate whether the user, group or everyone has read (`r`), write (`w`) or execute (`x`) permission. Other uses such as `setuid` are covered later.
- The next field is a number which tells us the number of *hard links* to the file. We said that the inode contains information about the file. The file's directory entry contains a hard link (or pointer) to the inode for the file, so every entry listed should have at least one hard link. Directory entries have an additional one for the `.` entry and one for each subdirectory's `..` entry. So we can see from the above listing that my home directory has

quite a few subdirectories

- The next two fields are the file's owner and the owner's primary group. Some systems, such as the Red Hat system, default to providing a separate group for each user. On other systems, all users may be in one or perhaps a few groups.
- The next field contains the length of the file.
- The penultimate field contains the timestamp of the last modification.
- And the final field contains the name of the file or directory.

The `-i` option of the `ls` command will display the inode numbers for you. We will use this later in this tutorial and also when we discuss hard and symbolic links in the tutorial for Topic 104.

Multiple files

You can also specify multiple parameters to the `ls` command, where each name is either that of a file or of a directory. If a name is that of a directory, then the `ls` command lists the contents of the directory rather than information about the entry itself. In our example, suppose we wanted information about the `lpi103` directory entry itself as it is listed in the parent directory. the command `ls -l ../lpi103` would give us a listing like the previous example. Listing 45 shows how to use `ls -ld` and also how to list entries for multiple files or directories.

Listing 45. Using `ls -d`

```
[ian@echidna lpi103]$ ls -ld ../lpi103 sedtab xaa
drwxrwxr-x  2 ian   ian      4096 Oct  2 18:49 ../lpi103
-rw-rw-r--  1 ian   ian       8 Sep 26 15:24 sedtab
-rw-rw-r--  1 ian   ian      15 Sep 23 14:11 xaa
```

Note that the modification time for `lpi103` is different to that in the previous listing. Also, as in the previous listing, it is different to the timestamps of any of the files in the directory. Is this what you would expect? Not normally. However, in developing this tutorial, I created some extra examples and then deleted them, so the directory time stamps reflect that fact. We will talk more about file times later in this section when we discuss [finding files](#).

Sorting the output

By default, `ls` lists files alphabetically. There are a number of options for sorting the output. For example, `ls -t` will sort by modification time (newest to oldest) while `ls -lS` will produce a long listing sorted by size (largest to smallest). Adding `-r` will reverse the sort order. for example, use `ls -lrt` to produce a long listing sorted from oldest to newest. Consult the man page for other ways you can list files and

directories.

Copy, move and delete

We have now learned some ways to create files, but suppose we want to make copies of files, rename files, move them around the filesystem hierarchy, or even delete them. We use three short commands for these purposes.

cp

is used to make a copy of one or more files. You **must** give at least two names, one (or more *source* names and one *target* name. If you specify two file names, then the first is copied to the second. Either source or target file name may include a path specification. If you specify a directory as the last name, then you may specify multiple files to be copied into it. All files will be copied from their existing locations and the copies will have the same file names as the original files. Note that there is no default assumption of the target being the current directory as in DOS and Windows operating systems.

mv

is used to *move* or *rename* one or more files or directories. In general, the names you may use follow the same rules as for copying with `cp`; you can rename a single file or move a set of files into a new directory. Since the name is only a directory entry that links to an inode, it should be no surprise that the inode number does not change **unless** the file is moved to another filesystem, in which case moving it behaves more like a copy followed by deleting the original.

rm

is used to *remove* one or more files. We will see how to remove directories shortly.

Listing 46 illustrates the use of `cp` and `mv` to make some backup copies of our text files. We also use `ls -i` to show inodes for some of our files.

1. We first make a copy of our `text1` file as `text1.bkp`.
2. We then decide to create a backup subdirectory using the `mkdir` command
3. We make a second backup copy of text 1, this time in the backup directory and show that all three files have different inodes.
4. We then move our `text1.bkp` to the backup directory and after that rename it to be more consistent with the second backup. While we could have done this with a single command we use two here for illustration.

5. We check the inodes again and confirm that text1.bkp with inode 2129019 is no longer in our lpi103 directory, but that the inode is that of text1.bkp.1 in the backup directory.

Listing 46. Copying and moving files

```
[ian@echidna lpi103]$ cp text1 text1.bkp
[ian@echidna lpi103]$ mkdir backup
[ian@echidna lpi103]$ cp text1 backup/text1.bkp.2
[ian@echidna lpi103]$ ls -i text1 text1.bkp backup
2128984 text1 2129019 text1.bkp

backup:
1564497 text1.bkp.2
[ian@echidna lpi103]$ mv text1.bkp backup
[ian@echidna lpi103]$ mv backup/text1.bkp backup/text1.bkp.1
[ian@echidna lpi103]$ ls -i text1 text1.bkp backup
ls: text1.bkp: No such file or directory
2128984 text1

backup:
2129019 text1.bkp.1 1564497 text1.bkp.2
```

Normally, the `cp` command will copy a file over an existing copy, if the existing file is writable. On the other hand, the `mv` will not move or rename a file if the target exists. There are several useful options relevant to this behavior of `cp` and `mv`.

-f or --force

will cause `cp` to attempt to remove an existing target file even if it is not writable.

-i or --interactive

will ask for confirmation before attempting to replace an existing file

-b or --backup

will make a backup of any files that would be replaced.

As usual, consult the man pages for full details on these and other options for copying and moving.

In Listing 47, we illustrate copying with backup and then file deletion.

Listing 47. Backup copies and file deletion

```
[ian@echidna lpi103]$ cp text2 backup
[ian@echidna lpi103]$ cp --backup=t text2 backup
[ian@echidna lpi103]$ ls backup
text1.bkp.1 text1.bkp.2 text2 text2.~1~
[ian@echidna lpi103]$ rm backup/text2 backup/text2.~1~
[ian@echidna lpi103]$ ls backup
text1.bkp.1 text1.bkp.2
```

Note that the `rm` command also accepts the `-i` (interactive) and `-f` (force options. Once you remove a file using `rm`, the filesystem no longer has access to it. Some systems default to setting an alias `alias rm='rm -i'` for the root user to help prevent inadvertent file deletion. This is also a good idea if you are nervous about what you might accidentally delete.

Before we leave this discussion, it should be noted that the `cp` command defaults to creating a new timestamp for the new file or files. The owner and group are also set to the owner and group of the user doing the copying. The `-p` option may be used to preserve selected attributes. Note that the root user may be the only user who can preserve ownership. See the man page for details.

Mkdir and rmdir

We have already seen how to create a directory with `mkdir`. Now we will look further at `mkdir` and introduce its analog for removing directories, `rmdir`.

Mkdir

Suppose we are in our `lpi103` directory and we wish to create subdirectories `dir1` and `dir2`. The `mkdir`, like the commands we have just been reviewing, will handle multiple directory creation requests in one pass as shown in Listing 48.

Listing 48. Creating multiple directories

```
[ian@echidna lpi103]$ mkdir dir1 dir2
```

Note that there is no output on successful completion, although you could use `echo $?` to confirm that the exit code is really 0.

If, instead, you wanted to create a nested subdirectory, such as `d1/d2/d3`, this would fail because the `d1` and `d2` directories do not exist. Fortunately, `mkdir` has a `-p` option which allows it to create any required parent directories. Listing 49 illustrates this.

Listing 49. Creating parent directories

```
[ian@echidna lpi103]$ mkdir d1/d2/d3
mkdir: cannot create directory `d1/d2/d3': No such file or directory
[ian@echidna lpi103]$ echo $?
1
[ian@echidna lpi103]$ mkdir -p d1/d2/d3
[ian@echidna lpi103]$ echo $?
0
```

Rmdir

Removing directories using the `rmdir` command is the opposite of creating them. Again, there is a `-p` option to remove parents as well. You can only remove a directory with `rmdir` if it empty as there is no option to force removal. We'll see another way to accomplish that particular trick when we look at [recursive manipulation](#). Once you learn that you will probably seldom use `rmdir` on the command line, but it is good to know about it nevertheless.

To illustrate directory removal, we copied our `text1` file into the directory `d1/d2` so that it is no longer empty. We then used `rmdir` to remove all the directories we just created with `mkdir`. As you can see, `d1` and `d2` were not removed because `d2` was not empty. The other directories were removed. Once we remove the copy of `text1` from `d2`, we can remove `d1` and `d2` with a single invocation of `rmdir -p`.

Listing 50. Removing directories

```
[ian@echidna lpi103]$ cp text1 d1/d2
[ian@echidna lpi103]$ rmdir -p d1/d2/d3 dir1 dir2
rmdir: `d1/d2': Directory not empty
[ian@echidna lpi103]$ ls . d1/d2
.:
backup  sedtab  text2   text4   text6   xab    yab
d1      text1   text3   text5   xaa     yaa

d1/d2:
text1
[ian@echidna lpi103]$ rm d1/d2/text1
[ian@echidna lpi103]$ rmdir -p d1/d2
```

Recursive manipulation

In the remaining few parts of this section we will look at various operations for handling multiple files and for recursively manipulating part of a directory tree.

Recursive listing

The `ls` command has a `-R` (note upper case 'R') option for listing a directory and all its subdirectories. The recursive option applies only to directory names; it will not find all the files called, say 'text1' in a directory tree. You may use other options that we have seen already along with `-R`. A recursive listing of our `lpi103` directory, including inode numbers, is shown in Listing 51.

Listing 51. Recursive directory listing

```
[ian@echidna lpi103]$ ls -iR ~/lpi103
/home/ian/lpi103:
1564496 backup  2128985 text2   2128982 text5   2128987 xab
2128991 sedtab  2128990 text3   2128995 text6   2128988 yaa
2128984 text1   2128992 text4   2128986 xaa     2128989 yab

/home/ian/lpi103/backup:
```

```
2129019 text1.bkp.1 1564497 text1.bkp.2
```

Recursive copy

You can use the `-r` (or `-R` or `--recursive`) option to cause the `cp` command to descend into source directories and copy the contents recursively. To prevent an infinite recursion, the source directory itself may not be copied. Listing 52 shows how to copy everything in our `lpi103` directory to a `copy1` subdirectory. We use `ls -R` to show the resulting directory tree.

Listing 52. Recursive copy

```
[ian@echidna lpi103]$ cp -pR . copy1
cp: cannot copy a directory, '.', into itself, `copy1'
[ian@echidna lpi103]$ ls -R
.:
backup  sedtab  text2   text4   text6   xab    yab
copy1   text1   text3   text5   xaa     yaa

./backup:
text1.bkp.1  text1.bkp.2

./copy1:
backup  text1  text3  text5  xaa  yaa
sedtab  text2  text4  text6  xab  yab

./copy1/backup:
text1.bkp.1  text1.bkp.2
```

Recursive deletion

We mentioned earlier that `rmdir` only removes empty directories. We can use the `-r` (or `-R` or `--recursive`) option to cause the `rm` command to remove both files **and** directories as shown in Listing 53 where we remove the `copy1` directory that we just created, along with its contents, including the `backup` subdirectory and its contents.

Listing 53. Recursive deletion

```
[ian@echidna lpi103]$ rm -r copy1
[ian@echidna lpi103]$ ls -R
.:
backup  text1  text3  text5  xaa  yaa
sedtab  text2  text4  text6  xab  yab

./backup:
text1.bkp.1  text1.bkp.2
```

If you have files that are not writable by you, you may need to add the `-f` option to force removal. This is often done by the root user when cleaning up, but be warned that you can lose valuable data if you are not careful.

Wildcards and globbing

Often, you may need to perform a single operation on many filesystem objects, without operating on the entire tree as we have just done with recursive operations. For example, you might want to find the modification times of all the text files we created in lpi103, without listing the split files. Although this is easy with our small directory, it is much harder in a large filesystem.

To solve this problem, use the wildcard support that is built in to the bash shell. This support, also called "globbing" (because it was originally implemented as a program called `/etc/glob`), lets you specify multiple files using wildcard pattern.

A string containing any of the characters '?', '*' or '[', is a *wildcard pattern*. Globbing is the process by which the shell (or possibly another program) expands these patterns into a list of pathnames matching the pattern. The matching is done as follows.

?

matches any single character.

*

matches any string, including an empty string.

[

introduces a *character class*. A character class is a non-empty string, terminated by a ']'. A match means matching any single character enclosed by the brackets. There are a few special considerations.

- The '*' and '?' characters match themselves. If you use these in filenames, you will need to be really careful about appropriate quoting or escaping.
- Since the string must be non-empty and terminated by ']', you must put ']' **first** in the string if you want to match it.
- The '-' character between two others represents a range which includes the two other characters and all between in the collating sequence. For example, [0-9a-fA-F] represents any upper or lower case hexadecimal digit. You can match a '-' by putting it either first or last within a range.
- The '!' character specified as the first character of a range complements the range so that it matches any character except the remaining characters. For example [!0-9] means any character except the digits 0 through 9. A '!' in any position other than the first matches itself. Remember that '!' is also used with the shell history function, so you need to be careful to properly escape it.

Globbing is applied separately to each component of a path name. You cannot

match a '/', nor include one in a range. You can use it anywhere that you might specify multiple file or directory names, for example in the `ls`, `cp`, `mv` or `rm` commands. In Listing 54, we first create a couple of oddly named files and then use the `ls` and `rm` commands with wildcard patterns.

Listing 54. Wildcard pattern examples

```
[ian@echidna lpil03]$ echo odd1>'text[*?!1]'
```

```
[ian@echidna lpil03]$ echo odd2>'text[2*?!]'
```

```
[ian@echidna lpil03]$ ls
```

```
backup  text1      text2      text3  text5  xaa  yaa
```

```
sedtab  text[*?!1]  text[2*?!]  text4  text6  xab  yab
```

```
[ian@echidna lpil03]$ ls text[2-4]
```

```
text2  text3  text4
```

```
[ian@echidna lpil03]$ ls text[!2-4]
```

```
text1  text5  text6
```

```
[ian@echidna lpil03]$ ls text*[2-4]*
```

```
text2  text[2*?!]  text3  text4
```

```
[ian@echidna lpil03]$ ls text*[!2-4]* # Surprise!
```

```
text1  text[*?!1]  text[2*?!]  text5  text6
```

```
[ian@echidna lpil03]$ ls text*[!2-4] # More surprise!
```

```
text1  text[*?!1]  text[2*?!]  text5  text6
```

```
[ian@echidna lpil03]$ echo text*>text10
```

```
[ian@echidna lpil03]$ ls *\!*
```

```
text[*?!1]  text[2*?!]
```

```
[ian@echidna lpil03]$ ls *[x\!]*
```

```
text1      text2      text3  text5  xaa
```

```
text[*?!1]  text[2*?!]  text4  text6  xab
```

```
[ian@echidna lpil03]$ ls *[y\!]*
```

```
text[*?!1]  text[2*?!]  yaa  yab
```

```
[ian@echidna lpil03]$ ls tex?[[]*
```

```
text[*?!1]  text[2*?!]
```

```
[ian@echidna lpil03]$ rm tex?[[]*
```

```
[ian@echidna lpil03]$ ls *b*
```

```
sedtab  xab  yab
```



```
backup:
```

```
text1.bkp.1  text1.bkp.2
```

```
[ian@echidna lpil03]$ ls backup/*2
```

```
backup/text1.bkp.2
```

```
[ian@echidna lpil03]$ ls -d .*
```

```
.  ..
```

Notes:

1. Complementation in conjunction with '*' can lead to some surprises. The pattern '*[!2-4]' matches the longest part of a name that does not have 2, 3, or 4 following it, which is matched by **both** `text[*?!1]` and `text[2*?!]`. So now both surprises should be clear.
2. As with earlier examples of `ls`, if pattern expansion results in a name that is a directory name and the `-d` option is not specified, then the contents of that directory will be listed (as in our example above for the pattern '*b*').
3. If a filename starts with a period (.) then that character must be matched explicitly. Notice that only the last `ls` command listed the two special directory entries (. and ..).

Remember that any wildcard characters in a command are liable to be expanded by the shell which may lead to unexpected results. Furthermore, If you specify a pattern that does not match any filesystem objects then POSIX requires that the original pattern string be passed to the command. We illustrate this in Listing 55. Some earlier implementations passed a null list to the command, so you may run into old scripts that give unusual behavior. We illustrate these points in Listing 55.

Listing 55. Wildcard pattern surprises

```
[ian@echidna lpi103]$ echo text*
text1 text2 text3 text4 text5 text6
[ian@echidna lpi103]$ echo "text*"
text*
[ian@echidna lpi103]$ echo text[[\!?]z??
text[[!?]z??
```

For more information on globbing, look at `man 7 glob`. You will need the section number as there is also `glob` information in section 3. The best way to understand all the various shell interactions is by practice, so try these wildcards out whenever you have a chance. Remember to try `ls` to check your wildcard pattern before committing it to whatever `cp`, `mv` or worse, `rm` might unexpectedly do for you.

Touching files

We will now look at the `touch` command which can update file access and modification times or create empty files. In the next part we will see how to use this information to find files and directories. We will use the `lpi103` directory that we created earlier in this tutorial. We will also look

touch

The `touch` command with no options takes one or more filenames as parameters and updates the **modification** time of the files. This is the same timestamp normally displayed with a long directory listing. In Listing 56, we use our old friend `echo` to create a small file called `f1`, and then use a long directory listing to display the modification time (or *mtime*). In this case, it happens also to be the time the file was created. We then use the `sleep` command to wait for 60 seconds and run `ls` again. Note that the timestamp for the file has changed by a minute.

Listing 56. Updating modification time with touch

```
[ian@echidna lpi103]$ echo xxx>f1; ls -l f1; sleep 60; touch f1; ls -l f1
-rw-rw-r--  1 ian      ian          4 Nov  4 15:57 f1
-rw-rw-r--  1 ian      ian          4 Nov  4 15:58 f1
```

If you specify a filename for a file that does not exist, then `touch` will normally

create an empty file for you, unless you specify the `-c` or `--no-create` option. Listing 57 illustrates both these commands. Note that only `f2` is created.

Listing 57. Creating empty files with touch

```
[ian@echidna lpil03]$ touch f2; touch -c f3; ls -l f*
-rw-rw-r-- 1 ian ian 4 Nov 4 15:58 f1
-rw-rw-r-- 1 ian ian 0 Nov 4 16:12 f2
```

The `touch` command can also set a file's mtime to a specific date and time using either the `-d` or `-t` options. The `-d` is very flexible in the date and time formats that it will accept, while the `-t` option needs at least an MMDDhhmm time with optional year and seconds values. Listing 58 shows some examples.

Listing 58. Setting mtime with touch

```
[ian@echidna lpil03]$ touch -t 200511051510.59 f3
[ian@echidna lpil03]$ touch -d 11am f4
[ian@echidna lpil03]$ touch -d "last fortnight" f5
[ian@echidna lpil03]$ touch -d "yesterday 6am" f6
[ian@echidna lpil03]$ touch -d "2 days ago 12:00" f7
[ian@echidna lpil03]$ touch -d "tomorrow 02:00" f8
[ian@echidna lpil03]$ touch -d "5 Nov" f9
[ian@echidna lpil03]$ ls -lrt f*
-rw-rw-r-- 1 ian ian 0 Oct 24 12:32 f5
-rw-rw-r-- 1 ian ian 4 Nov 4 15:58 f1
-rw-rw-r-- 1 ian ian 0 Nov 4 16:12 f2
-rw-rw-r-- 1 ian ian 0 Nov 5 00:00 f9
-rw-rw-r-- 1 ian ian 0 Nov 5 12:00 f7
-rw-rw-r-- 1 ian ian 0 Nov 5 15:10 f3
-rw-rw-r-- 1 ian ian 0 Nov 6 06:00 f6
-rw-rw-r-- 1 ian ian 0 Nov 7 11:00 f4
-rw-rw-r-- 1 ian ian 0 Nov 8 2005 f8
```

If you're not sure what date a date expression might resolve to, you can use the `date` command to find out. It also accepts the `-d` option and can resolve the same kind of date formats that `touch` can.

You can use the `-r` (or `--reference`) option along with a *reference filename* to indicate that `touch` (or `date`) should use the timestamp of an existing file. Listing 59 shows some examples.

Listing 59. Timestamps from reference files

```
[ian@echidna lpil03]$ date
Mon Nov 7 12:40:11 EST 2005
[ian@echidna lpil03]$ date -r f1
Fri Nov 4 15:58:27 EST 2005
[ian@echidna lpil03]$ touch -r f1 f1a
[ian@echidna lpil03]$ ls -l f1*
-rw-rw-r-- 1 ian ian 4 Nov 4 15:58 f1
-rw-rw-r-- 1 ian ian 0 Nov 4 15:58 f1a
```

A Linux system records both a file *modification* time and a file *access* time. Both timestamps are set to the same value when a file is created, and both are reset when it is modified. If a file is accessed at all, then the access time is updated, even if the file is not modified. For our last example with `touch`, we will look at file access times. The `-a` (or `--time=atime`, `--time=access` or `--time=use`) option specify that the access time should be updated. Listing 60 uses the `cat` command to access the `f1` file and display its contents. We then use `ls -l` and `ls -lu` to display the modification and access times respectively for `f1` and `f1a`, which we created using `f1` as a reference file. We then reset the access time of `f1` to that of `f1a` using `touch -a`.

Listing 60. Access time and modification time

```
[ian@echidna lpi103]$ cat f1
xxx
[ian@echidna lpi103]$ ls -lu f1*
-rw-rw-r-- 1 ian   ian           4 Nov  7 14:13 f1
-rw-rw-r-- 1 ian   ian           0 Nov  4 15:58 f1a
[ian@echidna lpi103]$ ls -l f1*
-rw-rw-r-- 1 ian   ian           4 Nov  4 15:58 f1
-rw-rw-r-- 1 ian   ian           0 Nov  4 15:58 f1a
[ian@echidna lpi103]$ cat f1
xxx
[ian@echidna lpi103]$ ls -l f1*
-rw-rw-r-- 1 ian   ian           4 Nov  4 15:58 f1
-rw-rw-r-- 1 ian   ian           0 Nov  4 15:58 f1a
[ian@echidna lpi103]$ ls -lu f1*
-rw-rw-r-- 1 ian   ian           4 Nov  7 14:13 f1
-rw-rw-r-- 1 ian   ian           0 Nov  4 15:58 f1a
[ian@echidna lpi103]$ touch -a -r f1a f1
[ian@echidna lpi103]$ ls -lu f1*
-rw-rw-r-- 1 ian   ian           4 Nov  4 15:58 f1
-rw-rw-r-- 1 ian   ian           0 Nov  4 15:58 f1a
```

For more complete information on the many allowable time and date specifications see the `man` or `info` pages for the `touch` and `date` commands.

Finding files

In the final topic for this part of the tutorial, we will look at the `find` command which is used to find files in one or more directory trees, based on criteria such as name, timestamp, or size. Again, we will use the `lpi103` directory that we created earlier in this tutorial.

find

The `find` command will search for files or directories using all or part of the name, or by other search criteria, such as size, type, file owner, creation date, or last access date. The most basic `find` is a search by name or part of a name. Listing 61 shows an example from our `lpi103` directory where we first search for all files that have either a 'l' or a 'k' in their name, then perform some path searches that we will

explain in the notes below.

Listing 61. Finding files by name

```
[ian@echidna lpil03]$ find . -name "[l*k]*"
./text1
./f1
./backup
./backup/text1.bkp.2
./backup/text1.bkp.1
./f1a
[ian@echidna lpil03]$ find . -ipath "*ACK*1"
./backup/text1.bkp.1
[ian@echidna lpil03]$ find . -ipath "*ACK*/*1"
./backup/text1.bkp.1
[
```

Notes:

1. The patterns that you may use are shell wildcard patterns like those we saw earlier when we discussed under [Wildcards and globbing](#).
2. You can use `-path` instead of `-name` to match full paths instead of just base file names. In this case, the pattern **may** span path components.
3. If you want case-insensitive search as shown in the use of `ipath` above, precede the `find` options that search on a string or pattern with an `'i'`
4. If you want to find a file or directory whose name begins with a dot, such as `.bashrc` or the current directory (`.`), then you **must** specify a leading dot as part of the pattern. Otherwise, name searches will ignore these files or directories.

In the first example above, we found both files and a directory (`./backup`). Use the `-type` parameter along with one-letter type to restrict the search. Use `'f'` for regular files, `'d'` for directories, and `'l'` for symbolic links. See the man page for `find` for other possible types. Listing 62 shows the result of searching for directories (`-type d`).

Listing 62. Find files by type

```
[ian@echidna lpil03]$ find . -type d
.
./backup
[ian@echidna lpil03]$ find . -type d -name "*"
./backup
```

Note that the `-type d` specification without any form of name specification displays directories that have a leading dot in their names (only the current directory in this case).

We can also search by file size, either for a specific size (n) or for files that are either larger (+n) or smaller than a given value (-n). By using both upper and lower size bounds, we can find files whose size is within a given range. By default the `-size` option of `find` assumes a unit of 'b' for 512-byte blocks. Among other choices, specify 'c' for bytes, or 'k' for kilobytes. In Listing 63 we first find all files with size 0, and then all with size of either 24 or 25 bytes. Note that specifying `-empty` instead of `-size 0` also finds empty files.

Listing 63. Finding files by size

```
[ian@echidna lpil03]$ find . -size 0
./f2
./f3
./f4
./f5
./f6
./f7
./f8
./f9
./fla
[ian@echidna lpil03]$ find . -size -26c -size +23c -print
./text1
./text2
./text5
./backup/text1.bkp.2
./backup/text1.bkp.1
```

Listing 63 introduces the `-print` option which is an example of an *action* that may be taken on the results returned by the search. In the bash shell, this is the default action if no action is specified. On some systems and some shells, an action is required, otherwise there is no output.

Other actions include `-ls` which prints file information equivalent to that from the `ls -lids` command, or `-exec` which executes a command for each file. The `-exec` must be terminated by a semi-colon which must be escaped to avoid the shell interpreting it first. Also specify `{}` wherever you want the returned file used in the command. as we saw above, the curly braces also have meaning to the shell and should be escaped (or quoted). Listing 64 shows how the `-ls` and the `-exec` options can be used to list file information.

Listing 64. Finding and acting on files

```
[ian@echidna lpil03]$ find . -size -26c -size +23c -ls
2128984  4 -rw-rw-r--  1 ian  ian    24 Sep 23 12:27 ./text1
2128985  4 -rw-rw-r--  1 ian  ian    25 Sep 23 13:39 ./text2
2128982  4 -rw-rw-r--  1 ian  ian    24 Sep 26 12:46 ./text5
1564497  4 -rw-rw-r--  1 ian  ian    24 Oct  4 09:45 ./backup/text1.bkp.2
2129019  4 -rw-rw-r--  1 ian  ian    24 Oct  4 09:43 ./backup/text1.bkp.1
[ian@echidna lpil03]$ find . -size -26c -size +23c -exec ls -l '{}' \;
```

The `-exec` option can be used for as many purposes as your imagination can dream up. For example:

```
find . -empty -exec rm '{}' \;
```

removes all the empty files in a directory tree, while

```
find . -name "*.htm" -exec mv '{}' '{}1' \;
```

renames all .htm file to .html files.

For our final examples, we use the timestamps described with the `touch` command to locate files having particular timestamps. Listing 65 shows three examples:

1. When used with `-mtime -2`, the `find` command finds all files modified within the last two days. A day in this case is a 24-hour period relative to the current date and time. Note that you would use `-atime` if you wanted to find files based on access time rather than modification time.
2. Adding the `-daystart` option means that we want to consider days as calendar days, starting at midnight. Now the `f3` file is excluded from the list.
3. Finally, we show how to use a time range in minutes rather than days to find files modified between one hour (60 minutes) and 10 hours (600 minutes) ago.

Listing 65. Finding files by timestamp

```
[ian@echidna lpil03]$ date
Mon Nov  7 14:59:02 EST 2005
[ian@echidna lpil03]$ find . -mtime -2 -type f -exec ls -l '{}' \;
-rw-rw-r--  1 ian   ian           0 Nov  5 15:10 ./f3
-rw-rw-r--  1 ian   ian           0 Nov  7 11:00 ./f4
-rw-rw-r--  1 ian   ian           0 Nov  6 06:00 ./f6
-rw-rw-r--  1 ian   ian           0 Nov  8 2005 ./f8
[ian@echidna lpil03]$ find . -daystart -mtime -2 -type f -exec ls -l '{}' \;
-rw-rw-r--  1 ian   ian           0 Nov  7 11:00 ./f4
-rw-rw-r--  1 ian   ian           0 Nov  6 06:00 ./f6
-rw-rw-r--  1 ian   ian           0 Nov  8 2005 ./f8
[ian@echidna lpil03]$ find . -mmin -600 -mmin +60 -type f -exec ls -l '{}' \;
-rw-rw-r--  1 ian   ian           0 Nov  7 11:00 ./f4
```

The man pages for the `find` command can help you learn the extensive range of options that we cannot cover in this brief introduction.

Section 5. Streams, pipes, and redirects

This section covers material for topic 1.103.4 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn about the following topics:

- Redirecting the standard IO streams: standard input, standard output, and standard error
- Piping output from one command to the input of another
- Sending output to both stdout and a file
- Using command output as arguments to another command

Redirecting standard IO

Recall that shells use three standard I/O *streams*.

1. *stdout* is the *standard output stream* which displays output from commands. It has file descriptor 1.
2. *stderr* is the *standard error stream* which displays error output from commands. It has file descriptor 2.
3. *stdin* is the *standard input stream* which provides input to commands. It has file descriptor 0.

Input streams provide input to programs, usually from terminal keystrokes. Output streams print text characters, usually to the terminal. The terminal was originally an ASCII typewriter or display terminal, but is now more often a window on a graphical desktop.

As we saw in [Text streams and filters](#) we can redirect standard output to a file or to the standard input of another command and we can redirect standard input from a file or from the output of another command.

Redirecting output

There are two ways to redirect output:

***n*>**

redirects output from file descriptor *n* to a file. You must have write authority to the file. If the file does not exist, it is created. If it does exist, the existing contents are usually lost without any warning.

***n*>>**

also redirects output from file descriptor *n* to a file. Again, you must have write

authority to the file. If the file does not exist, it is created. If it does exist, the output is appended to the existing file.

The *n* in *n>* or *n>>* refers to the *file descriptor*. If it omitted, then standard output is assumed. Listing 66 illustrates using redirection to separate the standard output and standard error from the `ls` command using files we created earlier in our `lpi103` directory. We also illustrate appending output to existing files.

Listing 66. Output redirection

```
[ian@echidna lpi103]$ ls x* z*
ls: z*: No such file or directory
xaa xab
[ian@echidna lpi103]$ ls x* z* >stdout.txt 2>stderr.txt
[ian@echidna lpi103]$ ls w* y*
ls: w*: No such file or directory
yaa yab
[ian@echidna lpi103]$ ls w* y* >>stdout.txt 2>>stderr.txt
[ian@echidna lpi103]$ cat stdout.txt
xaa
xab
yaa
yab
[ian@echidna lpi103]$ cat stderr.txt
ls: z*: No such file or directory
ls: w*: No such file or directory
```

We said that output redirection using *n>* usually overwrites existing files. You can control this with the `noclobber` option of the `set` builtin. If it has been set, you can override it using *n>|* as shown in Listing 67.

Listing 67. Output redirection with noclobber

```
[ian@echidna lpi103]$ set -o noclobber
[ian@echidna lpi103]$ ls x* z* >stdout.txt 2>stderr.txt
-bash: stdout.txt: cannot overwrite existing file
[ian@echidna lpi103]$ ls x* z* >|stdout.txt 2>|stderr.txt
[ian@echidna lpi103]$ cat stdout.txt
xaa
xab
[ian@echidna lpi103]$ cat stderr.txt
ls: z*: No such file or directory
[ian@echidna lpi103]$ set +o noclobber #restore original noclobber setting
```

Sometimes you may want to redirect both standard output and standard error into a file. This is often done for automated processes or background jobs so that you can review the output later. Use `&>` or `&>>` to redirect both standard output and standard error to the same place. Another way of doing this is to redirect file descriptor *n* and then redirect file descriptor *m* to the same place using the construct `m>&n` or `m>>&n`. The order in which outputs are redirected is important. For example, command `2>&1 >output.txt` is not the same as command `>output.txt 2>&1`

We illustrate these redirections in Listing 68. Notice in the last command that standard output was redirected after standard error, so the standard error output still goes to the terminal window.

Listing 68. Redirecting two streams to one file

```
[ian@echidna lpil03]$ ls x* z* &>output.txt
[ian@echidna lpil03]$ cat output.txt
ls: z*: No such file or directory
xaa
xab
[ian@echidna lpil03]$ ls x* z* >output.txt 2>&1
[ian@echidna lpil03]$ cat output.txt
ls: z*: No such file or directory
xaa
xab
[ian@echidna lpil03]$ ls x* z* 2>&1 >output.txt
ls: z*: No such file or directory
[ian@echidna lpil03]$ cat output.txt
xaa
xab
```

At other times you may want to ignore either standard output or standard error entirely. To do this, redirect the appropriate stream to `/dev/null`. In Listing 69 we show how to ignore error output from the `ls` command.

Listing 69. Ignoring output using `/dev/null`

```
[ian@echidna lpil03]$ ls x* z* 2>/dev/null
xaa xab
[ian@echidna lpil03]$ cat /dev/null
```

Redirecting input

Just as we can redirect the `stdout` and `stderr` streams, so too we can redirect `stdin` from a file, using the `<` operator. If you recall, in our discussion of [sort and uniq](#) that we used the `tr` command to replace the spaces in our `text1` file with tabs. In that example we used the output from the `cat` command to create standard input for the `tr` command. Instead of needlessly calling `cat`, we can now use input redirection to translate the spaces to tabs, as shown in Listing 70.

Listing 70. Input redirection

```
[ian@echidna lpil03]$ tr ' ' '\t'<text1
1      apple
2      pear
3      banana
```

Shells, including `bash`, also have the concept of a *here-document*, which is another form of input redirection. This uses the `<<` along with a word, such as `END`, for a

marker or sentinel to indicate the end of the input. We illustrate this in Listing 71.

Listing 71. Input redirection with a here-document

```
[ian@echidna lpil03]$ sort -k2 <<END
> 1 apple
> 2 pear
> 3 banana
> END
1 apple
3 banana
2 pear
```

Remember how you created the `text2` file back in [Listing 23](#)? You may wonder why you couldn't have typed just `sort -k2`, entered your data, and then pressed **Ctrl-d** to signal end of input. And the short answer is that you could, but you would not have learned about here-documents. The real answer is that here-documents are more often used in shell scripts (which are covered in the tutorial for topic 109 on shells, scripting, programming, and compiling). A script doesn't have any other way of signaling which lines of the script should be treated as input. Because shell scripts make extensive use of tabbing to provide indenting for readability, there is another twist to here-documents. If you use `<<-` instead of just `<<`, then leading tabs are stripped. In Listing 72 we use the same technique for creating a captive tab character that we used in [Listing 42](#). We then create a very small shell script containing two `cat` commands which each read from a here-document. Finally, we use the `.` (dot) command to *source* the script, which means to run it in the current shell context.

Listing 72. Input redirection with a here-document

```
[ian@echidna lpil03]$ ht=$(echo -en "\t")
[ian@echidna lpil03]$ cat<<END>ex-here.sh
> cat <<-EOF
> apple
> EOF
> ${ht}cat <<-EOF
> ${ht}pear
> ${ht}EOF
> END
[ian@echidna lpil03]$ cat ex-here.sh
cat <<-EOF
apple
EOF
    cat <<-EOF
    pear
    EOF
[ian@echidna lpil03]$ . ex-here.sh
apple
pear
```

Pipelines

In the section on [Text streams and filters](#) we described text *filtering* as the process of taking an input stream of text and performing some conversion on the text before sending it to an output stream. We also said that filtering is most often done by constructing a *pipeline* of commands where the output from one command is *piped* or *redirected* to be used as input to the next. Using pipelines in this way is not restricted to text streams., although that is often where they are used.

Piping stdout to stdin

As we have already seen, we use the | (pipe) operator between two commands to direct the stdout of the first to the stdin of the second. We construct longer pipelines by adding more commands and more pipe operators as shown in Listing 73.

Listing 73. Piping output through several commands

```
command1 | command2 | command3
```

One thing to note is that pipelines **only** pipe stdout to stdin. You cannot use 2| to pipe stderr alone, at least, not with the tools we have learned so far. If stderr has been redirected to stdout, then both streams will be piped. We illustrate this in Listing 74 where we use a pipe to sort both the error and normal output messages from an unlikely `ls` command with four wildcard arguments that are not in alphabetical order.

Listing 74. Piping two output streams

```
[ian@echidna lpi103]$ ls y* x* z* u* q* 2>&1 |sort
ls: q*: No such file or directory
ls: u*: No such file or directory
ls: z*: No such file or directory
xaa
xab
yaa
yab
```

Any of the commands may have options or arguments. Many commands use a hyphen (-) used in place of a filename as an argument to indicate when the input should come from stdin rather than a file. Check the man pages for the command to be sure. Constructing long pipelines of commands that each have limited capability is a common Linux and UNIX way of accomplishing tasks.

One advantage of pipes on Linux and UNIX systems is that, unlike some other popular operating systems, there is no intermediate file involved with a pipe. The stdout of the first command is **not** written to a file and then read by the second command. If your particular version of `tar` doesn't happen to support unzipping files compressed using `bzip2`, that's no problem. As we saw in the tutorial for Topic 102, you can just use a pipeline like

`bunzip2 -c drgeo-1.1.0.tar.bz2 | tar -xvf -`
to do the task.

Output as arguments

In the section [Using the command line](#) we learned about command substitution and how to use the output from a command as part of another. In the previous section on [Basic file management](#) we learned how to use the `-i` option of the `find` command to use the output of the `find` command as input for another command. Listing 75 shows three ways of displaying the contents of our `text1` and `text2` files using these techniques.

Listing 75. Using output as arguments with command substitution and `find -exec`

```
[ian@echidna lpil03]$ cat `ls text[12]`
1 apple
2 pear
3 banana
9     plum
3     banana
10    apple
[ian@echidna lpil03]$ cat $(find . -name "text[12]")
1 apple
2 pear
3 banana
9     plum
3     banana
10    apple
[ian@echidna lpil03]$ find . -name "text[12]" -exec cat '{}' \;
1 apple
2 pear
3 banana
9     plum
3     banana
10    apple
```

The above methods are fine as far as they go, but they have some limitations. Let's consider the case of a filename containing whitespace (a blank in this case). Look at Listing 76 and see if you can understand what is happening with each of the commands before reading the explanations below.

Listing 76. Using output as arguments with command substitution and `find -exec`

```
[ian@echidna lpil03]$ echo grapes>"text sample2"
[ian@echidna lpil03]$ cat `ls text*le2`
cat: text: No such file or directory
cat: sample2: No such file or directory
[ian@echidna lpil03]$ cat "`ls text*le2`"
grapes
[ian@echidna lpil03]$ cat "`ls text*2`"
cat: text2
```

```
text sample2: No such file or directory
```

Here is what we did.

- We created a file called "text sample2" containing one line with the word "grapes"
- We attempted to use command substitution to display the contents of "text sample2". This failed because the shell passed **two** parameters to `cat`, namely `text` and `sample2`.
- Being smarter than the shell, we decided to put quotes around the command substitution values. This works
- Finally, we changed the wildcard expression and the output is a very strange looking error. What is happening here is that the shell is giving the `cat` command a **single** parameter which is equivalent to the string that would result from `echo -e "text2\ntext sample2"`
If this seems strange, try it yourself!

What we need here is some way of delineating the individual file names regardless of whether they consist of a single word or multiple words. We haven't mentioned it previously, but when the output of commands such as `ls` is used in a pipe or command substitution, the output is usually delivered one item per line. One method to handle this is with a `read` builtin in a loop with the `while` builtin. Although beyond the scope of this objective, we illustrate it to whet your appetite and as a lead-in to the solution we present next.

Listing 77. Using `while` and `read` in a loop

```
[ian@echidna lpil03]$ ls text*2 | while read l; do cat "$l";done
9      plum
3      banana
10     apple
grapes
```

xargs

Most of the time we want to process lists of files, so what we really need is some way of finding them and then processing them. Fortunately, the `find` command has an option, `-print0`, which separates output filenames with a null character instead of a newline. Commands such as `tar` and `xargs` have a `-0` (or `--null` option that allows them to understand this form of parameter. We have already seen `tar`. The `xargs` command works a little like the `-exec` option of `find`, but there are some important differences as we shall see. First let's look at an example.

Listing 78. Using xargs with -0

```
[ian@echidna lpil03]$ find . -name "text*2" -print0 |xargs -0 cat
9      plum
3      banana
10     apple
1 apple
2 pear
3 banana
grapes
```

Notice that we now pipe the output from `find` to `xargs`. You don't need the delimited semi-colon on the end of the command and, by default, `xargs` appends the arguments to the command string. However, we seem to have seven lines of output rather than the expected four. What went wrong?

find again

We can use the `wc` command to check that there are only four lines of output in the two files we thought we were printing. The answer to our problem lies in the fact that `find` is searching our backup directory where it also finds `backup/text1.bkp.2` which matches our wildcard pattern. To solve this, we use the `-maxdepth` option of `find` to restrict the search to a depth of one directory, the current one. There's also a corresponding `-mindepth` option which allows you to be quite specific in where you search. Listing 79 illustrates our final solution.

Listing 79. Restricting find to our four desired lines

```
[ian@echidna lpil03]$ ls text*2
text2  text sample2
[ian@echidna lpil03]$ wc text*2
  3      6    25 text2
  1      1     7 text sample2
  4      7    32 total
[ian@echidna lpil03]$ find . -name "text*2" -maxdepth 1 -print0 |xargs -0 cat
9      plum
3      banana
10     apple
grapes
```

More on xargs

There are some other differences between `xargs` and `find -exec`.

- The `xargs` command defaults to passing as many arguments as possible to the command. You may limit the number of input lines used with `-l` or `--max-lines` and a number. Alternatively, you may use `-n` or `--max-args` to limit the number of arguments passed, or `-s` or `--max-chars` to limit the maximum number of characters used in the argument string. If your command can process multiple arguments, it is generally more efficient to process as many as possible at a time.

- You may use '{}' as you did for `find -exec` if you specify the `-i` or `--replace` option. You may change the default of '{}' for the string that indicates where to substitute the input parameter by specifying a value for `-i`. This implies `-l 1`.

Our final examples for `xargs` are shown in Listing 80.

Listing 80. Additional xargs examples

```
[ian@echidna lpil03]$ # pass all arguments at once
[ian@echidna lpil03]$ find . -name "text*2" |xargs echo
./text2 ./backup/text1.bkp.2 ./text sample2
[ian@echidna lpil03]$ # show the files we created earlier with the touch command
[ian@echidna lpil03]$ ls f[0-n]*|xargs echo
f1 f1a f2 f3 f4 f5 f6 f7 f8 f9
[ian@echidna lpil03]$ # remove them in one stroke
[ian@echidna lpil03]$ ls f[0-n]*|xargs rm
[ian@echidna lpil03]$ # Use a replace string
[ian@echidna lpil03]$ find . -name "text*2" |xargs -i echo - '{}' -
- ./text2 -
- ./backup/text1.bkp.2 -
- ./text sample2 -
[ian@echidna lpil03]$ # Limit of one input line per invocation
[ian@echidna lpil03]$ find . -name "text*2" |xargs -l1 echo
./text2
./backup/text1.bkp.2
./text sample2
[ian@echidna lpil03]$ # Limit of one argument per invocation
[ian@echidna lpil03]$ find . -name "text*2" |xargs -n1 echo
./text2
./backup/text1.bkp.2
./text
sample2
```

Note that we did not use `-print0` here. Does that explain the final example in Listing 80?

Splitting output

This section wraps up with a brief discussion of one more command. Sometimes you may want to see output on your screen while saving a copy for later. While you **could** do this by redirecting the command output to a file in one window and then using `tail -fn1` to follow the output in another screen, using the `tee` command is easier.

You use `tee` with a pipeline. The arguments are a file (or multiple files) for standard output. The `-a` option appends rather than overwriting files. As we saw earlier in our discussion of pipelines, you need to redirect `stderr` to `stdout` before piping to `tee` if you want to save both. Listing 81 shows `tee` being used to save output in two files, `f1` and `f2`.

Listing 81. Splitting stdout with tee

```
[ian@echidna lpi103]$ ls text[1-3]|tee f1 f2
text1
text2
text3
[ian@echidna lpi103]$ cat f1
text1
text2
text3
[ian@echidna lpi103]$ cat f2
text1
text2
text3
```

Section 6. Create, monitor, and kill processes

This section covers material for topic 1.103.5 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn about the following topics:

- Foreground and background jobs
- Starting processes without a terminal for I/O
- Monitoring and displaying processes
- Sending signals to processes
- Identifying and killing processes

If you stop and reflect for a moment, it is pretty obvious that lots of things are running on your computer, other than the terminal programs we have been running. Indeed, if you are using a graphical desktop, you may have opened more than one terminal window at one time, or perhaps have opened a file browser, internet browser, game, spreadsheet, or other application. So far, our examples have entered commands at a terminal window. The command runs and we wait for it to complete before we do anything else. In the section [Using the command line](#), we encountered the `ps` command which displayed process status and we saw that processes have a Process ID (PID) and a Parent Process id (PPID). In this section, you learn how to do more than one thing at a time using your terminal window.

Foreground and background jobs

When you run a command in your terminal window, such as we have done to this

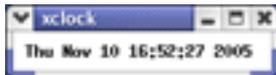
point, you are running it in the *foreground*. Our commands to date have run quickly, but suppose we are running a graphical desktop and would like a digital clock displayed on the desktop. For now, let's ignore the fact that most desktops already have one; we're just using this as an example.

If you have the X Window System installed, you probably also have some utilities such as `xclock` or `xeyes`. Either works for this exercise, but we'll use `xclock`. The man page explains that you can launch a digital clock on your graphical desktop using the command

```
xclock -d -update 1
```

The `-update 1` part requests updates every second, otherwise the clock updates only every minute. So let's run this in a terminal window. We should see a clock like Figure 2, and our terminal window should look like Listing 82. If you don't have `xclock` or the X Window System, we'll show you shortly how to create a poor man's digital clock with your terminal, so you might want to follow along for now and then retry these exercises with that clock.

Figure 2. A digital clock with `xclock`



Listing 82. Starting `xclock`

```
[ian@echidna ian]$ xclock -d -update 1
```

Unfortunately, your terminal window no longer has a prompt, so we really need to get control back. Fortunately, the Bash shell has a *suspend* key, `Ctrl-z`. Pressing this key combination gets you a terminal prompt again as shown in Listing 83

Listing 83. Suspending `xclock` with `Ctrl-z`

```
[ian@echidna ian]$ xclock -d -update 1
[1]+  Stopped                  xclock -d -update 1
[ian@echidna ian]$
```

The clock is still on your desktop, but it has stopped running. Suspending it did exactly that. In fact, if you drag another window over part of it, that part won't even redraw. You also see a terminal output message indicating "[1]+ Stopped". The 1 in this message is a *job number*. You can restart the clock by typing `fg %1`. You could also use the command name or part of it by using `fg %xclock` or `fg %?clo`. Finally, if you just use `fg` with no parameters, you can restart the most recently stopped job, job 1 in this case. Restarting it with `fg` also brings the job right back to the foreground, and you no longer have a shell prompt. What you need to do is place the job in the *background*; a `bg` command takes the same type of job specification as the `fg` command and does exactly that. Listing 84 shows how to

bring the `xclock` job back to the foreground and suspend it using two forms of the `fg` command. You can suspend it again and place it in the background; the clock continues to run while you do other work at your terminal.

Listing 84. Placing `xclock` in the background

```
[ian@echidna ian]$ fg %1
xclock -d -update 1

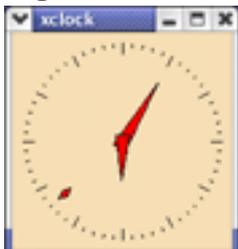
[1]+  Stopped                  xclock -d -update 1
[ian@echidna ian]$ fg %?clo
xclock -d -update 1

[1]+  Stopped                  xclock -d -update 1
[ian@echidna ian]$ bg
[1]+  xclock -d -update 1 &
[ian@echidna ian]$
```

Using "&"

You may have noticed that when we placed the `xclock` job in the background, the message no longer said "Stopped" and that it was terminated with an ampersand (&). In fact, you don't need to suspend the process to place it in the background at all. You can simply append an ampersand to the command and the shell will start the command (or command list) in the background. Let's start an analog clock with a wheat background and red hands using this method. You should see a clock like that in Figure 3 and terminal output like Listing 85.

Figure 3. An analog clock with `xclock`



Listing 85. Starting `xclock` in background with &

```
[ian@echidna ian]$ xclock -bg wheat -hd red -update 1&
[2] 5659
```

Notice that the message is slightly different this time. It represents a job number and a process id (PID). We will cover PIDs and more about status in a moment. For now, let's use the `jobs` command to find out what jobs are running. Add the `-l` option to list PIDs, and you see that job 2 indeed has PID 5659 as shown in Listing 86. Note also that job 2 has a plus sign (+) beside the job number, indicating that it is the *current job*. This job will come to the foreground if no job specification is given with the `fg` command.

Listing 86. Displaying job and process information

```
[ian@echidna ian]$ jobs -l
[1]-  4234 Running                  xclock -d -update 1 &
[2]+  5659 Running                  xclock -bg wheat -hd red -update 1 &
```

Before we address some other issues related to background jobs, let's create a poor man's digital clock. We use the `sleep` command to cause a delay for two seconds and we use the `date` command to print the current date and time. We wrap these commands in a `while` loop with a `do/done` block to create an infinite loop. Finally we put the whole lot in parentheses to make a command list and put the entire list in the background using an ampersand.

Listing 87. Poor man's digital clock

```
[ian@echidna ian]$ (while sleep 2; do date;done) &
[1] 16291
[ian@echidna ian]$ Thu Nov 10 22:58:02 EST 2005
Thu Nov 10 22:58:04 EST 2005
Thu Nov 10 22:58:06 EST 2005
Thu Nov 10 22:58:08 EST 2005
fThu Nov 10 22:58:10 EST 2005
Thu Nov 10 22:58:12 EST 2005
gThu Nov 10 22:58:14 EST 2005

( while sleep 2; do
  date;
done )
Thu Nov 10 22:58:16 EST 2005
Thu Nov 10 22:58:18 EST 2005
```

As expected, our list is running as job 1 with PID 16291. Every two seconds, the `date` command runs and a date and time are printed on the terminal. The input that you type is highlighted. A slow typist will have characters interspersed with several lines of output before a full command can be typed. In fact, notice how the 'f' 'g' that we type in to bring the command list to foreground are a couple of lines apart. When we finally get the `f g` command entered, bash displays the command that is now running in our shell, namely, the command list, which is still happily printing the time every two seconds.

Once we succeed in getting the job into the foreground, we can either terminate (or *kill*), or take some other action, In this case, we user Ctrl-c to terminate our 'clock'.

Standard IO and background processes

The output from the `date` command in our previous example is interspersed with echoed characters for the `f g` command that we are trying to type. This raises an interesting issue. What happens to a process if it needs input from stdin?

The terminal process under which we start a background application is called the

controlling terminal. Unless redirected elsewhere, the stdout and stderr streams from the background process are directed to the controlling terminal. Similarly, the background task expects input from the controlling terminal, but the controlling terminal has no way of directing characters you type to the stdin of a background process. In such a case, the Bash shell suspends the process, so that it is no longer executing. You may bring it to the foreground and supply the necessary input. Listing 88 illustrates a simple case where you can put a command list in the background. After a moment, press **Enter** and the process stops. Bring it to the foreground and provide a line of input followed by Ctrl-d to signal end of input file. The command list completes and we display the file we created.

Listing 88. Waiting for stdin

```
[ian@echidna ian]$ (date; cat - >bginput.txt; date)&
[1] 18648
[ian@echidna ian]$ Fri Nov 11 00:03:28 EST 2005

[1]+  Stopped                  ( date; cat - >bginput.txt; date )
[ian@echidna ian]$ fg
( date; cat - >ginput.txt; date )
input data
Fri Nov 11 00:03:53 EST 2005
[ian@echidna ian]$ cat bginput.txt
input data
```

Jobs without terminals

In practice, we probably want to have standard IO streams for background processes redirected to or from a file. There is another related question; what happens to the process if the controlling terminal closes or the user logs off? The answer depends on the shell in use. If the shell sends a SIGHUP (or hangup) signal, then the application is likely to close. We cover signals shortly, but for now we'll consider another way around this problem.

nohup

The `nohup` command is used to start a command that will ignore hangup signals and will append stdout and stderr to a file. The default file is either `nohup.out` or `$HOME/nohup.out`. If the file cannot be written, then the command will not run. If you want output to go somewhere else, redirect stdout, or stderr as we learned in the previous section of this tutorial.

One other aspect of `nohup` is that it will not execute a pipeline or a command list. In the topic [Redirecting standard IO](#) we showed how to save a set of commands in a shell script and source it. You can save a pipeline or list in a file and then run it using the `sh` (default shell) or the `bash` command, although you can't use the `.` or `source` command as we did in the earlier example. The next tutorial in this series (on topic

104, covering Devices, Linux filesystems, and the Filesystem Hierarchy Standard) shows how to make the script file executable, but for now we'll stick to running scripts by sourcing them or by using the `sh` or the `bash` command, Listing 89 shows how we might do this for our poor man's digital clock. Needless to say, having the time written to a file isn't particularly useful, and the file will keep growing, so we'll set the clock to update every 30 seconds instead of every second.

Listing 89. Using `nohup` with a command list in a script

```
[ian@echidna ian]$ echo "while sleep 30; do date;done">pmc.sh
[ian@echidna ian]$ nohup . pmc.sh&
[1] 21700
[ian@echidna ian]$ nohup: appending output to `nohup.out'

[1]+  Exit 126                  nohup . pmc.sh
[ian@echidna ian]$ nohup sh pmc.sh&
[1] 21709
[ian@echidna ian]$ nohup: appending output to `nohup.out'

[ian@echidna ian]$ nohup bash pmc.sh&
[2] 21719
[ian@echidna ian]$ nohup: appending output to `nohup.out'
```

If we display the contents of `nohup.out`, we see that the first line indicates why we got an exit code of 126 on our first attempt above. Subsequent lines will be output from the two versions of `pmc.sh` that are now running in background. This is illustrated in Listing 90.

Listing 90. Output from `nohup` processes

```
[ian@echidna ian]$ cat nohup.out
/bin/nice: .: Permission denied
Fri Nov 11 15:30:03 EST 2005
Fri Nov 11 15:30:15 EST 2005
Fri Nov 11 15:30:33 EST 2005
Fri Nov 11 15:30:45 EST 2005
Fri Nov 11 15:31:03 EST 2005
```

Now let's turn our attention to the status of our processes. If you are following along and planning to take a break at this point, please stay around as you now have two jobs that are creating ever larger files in your file system. You can use the `fg` command to bring each to foreground, and then use `Ctrl-c` to terminate it, but if you let them run for a little longer we'll see other ways to monitor and interact with them.

Process status

In the previous part of this section, we had a brief introduction to the `jobs` command and saw how to use it to list the Process IDs (or PIDs) of our jobs.

`ps`

There is another command, the `ps` command, which we use to display various pieces of process status information. Remember "ps" as an acronym for "process status". The `ps` command accepts zero or more PIDs as argument and displays the associated process status. If we use the `jobs` command with the `-p` option, the output is simply the PID of the *process group leader* for each job. We'll use this output as arguments to the `ps` command as shown in Listing 91.

Listing 91. Status of background processes

```
[ian@echidna ian]$ jobs
[1]-  Running                nohup sh pmc.sh &
[2]+  Running                nohup bash pmc.sh &
[ian@echidna ian]$ jobs -p
21709
21719
[ian@echidna ian]$ ps `jobs -p`
  PID TTY          STAT       TIME COMMAND
 21709 pts/3        SN           0:00 sh pmc.sh
 21719 pts/3        SN           0:00 bash pmc.sh
```

If we use `ps` with no options we see a list of processes that have our terminal as their controlling terminal as shown in Listing 92.

Listing 92. Displaying status with ps

```
[ian@echidna ian]$ ps
  PID TTY          STAT       TIME CMD
 20475 pts/3        SN           0:00 bash
 21709 pts/3        SN           0:00 sh
 21719 pts/3        SN           0:00 bash
 21922 pts/3        SN           0:00 sleep
 21930 pts/3        SN           0:00 sleep
 21937 pts/3        SN           0:00 ps
```

Several options, including `-f` (full), `-j` (jobs), and `-l` (long) give control of how much information is displayed. If we do not specify any PIDs, then another useful option is the `--forest` option, which displays the commands in a tree hierarchy, showing us which process has which other process as a parent. In particular, we see that the `sleep` commands of the previous listing are children of the scripts we have running in background. If we happened to run the command at a different instant, we might see the `date` command listed in the process status instead, but the odds are very small with this script. We illustrate some of these in Listing 93.

Listing 93. More status information

```
[ian@echidna ian]$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
ian          20475 20474  0  15:02 pts/3        00:00:00 -bash
ian          21709 20475  0  15:29 pts/3        00:00:00 sh pmc.sh
ian          21719 20475  0  15:29 pts/3        00:00:00 bash pmc.sh
ian          21945 21709  0  15:34 pts/3        00:00:00 sleep 30
ian          21953 21719  0  15:34 pts/3        00:00:00 sleep 30
```

```

ian      21954 20475  0 15:34 pts/3      00:00:00 ps -f
[ian@echidna ian]$ ps -j --forest
  PID  PGID  SID  TTY      TIME  CMD
20475 20475 20475 pts/3    00:00:00 bash
21709 21709 20475 pts/3    00:00:00 sh
21945 21709 20475 pts/3    00:00:00 \_ sleep
21719 21719 20475 pts/3    00:00:00 bash
21953 21719 20475 pts/3    00:00:00 \_ sleep
21961 21961 20475 pts/3    00:00:00 ps

```

Listing other processes

The `ps` commands we have used so far only list processes that were started from your terminal session (note the `SID` column in the second example of Listing 93). To see all the processes with controlling terminals use the `-a` option. The `-x` option displays processes without a controlling terminal, and the `-e` option displays information for **every** process. Listing 94 shows full format for all the processes with a controlling terminal.

Listing 94. Displaying other processes

```

[ian@echidna ian]$ ps -af
UID      PID  PPID  C  STIME  TTY      TIME  CMD
ian      4234 32537  0  Nov10 pts/0    00:00:00 xclock -d -update 1
ian      5659 32537  0  Nov10 pts/0    00:00:00 xclock -bg wheat -hd red -update
ian      21709 20475  0  15:29 pts/3    00:00:00 sh pmc.sh
ian      21719 20475  0  15:29 pts/3    00:00:00 bash pmc.sh
ian      21969 21709  0  15:35 pts/3    00:00:00 sleep 30
ian      21977 21719  0  15:35 pts/3    00:00:00 sleep 30
ian      21978 20475  0  15:35 pts/3    00:00:00 ps -af

```

Note that this listing includes the two `xclock` processes that we started earlier from the main graphical terminal of this system (indicated here by `pts/0`), while the remaining processes displayed are those associated with an `ssh` (Secure Shell) connection (`pts/3` in this case).

There are many more options for `ps`, including a number which provide significant control over what fields are displayed and how they are displayed. Others provide control over the selection of processes for display, for example, by selecting those processes for a particular user. See the man pages for `ps` for full details, or get a brief summary by using `ps --help`.

top

If you run `ps` several times in a row, to see what is changing, you probably need the `top` command instead. It displays a continuously updated process list, along with useful summary information. See the man pages for `top` for full details on options, including how to sort by memory usage or other criteria. Listing 95 shows the first few lines of a `top` display.

Listing 95. Displaying other processes

```

3:37pm up 46 days, 5:11, 2 users, load average: 0.01, 0.17, 0.19
96 processes: 94 sleeping, 1 running, 0 zombie, 1 stopped
CPU states: 0.1% user, 1.0% system, 0.0% nice, 0.9% idle
Mem: 1030268K av, 933956K used, 96312K free, 0K shrd, 119428K buff
Swap: 1052216K av, 1176K used, 1051040K free 355156K cached

```

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	COMMAND
22069	ian	17	0	1104	1104	848	R	0.9	0.1	0:00	top
1	root	8	0	500	480	444	S	0.0	0.0	0:04	init
2	root	9	0	0	0	0	SW	0.0	0.0	0:00	keventd
3	root	9	0	0	0	0	SW	0.0	0.0	0:00	kapmd
4	root	19	19	0	0	0	SWN	0.0	0.0	0:00	ksoftirqd_CPU0
5	root	9	0	0	0	0	SW	0.0	0.0	0:00	kswapd

Signals

Let's now look at Linux *signals*, which are an asynchronous way to communicating with processes. We have already mentioned the SIGHUP signal and we have used both Ctrl-c and Ctrl-z, which are another way of sending a signal to processes. The general way to send a signal is with the `kill` command.

Sending signals using kill

The `kill` command sends a signal to a specified job or process. Listing 96 shows the use of the SIGTSTP and SIGCONT signals to stop and resume a background job. Use of the SIGTSTP signal is equivalent to using the `fg` command to bring the job to the foreground and then Ctrl-z to suspend it. Using SIGCONT is like using the `bg` command.

Listing 96. Stopping and restarting background jobs

```

[ian@echidna ian]$ kill -s SIGTSTP %1
[ian@echidna ian]$ jobs -l
[1]+ 21709 Stopped                  nohup sh pmc.sh
[2]- 21719 Running                  nohup bash pmc.sh &
[ian@echidna ian]$ kill -s SIGCONT %1
[ian@echidna ian]$ jobs -l
[1]+ 21709 Running                  nohup sh pmc.sh &
[2]- 21719 Running                  nohup bash pmc.sh &

```

We used the job specification (%1) in this example, but you can also send signals to a process id (such as 21709 which is the PID of job %1). If you use the `tail` command while job %1 is stopped, only one process is updating the `nohup.out` file.

There are a number of other possible signals which you can display on your system using `kill -l`. Some are used to report errors such as illegal operation codes, floating point exceptions, or attempts to access memory that a process does not have access to. Notice that signals have both a number such as 20, and a name, such as SIGTSTP. You may use either the number or the name with the `-s` option.

You should always check the signal numbers on your system before assuming which number belongs to which signal.

Signal handlers and process termination

We have seen that Ctrl-c terminates a process. In fact, it sends a SIGINT (or interrupt) signal to the process. If you use kill without any signal name, it sends a SIGTERM signal. For most purposes, these two signals are equivalent.

We said that the nohup command makes a process immune to the SIGHUP signal. In general, a process can implement a *signal handler* to catch signals. So a process could implement a signal handler to catch either SIGINT or SIGTERM. Since the signal handler knows what signal was sent, it may choose to ignore SIGINT and only terminate when it receives SIGTERM, for example. Listing 97 shows how to send the SIGTERM signal to job %1. Notice that the process status shows as "Terminated" right after we send the signal. This would show as "Interrupt" if we used SIGINT instead. After a few moments, the process cleanup has occurred and the job no longer shows in the job list.

Listing 97. Terminating a process with SIGTERM

```
[ian@echidna ian]$ kill -s SIGTERM %1
[ian@echidna ian]$ jobs -l
[1] 21709 Terminated          nohup sh pmc.sh
[2]- 21719 Running             nohup bash pmc.sh &
[ian@echidna ian]$ jobs -l
[2]+ 21719 Running             nohup bash pmc.sh &
```

Signal handlers give a process great flexibility in that a process can do its normal work and be interrupted by a signal for some special purpose. Besides allowing a process to catch termination requests and take possible action such as closing files or checkpointing transactions in progress, signals are often used to tell a daemon process to reread its configuration file and possibly restart operation. You might do this for the inetd process when you change network parameters, or the line printer daemon (lpd) when you add a new printer.

Terminating processes unconditionally

Some signals cannot be caught, such as some hardware exceptions. SIGKILL, the most likely one you will use, cannot be caught by a signal handler and unconditionally terminates a process. In general, you should need this only if all other means of terminating the process have failed.

Logout and nohup

Remember that we said that using nohup would allow our processes to keep

running after we log out. Well, let's do that and then log back in again. After we log back in, let's check our remaining poor man's clock process using `jobs` and `ps` as we have done above. The output is shown in Listing 98.

Listing 98. Logging back in

```
[ian@echidna ian]$ jobs
[ian@echidna ian]$ ps -a
  PID TTY          TIME CMD
 4234 pts/0    00:00:00 xclock
 5659 pts/0    00:00:00 xclock
27217 pts/4    00:00:00 ps
```

We see that we are running on pts/4 this time, but there is no sign of our jobs, just the `ps` command and the two `xclock` processes we started from the graphical terminal (pts/0). Not what we were expecting perhaps. However, all is not lost. In Listing 99 we show one way to find our missing job using the `-s` option for session ID, along with the session ID of 20475 that we saw in Listing 93. Think about other ways you might have found them if you didn't happen to have the session ID available.

Listing 99. Logging back in

```
[ian@echidna ian]$ ps -js 20475
  PID  PGID  SID  TTY          TIME CMD
21719 21719 20475 ?           00:00:00 bash
27335 21719 20475 ?           00:00:00 sleep
```

Given what we have now learned about killing processes, you should be able to kill these processes using their PID and the `kill` command.

Section 7. Process execution priorities

This section covers material for topic 1.103.6 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn about the following topics:

- Process execution priorities
- Setting priorities
- Changing priorities

Priorities

As we have seen in the previous section, Linux, like most modern operating systems can run multiple processes. It does this by sharing the CPU and other resources among the processes. If some process can use 100% of the CPU, then other processes may become unresponsive. When we looked at [Process status](#) in the previous section, we saw that the `top` command's default output was to list processes in descending order of CPU usage. If we ran the `top` command with our Poor Man's Clock script, that process probably wouldn't make it onto the list because the process spends most of its time not using the CPU.

Your system may have many commands that are capable of using lots of CPU. Examples include movie editing tools, and programs to convert between different image types or between different sound encoding, such as mp3 to ogg.

We'll create a small script that just uses CPU and does little else. The script takes two inputs, a count and a label. It prints the label and the current date and time, then decrements the count till it reaches 0, then prints the label and the date again. This script has no error checking and is not very robust, but it illustrates our point.

Listing 100. CPU-intensive script

```
[ian@echidna ian]$ echo 'x="$1"'>count1.sh
[ian@echidna ian]$ echo 'echo "$2" $(date)'>>count1.sh
[ian@echidna ian]$ echo 'while [ $x -gt 0 ]; do let x=$x-1;done'>>count1.sh
[ian@echidna ian]$ echo 'echo "$2" $(date)'>>count1.sh
[ian@echidna ian]$ cat count1.sh
x="$1"
echo "$2" $(date)
while [ $x -gt 0 ]; do let x=$x-1;done
echo "$2" $(date)
```

If you run this on your own system, you might see output such as is shown in Listing 101. This script uses lots of CPU, as we'll see in a moment. If you are not using your own workstation, make sure that it is OK to use lots of CPU before you run the script.

Listing 101. Running count1.sh

```
[ian@echidna ian]$ sh count1.sh 10000 A
A Mon Nov 14 07:14:04 EST 2005
A Mon Nov 14 07:14:05 EST 2005
[ian@echidna ian]$ sh count1.sh 99000 A
A Mon Nov 14 07:14:26 EST 2005
A Mon Nov 14 07:14:32 EST 2005
```

So far, so good. Now let's use some of the other things we learned in this tutorial to create a list of commands so we can run the script in background and launch the

`top` command to see how much CPU the script is using. The command list is shown in Listing 102 and the output from `top` in Listing 103.

Listing 102. Running `count1.sh` and `top`

```
[ian@echidna ian]$ (sh count1.sh 99000 A&);top
```

Listing 103. Using lots of CPU

```
7:20am up 48 days, 20:54, 2 users, load average: 0.05, 0.05, 0.00
91 processes: 88 sleeping, 3 running, 0 zombie, 0 stopped
CPU states: 0.1% user, 0.0% system, 0.0% nice, 0.9% idle
Mem: 1030268K av, 1002864K used, 27404K free, 0K shrd, 240336K buff
Swap: 1052216K av, 118500K used, 933716K free, 605152K cached

  PID USER      PRI  NI  SIZE  RSS SHARE STAT  %CPU %MEM    TIME COMMAND
 8684 ian        20   0  1044  1044   932 R    98.4  0.1   0:01 sh
```

Not bad. We are using 98.4% of the CPU with a simple script.

Displaying and setting priorities

If we had a long running job such as this, we might find that it interfered with our ability (or the ability of other users) to do other work on our system. Linux and UNIX systems use a priority system with 40 priorities, ranging from -20 (highest priority) to 19 (lowest priority).

`nice`

Processes started by regular users usually have priority 0. The `nice` command displays our default priority. The `ps` command can also display the priority (nice, or NI, level), for example using the `-l` option. We illustrate this in Listing 104, where we have highlighted the nice value of 0.

Listing 104. Displaying priority information

```
[ian@echidna ian]$ nice
0
[ian@echidna ian]$ ps -l
  F S  UID  PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  TTY          TIME CMD
000 S  500  7283  7282  0  70   0  -   1103 wait4 pts/2    00:00:00 bash
000 R  500  9578  7283  0  72   0  -    784 -      pts/2    00:00:00 ps
```

The `nice` command can also be used to start a process with a different priority. You use the `-n` or (`--adjustment`) option with a positive value to increase the priority value and a negative value to decrease it. Remember that processes with the lowest priority value run at highest scheduling priority, so think of increasing the priority

value as being *nice* to other processes. Note that you usually need to be the superuser (root) to specify negative priority adjustments. In other words, regular users can usually only make their processes nicer. In Listing 105, we run two copies of the `count1.sh` script in background with different scheduling priorities. Notice that there is about a 5 second gap between the completion times. Try experimenting with different `nice` values, or by running the first process with a priority adjustment instead of the second one to demonstrate the different possibilities for yourself.

Listing 105. Using `nice` to set priorities

```
[ian@echidna ian]$ (sh count1.sh 99000 A&);\
> (nice -n 19 sh count1.sh 99000 B&);\
> sleep 2;ps -l;sleep 20
B Mon Nov 14 08:17:36 EST 2005
A Mon Nov 14 08:17:36 EST 2005
  F S  UID  PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  TTY          TIME CMD
000 S  500  7283  7282  0  70   0  -   1104 wait4  pts/2    00:00:00 bash
000 R  500  10765    1  84  80   0  -   1033 -      pts/2    00:00:01 sh
000 R  500  10767    1  14  79  19  -   1033 -      pts/2    00:00:00 sh
000 R  500  10771  7283  0  72   0  -    784 -      pts/2    00:00:00 ps
A Mon Nov 14 08:17:43 EST 2005
B Mon Nov 14 08:17:48 EST 2005
```

Note that, as with the `nohup` command, you cannot use a command list or a pipeline as the argument of `nice`.

Changing priorities

`renice`

If you happen to start a process and realize that it should run at a different priority, there is a way to change it after it has started, using the `renice` command. You specify an absolute priority (and not an adjustment) for the process or processes to be changed as shown in Listing 106.

Listing 106. Using `renice` to change priorities

```
[ian@echidna ian]$ sh count1.sh 299000 A&
[1] 11322
[ian@echidna ian]$ A Mon Nov 14 08:30:29 EST 2005

[ian@echidna ian]$ renice +1 11322;ps -l
11322: old priority 0, new priority 1
  F S  UID  PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  TTY          TIME CMD
000 S  500  7283  7282  0  75   0  -   1104 wait4  pts/2    00:00:00 bash
000 R  500  11322  7283  96  77   1  -   1032 -      pts/2    00:00:11 sh
000 R  500  11331  7283  0  76   0  -    786 -      pts/2    00:00:00 ps
[ian@echidna ian]$ renice +3 11322;ps -l
11322: old priority 1, new priority 3
  F S  UID  PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  TTY          TIME CMD
000 S  500  7283  7282  0  75   0  -   1104 wait4  pts/2    00:00:00 bash
000 R  500  11322  7283  93  76   3  -   1032 -      pts/2    00:00:16 sh
000 R  500  11339  7283  0  76   0  -    785 -      pts/2    00:00:00 ps
```

You can find more information on `nice` and `renice` in the man pages.

Section 8. Searching with regular expressions

This section covers material for topic 1.103.7 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn about the following topics:

- Regular expressions
- Searching files and filesystems using regular expressions
- Using regular expressions with `sed`

Regular expressions

Regular expressions have their roots in computer language theory. Most students of computer science learn that the languages that can be denoted by regular expressions are precisely the same as those accepted by finite automata. The regular expressions covered in this section are capable of expressing more complexity and so are **not** the same as those you may have learned about in computer science classes, although the heritage is clear.

A regular expression (also called a "regex" or "regexp") is a way of describing a text string or *pattern* so that a program can *match* the pattern against arbitrary text strings, providing an extremely powerful search capability. The `grep` (for *generalized regular expression processor*) is a standard part of any Linux or UNIX programmer's or administrator's toolbox, allowing regular expressions to be used in searching files, or command output. In the section on [text streams and filters](#), we introduced `sed`, the stream editor, which is another standard tool that uses regular expressions extensively for finding and replacing text in files or text streams. This section helps you better understand the regular expressions used by both `grep` and `sed`. Another program that uses regular expression extensively is `awk`, which is part of the material for exam 201 for LPIC-2 certification.

As with other parts of this tutorial series, whole books have been written on regular expressions and computer language theory. See [Resources](#) for some suggestions.

As you learn about regular expressions, you will see similarities between regular

expression syntax and the wildcard (or globbing) syntax discussed under [Wildcards and globbing](#). The similarity is only superficial.

Basic building blocks

Two forms of regular expression syntax are used with the GNU `grep` program found on most Linux systems: *basic* and *extended*. With GNU `grep`, there is no difference in functionality. Basic syntax is described here, along with the differences between it and extended syntax.

Regular expressions are built from *characters* and *operators*, augmented by *metacharacters*. Most characters match themselves, and most metacharacters must be escaped using a backslash (`\`). The fundamental operations are

Concatenation

Concatenating two regular expressions creates a longer expression. For example, the regular expression `a` will match the string `abcdcba` twice (the first and last `a`) and so will the regular expression `b`. However, `ab` will only match `abcdcba`, while `ba` will only match `abcdcba`.

Repetition

The Kleene `*` or repetition operator will match zero or more occurrences of the preceding regular expression. Thus an expression like `a*b` will match any string of `a`'s terminated by a `b`, including just `b` itself. The Kleene `*` does not have to be escaped, so an expression in which you want to match a literal asterisk (`*`) must have the asterisk escaped.

Alternation

The alternation operator (`|`) matches either the preceding or following expression. It must be escaped in basic syntax. For example, the expression `a*\|b*c` will match a string consisting of any number of `a`'s or any number of `b`'s (but not both) terminated by a single `c`. Again, the single character `c` would match.

You often need to quote your regular expressions to avoid shell expansion.

We will use the text files that we created earlier in the `lpi103` directory as examples. Study the simple examples of Listing 107. Note that `grep` takes a regular expression as a required parameter and a list of zero or more files to search. If no files are given, `grep` searches `stdin`, which makes it a filter that can be used in pipelines. If no lines match, there is no output from `grep`, although its exit code can be tested.

Listing 107. Simple regular expressions

```
[ian@echidna lpi103]$ grep p text1
1 apple
```

```

2 pear
[ian@echidna lpil03]$ grep pea text1
2 pear
[ian@echidna lpil03]$ grep "p*" text1
1 apple
2 pear
3 banana
[ian@echidna lpil03]$ grep "pp*" text1
1 apple
2 pear
[ian@echidna lpil03]$ grep "x" text1
[ian@echidna lpil03]$ grep "x*" text1
1 apple
2 pear
3 banana
[ian@echidna lpil03]$ cat text1 | grep "l\\|n"
1 apple
3 banana
[ian@echidna lpil03]$ echo -e "find an\\n* here" | grep "\\*"
* here

```

As you can see from these examples, you may sometimes get surprising results, particularly with repetition. You may have expected **p*** or at least **pp*** to match a couple of **p**'s, but **p***, and **x*** too, match every line in the file because the ***** operator matches **zero** or more of the preceding regular expression.

First shortcuts

Now that you have the basic building blocks of regular expressions, let's look at a few convenient shortcuts.

+

The **+** operator is like the ***** operator, except that it matches **one** or more occurrences of the preceding regular expression. It must be escaped for basic expressions.

?

The **?** indicates that the preceding expression is optional, so it represents zero or one occurrences of it. This is not the same as the **?** used in globbing.

.

The **.** (dot) is a metacharacter that stands for any character. One of the most commonly used patterns is **.***, which matches an arbitrary length string containing any characters (or no characters at all). Needless to say, you will find this used as part of a longer expression. Compare a single dot with the **?** used in globbing and **.*** with the ***** used in globbing.

Listing 108. More regular expressions

```

[ian@echidna lpil03]$ grep "pp\+" text1 # at least two p's
1 apple
[ian@echidna lpil03]$ grep "pl\?e" text1
1 apple

```

```

2 pear
[ian@echidna lpi103]$ grep "pl\?e" text1 # pe with optional l between
1 apple
2 pear
[ian@echidna lpi103]$ grep "p.*r" text1 # p, some string then r
2 pear
[ian@echidna lpi103]$ grep "a.." text1 # a followed by two other letters
1 apple
3 banana

```

Matching beginning or end of line

The `^` (caret) matches the beginning of a line while the `$` (dollar sign) matches the end of line. So `^..b` matches any two characters at the beginning of a line followed by a **b**, while `ar$` matches any line ending in **ar**. The regular expression `^$` matches an empty line.

More complex expressions

So far, we have seen repetition applied to a single character. If you wanted to search for one or more occurrences of a multicharacter string such as the **an** that occurs twice in **banana**, use parentheses, which must be escaped in basic syntax. Similarly, you might want to search for a few characters, without using something as general as the `.` or as long-winded as alternation. You can do this too, by enclosing the alternatives in square brackets (`[]`), which do not need to be escaped for regular syntax. Expressions in square brackets constitute a *character class*. With a few exceptions covered later, using square brackets also eliminates the need for escaping special characters such as `.` and `*`.

Listing 109. Parentheses and character classes

```

[ian@echidna lpi103]$ grep "\(an\)\+" text1 # find at least 1 an
3 banana
[ian@echidna lpi103]$ grep "an\(an\)\+" text1 # find at least 2 an's
3 banana
[ian@echidna lpi103]$ grep "[3p]" text1 # find p or 3
1 apple
2 pear
3 banana
[ian@echidna lpi103]$ echo -e "find an\n* here\nsomewhere." | grep "[.*]"
* here
somewhere.

```

There are several other interesting possibilities for character classes.

Range expression

A range expression is two characters separated by a `-` (hyphen), such as `0-9` for digits, or `0-9a-fA-F` for hexadecimal digits. Note that ranges are locale-dependent.

Named classes

Several named classes provide a convenient shorthand for commonly used

classes. Named classes open with [: and close with :]. Some examples:

[:alnum:]

Alphanumeric characters

[:blank:]

Space and tab characters

[:digit:]

The digits 0 through 9 (equivalent to 0-9)

[:upper:] and [:lower:]

Upper and lower case letters respectively.

^ (negation)

When used as the first character in a square brackets, the ^ (caret) negates the sense of the remaining characters so the match occurs only if a character (except the leading ^) is **not in the class**.

Given the special meanings above, if you want to match a literal - (hyphen) within a character class you must put it first or last. If you want to match a literal ^ (caret) it must not be first. And a] (right square bracket) closes the class, unless it is placed first.

Character classes are one area where regular expressions and globbing **are** similar, although the negation differs (^ vs. !). Listing 108 shows some examples of character classes.

Listing 110. More character classes

```
[ian@echidna lpi103]$ # Match on range 3 through 7
[ian@echidna lpi103]$ echo -e "123\n456\n789\n0" | grep "[3-7]"
123
456
789
[ian@echidna lpi103]$ # Find digit followed by no n or r till end of line
[ian@echidna lpi103]$ grep "[[:digit:]][^nr]*$" text1
1 apple
```

Using regular expressions with sed

The brief introduction to [Sed](#) mentioned that sed uses regular expressions. Regexp's can be used in address expressions as well as in substitution expressions. So the expression `/abc/s/xyz/XYZ/g` means to apply the substitution command that will substitute XYZ for every occurrence of xyz **only** to lines that contain abc. Listing 1111 shows two examples with our text1 file and one where we change the last word before a period (.) to the string LAST WORD. Notice that the string First was not

changed as it was not preceded by a blank.

Listing 111. Regular expressions in sed

```
[ian@echidna lpil03]$ sed -e '\(a.*a\)|\(p.*p\) /s/a/A/g' text1
1 Apple
2 pear
3 bAnAnA
[ian@echidna lpil03]$ sed -e '^[^lmnXYZ]*$/s/ear/each/g' text1
1 apple
2 peach
3 banana
[ian@echidna lpil03]$ echo "First. A phrase. This is a sentence." | \
> sed -e 's/ [^ ]*\./ LAST WORD./g'
First. A LAST WORD. This is a LAST WORD.
```

Extended regular expressions

Extended regular expression syntax eliminates the need to escape several characters when used as we have used them in basic syntax, including parentheses, '?', '+', '|', and '{'. This means that they must be escaped if you want them interpreted as characters. You may use the `-E` (or `--extended-regexp` option of `grep` to indicate that you are using extended regular expression syntax. Alternatively, the `egrep` command does this for you. Some older versions of `sed` do not support extended regular expressions. If your version of `sed` does support extended regexps, use the `-r` option to tell `sed` that you are using extended syntax. Listing 112 shows an example used earlier in this section along with the corresponding extended expression used with `egrep`.

Listing 112. Extended regular expressions

```
[ian@echidna lpil03]$ grep "an\(an\)+" text1 # find at least 2 an's
3 banana
[ian@echidna lpil03]$ egrep "an(an)+" text1 # find at least 2 an's
3 banana
```

Finding stuff in files

This section wraps up by giving you some taste of the power of `grep` and `find` to hunt down things in your filesystem. Again, the examples are relatively simple; they use the files created in the `lpil03` directory and its children.

First, `grep` can search multiple files at once. If you add the `-n` option, it tells you what line numbers matched. If you simply want to know how many lines matched, use the `-c` option, and if you want just a list of files with matches, use the `-l` option. Listing 113 shows some examples.

Listing 113. Grepping multiple files

```
[ian@echidna lpil03]$ grep plum *
text2:9 plum
text6:9 plum
text6:9 plum
yaa:9 plum
[ian@echidna lpil03]$ grep -n banana text[1-4]
text1:3:3 banana
text2:2:3 banana
[ian@echidna lpil03]$ grep -c banana text[1-4]
text1:1
text2:1
text3:0
text4:0
[ian@echidna lpil03]$ grep -l pear *
ex-here.sh
nohup.out
text1
text5
text6
xaa
```

The final example uses `find` to locate all the regular file in the current directory and its children and then `xargs` to pass the file list to `grep` to determine the number of occurrences of **banana** in each file. Finally, this output is filtered through another invocation of `grep`, this time with the `-v` option to find all files that do **not** have zero occurrences of the search string.

Listing 114. Hunting down files with at least one banana

```
[ian@echidna lpil03]$ find . -type f -print0 | xargs -0 grep -c banana | grep -v ":0$"
./text1:1
./text2:1
./xab:1
./yaa:1
./text5:1
./text6:4
./backup/text1.bkp.2:1
./backup/text1.bkp.1:1
```

This section has only scratched the surface of what you can do with the Linux command line and regular expressions. Use the man pages to learn more about these invaluable tools.

Section 9. File editing with vi

This section covers material for topic 1.103.8 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

In this section, you learn about the following topic:

- Editing text with vi

Using vi

The vi editor is almost certainly on every Linux and UNIX system. In fact, if a system has just one editor, it's probably vi, so it's worth knowing your way around in vi. This section introduces you to some basic vi editing commands, but for a full vi tutorial, check out our "vi intro -- the cheat sheet method" (see [Resources](#)), or consult the man pages or one of the many excellent books that are available.

Starting vi

Most Linux distributions now ship with the vim (for **Vi IMproved**) editor rather than classic vi. Vim is upward compatible with vi and has a graphical mode available (gvim) as well as the standard vi text mode interface. The `vi` command is usually an alias or symbolic link to the vim program. Refer back to the topic [Where does the shell find commands?](#) to learn exactly what command is used.

If you recall the section on [changing priorities](#), we wanted to change the priority of our running `count1.sh` shell script. Perhaps you tried this yourself and found that the command ran so fast that you didn't have enough time to accomplish the priority change with `renice`. So let's start by using the vi editor to add a line at the beginning of the file to sleep for 20 seconds so we have some time to change priorities.

To start the vi editor, use the `vi` command with a filename as a parameter. You have many options to choose from, so see the man pages for details. Use the command

```
vi count1.sh
```

You should see a display similar to Listing 115. If you are using vim, some of the words may be in other colors. Vim has a syntax highlighting mode (which was not part of the original vi editor), and it may be turned on by default on your system.

Listing 115. Editing count1.sh using vi

```
x="$1"
echo "$2" $(date)
while [ $x -gt 0 ]; do let x=$x-1;done
echo "$2" $(date)
~
~
~
~
~
~
"count1.sh" 4L, 82C
```

vi modes

The vi editor has two modes of operation:

Command mode

In command mode, you move around the file and perform editing operations such as searching for text, deleting text, changing text, and so on. You usually start in command mode.

Insert mode

In insert mode, you type new text into the file at the insertion point. To return to command mode, press the **Esc** (Escape) key.

These two modes determine way the editor behaves. Vi dates from the time when not all terminal keyboards had cursor movement keys, so everything you can do in vi can be done with the keys typically found on a standard typewriter plus a couple of keys such as **Esc** and **Insert**. However, you can configure vi to use additional keys if they are available; most of the keys on your keyboard do something useful in vi. Because of this legacy and the slow nature of early terminal connections, vi has a well-deserved reputation for using very brief and cryptic commands.

Getting out of vi

One of the first things I like to learn about a new editor is how to get out of it before I do anything I shouldn't have done. The following ways to get out of vi include saving or abandoning your changes, or restarting from the beginning. If these commands don't seem to work for you, you may be in insert mode, so press **Esc** to leave insert mode and return to command mode.

:q!

Quit editing the file and abandon all changes. This is a very common idiom for getting out of trouble.

:w!

Write the file (whether modified or not). Attempt to overwrite existing files or read-only or other unwriteable files. You may give a filename as a parameter, and that file will be written instead of the one you started with. It's generally safer to omit the ! unless you know what you're doing here.

ZZ

Write the file if it has been modified. Then exit. This is a very common idiom for normal vi exit.

:e!

Edit the current disk copy of the file. This will reload the file, abandoning

changes you have made. You may also use this if the disk copy has changed for some other reason and you want the latest version.

:!

Run a shell command. Type the command and press **Enter**. When the command completes, you will see the output and a prompt to return to vi editing.

Notes:

1. When you type the colon (:), your cursor will move to the bottom line of your screen where you can type in the command and any parameters.
2. If you omit the exclamation point from the above commands, you may receive an error message such as one saying changes have not been saved, or the output file cannot be written (for example, you are editing a read-only file).
3. The : commands have longer forms (:quit, :write, :edit), but the longer forms are seldom used.

Moving around

The following commands help you move around in a file:

h

Move left one character on the current line

j

Move down to the next line

k

Move up to the previous line

l

Move right one character on the current line

w

Move to the next word on the current line

e

Move to the next end of word on the current line

b

Move to the previous beginning of the word on the current line

Ctrl-f

Scroll forward one page

Ctrl-b

Scroll backward one page

If you type a number before any of these commands, then the command will be executed that many times. This number is called a *repetition count* or simply *count*. For example, 5h will move left five characters. You can use repetition counts with many vi commands.

Moving to lines

The following commands help you move to specific lines in your file:

G

Moves to a specific line in your file. For example, 3G moves to line 3. With no parameter, G moves to the last line of the file.

H

Moves relative to the top line on the screen. For example, 3H moves to the line currently 3rd from the top of your screen.

L

Is like H, except that movement is relative to the last line on screen. Thus, 2L moves to the second to last line on your screen.

Searching

You can search for text in your file using regular expressions:

/

Use / followed by a regular expression to search forward in your file.

?

Use ? followed by a regular expression to search backward in your file.

n

Use n to repeat the last search in either direction.

You may precede any of the above search commands with a number indicating a repetition count. So 3/x will find the third occurrence of x from the current point, as will /x followed by 2n.

Modifying text

Use the following commands if you need to insert, delete, or modify text:

i

Enter insert mode before the character at the current position. Type your text and press **Esc** to return to command mode. Use **I** to insert at the beginning of the current line.

a

Enter insert mode after the character at the current position. Type your text and press **Esc** to return to command mode. Use **A** to insert at the end of the current line.

c

Use **c** to change the current character and enter insert mode to type replacement characters.

o

Open a new line for text insertion below the current line. Use **O** to open a line above the current line.

cw

Delete the remainder of the current word and enter insert mode to replace it. Use a repetition count to replace multiple words. Use **c\$** to replace to end of line.

dw

As for **cw** (and **c\$**) above, except that insert mode is not entered.

dd

Delete the current line. Use a repetition count to delete multiple lines.

x

Delete the character at the cursor position. Use a repetition count to delete multiple characters.

p

Put the last deleted text after the current character. Use **P** to put it before the current character.

xp

This combination of **x** and **p** is a useful idiom. This swaps the character at the cursor position with the one on its right.

Putting it together

We set out to add a line to our `count1.sh` file. To keep the original and save the modified version as `count2.sh`, we could use these `vi` commands once we open the file with `vi`. Note that `<Esc>` means to press the **Esc** key.

Listing 116. Editor commands to add a line to count1.sh

```
1G
O
sleep 20<Esc>
:w! count2.sh
:q
```

Simple when you know how.

The next tutorial in this series covers [Topic 104 on Devices, Linux filesystems, and the Filesystem Hierarchy Standard \(FHS\)](#).

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- In the three-part series "[Sed by example](#)" (developerWorks, October and November 2000), Daniel Robbins shows you how to use the powerful (but often forgotten) UNIX stream editor, sed. Sed is an ideal tool for batch-editing files or for creating shell scripts to modify existing files in powerful ways.
- [sed . . . the stream editor](#) is a useful site maintained by Eric Pement with lots of sed links, a sed FAQ, and a handy set of sed one-liners.
- In our tutorial on using vi, "[vi intro -- the cheat sheet method](#)" (developerWorks, December 2000), Daniel Robbins shows you how to use the vi editor to edit text, use insert mode, copy and paste text, and use important vim extensions like visual mode and multi-window editing.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- In the [Linux Man Page Howto](#), learn how man pages work.
- Visit the home of the [Filesystem Hierarchy Standard](#) (FHS).
- At the [LSB home page](#), learn about The Linux Standard Base (LSB), a Free Standards Group (FSG) project to develop a standard binary operating environment.
- *[Introduction to Automata Theory, Languages, and Computation \(2nd Edition\)](#)* (Addison-Wesley, 2001) is a good source of information on regular expressions and finite state machines.
- *[Mastering Regular Expressions, Second Edition](#)* (O'Reilly Media, Inc., 2002) covers regular expressions as used in grep, sed, and other programming environments.
- *[LPI Linux Certification in a Nutshell](#)* (O'Reilly, 2001) and *[LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#)* (Que, 2004) are references for those who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content.](#)
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard

Junior Level Administration (LPIC-1) topic 104

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

28 Dec 2005

Updated 16 Apr 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 101. In this fourth in a [series of five tutorials](#), Ian introduces you to Linux® devices, filesystems, and the Filesystem Hierarchy Standard. By the end of this tutorial, you will know how to create and format partitions with different Linux filesystems and how to manage and maintain those systems.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 101, the five topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 101: Tutorials and topics		
LPI exam 101 topic	developerWorks tutorial	Tutorial summary
Topic 101	LPI exam 101 prep (topic 101): Hardware and architecture	Learn to configure your system hardware with Linux. By the end of this tutorial, you will know how Linux configures the hardware found on a modern PC and where to look if you have problems.
Topic 102	LPI exam 101 prep: Linux installation and package management	Get an introduction to Linux installation and package management. By the end of this tutorial, you will know how Linux uses disk partitions, how Linux boots, and how to install and manage software packages.
Topic 103	LPI exam 101 prep: GNU and UNIX commands	Get an introduction to common GNU and UNIX commands. By the end of this tutorial, you will know how to use commands in the bash shell, including how to use text processing commands and filters, how to search files and directories, and how to manage processes.
Topic 104	LPI exam 104 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard.	(This tutorial). Learn how to create filesystems on disk partitions, as well as how to make them accessible to users, manage file ownership and user quotas, and repair filesystems as needed. Also learn about hard and symbolic links, and how to locate files in your filesystem and where files should be placed. See detailed objectives below.
Topic 110	LPI exam 110 prep: The X Window system	Learn about the X Window System on Linux. By the end of this tutorial, you will know how to install and maintain the X Window System. This tutorial covers both major

packages for X on Linux:
XFree86 and X.Org.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Devices, Linux filesystems, and the Filesystem Hierarchy Standard", the fourth of five tutorials designed to prepare you for LPI exam 101. In this tutorial, you learn how to create and manage partitions. You also learn about the *Filesystem Hierarchy Standard (FHS)*, which recommends where different types of data can be found and should be stored on a Linux system.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Devices, Linux filesystems, and the Filesystem Hierarchy Standard: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
1.104.1 Create partitions and filesystems	Weight 3	Learn to configure disk partitions and create filesystems on media such as hard disks. Use various <code>mkfs</code> commands to set up filesystems, including ext2, ext3, reiserfs, vfat, and xfs.
1.104.2 Maintain the integrity of filesystems	Weight 3	Learn to verify the integrity of filesystems, monitor free space and inodes, and repair simple filesystem problems. Learn to maintain standard filesystems and journaling

		filesystems.
1.104.3 Mount and unmount filesystems	Weight 3	Learn to mount and unmount filesystems manually. Also learn to configure filesystem mounting on bootup, and configure removable filesystems, such as tape drives, floppies, and CDs, so that ordinary users can mount and unmount them.
1.104.4 Manage disk quota	Weight 5	Learn to manage disk quotas for users, including setting up quotas for a filesystem, and editing, checking, and generating user quota reports.
1.104.5 Use file permissions to control access to files	Weight 5	Learn to control file access through permissions, including access permissions on regular and special files as well as directories. Also learn about access modes such as suid, sgid, and the sticky bit; the use of the group field to grant file access to workgroups; the immutable flag; and the default file creation mode.
1.104.6 Manage file ownership	Weight 1	Learn to control user and group ownership of files, including how to change the user and group owner of a file as well as the default group owner for new files.
1.104.7 Create and change hard and symbolic links	Weight 1	Learn to create and manage hard and symbolic links to a file, including creating and identifying links. Learn to copy files through links, and use linked files to support system administration tasks.
1.104.8 Find system files and place files in the correct location	Weight 5	Learn about the Filesystem Hierarchy Standard, including typical file locations and directory classifications. Learn to find files and commands on a Linux system.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

This tutorial builds on content covered in the previous three tutorials in this series, so you may want to first review the [tutorials for topics 101, 102, and 103](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Creating partitions and filesystems

This section covers material for topic 1.104.1 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn how to:

- Configure disk partitions
- Create filesystems on hard disks and other media
- Use `mkfs` commands to set up ext2, ext3, reiserfs, vfat, and xfs partitions

First, a quick review. In the tutorial for topic 101, "[LPI exam 101 prep \(topic 101\): Hardware and architecture](#)," you learned about IDE and SCSI hard drives such as `/dev/hda` and `/dev/sdb`, and partitions on these drives, such as `/dev/hda1`, `/dev/hda5` and `/dev/sda1`.

In the tutorial for topic 102, "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)," you learned more about partitions, including *primary*, *extended*, and *logical* partitions. You also learned that a Linux filesystem contains *files* that are arranged on a disk or other *block storage device* in *directories*. As with many other systems, directories on a Linux system may contain other directories called *subdirectories*. That tutorial also discussed the considerations that guide you in making choices about partitioning.

This section reviews block devices and partitions, and then introduces you to the `fdisk` command, which is used to create, modify, or delete partitions on block devices. It also introduces the various forms of the `mkfs` command (`mkfs` stands for *make filesystem*); these commands are used to format partitions as a particular filesystem type.

Note: In addition to the tools and filesystems required for the LPI exams, you may

encounter or need other tools and filesystems. Find a brief summary of some other available tools in [Other tools and filesystems](#).

Block devices and partitions

Let's quickly review block devices and partitions. If you need more information, refer back to the tutorials for [topic 101](#) and [topic 102](#).

Block devices

A *block device* is an abstraction layer for any storage device that can be formatted in fixed-size *blocks*; individual blocks may be accessed independently of access to other blocks. Such access is often called *random access*.

The abstraction layer of randomly accessible fixed-size blocks allows programs to use these block devices without worrying about whether the underlying device is a hard drive, floppy, CD, network drive, or some type of virtual device such as an in-memory file system.

Examples of block devices include the first IDE hard drive on your system (`/dev/hda`) or the second SCSI drive (`/dev/sdb`). Use the `ls -l` command to display `/dev` entries. The first character on each output line is **b** for a **block** device, such as floppy, CD drive, IDE hard drive, or SCSI hard drive; and **c** for a **character** device, such as a tape drive or terminal. See the examples in Listing 1.

Listing 1. Linux block and character devices

```
[ian@lyrebird ian]$ ls -l /dev/fd0 /dev/hda /dev/sdb /dev/st0 /dev/tty0
brw-rw---- 1 ian floppy 2, 0 Jun 24 2004 /dev/fd0
brw-rw---- 1 root disk 3, 0 Jun 24 2004 /dev/hda
brw-rw---- 1 root disk 8, 16 Jun 24 2004 /dev/sdb
crw-rw---- 1 root disk 9, 0 Jun 24 2004 /dev/st0
crw--w---- 1 root root 4, 0 Jun 24 2004 /dev/tty0
```

Partitions

For some block devices, such as floppy disks and CD or DVD discs, it is common to use the whole media as a single filesystem. However, with large hard drives, and even with smaller USB memory keys, it is more common to divide, or partition, the available space into several different *partitions*.

Partitions can be different sizes, and different partitions may have different filesystems on them, so a single disk can be used for many purposes, including sharing it between multiple operating systems. For example, I use test systems with several different Linux distributions and sometimes a Windows® system, all sharing one or two hard drives.

You will recall from the 101 and 102 tutorials that hard drives have a *geometry*, defined in terms of cylinders, heads, and sectors. Even though modern drives use *logical block addressing (LBA)*, which renders geometry largely irrelevant, the fundamental allocation unit for partitioning purposes is still the cylinder.

Displaying partition information

Partition information is stored in a *partition table* on the disk. The table lists information about the start and end of each partition, information about its *type*, and whether it is marked bootable or not. To create and delete partitions, you edit the partition table using a program specially designed for the job. For the LPI exam, you need to know about the `fdisk` program, so that is what is covered here, although several other tools exist.

The `fdisk` command with the `-l` option is used to list partitions. Add a device name, such as `/dev/hda`, if you want to look at the partitions on a particular drive. Note that partitioning tools require root access. Listing 2 shows the partitions on one of my hard drives.

Listing 2. Listing partitions with fdisk

```
[root@lyrebird root]# fdisk -l /dev/hda

Disk /dev/hda: 160.0 GB, 160041885696 bytes
255 heads, 63 sectors/track, 19457 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1    *           1         2078    16691503+   7  HPFS/NTFS
/dev/hda2                2079         3295     9775552+   c  Win95 FAT32 (LBA)
/dev/hda3                3296         3422     1020127+  83  Linux
/dev/hda4                3423        19457   128801137+  f  Win95 Ext 'd (LBA)
/dev/hda5                3423         3684     2104483+  82  Linux swap
/dev/hda6                3685         6234    20482843+  83  Linux
/dev/hda7                6235         7605     11012526   83  Linux
/dev/hda8                7606         9645    16386268+  83  Linux
/dev/hda9                9646        12111    19808113+  83  Linux
/dev/hda10           12112        15680    28667961   83  Linux
/dev/hda11           15681        19457    30338721   83  Linux
```

Notes:

1. The header information shows the disk size and geometry. Most large disks using LBA have 255 heads per cylinder and 63 sectors per track, making a total of 16065 sectors, or 8225280 bytes per cylinder.
2. In this example, the first primary partition (`/dev/hda1`) is marked *bootable* (or *active*). As you saw in the tutorial for topic 102, this enables the standard DOS PC master boot record to boot the partition. This flag has

no significance for the LILO or GRUB boot loaders.

3. The *Start* and *End* columns show the starting and ending cylinder for each partition. These must not overlap and should generally be contiguous, with no intervening space.
4. The *Blocks* column shows the number of 1K (1024 byte) blocks in the partition. The maximum number of blocks in a partition is therefore half of the product of the number of cylinders ($End + 1 - Start$) and the number of sectors per cylinder. A trailing + sign indicates that not all sectors in the partition are used.
5. The *Id* field indicates the intended use of the partition. Type 82 is a Linux swap partition, and type 83 is a Linux data partition. There are approximately 100 different partition types defined. This particular disk is shared between several operating systems, including Windows/XP, hence the presence of Windows NTFS (and FAT32) partitions.

Partitioning with fdisk

You have just seen how to display partition information using the `fdisk` command. This command also provides an interactive environment for editing the partition table to create or remove partitions.

Warnings

Before you start modifying partitions, there are some important things to remember. You risk **losing your existing data** if you do not follow these guidelines.

1. **Do not change partitions that are in use.** Plan your actions and execute them carefully.
2. **Know your tool.** The `fdisk` command does not commit any changes to your disk until you tell it to. Other tools, including `parted`, may commit changes as you go.
3. **Back up important data before you start,** as with any operation that may cause data loss.
4. Partitioning tools write the partition table. Unless the tool you are using also includes the ability to move, resize, format, or otherwise write to the data area of your disk, your data will not be touched. If you do make an accidental mistake, stop as quickly as possible and seek help. You may still be able to recover your partitions and data.

Start fdisk

To start `fdisk` in interactive mode, simply give the name of a disk, such as `/dev/hda` or `/dev/sdb`, as a parameter. The following example boots a Knoppix live CD. You will need root authority, and you will see output similar to Listing 3.

Listing 3. Starting interactive fdisk

```
root@ttypl[knoppix]# fdisk /dev/hda

The number of cylinders for this disk is set to 14593.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
 1) software that runs at boot time (e.g., old versions of LILO)
 2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):
```

Most modern disks have more than 1024 cylinders, so you will usually see the warning as shown in Listing 3. Type `m` to display a list of available one-letter commands as shown in Listing 4.

Listing 4. Help in fdisk

```
Command action
 a  toggle a bootable flag
 b  edit bsd disklabel
 c  toggle the dos compatibility flag
 d  delete a partition
 l  list known partition types
 m  print this menu
 n  add a new partition
 o  create a new empty DOS partition table
 p  print the partition table
 q  quit without saving changes
 s  create a new empty Sun disklabel
 t  change a partition's system id
 u  change display/entry units
 v  verify the partition table
 w  write table to disk and exit
 x  extra functionality (experts only)

Command (m for help):
```

Use the `p` command to display the existing partition on this particular disk; Listing 5 shows the output.

Listing 5. Displaying the existing partition table

```
Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1         2611     20972826   7   HPFS/NTFS

Command (m for help):
```

This particular disk is a 120GB disk with a Windows/XP partition of approximately 20GB. It is a primary partition, and it is marked bootable, as is typical for a Windows system.

Our workstation layout

Let's now use part of the free space to set up a simple workstation with the following additional partitions. In practice, it's unlikely that you would mix this many different filesystem types, but we'll do it here for illustration purposes.

1. Another primary partition for our boot files. This will be mounted as `/boot` and will contain kernel files and initial ramdisk files. If you use the GRUB boot loader, GRUB files will also be located here. From the tutorial for topic 102, our guideline is for approximately 100MB. We see from Listing 5 that a cylinder contains approximately 8MB of data, so we will use 13 cylinders for `/boot`. This will be `/dev/hda2`.
2. We will create an extended partition to house logical partitions spanning the rest of the free space. This will be `/dev/hda3`.
3. We will create a 500MB swap partition as `/dev/hda5`. We will use 64 cylinders for this.
4. We will create a logical partition of approximately 20GB for our Linux system. This will be `/dev/hda6`.
5. We will create a separate 10GB partition for user data. This will eventually be mounted as `/home`. For now, it will simply be `/dev/hda7`.
6. Finally, we will create a small 2GB partition for sharing data between the Linux and Windows systems. This will eventually be formatted as FAT32 (or `vfat`). This will be `/dev/hda8`.

Creating our partitions

Let's start by using the `n` command to create a new partition; see Listing 6.

Listing 6. Creating our first partition

```
Command (m for help): n
```

```

Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (2612-14593, default 2612):
Using default value 2612
Last cylinder or +size or +sizeM or +sizeK (2612-14593, default 14593): 2624

Command (m for help): p

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1    *           1         2611     20972826   7  HPFS/NTFS
/dev/hda2                2612        2624       104422+   83  Linux

Command (m for help):

```

We took the default for the first cylinder and specified the value of 2624 for the last cylinder, resulting in a 13-cylinder partition. You can see from Listing 6 that our partition is, indeed, approximately 100MB in size. Since it is a primary partition, it must be numbered from 1 through 4. It is a good idea to assign partition numbers sequentially; some tools complain if this is not done.

Notice also that our new partition was assigned a type of 83, for a Linux data partition. Think of this as an indicator to the operating system of the intended use of the partition. The eventual use should match this, but at this point we don't even have the partition formatted, let alone have any data on it.

We now create the extended partition, which is a container for the logical partitions on the disk. We assign partition number 3 (/dev/hda3). The interaction and result is shown in Listing 7. Note again that the partition type was assigned automatically.

Listing 7. Creating an extended partition

```

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
e
Partition number (1-4): 3
First cylinder (2625-14593, default 2625):
Using default value 2625
Last cylinder or +size or +sizeM or +sizeK (2625-14593, default 14593):
Using default value 14593

Command (m for help): p

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1    *           1         2611     20972826   7  HPFS/NTFS

```

```

/dev/hda2      2612      2624      104422+  83  Linux
/dev/hda3      2625      14593     96140992+  5  Extended

Command (m for help):

```

Now let's move on to creating a swap partition as a logical partition within our extended partition. We will use a value of +64 (cylinders) for the last cylinder, rather than performing the arithmetic ourselves. Note that this time we use the `t` command to assign a type of 82 (Linux swap) to the newly created partition. Otherwise, it would be another type 83 (Linux data) partition.

Listing 8. Creating a swap partition

```

Command (m for help): n
Command action
  l   logical (5 or over)
  p   primary partition (1-4)
l
First cylinder (2625-14593, default 2625):
Using default value 2625
Last cylinder or +size or +sizeM or +sizeK (2625-14593, default 14593): +64

Command (m for help): t
Partition number (1-5): 5
Hex code (type L to list codes): 82
Changed system type of partition 5 to 82 (Linux swap)

Command (m for help): p

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1    *           1         2611     20972826   7  HPFS/NTFS
/dev/hda2           2612         2624     104422+   83  Linux
/dev/hda3           2625        14593    96140992+   5  Extended
/dev/hda5           2625        2689     522081    82  Linux swap

Command (m for help):

```

Now let's define the main Linux partition and the `/home` partition. This time we will simply specify sizes of +20480M and +10240M, indicating 20GB and 10GB, respectively. We let `fdisk` calculate the number of cylinders for us. The results are shown in Listing 9.

Listing 9. Creating our main Linux partition

```

Command (m for help): n
Command action
  l   logical (5 or over)
  p   primary partition (1-4)
l
First cylinder (2690-14593, default 2690):
Using default value 2690
Last cylinder or +size or +sizeM or +sizeK (2690-14593, default 14593): +20480M

```

```

Command (m for help): n
Command action
  l   logical (5 or over)
  p   primary partition (1-4)
l
First cylinder (5181-14593, default 5181):
Using default value 5181
Last cylinder or +size or +sizeM or +sizeK (5181-14593, default 14593): +10240M

Command (m for help): p

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1    *           1         2611     20972826   7  HPFS/NTFS
/dev/hda2                2612         2624       104422+   83  Linux
/dev/hda3                2625        14593     96140992+   5  Extended
/dev/hda5                2625         2689       522081    82  Linux swap
/dev/hda6                2690         5180     20008926   83  Linux
/dev/hda7                5181         6426     10008463+   83  Linux

Command (m for help):

```

Our final partition is the FAT32 partition. We use the commands we have used above to create `/dev/hda8` using a size specification of `+2048M`, and then we change the partition type to `b` (for a W95 FAT32 partition). Next we save all our work.

Saving our partition table

So far, we have just been doing an in-memory edit of a partition table. We could use the `q` command to quit without saving changes. If something is not as we like, it, we can use the `d` command to delete one or more partitions so we can redefine them. If we are happy with our setup, we use the `v` command to verify our setup, and then the `w` command to write the new partition table and exit. See Listing 10. If you run `fdisk -l` again, you will see that Linux now knows about the new partitions. Unlike in some operating systems, it is not always necessary to reboot to see the changes. A reboot may be required if, for example, `/dev/hda3` became `/dev/hda2` because the original `/dev/hda2` was deleted. If a reboot is needed, `fdisk` should tell you to do so.

Listing 10. Saving the partition table

```

Command (m for help): v
127186915 unallocated sectors

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
root@tty0[knoppix]# fdisk -l /dev/hda

```

```
Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1	*	1	2611	20972826	7	HPFS/NTFS
/dev/hda2		2612	2624	104422+	83	Linux
/dev/hda3		2625	14593	96140992+	5	Extended
/dev/hda5		2625	2689	522081	82	Linux swap
/dev/hda6		2690	5180	20008926	83	Linux
/dev/hda7		5181	6426	10008463+	83	Linux
/dev/hda8		6427	6676	2008093+	b	W95 FAT32

More on fdisk

You may notice that we did not change the bootable flag on any partition. As our disk stands now, it still has the Windows Master Boot Record (MBR) and will therefore boot the first primary partition that is marked bootable (the NTFS partition in our example).

Neither LILO nor GRUB uses the bootable flag. If either of these is installed in the MBR, then it can boot the Windows/XP partition. You could also install LILO or GRUB into your /boot partition (/dev/hda2) and mark that partition bootable and remove the bootable flag from /dev/hda1. Leaving the original MBR can be useful if the machine is later returned to being a Windows-only machine.

You have now seen one way to set up a Linux workstation. Other choices you might make are covered later in this tutorial, under [Finding and placing system files](#).

Filesystem types

Linux supports several different filesystems. Each has strengths and weaknesses and its own set of performance characteristics. One important attribute of a filesystem is *journaling*, which allows for much faster recovery after a system crash. Generally, a journaling filesystem is preferred over a non-journaling one when you have a choice. Following is a brief summary of the types you need to know about for the LPI exam. See [Resources](#) for additional background information.

The ext2 filesystem

The ext2 filesystem (also known as the *second extended filesystem*) was developed to address shortcomings in the Minix filesystem used in early versions of Linux. It has been used extensively on Linux for many years. There is no journaling in ext2, and it has largely been replaced by ext3.

The ext3 filesystem

The ext3 filesystem adds journaling capability to a standard ext2 filesystem and is

therefore an evolutionary growth of a very stable filesystem. It offers reasonable performance under most conditions and is still being improved. Because it adds journaling on top of the proven ext2 filesystem, it is possible to convert an existing ext2 filesystem to ext3 and even convert back again if required.

The ReiserFS filesystem

ReiserFS is a B-tree-based filesystem that has very good overall performance, particularly for large numbers of small files. ReiserFS also scales well and has journaling.

The XFS filesystem

XFS is a filesystem with journaling. It comes with robust features and is optimized for scalability. XFS aggressively caches in-transit data in RAM, so an uninterruptible power supply is recommended if you use XFS.

The swap filesystem

Swap space must be formatted for use as swap space, but it is not generally considered a filesystem, otherwise.

The vfat filesystem

This filesystem (also known as *FAT32*) is not journaled and lacks many features required for a full Linux filesystem implementation. It is useful for exchanging data between Windows and Linux systems as it can be read by both Windows and Linux. Do **not** use this filesystem for Linux, except for sharing data between Windows and Linux. If you unzip or untar a Linux archive on a vfat disk, you will lose permissions, such as execute permission, and you will lose any symbolic links that may have been stored in the archive.

Both ext3 and ReiserFS are mature and used as the default filesystem on a number of distributions. These are recommended for general use.

Creating filesystems

Linux uses the `mkfs` command to create filesystems and `mkswap` command to make swap space. The `mkfs` command is actually a front end to several filesystem-specific commands such as `mkfs.ext3` for ext3 and `mkfs.reiserfs` for ReiserFS.

What filesystem support is already installed on your system? Use the `ls /sbin/mk*` command to find out. An example is shown in Listing 11.

Listing 11. Filesystem creation commands

```

root@tty0[knoppix]# ls /sbin/mk*
/sbin/mkdosfs      /sbin/mkfs.ext2    /sbin/mkfs.msdos   /sbin/mkraid
/sbin/mke2fs       /sbin/mkfs.ext3    /sbin/mkfs.reiserfs /sbin/mkreiserfs
/sbin/mkfs         /sbin/mkfs.jfs     /sbin/mkfs.vfat    /sbin/mkswap
/sbin/mkfs.cramfs  /sbin/mkfs.minix   /sbin/mkfs.xfs

```

You will notice various forms of some commands. For example, you will usually find that the files `mke2fs`, `mkfs.ext2` and `mkfs.ext3` are identical, as are `mkreiserfs` and `mkfs.reiserfs`.

There are a few common options for all `mkfs` commands. Options that are specific to the type of filesystem being created are passed to the appropriate creation command, based on the type of filesystem specified in the `-type` option. Our examples use `mkfs -type`, but you may use the other forms directly with equal effect. For example, you may use `mkfs -type reiserfs`, `mkreiserfs`, or `mkfs.reiserfs`. For the manual pages for a specific filesystem, use the appropriate `mkfs` command as the name, for example, `man mkfs.reiserfs`. Many of the values displayed in the output examples below can be controlled by options to `mkfs`.

Creating an ext3 filesystem

Listing 12. Creating an ext3 filesystem

```

root@tty0[knoppix]# mkfs -t ext3 /dev/hda8
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
251392 inodes, 502023 blocks
25101 blocks (5.00%) reserved for the super user
First data block=0
16 block groups
32768 blocks per group, 32768 fragments per group
15712 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 32 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.

```

A useful option for `ext2` and `ext3` filesystems is the `-L` option with a name, which assigns a label to the partition. This can be used instead of the device name when mounting filesystems; it provides some level of insulation against changes that may need to be reflected in various control files. To display or set a label for an existing `ext2` or `ext3` filesystem, use the `e2label` command. Labels are limited to a maximum size of 16 characters.

Note that a journal is created with ext3. If you wish to add a journal to an existing ext2 system, use the `tune2fs` command with the `-j` option.

Creating a ReiserFS filesystem

Listing 13. Creating a ReiserFS filesystem

```
.root@tty0[knoppix]# mkfs -t reiserfs /dev/hda6
mkfs.reiserfs 3.6.17 (2003 www.namesys.com)

A pair of credits:
Many persons came to www.namesys.com/support.html, and got a question answered
for $25, or just gave us a small donation there.

Jeremy Fitzhardinge wrote the teahash.c code for V3. Colin Plumb also
contributed to that.

Guessing about desired format. Kernel 2.4.26 is running.
Format 3.6 with standard journal
Count of blocks on the device: 5002224
Number of blocks consumed by mkreiserfs formatting process: 8364
Blocksize: 4096
Hash function used to sort names: "r5"
Journal Size 8193 blocks (first block 18)
Journal Max transaction length 1024
inode generation number: 0
UUID: 72e317d6-8d3a-45e1-bcda-ad7eff2b3b40
ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
          ALL DATA WILL BE LOST ON '/dev/hda6'!
Continue (y/n):y
Initializing journal - 0%....20%....40%....60%....80%....100%
Syncing..ok

Tell your friends to use a kernel based on 2.4.18 or later, and especially not a
kernel based on 2.4.9, when you use reiserFS. Have fun.

ReiserFS is successfully created on /dev/hda6.
```

You can label a ReiserFS system using the `-l` (or `--label` option with a name). You can use the `reiserfstune` command to add a label or display the label on an existing ReiserFS filesystem. Labels are limited to a maximum size of 16 characters.

Creating an XFS filesystem

Listing 14. Creating an XFS filesystem

```
root@tty0[knoppix]# mkfs -t xfs /dev/hda7
meta-data=/dev/hda7          isize=256      agcount=16, agsize=156382 blks
=                               sectsz=512
data      =                   bsize=4096   blocks=2502112, imaxpct=25
=                               sunit=0       swidth=0 blks, unwritten=1
naming    =version 2         bsize=4096
log       =internal log     bsize=4096   blocks=2560, version=1
=                               sectsz=512   sunit=0 blks
realtime  =none              extsz=65536  blocks=0, rtextents=0
```

You can label an XFS system using the `-L` option with a name. You can use the

`xfstool` command with the `-L` option to add a label to an existing XFS filesystem. Use the `-l` option of `xfstool` to display a label. Unlike `ext2`, `ext3` and `ReiserFS`, labels are limited to a maximum size of 12 characters.

Creating a vfat filesystem

Listing 15. Creating a vfat filesystem

```
root@tty0[knoppix]# mkfs -t vfat /dev/hda8
mkfs.vfat 2.10 (22 Sep 2003)
```

Label a FAT32 filesystems using the `-n` (for volume name) option. The `e2label` command will display or set the label on vfat as well as ext partitions. Labels on FAT32 are limited to a maximum size of 16 characters.

Creating swap space

Listing 16. Creating swap space

```
root@tty0[knoppix]# mkswap /dev/hda5
Setting up swap space version 1, size = 534605 kB
```

Unlike regular filesystems, swap partitions aren't mounted. Instead, they are enabled using the `swapon` command. Your Linux system's startup scripts will take care of automatically enabling your swap partitions.

Other tools and filesystems

The following tools and filesystems are not part of the LPI objectives for this exam. This very brief overview touches on some of the tools and filesystems that you may encounter.

Partitioning tools

Many Linux distributions include the `cedisk` and `sfdisk` commands. The `cedisk` command provides a more graphical interface than `fdisk`, using the `ncurses` library functions as shown in Figure 1. The `sfdisk` command is intended for programmer use and can be scripted. Use it only if you know what you are doing.

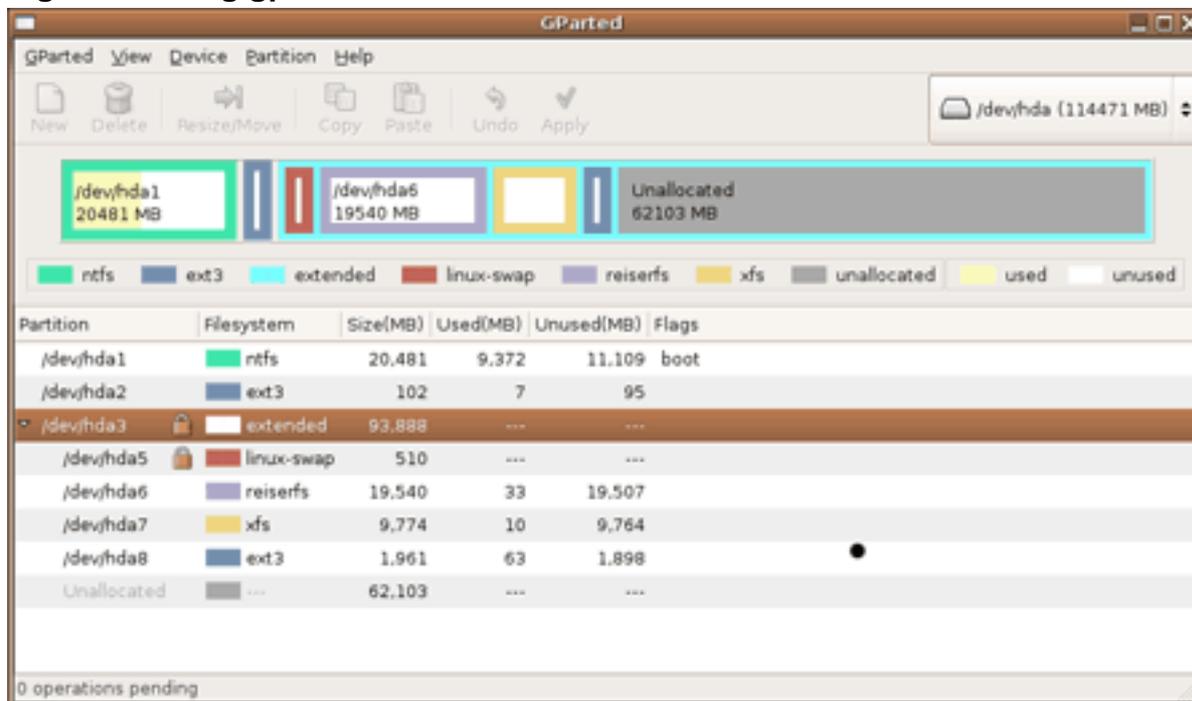
Figure 1. Using `cedisk`



Another popular tool for working with the partition table is parted, which can resize and format many partition types as well as create and destroy them. While parted cannot resize NTFS partitions, ntfsresize can. The gparted tool uses the Qt toolkit to provide a graphical interface. It includes the parted functions as well as ntfsresize functions.

The gparted tool is another graphical partitioning tool, designed for the GNOME desktop. It uses the GTK+GUI library, and is shown in Figure 2. (See Resources for links to both gparted and parted.)

Figure 2. Using gparted



Several commercial partitioning tools are available. Perhaps the best known one is

PartitionMagic, now sold by Symantec.

Many distributions allow you to partition your disk, and sometimes shrink an existing Windows NTFS or FAT32 partition, as part of the installation process. Consult the installation documentation for your distribution.

Logical volume manager

The logical volume manager (or LVM) for Linux allows you to combine multiple physical storage devices into a single *volume group*. For example, you might add a partition to an existing volume group, rather than having to carve out contiguous space large enough for your desired filesystem.

RAID

RAID (Redundant Array of Independent Disks) is a technology for providing a reliable data storage using low-cost disks that are much less expensive than the disks found on high-end systems. There are several different types of RAID, and RAID may be implemented in hardware or software. Linux supports both hardware and software RAID.

More filesystems

You will probably encounter filesystems besides those discussed above.

IBM's *Journalized File System (JFS)*, currently used in IBM enterprise servers, is designed for high-throughput server environments. It is available for Linux and is included in several distributions. To create JFS filesystems, use the `mkfs.jfs` command.

There are other filesystems too, such as the `cramfs` filesystem often used on embedded devices.

The next section shows you how to maintain integrity on filesystems and what to do when things go wrong.

Section 3. Filesystem integrity

This section covers material for topic 1.104.2 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn how to:

- Monitor free space and inodes
- Verify the integrity of filesystems
- Repair simple filesystem problems

Both standard and journaling filesystems are covered. The emphasis is on ext2 and ext3, but the tools for other filesystems are mentioned as well. Most of this material applies to both 2.4 and 2.6 kernels. The examples in this section mostly use Ubuntu 5.10 "Breezy Badger" (a distribution based on Debian), with a 2.6.12 kernel, which was installed on the filesystems we created in the previous section. Your results on other systems are likely to differ.

Monitoring free space

First, a review. In the tutorial for topic 103, "[LPI exam 101 prep: GNU and UNIX commands](#)," you learned that a file or directory is contained in a collection of *blocks*, and information about the file or directory is contained in an *inode*.

Both the data blocks and the inode blocks take space on filesystems, so you need to monitor the space usage to ensure that your filesystems have space for growth.

df

`df` displays information about mounted filesystems. (You will learn more about mounting filesystems in the next section, [Mounting and unmounting filesystems](#)). If you add the `-T` option, then the filesystem type is included in the display; otherwise, it is not. The output from `df` for the Ubuntu system that we installed on the filesystems created in the previous section is shown in Listing 17.

Listing 17. Displaying filesystem usage

```
ian@pinguino:~$ df -T
Filesystem      Type      1K-blocks      Used Available Use% Mounted on
/dev/hda6 reiserfs 20008280 1573976 18434304 8% /
tmpfs           tmpfs     1034188         0 1034188 0% /dev/shm
tmpfs           tmpfs     1034188      12588 1021600 2% /lib/modules/2.6.12-10-386/volatile
/dev/hda2      ext3      101105         19173 76711 20% /boot
/dev/hda8      vfat      2004156         8 2004148 1% /dos
/dev/hda7      xfs       9998208         3544 9994664 1% /home
/dev/hda1      ntfs     20967416 9594424 11372992 46% /media/hda1
```

You will notice that the output includes the total number of blocks as well as the number used and free. You will also notice the filesystem, such as `/dev/hda7`, and its mount point: `/home` for `/dev/hda7`. The two `tmpfs` entries are for virtual memory filesystems. These exist only in RAM or swap space and are created when mounted without need for a `mkfs` command. You can read more about `tmpfs` in "Common

threads: Advanced filesystem implementor's guide, Part 3" (see [Resources for a link](#)).

If you want specific information on inode usage, use the `-i` option on the `df` command. You can exclude certain filesystem types using the `-x` option or restrict information to just certain filesystem types using the `-t` option. Use these multiple time if necessary. See the examples in Listing 18.

Listing 18. Displaying inode usage

```
ian@pinguino:~$ df -i -x tmpfs
Filesystem          Inodes    IUsed    IFree  IUse%  Mounted on
/dev/hda6            0          0         0       -      /
/dev/hda2           26208       34    26174     1%    /boot
/dev/hda8            0          0         0       -      /dos
/dev/hda7          10008448     176 10008272     1%    /home
/dev/hda1           37532    36313    1219    97%    /media/hda1
ian@pinguino:~$ df -iT -t vfat -t ext3
Filesystem    Type    Inodes    IUsed    IFree  IUse%  Mounted on
/dev/hda2     ext3    26208       34    26174     1%    /boot
/dev/hda8     vfat     0          0         0       -      /dos
```

Perhaps you are not surprised to see that the FAT32 filesystem does not have inodes, but it may surprise you to see that the ReiserFS information shows no inodes. ReiserFS keeps metadata for files and directories in *stat items*. And since ReiserFS uses a balanced tree structure, there is no predetermined number of inodes as there are, for example, in ext2, ext3, or xfs filesystems.

There are several other options you may use with `df` to limit the display to local filesystems or control the format of output. For example, use the `-H` option to display human readable sizes, such as 1K for 1024, or use the `-h` (or `--si`) option to get sizes in powers of 10 (1K=1000).

If you aren't sure which filesystem a particular part of your directory tree lives on, you can give the `df` command a parameter of a directory name or even a filename as shown in Listing 19.

Listing 19. Human readable output for df

```
ian@pinguino:~$ df --si ~ian/index.html
Filesystem    Size    Used    Avail  Use%  Mounted on
/dev/hda7     11G    3.7M    11G    1%    /home
```

du

The `df` command gives information about only a whole filesystem. Sometimes you might want to know how much space is used by your home directory, or how big a partition you would need if you wanted to move `/usr` to its own filesystem. To answer this kind of question, use the `du` command.

The `du` command displays information about the filename (or filenames) given as parameters. If a directory name is given, then `du` recurses and calculates sizes for every file and subdirectory of the given directory. The result can be a lot of output. Fortunately, you can use the `-s` option to request just a summary for a directory. If you use `du` to get information for multiple directories, then add the `-c` option to get a grand total. You can also control output format with the same set of size options (`-h`, `-H`, `--si`, and so on) that are used for `df`. Listing 20 shows two views of my home directory on the newly installed Ubuntu system.

Listing 20. Using du

```
ian@pinguino:~$ du -hc *
0      Desktop
16K    index.html
16K    total
ian@pinguino:~$ du -hs .
3.0M   .
```

The reason for the difference between the 16K total from `du -c *` and the 3M summary from `du -s` is that the latter includes the entries starting with a dot, such as `.bashrc`, while the former does not.

One other thing to note about `du` is that you must be able to read the directories that you are running it against.

So now, let's use `du` to display the total space used by the `/usr` tree and each of its first-level subdirectories. The result is shown in Listing 21. Use root authority to make sure you have appropriate access permissions.

Listing 21. Using du on /usr

```
root@pinguino:~# du -shc /usr/*
66M   /usr/bin
0     /usr/doc
1.3M  /usr/games
742K  /usr/include
0     /usr/info
497M  /usr/lib
0     /usr/local
7.3M  /usr/sbin
578M  /usr/share
0     /usr/src
14M   /usr/X11R6
1.2G  total
```

Checking filesystems

Sometimes your system may crash or lose power. In these cases, Linux will not be able to cleanly unmount your filesystems, and they may be left in an inconsistent

state, with some changes completed and some not. Operating with a damaged filesystem is not a good idea as you are likely to further compound any existing errors.

The main tool for checking filesystems is `fsck`, which, like `mkfs`, is really a front end to filesystem checking routines for the various filesystem types. Some of the underlying check routines are shown in Listing 22.

Listing 22. Some of the fsck programs

```
ian@pinguino:~$ ls /sbin/*fsck*
/sbin/dosfsck      /sbin/fsck.ext3    /sbin/fsck.reiser4  /sbin/jfs_fscklog
/sbin/e2fsck       /sbin/fsck.jfs     /sbin/fsck.reiserfs /sbin/reiserfsck
/sbin/fsck         /sbin/fsck.minix   /sbin/fsck.vfat
/sbin/fsck.cramfs  /sbin/fsck.msdos   /sbin/fsck.xfs
/sbin/fsck.ext2    /sbin/fsck.nfs     /sbin/jfs_fsck
```

The system boot process use `fsck` to check the root filesystem and any other filesystems that are specified in the `/etc/fstab` control file. If the filesystem was not cleanly unmounted, a consistency check is performed. This is controlled by the `pass` (or `passno`) field (the sixth field) of the `/etc/fstab` entry. Filesystems with `pass` set to zero are not checked at boot time. The root filesystem has a `pass` value of 1 and is checked first. Other filesystems will usually have a `pass` value of 2 (or higher), indicating the order in which they should be checked. Multiple `fsck` operations can run in parallel, so different filesystems are allowed to have the same `pass` value, as is the case for our example `/boot` and `/home` filesystems.

Listing 23. Boot checking of filesystems with fstab entries

#	<file system>	<mount point>	<type>	<options>	<dump>	<pass>
	proc	/proc	proc	defaults	0	0
	/dev/hda6	/	reiserfs	defaults	0	1
	/dev/hda2	/boot	ext3	defaults	0	2
	/dev/hda8	/dos	vfat	defaults	0	0
	/dev/hda7	/home	xfs	defaults	0	2

Note that some journaling filesystems, such as ReiserFS and xfs, might have a `pass` value of 0 because the journaling code, rather than `fsck`, does the filesystem consistency check and repair.

Repairing filesystems

If the automatic boot time check of filesystems is unable to restore consistency, you are usually dumped into a single user shell with some instructions to run `fsck` manually. For an `ext2` filesystem, which is not journaled, you may be presented with a series of requests asking you to confirm proposed actions to fix particular blocks on the filesystem. You should generally allow `fsck` to attempt to fix problems, by

responding `y` (for yes). When the system reboots, check for any missing data or files.

If you suspect corruption, or want to run a check manually, most of the checking programs require the filesystem to be unmounted first. Since you can't unmount the root filesystem on a running system, the best you can do is drop to single user mode (using `telinit 1`) and then remount the root filesystem read-only, at which time you should be able to perform a consistency check. (Mounting filesystems is covered in the next section; [Mounting and unmounting filesystems](#).) A better way to check a filesystem is to boot a recovery system, such as a live CD or a USB memory key, and perform the check of your unmounted filesystems from that.

Why journal?

An `fsck` scan of an ext2 disk can take quite a while to complete, because the internal data structure (or *metadata*) of the filesystem must be scanned completely. As filesystems get larger and larger, this takes longer and longer, even though disks also keep getting faster, so a full check may take one or more hours.

This problem was the impetus for *journaled* or *journaling* filesystems. Journaled filesystems keep a log of recent changes to the filesystem metadata. After a crash, the filesystem driver inspects the log in order to determine which recently changed parts of the filesystem may possibly have errors. With this design change, checking a journaled filesystem for consistency typically takes just a matter of seconds, regardless of filesystem size. Furthermore, the filesystem driver will usually check the filesystem on mounting, so an external `fsck` check is generally not required. In fact, for the xfs filesystem, `fsck` does nothing!

If you do run a manual check of a filesystem, check the man pages for the appropriate `fsck` command (`fsck.ext3`, `e2fsck`, `reiserfsck`, and so on) to determine the appropriate parameters. Some examples are in Listing 24, using a Ubuntu live CD image to run the `fsck` commands.

Listing 24. Running fsck manually

```
root@ubuntu:~# fsck -p /dev/hda6
fsck 1.38 (30-Jun-2005)
Reiserfs super block in block 16 on 0x306 of format 3.6 with standard journal
Blocks (total/free): 5002224/4608416 by 4096 bytes
Filesystem is clean
Replaying journal..
Reiserfs journal '/dev/hda6' in blocks [18..8211]: 0 transactions replayed
Checking internal tree..finished
root@ubuntu:~# fsck -p /dev/hda2
fsck 1.38 (30-Jun-2005)
BOOT: clean, 34/26208 files, 22488/104420 blocks
root@ubuntu:~# fsck -p /dev/hda7
fsck 1.38 (30-Jun-2005)
root@ubuntu:~# fsck -a /dev/hda8
fsck 1.38 (30-Jun-2005)
dosfsck 2.11, 12 Mar 2005, FAT32, LFN
```

```
/dev/hda8: 1 files, 2/501039 clusters
```

Advanced tools

There are several more advanced tools that you can use to examine or repair a filesystem. Check the man pages for the correct usage and the Linux Documentation Project (see [Resources](#)) for how-to information. Almost all of these commands require a filesystem to be unmounted, although some functions can be used on filesystems that are mounted read-only. A few of the commands are described below.

You should always back up your filesystem before attempting any repairs.

Tools for ext2 and ext3 filesystems

tune2fs

Adjusts parameters on ext2 and ext3 filesystems. Use this to add a journal to an ext2 system, making it an ext3 system, as well as display or set the maximum number of mounts before a check is forced. You can also assign a label and set or disable various optional features.

dumpe2fs

Prints the super block and block group descriptor information for an ext2 or ext3 filesystem.

debugfs

Is an interactive file system debugger. Use it to examine or change the state of an ext2 or ext3 file system.

Tools for Reiserfs filesystems

reiserfstune

Displays and adjusts parameters on ReiserFS filesystems.

debugreiserfs

Performs similar functions to dumpe2fs and debugfs for ReiserFS filesystems.

Tools for XFS filesystems

xfs_info

Displays XFS filesystem information.

xfs_growfs

Expands an XFS filesystem (assuming another partition is available).

xfs_admin

Changes the parameters of an XFS filesystem.

xfs_repair

Repairs an XFS filesystem when the mount checks are not sufficient to repair the system.

xfs_db

Examines or debugs an XFS filesystem.

Section 4. Mounting and unmounting filesystems

This section covers material for topic 1.104.3 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn how to:

- Mount filesystems
- Unmount filesystems
- Configure filesystem mounting on bootup
- Configure user-mountable, removable filesystems such as tape drives, floppies, and CDs

Mounting filesystems

The Linux filesystem is one big tree rooted at /, and yet we have filesystems on different devices and partitions. Now we resolve this apparent incongruity. The root (/) filesystem is mounted as part of the initialization process. Each of the other filesystems that we have created is not usable by your Linux system until it is *mounted at a mount point*.

A mount point is simply a directory in the current set of mounted filesystems at which point the filesystem on a device is grafted into the tree. Mounting is the process of making the filesystem on the device part of your accessible Linux filesystem. For example, you might mount filesystems on hard drive partitions as /boot, /tmp, or /home, and you might mount the filesystem on a floppy drive as /mnt/floppy and the filesystem on a CD-ROM as /media/cdrom1.

Besides filesystems on partitions, floppy disks, and CDs, there are other types of

filesystems. We alluded briefly to the `tmpfs` filesystem, which is a virtual memory filesystem. It is also possible to mount filesystems from one system on another system using a networked filesystem such as NFS or AFS. You can also create a file in an existing filesystem and format that as a, possibly different, kind of filesystem and mount that too.

While the mount process actually mounts the *filesystem* on some device (or other resource), it is common to simply say that you "mount the device", which is understood to mean "mount the filesystem on the device".

The basic form of the `mount` command takes two parameters: the device (or other resource) containing the filesystem to be mounted, and the mount point. For example, we mount our FAT32 partition `/dev/hda8` at the mount point `/dos` as shown in Listing 25.

Listing 25. Mounting /dos

```
root@pinguino:~# mount /dev/hda8 /dos
```

The mount point must exist before you mount anything over it. When you do, the files on the filesystem you are mounting become the files and subdirectories of the mount point. If the mount point directory already contained files or subdirectories, they are no longer visible until the mounted filesystem is unmounted, at which point they become visible again. It is a good idea to avoid this problem by using only empty directories as mount points.

After mounting a filesystem, any files or directories created or copied to the mount point or any directory below it will be created on the mounted filesystem. So a file such as `/dos/sampdir/file.txt` will be created on the FAT32 filesystem that we mounted at `/dos` in our example.

Usually, the `mount` command will automatically detect the type of filesystem being mounted. Occasionally you may need to specify the filesystem type explicitly using the `-t` option as shown in Listing 26.

Listing 26. Mounting with explicit filesystem type

```
root@pinguino:~# mount -t vfat /dev/hda8 /dos
```

To see what filesystems are mounted, use the `mount` command with no parameters. Listing 27 shows our example system.

Listing 27. Displaying mounted filesystems

```

/dev/hda6 on / type reiserfs (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
tmpfs on /lib/modules/2.6.12-10-386/volatile type tmpfs (rw,mode=0755)
/dev/hda2 on /boot type ext3 (rw)
/dev/hda8 on /dos type vfat (rw)
/dev/hda7 on /home type xfs (rw)
/dev/hda1 on /media/hda1 type ntfs (rw)
tmpfs on /dev type tmpfs (rw,size=10M,mode=0755)

```

You can also view similar information by displaying `/proc/mounts` or `/etc/mtab`, both of which contain information about mounted filesystems.

Mount options

The `mount` command has several options that override the default behavior. For example, you can mount a filesystem read-only by specifying `-o ro`. If the filesystem is already mounted, add `remount` as shown in Listing 28.

Listing 28. Remounting read-only

```

root@pinguino:~# mount -o remount,ro /dos

```

Notes:

- Separate multiple options with commas.
- When remounting an already mounted filesystem, it suffices to specify either the mount point or the device name. It is not necessary to specify both.
- You cannot mount a read-only filesystem as read-write. Media that cannot be modified, such as CD-ROM discs, will automatically be mounted read-only.
- To remount a writable device read-write, specify `-o remount,rw`

Remount commands will not complete successfully if any process has open files or directories in the filesystem being remounted. Use the `lsof` command to determine what files are open. Check the man pages to learn about additional mount options and `lsof`.

fstab

In the tutorial for topic 102, "[LPI exam 101 prep \(topic 102\)d: Linux installation and](#)

[package management](#)," you learned how to use the `root=` parameter in both GRUB and LILO to tell the boot loader what filesystem should be mounted as root. Once this filesystem is mounted, the initialization process runs `mount` with the `-a` option to automatically mount a set of filesystems. The set is specified in the file `/etc/fstab`. Listing 29 shows `/etc/fstab` for the sample Ubuntu system that we installed using the filesystems created earlier in this tutorial.

Listing 29. An example `fstab`

```
root@pinguino:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
#<file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
/dev/hda6 / reiserfs defaults 0 1
/dev/hda2 /boot ext3 defaults 0 2
/dev/hda8 /dos vfat defaults 0 0
/dev/hda7 /home xfs defaults 0 2
/dev/hda1 /media/hda1 ntfs defaults 0 0
/dev/hda5 none swap sw 0 0
/dev/hdc /media/cdrom0 udf,iso9660 user,noauto 0 0
/dev/fd0 /media/floppy0 auto rw,user,noauto 0 0
```

Lines starting with a `#` character are comments. Remaining lines contain six fields. Since the fields are positional, they must all be specified.

file system

For the examples used so far, this will be a device name such as `/dev/hda1`.

mount point

This is the mount point we discussed in [Mounting filesystems](#) above. For swap space, this should be the value `none`. For `ext2`, `ext3`, and `xfs` filesystems, you may also specify a label such as `LABEL=XFSHOME`. This makes your system more robust when devices are added or removed.

type

Specifies the type of filesystem. CD/DVD drives will often support either ISO9660 or UDF filesystems, so you may specify multiple possibilities in a comma-separated list. If you want `mount` to automatically determine the type, specify `auto` as is done in the last line for the floppy drive.

option

Specifies the mount options. Specify `defaults` if you want default mount options. Some options you will want to know about are:

- `rw` and `ro` specify whether the filesystem should be mounted read-write or read-only.
- `noauto` specifies that this filesystem should not be automatically mounted at boot time or whenever `mount -a` is issued. In our example, this is done for the removable drives.

- `user`
- Specifies that a non-root user is permitted to mount and unmount the filesystem. This is especially useful for removable media. This option must be specified in `/etc/fstab` rather than on the `mount` command.
- `exec` and `noexec` specify whether or not to allow execution of files from the mounted filesystem. User-mounted filesystems default to `noexec` unless `exec` is specified **after** `user`.
- `noatime` will disable recording of access times. Not using access times may improve performance.

dump

Specifies whether the `dump` command should consider this `ext2` or `ext3` filesystem for backups. A value of 0 tells `dump` to ignore this filesystem.

pass

Non-zero values of `pass` specify the order of checking filesystems at boot time, as discussed in [Checking filesystems](#).

For filesystems that are listed in `/etc/fstab`, it suffices to give either the device name or the mount point when mounting the filesystem. You do not need to give both.

Consult the man pages for `fstab` and `mount` for additional information, including options not covered here.

Unmounting filesystems

All mounted filesystems are usually unmounted automatically by the system when it is rebooted or shut down. When a filesystem is unmounted, any cached filesystem data in memory is flushed to the disk.

You may also unmount filesystems manually. Indeed, you **should** do this when removing writable media such as diskettes or USB drives or memory keys. Before unmounting a filesystem, make sure that there are no processes running that have open files on the filesystem. Then, use the `umount` command, specifying *either* the device name or mount point as an argument. Some successful and unsuccessful examples are shown in Listing 30.

Listing 30. Unmounting filesystems

```
root@pinguino:~# lsof /dos
root@pinguino:~# umount /dos
root@pinguino:~# mount /dos
```

```
root@pinguino:~# umount /dev/hda8
root@pinguino:~# umount /boot
umount: /boot: device is busy
umount: /boot: device is busy
root@pinguino:~# lsof /boot
COMMAND  PID USER  FD   TYPE DEVICE   SIZE NODE NAME
klogd    6498 klog   1r   REG   3,2 897419 6052 /boot/System.map-2.6.12-10-386
```

After a filesystem is unmounted, any files in the directory used for the mount point are visible again.

Swap space

You may have noticed in the discussion of `fstab` above that swap space does not have a mount point. The boot process usually enables swap space defined in `/etc/fstab` unless the `noauto` option is specified. To manually control swap space on a running system -- for example, if you added a new swap partition -- use the `swapon` and `swapoff` commands. See the man pages for details.

You can view the currently enabled swap devices with `cat /proc/swaps`.

Section 5. Disk quotas

This section covers material for topic 1.104.4 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn how to:

- Enable quotas
- Set quota limits
- Check quotas
- Generate quota reports

Quotas allow you to control disk usage by user or by group. Quotas prevent individual users and groups from using a larger portion of a filesystem than they are permitted, or from filling it up altogether. Quotas must be enabled and managed by the root user. They are often used on multi-user systems, but less often on single-user workstations.

Enabling quotas

Quotas require kernel support. Generally, a recent 2.4 or 2.6 kernel should have all the support you need. Earlier versions may have incomplete quota support, requiring you to build your own kernel. In current implementations you will probably find quota support implemented as kernel modules. There are three different types of quota support; `vfsold` (version 1 quota), `vfsv0` (version 2 quota), and `xfs` (quota on XFS filesystems). This section covers version 2 quota on non-XFS filesystems and `xfs` quota on XFS filesystems.

The first step to enable quotas is to add the `usrquota`, or `grpquota` options to the filesystem definitions in `/etc/fstab`, according to whether you want to implement user quotas, group quotas, or both. Suppose you want to add both types of quota to the XFS filesystem used for home directories in our example and also to the `/boot` filesystem so you can see how it works on two different filesystems. Do this as shown in Listing 31.

Listing 31. Enabling quota support in `/etc/fstab`

```
/dev/hda2    /boot    ext3    defaults,usrquota,grpquota    0    2
/dev/hda7    /home    xfs     defaults,usrquota,grpquota    0    2
```

For XFS filesystems, quota data is considered part of the filesystem metadata. For other filesystems, user quota information is stored in the `aquota.user` file in the root of the filesystem, and group quota is similarly stored in `aquota.group`. Version 1 quotas used `quota.user` and `quota.group`.

After you edit `/etc/fstab` and add quotas, you need to remount the filesystems and, for non-XFS filesystems, create the quota files and enable quota checking. The `quotacheck` command checks the quotas on all filesystems and creates the required `aquota.user` and `aquota.group` files if they do not exist. It can also repair damaged quota files. See the man pages for more information. The `quotaon` command turns on quota checking. Listing 32 shows an example. The following options are used on both commands:

- a** For all filesystems in `/etc/fstab` that are enabled for automount
- u** For user quotas (this is the default)
- g** For group quotas
- v** For verbose output

Listing 32. Creating quota files and turning quota on

```

root@pinguino:~# quotacheck -augv
quotacheck: Scanning /dev/hda2 [/boot] quotacheck: Cannot stat old user quota
file: No such file or directory
quotacheck: Cannot stat old group quota file: No such file or directory
quotacheck: Cannot stat old user quota file: No such file or directory
quotacheck: Cannot stat old group quota file: No such file or directory
done
quotacheck: Checked 4 directories and 23 files
quotacheck: Old file not found.
quotacheck: Old file not found.
quotacheck: Skipping /dev/hda7 [/home]
root@pinguino:~# quotaon -ugva
/dev/hda2 [/boot]: group quotas turned on
/dev/hda2 [/boot]: user quotas turned on

```

Checking quotas on boot

The `quotacheck` and `quotaon` commands are usually included in initialization scripts so that quotas are enabled whenever you reboot the system. The Quota Mini HOWTO (see [Resources](#) for a link) has additional information.

The `quotaoff` command disables quotas, should you ever need to do so.

Setting quota limits

As you have seen, quotas are controlled either through binary files in the root of the filesystem or through filesystem metadata. In order to set a quota for a particular user, use the `edquota` command. This command extracts the quota information for the user from the various filesystems with quotas enabled, creates a temporary file, and opens an editor for you to adjust the quotas. See the man pages for `edquota` to find out which editor is used. You must be root to edit quotas. The information displayed will look something like Listing 33.

Listing 33. Running edquota

```

Disk quotas for user ian (uid 1000):
Filesystem      blocks      soft      hard    inodes      soft      hard
/dev/hda2         0            0         0           0            0         0
/dev/hda7       2948         0         0          172           0         0

```

As you can see, `edquota` displays my current usage of both 1K blocks and inodes on each of the filesystems that have quota turned on. There are also soft and hard limits for both block and inode usage. In this example, these are 0, meaning no quota limit is enforced.

The soft limit is the value at which a user will receive e-mail warnings about being

over quota. The hard limit is the value that a user may not exceed. You can think of block limits as a limit on the amount of data that a user may store, and inode limits as a limit on the number of files and directories.

Changing quota limits

You change the quota limits by changing the values in the temporary file and then saving the file. Quit the file without saving if you do not want to make changes. Assume you want to set my quota to 10MB of data and 1000 files on the /home filesystem. Allowing 10% additional for hard limits, you would set values as in Listing 34.

Listing 34. Setting limits

```
Disk quotas for user ian (uid 1000):
Filesystem      blocks      soft      hard      inodes      soft      hard
/dev/hda2        0            0          0          0            0          0
/dev/hda7        2948        10240     11264     172          1000      1100
```

Save the file, and the new quotas will take effect. In this example, no changes were made to the quota for user ian on the /boot filesystem, since ian cannot write to this filesystem. Note also that any changes you make to the used blocks or inodes values will be ignored.

Copying quotas

Now suppose you are creating ids for people enrolled in a class. Assume you have users gretchen, tom, and greg and you'd like them all to have the same quota as ian. You do this using the `-p` option of `edquota`, which uses the quota values of ian as a prototype for those of the other users as shown in Listing 35.

Listing 35. Setting quotas from a prototype

```
root@pinguino:~# edquota -p ian gretchen tom greg
```

Group limits

You can also use `edquota` to restrict the allocation of disk space based on the group ownership of files. For example, the three class attendees above are set up in primary group `xml-101`. To limit the total amounts used by the all members of the group to 25MB and 2500 files, use the command `edquota -g xml-101` and set the values as shown in Listing 36.

Listing 36. Setting quotas for a group

```
Disk quotas for group xml-101 (gid 1001):
```

Filesystem	blocks	soft	hard	inodes	soft	hard
/dev/hda2	0	0	0	0	0	0
/dev/hda7	28	25600	28160	10	2500	2750

The grace period

Users may exceed their soft limit for a *grace period*, which defaults to 7 days. After the grace period, the soft limit is enforced as a hard limit. Set grace periods with the `-y` option of `edquota`. Again, you will be placed in an editor with data similar to that of Listing 37. As before, save changes to update the values. Be sure to leave your users enough time to receive their warning e-mail and delete some files.

Listing 37. Setting grace periods

```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
Filesystem      Block grace period   Inode grace period
/dev/hda2       7days                7days
/dev/hda7       7days                7days
```

Checking quotas

The `quota` command with no options displays the quotas for the invoking user on any filesystems for which the user has quotas set. The `-v` option displays the information for all filesystems that have quota enabled. The root user may also add a user name to the command to view quotas for a particular user. These commands are shown in Listing 38.

Listing 38. Displaying quotas

```
root@pinguino:~# quota
Disk quotas for user root (uid 0): none
root@pinguino:~# quota -v
Disk quotas for user root (uid 0):
Filesystem  blocks  quota  limit  grace  files  quota  limit  grace
/dev/hda2   19173   0      0      0      26     0      0
/dev/hda7   16      0      0      0      5      0      0
root@pinguino:~# quota -v ian
Disk quotas for user ian (uid 1000):
Filesystem  blocks  quota  limit  grace  files  quota  limit  grace
/dev/hda2   0      0      0      0      0      0      0
/dev/hda7   2948   10240 11264  0      172    1000  1100
```

Along with the statistics on current usage, you see the soft and hard quota limits displayed. Listing 39 shows what happens if you exceed the soft limit and then what happens if you attempt to exceed the hard limit. In this example, a file of approximately 4MB is created and then a copy is made. Along with the original usage of approximately 3MB, this is sufficient to exceed the soft limit. Notice how the soft limit has an asterisk beside it indicating that the user is over quota. Note also

that the grace period columns now indicate how long the user has to correct the problem.

Listing 39. Exceeding quotas

```
ian@pinguino:~$ dd if=/dev/zero of=big1 bs=512 count=8000
8000+0 records in
8000+0 records out
4096000 bytes transferred in 0.019915 seconds (205674545 bytes/sec)
ian@pinguino:~$ cp big1 big2
ian@pinguino:~$ quota
Disk quotas for user ian (uid 1000):
    Filesystem blocks quota limit grace files quota limit grace
    /dev/hda7  10948* 10240 11264 7days  174  1000  1100
ian@pinguino:~$ cp big1 big3
cp: writing `big3': Disk quota exceeded
```

Generating quota reports

Checking user quotas one user at a time is not very useful, so you will want to use the `repquota` command to generate quota reports. Listing 40 shows how to see the quotas for all users and groups on `/home`.

Listing 40. Exceeding quotas

```
root@pinguino:~# repquota -ug /home
*** Report for user quotas on device /dev/hda7
Block grace time: 7days; Inode grace time: 7days
      Block limits
User      used  soft  hard  grace  used  soft  hard  grace
-----
root      --    16    0     0
ian       +-  11204 10240 11264 6days  175  1000  1100
tom       --    8     10240 11264
gretchen  --    8     10240 11264
greg      --   12    10240 11264

*** Report for group quotas on device /dev/hda7
Block grace time: 7days; Inode grace time: 7days
      Block limits
Group     used  soft  hard  grace  used  soft  hard  grace
-----
root      --    16    0     0
ian       --  11204 0     0
xml-101   --   28   25600 28160  10  2500  2750
```

Note the plus sign in the listing for user `ian`, indicating that `ian` is over quota.

As with other quota commands, the `-a` option produces a report for all mounted filesystems that have quota enabled. The `-v` option produces more verbose output. And the `-n` option produces the listing by numeric user number rather than resolving the user number to a name. This may provide a performance boost for large reports, but is generally less useful to human readers.

Warning users

The `warnquota` command is used to send e-mail warnings to users who are over quota. When a group is over quota, the e-mail is sent to the user specified in `/etc/quotagrpadmins` for the group. Normally `warnquota` is run periodically as a `cron` job. See the man pages for `cron` and `warnquota` for more information.

Section 6. File permissions and access control

This section covers material for topic 1.104.5 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn about:

- Users and groups
- Permissions on files and directories
- Changing permissions
- Access modes
- Immutable files
- Default file creation modes

User and groups

By now, you know that Linux is a multi-user system and that each user belongs to one *primary* group and possibly additional groups. It is also possible to log in as one user and become another user using the `su` or `sudo -s` commands. Ownership of files in Linux is closely related to user ids and groups, so let's review some basic user and group information.

Who am I?

If you have not become another user, your id is still the one you used to log in. If you have become another user, your prompt may include your user id, as most of the examples in this tutorial do. If your prompt does not include your user id, then you can use the `whoami` command to check your current effective id. Listing 41 shows some examples where the prompt strings (from the `PS1` environment variable) are

different from the other examples in this tutorial.

Listing 41. Determining effective user id

```
/home/ian$ whoami
tom
/home/ian$ exit
exit
$ whoami
ian
```

What groups am I in?

Similarly, you can find out what groups you are in by using the `groups` command. You can find out both user and group information using the `id` command. Add a user id parameter to either `groups` or `id` to see information for that user id instead of the current user id. See Listing 42 for some examples.

Listing 42. Determining group membership

```
$ su tom
Password:
/home/ian$ groups
xml-101
/home/ian$ id
uid=1001(tom) gid=1001(xml-101) groups=1001(xml-101)
/home/ian$ exit
$ groups
ian adm dialout cdrom floppy audio dip video plugdev lpadmin scanner admin xml-101
$ id
uid=1000(ian) gid=1000(ian) groups=4(adm),20(dialout),24(cdrom),25(floppy),
29(audio),30(dip),44(video),46(plugdev),104(lpadmin),105(scanner),106(admin),
1000(ian),1001(xml-101)
$ groups tom
tom : xml-101
```

File ownership and permissions

Just as every user has an id and is a member of one primary group, so every file on a Linux system has one owner and one group associated with it.

Ordinary files

Use the `ls -l` command to display the owner and group.

Listing 43. Determining file ownership

```
gretchen@pinguino:~$ ls -l /bin/bash .bashrc
-rw-r--r-- 1 gretchen xml-101 2227 Dec 20 10:06 .bashrc
-rwxr-xr-x 1 root root 645140 Oct 5 08:16 /bin/bash
```

In this particular example, user gretchen's `.bashrc` file is owned by her and is in the `xml-101` group, which is her primary group. Similarly, `/bin/bash` is owned by user `root` and is in the group `root`. User names and groups names come from separate namespaces, so a group name may be the same as a user name. In fact, many distributions default to creating a matching group for each new user.

The Linux permissions model has three types of permission for each filesystem object. The permissions are read (`r`), write (`w`), and execute (`x`). Write permission includes the ability to alter or delete an object. In addition, these permissions are specified separately for the file's owner, members of the file's group, and everyone else.

Referring back to the first column of Listing 43, notice that it contains a ten-character string. The first character describes the type of object (`-` for an ordinary file in this example) and the remaining nine characters represent three groups of three characters. The first group indicates the read, write, and execute permissions for the file's owner. A `-` indicates that the corresponding permission is not granted. So user gretchen can read and write the `.bashrc` file, but not execute it; while root can read, write, **and** execute the `/bin/bash` file. The second group indicates the read, write, and execute permissions for the file's group. Members of the `xml-101` group can read gretchen's `.bashrc` file, but not write it, as can everyone else. Similarly, members of the `root` group and everyone else can read or execute the `/bin/bash` file.

Directories

Directories use the same permissions flags as regular files but they are interpreted differently. Read permission for a directory allows a user with that permission to list the contents of the directory. Write permission means a user with that permission can create or delete files in the directory. Execute permission allows the user to enter the directory and access any subdirectories. Without execute permission, the filesystem objects inside a directory are not accessible. Without read permission, the filesystem objects inside a directory are not viewable, but these objects can still be accessed as long as you know the full path to the object on disk. Listing 44 is a somewhat artificial example that illustrates these points.

Listing 44. Permissions and directories

```
ian@pinguino:~$ ls -l /home
total 8
drwxr-x---  2 greg      xml-101   60 2005-12-20 11:37 greg
drwx----- 13 gretchen  xml-101 4096 2005-12-21 12:22 gretchen
drwxr-xr-x 15 ian       ian      4096 2005-12-21 10:25 ian
d-wx--x--x  2 tom        xml-101   75 2005-12-21 11:05 tom
ian@pinguino:~$ ls -a ~greg
.  ..  .bash_history  .bash_profile  .bashrc
ian@pinguino:~$ ls -a ~gretchen
ls: /home/gretchen: Permission denied
ian@pinguino:~$ ls -a ~tom
ls: /home/tom: Permission denied
ian@pinguino:~$ head -n 3 ~tom/.bashrc
```

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples
```

The first character of a long listing describes the type of object (d for a directory). User greg's home directory has read and execute permission for members of the xml-101 group, so user ian can list the directory. User gretchen's home directory has neither read nor execute permission, so user ian cannot access it. User's tom's home has execute but not read permission, so user ian cannot list the contents, but can access objects within the directory if he knows they exist.

Other filesystem objects

A long listing may contain filesystem objects other than files and directories as shown by the first character in the listing. We will see more of these later in this section, but for now, note the possible types of object.

Code	Object type
-	Regular file
d	Directory
l	Symbolic link
c	Character special device
b	Block special device
p	FIFO
s	Socket

Changing permissions

Adding permissions

Suppose you create a "Hello world" shell script. When you first create the script, it will usually not be executable. Use the `chmod` command with the `+x` option to add the execute permissions as shown in Listing 45.

Listing 45. Creating an executable shell script

```
ian@pinguino:~$ echo 'echo "Hello world!"'>hello.sh
ian@pinguino:~$ ls -l hello.sh
-rw-r--r-- 1 ian ian 20 2005-12-22 12:57 hello.sh
ian@pinguino:~$ ./hello.sh
-bash: ./hello.sh: Permission denied
```

```
ian@pinguino:~$ chmod +x hello.sh
ian@pinguino:~$ ./hello.sh
Hello world!
ian@pinguino:~$ ls -l hello.sh
-rwxr-xr-x 1 ian ian 20 2005-12-22 12:57 hello.sh
```

You can use `r` to set the read permissions, and `w` to set the write permissions in a similar manner. In fact, you can use any combination of `r`, `w`, and `x` together. For example, using `chmod +rwx` would set all the read, write, and execute permissions for a file. This form of `chmod` adds permissions that are not already set.

Being selective

You may have noticed in the above example that execute permission was set for the owner, group, **and** others. To be more selective, you may prefix the mode expression with `u` to set the permission for users, `g` to set it for groups, and `o` to set it for others. Specifying `a` sets the permission for all users, which is equivalent to omitting it. Listing 46 shows how to add user and group write and execute permissions to another copy of the shell script.

Listing 46. Selectively adding permissions

```
ian@pinguino:~$ echo 'echo "Hello world!"'>hello2.sh
ian@pinguino:~$ chmod ug+xw hello2.sh
ian@pinguino:~$ ls -l hello2.sh
-rwxrwxr-- 1 ian ian 20 2005-12-22 13:17 hello2.sh
```

Removing permissions

Sometimes you need to remove permissions rather than add them. Simply change the `+` to a `-`, and you remove any of the specified permissions that are set. Listing 47 shows how to remove all permissions for other users on the two shell scripts.

Listing 47. Removing permissions

```
ian@pinguino:~$ ls -l hello*
-rwxrwxr-- 1 ian ian 20 2005-12-22 13:17 hello2.sh
-rwxr-xr-x 1 ian ian 20 2005-12-22 12:57 hello.sh
ian@pinguino:~$ chmod o-xrw hello*
ian@pinguino:~$ ls -l hello*
-rwxrwx--- 1 ian ian 20 2005-12-22 13:17 hello2.sh
-rwxr-x--- 1 ian ian 20 2005-12-22 12:57 hello.sh
```

Note that you can change permissions on more than one file at a time. As with some other commands you met in the tutorial for topic 103, you can even use the `-R` (or `--recursive`) option to operate recursively on directories and files.

Setting permissions

Now that you can add or remove permissions, you may wonder how to set just a specific set of permissions. Do this using = instead of + or -. To set the permissions on the above scripts so that other users have no access rights, you could use `chmod o= hello*`, instead of the command we used to remove permissions.

If you want to set different permissions for user, group, or other, you can separate different expressions by commas; for example, `ug=rwx,o=rx`, or you can use numeric permissions, which are described next.

Octal permissions

So far you have used symbols (ugo and rwx) to specify permissions. There are three possible permissions in each group. You can also set permissions using octal numbers instead of symbols. Permissions set in this way use up to four octal digits. We will look at the first digit when we discuss attributes. The second digit defines user permissions, the third group permissions and the fourth other permissions. Each of these three digits is constructed by adding the desired permissions settings: read (4), write (2), and execute (1). In the example for `hello.sh` in Listing 45, the script was created with permissions `-rw-r--r--`, corresponding to octal 644. Setting execute permission for everyone changed the mode to 755.

Using numeric permissions is very handy when you want to set all the permissions at once without giving the same permissions to each of the groups. Use Table 4 as a handy reference for octal permissions.

Symbolic	Octal
<code>rwx</code>	7
<code>rw-</code>	6
<code>r-x</code>	5
<code>r--</code>	4
<code>-wx</code>	3
<code>-w-</code>	2
<code>--x</code>	1
<code>---</code>	0

Access modes

When you log in, the new shell process runs with your user and group ids. These are the permissions that govern your access to any files on the system. This usually means that you cannot access files belonging to others and cannot access system files. In fact, we as users are totally dependent on other programs to perform

operations on our behalf. Because the programs you start inherit *your* user id, they cannot access any filesystem objects for which you haven't been granted access.

An important example is the `/etc/passwd` file, which cannot be changed by normal users directly, because write permission is enabled only for root: However, normal users need to be able to modify `/etc/passwd` somehow, whenever they need to change their password. So, if the user is unable to modify this file, how can this be done?

suid and sgid

The Linux permissions model has two special access modes called suid (set user id) and sgid (set group id). When an executable program has the suid access modes set, it will run as if it had been started by the file's owner, rather than by the user who really started it. Similarly, with the sgid access modes set, the program will run as if the initiating user belonged to the file's group rather than to his own group. Either or both access modes may be set.

Listing 48 shows that the `passwd` executable is owned by root:

Listing 48. suid access mode on `/usr/bin/passwd`

```
ian@pinguino:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 25648 2005-10-11 12:14 /usr/bin/passwd
```

Note that in place of an `x` in the user's permission triplet, there's an `s`. This indicates that, for this particular program, the suid and executable bits are set. So when `passwd` runs, it will execute as if the root user had launched it, with full superuser access, rather than that of the user who ran it. Because `passwd` runs with `root` access, it can modify `/etc/passwd`.

The suid and sgid bits occupy the same space as the `x` bits in a long directory listing. If the file is executable, the suid or sgid bits, if set, will be displayed as lowercase `s`, otherwise they are displayed as uppercase `S`.

While suid and sgid are handy, and even necessary in many circumstances, improper use of these access mode can allow breaches of a system's security. You should have as few suid programs as possible. The `passwd` command is one of the few that **must** be suid.

Setting suid and sgid

The suid and sgid bits are set and reset symbolically using the letter `s`; for example, `u+s` sets the suid access mode, and `g-s` removes the sgid mode. In the octal format, suid has the value 4 in the first (high order) digit, while sgid has the value 2.

Directories and sgid

When a directory has the sgid mode enabled, any files or directories created in it will inherit the group id of the directory. This is particularly useful for directory trees that are used by a group of people working on the same project. Listing 49 shows how user greg could set up a directory that all users of the xml-101 group could use, along with an example of how user gretchen could create a file in the directory.

Listing 49. sgid access mode and directories

```
greg@pinguino:~$ mkdir xml101
greg@pinguino:~$ chmod g+ws xml101
greg@pinguino:~$ ls -ld xml101
drwxrwsr-x  2 greg xml-101 6 Dec 25 22:01 xml101
greg@pinguino:~$ su - gretchen
Password:
gretchen@pinguino:~$ touch ~greg/xml101/gretchen.txt
gretchen@pinguino:~$ ls -l ~greg/xml101/gretchen.txt
-rw-r--r--  1 gretchen xml-101 0 Dec 25 22:02 /home/greg/xml101/gretchen.txt
```

Any member of group xml-101 can now create files in user greg's xml101 directory. As Listing 50 shows, other members of the group cannot update the file gretchen.txt, but they do have write permission to the directory and can therefore delete the file.

Listing 50. sgid access mode and file ownership

```
gretchen@pinguino:~$ su - tom
Password:
~$ cat something >> ~greg/xml101/gretchen.txt
~su: /home/greg/xml101/gretchen.txt: Permission denied
~$ rm ~greg/xml101/gretchen.txt
rm: remove write-protected regular empty file `~/home/greg/xml101/gretchen.txt'? y
~$ ls -l ~greg/xml101
total 0
```

The sticky bit

You have just seen how anyone with write permission to a directory can delete files in it. This might be acceptable for a workgroup project, but is not desirable for globally shared file space such as the /tmp directory. Fortunately, there is a solution.

The remaining access mode bit is called the *sticky* bit. It is represented symbolically by t and numerically as a 1 in the high-order octal digit. It is displayed in a long directory listing in the place of the executable flag for other users (the last character), with the same meaning for upper and lower case as for suid and sgid. If set for a directory, it permits only the owning user or the superuser (root) to delete or unlink a file. Listing 51 shows how user greg could set the sticky bit on his xml101 directory and also shows that this bit is set for /tmp.

Listing 51. Sticky directories

```

greg@pinguino:~$ chmod +t xml101
greg@pinguino:~$ ls -l xml101
total 0
greg@pinguino:~$ ls -ld xml101
drwxrwsr-t  2 greg xml-101 6 Dec 26 09:41 xml101
greg@pinguino:~$ ls -ld xml101 /tmp
drwxrwxrwt 13 root root   520 Dec 26 10:03 /tmp
drwxrwsr-t  2 greg xml-101  6 Dec 26 09:41 xml101

```

On a historical note, UNIX® systems used to use the sticky bit on files to hoard executable files in swap space and avoid reloading. Modern Linux kernels ignore the sticky bit if it is set for files.

Access mode summary

Table 5 summarizes the symbolic and octal representation for the three access modes discussed here.

Table 5. Access modes		
Access mode	Symbolic	Octal
suid	s with u	4000
sgid	s with g	2000
sticky	t	1000

Combining this with the earlier permission information, you can see that the full octal representation corresponding to greg's xml101 permissions and access modes of drwxrwsr-t is 1775.

Immutable files

The access modes and permissions provide extensive control over who can do what with files and directories. However, they do not prevent inadvertent deletion of files by the root user. There are some additional *attributes* available on various filesystems that provide additional capabilities. One of these is the *immutable* attribute. If this is set, even root cannot delete the file until the attribute is unset.

Use the `lsattr` command to see whether the immutable flag (or any other attribute) is set for a file or directory. To make a file immutable, use the `chattr` command with the `-i` flag.

Listing 52 shows that user root can create an immutable file but cannot delete it until the immutable flag is removed.

Listing 52. Immutable files

```
root@pinguino:~# touch keep.me
root@pinguino:~# chattr +i keep.me
root@pinguino:~# lsattr keep.me
----i----- keep.me
root@pinguino:~# rm -f keep.me
rm: cannot remove `keep.me': Operation not permitted
root@pinguino:~# chattr -i keep.me
root@pinguino:~# rm -f keep.me
```

Changing the immutable flag requires root authority, or at least the `CAP_LINUX_IMMUTABLE` capability. Making files immutable is often done as part of a security or intrusion detection effort. See the capabilities man page (`man capabilities`) for more information.

The umask

When a new file is created, the creation process specifies the permissions that the new file should have. Often, the mode requested is 0666, which makes the file readable and writable by anyone. However, this permissive creation is affected by a *umask* value, which specifies what permissions a user does **not** want to grant automatically to newly created files or directories. The system uses the umask value to reduce the originally requested permissions. You can view your umask setting with the `umask` command, as shown in Listing 53.

Listing 53. Displaying octal umask

```
ian@pinguino:~$ umask
0022
```

Remember that the umask specifies which permissions should **not** be granted. On Linux systems, the umask normally defaults to 0022, which **removes** group and other write permission from new files. Use the `-S` option to display the umask symbolically, in a form that shows which are the permissions that **are** allowed.

You can use the `umask` command to set a umask as well as display one. So, if you would like to keep your files more private and disallow all group or other access to newly created files, you would use a umask value of 0077. Or set it symbolically using `umask u=rwx,g=,o=`, as illustrated in Listing 54.

Listing 54. Setting the umask

```
ian@pinguino:~$ umask
0022
ian@pinguino:~$ umask -S
u=rwx,g=rx,o=rx
```

```
ian@pinguino:~$ umask u=rwx,g=,o=
ian@pinguino:~$ umask
0077
ian@pinguino:~$ touch newfile
ian@pinguino:~$ ls -l newfile
-rw----- 1 ian ian 0 2005-12-26 12:49 newfile
```

The next section shows you how to change the owner and group of an existing filesystem object.

Section 7. Setting file owner and group

This section covers material for topic 1.104.6 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

In this section, you learn about:

- Changing a file's group
- Default group for new files
- Changing the owner of a file

In the previous section you learned how every filesystem object has an owner and a group. In this section you learn how to change the owner or group of an existing file and how the default group for new files can be set.

File group

To change the group of a file, use the `chgrp` command with a group name and one or more filenames. You may also use the group number if you prefer. An ordinary user must be a member of the group to which the file's group is being changed. The root user may change files to any group. Listing 55 shows an example.

Listing 55. Changing group ownership

```
ian@pinguino:~$ touch file1 file2
ian@pinguino:~$ ls -l file*
-rw-r--r-- 1 ian ian 0 2005-12-26 14:09 file1
-rw-r--r-- 1 ian ian 0 2005-12-26 14:09 file2
ian@pinguino:~$ chgrp xml-101 file1
ian@pinguino:~$ chgrp 1001 file2
ian@pinguino:~$ ls -l file*
-rw-r--r-- 1 ian xml-101 0 2005-12-26 14:09 file1
-rw-r--r-- 1 ian xml-101 0 2005-12-26 14:09 file2
```

As with many of the commands covered in this tutorial, `chgrp` has a `-R` option to allow changes to be applied recursively to all selected files and subdirectories.

Default group

In the [previous section](#) you learned how setting the `sgid` mode on a directory causes new files created in that directory to belong to the group of the directory rather than to the group of the user creating the file.

You may also use the `newgrp` command to temporarily change your primary group to another group of which you are a member. A new shell will be created, and when you exit the shell, your previous group will be reinstated, as shown in Listing 56.

Listing 56. Using `newgrp` to temporarily change default group

```
ian@pinguino:~$ newgrp xml-101
ian@pinguino:~$ groups
xml-101 adm dialout cdrom floppy audio dip video plugdev lpadmin scanner admin ian
ian@pinguino:~$ touch file3
ian@pinguino:~$ ls -l file3
-rw-r--r--  1 ian xml-101 0 2005-12-26 14:34 file3
ian@pinguino:~$ exit
ian@pinguino:~$ groups
ian adm dialout cdrom floppy audio dip video plugdev lpadmin scanner admin xml-101
```

File owner

The root user can change the ownership of a file using the `chown` command. In its simplest form, the syntax is like the `chgrp` command, except that a user name or numeric id is used instead of a group name or id. The file's group may be changed at the same time by adding a colon and a group name or id right after the user name or id. If only a colon is given, then the user's default group is used. Naturally, the `-R` option will apply the change recursively. Listing 57 shows an example.

Listing 57. Using `newgrp` to temporarily change default group

```
root@pinguino:~# ls -l ~ian/file4
-rw-r--r--  1 ian ian 0 2005-12-26 14:44 /home/ian/file4
root@pinguino:~# chown greg ~ian/file4
root@pinguino:~# ls -l ~ian/file4
-rw-r--r--  1 greg ian 0 2005-12-26 14:44 /home/ian/file4
root@pinguino:~# chown tom: ~ian/file4
root@pinguino:~# ls -l ~ian/file4
-rw-r--r--  1 tom xml-101 0 2005-12-26 14:44 /home/ian/file4
```

An older form of specifying both user and group used a dot instead of a colon. This

is no longer recommended as it may cause confusion when names include a dot.

Section 8. Hard and symbolic links

This section covers material for topic 1.104.7 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 1.

In this section, you learn about:

- Hard links
- Symbolic links

Hard links

In the tutorial for topic 103, "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)," you learned that a file or directory is contained in a collection of *blocks* and that information about the file or directory is contained in an *inode*.

A *hard link* is a pointer to an inode. So, a file name is really a link to the inode that contains information about the file. As you learned, you can use the `-i` option of the `ls` command to display inode numbers for file and directory entries.

You can use the `ln` command to create additional hard links to an existing file (but not to a directory, even though the system sets up `.` and `..` as hard links). If there are multiple hard links to an inode, then the inode is deleted only when the link count goes to zero.

Listing 58 shows how to create a file and then a hard link to it. It also shows that even though the original file name is removed, the second hard link prevents the inode from being erased when the first filename is removed.

Listing 58. Hard links

```
ian@pinguino:~$ echo testing > file1
ian@pinguino:~$ ls -l file*
-rw-r--r-- 1 ian ian 8 2005-12-26 15:35 file1
ian@pinguino:~$ ln file1 file2
ian@pinguino:~$ ls -l file*
-rw-r--r-- 2 ian ian 8 2005-12-26 15:35 file1
-rw-r--r-- 2 ian ian 8 2005-12-26 15:35 file2
ian@pinguino:~$ rm file1
ian@pinguino:~$ ls -l file*
-rw-r--r-- 1 ian ian 8 2005-12-26 15:35 file2
```

```
ian@pinguino:~$ cat file2
testing
```

Hard links may exist only within a particular filesystem. They cannot cross filesystems, since they refer to an inode by number, and inode numbers are only unique within a filesystem.

Finding hard links

If you need to find which files link to a particular inode, you can use the `find` command and the `-samefile` option with a filename or the `-inum` option with an inode number, as shown in Listing 59.

Listing 59. Finding hard links

```
ian@pinguino:~$ ln file2 file3
ian@pinguino:~$ ls -il file2
172 -rw-r--r--  2 ian ian 8 2005-12-26 15:35 file2
ian@pinguino:~$ find . -samefile file2
./file2
./file3
ian@pinguino:~$ find . -inum 172
./file2
./file3
```

Symbolic links

Another form of filesystem link that is used in Linux systems is a *symbolic link* (often called simply a *symlink*). In this case, the link refers to the name of another filesystem object rather than its inode. Symbolic links can refer to directories and can refer to files on other filesystems. They are frequently used to provide aliases for system commands. Using a long directory listing, you can see whether an object is a symbolic link when its first character is the lowercase letter `l`, as shown in Listing 60.

Listing 60. Symbolic link examples

```
ian@pinguino:~$ ls -l /sbin/mkfs.*
-rwxr-xr-x  1 root root  14160 2005-09-20 12:43 /sbin/mkfs.cramfs
-rwxr-xr-x  3 root root  31224 2005-08-23 09:25 /sbin/mkfs.ext2
-rwxr-xr-x  3 root root  31224 2005-08-23 09:25 /sbin/mkfs.ext3
-rwxr-xr-x  2 root root  55264 2005-06-24 07:48 /sbin/mkfs.jfs
-rwxr-xr-x  1 root root  13864 2005-09-20 12:43 /sbin/mkfs.minix
lrwxrwxrwx  1 root root    7 2005-12-14 07:40 /sbin/mkfs.msdos -> mkdosfs
-rwxr-xr-x  2 root root 241804 2005-05-11 09:40 /sbin/mkfs.reiser4
-rwxr-xr-x  2 root root 151020 2004-11-25 21:09 /sbin/mkfs.reiserfs
lrwxrwxrwx  1 root root    7 2005-12-14 07:40 /sbin/mkfs.vfat -> mkdosfs
-rwxr-xr-x  1 root root 303788 2005-04-14 01:27 /sbin/mkfs.xfs
```

In addition to the type of `l`, you can see on the right a `->` followed by the name that the link refers to. For example, the `mkfs.vfat` command is a symbolic link to the

`mkdosfs` command. You will find many other such links in `/sbin` and other system directories. Another tipoff is that the size is the number of characters in the link target's name.

You create a symlink using the `ln` command with the `-s` option as shown in Listing 61.

Listing 61. Creating symlinks

```
ian@pinguino:~$ touch file5
ian@pinguino:~$ ln -s file5 file6
ian@pinguino:~$ ln -s file5 file7
ian@pinguino:~$ ls -l file*
-rw-r--r--  2 ian ian  8 2005-12-26 15:35 file2
-rw-r--r--  2 ian ian  8 2005-12-26 15:35 file3
-rw-r--r--  1 ian ian  0 2005-12-26 17:40 file5
lrwxrwxrwx  1 ian ian  5 2005-12-26 17:40 file6 -> file5
lrwxrwxrwx  1 ian ian  5 2005-12-26 17:40 file7 -> file5
```

Note that the link counts in the directory listing are not updated. Deleting the link does not affect the target file. Symlinks do not prevent a file from being deleted; if the target file is moved or deleted, then the symlink will be broken. For this reason, many systems use colors in directory listings, often pale blue for a good link and red for a broken one.

Finding symbolic links

If you need to find which files link symbolically to a particular file, you can use the `find` command and the `-lname` option with a filename, as illustrated in Listing 62. Links may use a relative or absolute path, so you probably want a leading asterisk in the name to be matched.

Listing 62. Finding symbolic links

```
ian@pinguino:~$ mkdir linktest1
ian@pinguino:~$ ln -s ~/file3 linktest1/file8
ian@pinguino:~$ find . -lname "*file3"
./linktest1/file8
ian@pinguino:~$ find . -lname "*file5"
./file7
./file6
```

Paths and symlinks

In most of the examples we have seen so far, the symlink has been in the same directory as the target, and the paths in the link have been implicitly relative paths. In Listing 62 we created a link in the `linktest1` subdirectory, which used an absolute target (`~/file3`). When creating symlinks, you need to consider whether to use relative or absolute paths, since you may use either. Figure 3 illustrates the effect of moving a set of files and symlinks into a subdirectory.

Figure 3. Symlinks and paths

```

ian@pinguino:~$ mkdir linktest2
ian@pinguino:~$ ls file?
file2 file3 file5 file6 file7
ian@pinguino:~$ mv file? linktest2
ian@pinguino:~$ ls -l linktest1 linktest2
linktest1:
total 0
lrwxrwxrwx  1 ian ian 15 2005-12-26 18:07 file8 -> /home/ian/file3

linktest2:
total 8
-rw-r--r--  2 ian ian 8 2005-12-26 15:35 file2
-rw-r--r--  2 ian ian 8 2005-12-26 15:35 file3
-rw-r--r--  1 ian ian 0 2005-12-26 17:40 file5
lrwxrwxrwx  1 ian ian 5 2005-12-26 17:40 file6 -> file5
lrwxrwxrwx  1 ian ian 5 2005-12-26 17:40 file7 -> file5

```

The red color indicates that the linktest1/file8 link is now broken. This is not surprising as there is no longer a ~/file3. However, the two symlinks to file5 are still good as the file is still in the same relative location, even though it and the two links to it have moved. There are no hard and fast rules about whether to use relative or absolute paths in symbolic links; it depends somewhat on whether the link or the target is more likely to be moved. Just remember to consider the issue when making symbolic links.

Broken symlinks

One final point on our broken symbolic link. Attempts to read the file will fail as it does not exist. However, attempts to write it will work if you have the appropriate permission on the target file, as shown in Listing 63.

Listing 63. Reading from and writing to a broken symlink

```

ian@pinguino:~$ cat linktest1/file8
cat: linktest1/file8: No such file or directory
ian@pinguino:~$ echo "test file 8" >> linktest1/file8
ian@pinguino:~$ cat linktest1/file8
test file 8
ian@pinguino:~$ find . -name file3
./linktest2/file3
./file3

```

Since I can create files in my home directory, writing to the broken link created the missing target file.

Section 9. Finding and placing system files

This section covers material for topic 1.104.8 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn about:

- The Filesystem Hierarchy Standard and how files and directories are classified
- Finding files and commands

Filesystem Hierarchy Standard

The Filesystem Hierarchy Standard is a document that specifies the layout of directories on a Linux or other UNIX-like system. It was created to provide a common layout to simplify distribution-independent software development, by placing files in the same general place across Linux distributions. It is also used in the Linux Standard Base (see [Resources](#)).

The two independent FHS categories

At the core of the FHS are two independent characteristics of files:

Shareable vs. unshareable

Shareable files can be located on one system and used on another, while unshareable files must reside on the system on which they are used.

Variable vs. static

Static files include documentation, libraries, and binaries that do not change without system administrator intervention. Files that are not static are variable.

These distinctions allow files with different sets of characteristics to be stored on different filesystems. Table 6 is an example from the FHS document showing a layout that would be FHS-compliant.

	Shareable	Unshareable
Static	/usr /opt	/etc /boot
Variable	/var/mail /var/spool/news	/var/run /var/lock

The root filesystem

The FHS goal is to keep the root filesystem as small as possible. It must contain all the files necessary to boot, restore, recover, or repair the system, including the utilities that an experienced administrator would need for these tasks. Note that booting a system requires that enough be on the root filesystem to permit mounting of other filesystems.

Directories in the root

Table 7 shows the purpose of the directories that the FHS requires in the root (or /) filesystem. Either the directory or a symbolic link to it must be present, except for those marked optional, which are required only if the corresponding subsystem is present.

Directory	Purpose
bin	Essential command binaries
boot	Static files of the boot loader
dev	Device files
etc	Host-specific system configuration
lib	Essential shared libraries and kernel modules
media	Mount point for removable media
mnt	Mount point for mounting a filesystem temporarily
opt	Add-on application software packages
sbin	Essential system binaries
srv	Data for services provided by this system
tmp	Temporary files
usr	Secondary hierarchy
var	Variable data
home	User home directories (optional)
lib<qual>	Alternate format essential shared libraries (optional)
root	Home directory for the root user (optional)

/usr and /var

The /usr and /var hierarchies are complex enough to have complete sections of the FHS devoted to them. The /usr filesystem is the second major section of the filesystem, containing shareable, read-only data. It can be shared between systems,

although present practice does not often do this. The `/var` filesystem contains variable data files, including spool directories and files, administrative and logging data, and transient and temporary files. Some portions of `/var` are not shareable between different systems, but others, such as `/var/mail`, `/var/cache/man`, `/var/cache/fonts`, and `/var/spool/news` may be shared.

To fully understand the standard, read the FSH document (see [Resources](#)).

Where's that file?

Linux systems often contain hundreds of thousands of files. The newly installed Ubuntu system that we have been using in this tutorial has nearly 50000 files in the `/usr` hierarchy alone. A Fedora system that I have been using for some time has about 175000. The remainder of this section looks at tools to help you find files, particularly programs, in this vast sea of data.

Your PATH

If you have used several Linux systems, you may have noticed that if you log in as root, you are able to execute commands such as `fdisk`, which you apparently cannot execute if you are a user. What happens is that when you run a program at the command line, the bash (or other) shell searches through a list of directories to find the program you requested. The list of directories is specified in your `PATH` environment variable, and it is not uncommon for root's path to include `/sbin`, while non-root user paths do not. Listing 64 shows two different user path examples, as well as a root path example.

Listing 64. Some PATH examples

```
ian@pinguino:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11:/usr/games
[ian@attic4 ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/ian/bin
[ian@attic4 ~]$ su -
Password:
[root@attic4 ~]# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:
/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin
```

As you can see, the `PATH` variable is just a list of directory names, separated by colons. Since the `fdisk` command is actually located in `/sbin/fdisk`, only the first and last of these paths would allow the user to run it by typing `fdisk` without providing a fully qualified name (`/sbin/fdisk`).

Usually, your path is set in an initialization file such as `.bash_profile` or `.bashrc`. You can change it for the current session by specifying a new path. Remember to export

the PATH variable if you want the new value to be available to other processes that you start. An example is shown in Listing 65.

Listing 65. Changing your PATH

```
[ian@attic4 ~]$ fdisk
-bash: fdisk: command not found
[ian@attic4 ~]$ export PATH=/sbin:$PATH
[ian@attic4 ~]$ fdisk

Usage: fdisk [-l] [-b SSZ] [-u] device
E.g.: fdisk /dev/hda (for the first IDE disk)
      or: fdisk /dev/sdc (for the third SCSI disk)
      or: fdisk /dev/eda (for the first PS/2 ESDI drive)
      or: fdisk /dev/rd/c0d0 or: fdisk /dev/ida/c0d0 (for RAID devices)
      ...
```

which, type, and whereis

In the previous example we discovered that the `fdisk` command was not available only by attempting to run it. There are several commands that can help you do this too.

which

You can use the `which` command to search your path and find out which command will be executed (if any) when you type a command. Listing 66 shows an example of finding the `fdisk` command.

Listing 66. Using which

```
[ian@attic4 ~]$ which fdisk
/usr/bin/which: no fdisk in (/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:
/usr/X11R6/bin:/home/ian/bin)
[ian@attic4 ~]$ export PATH=/sbin:$PATH
[ian@attic4 ~]$ which fdisk
/sbin/fdisk
```

The `which` command shows you the first occurrence of a command in your path. If you want to know if there are multiple occurrences, then add the `-a` option as shown in Listing 67.

Listing 67. Using which to find multiple occurrences

```
[root@attic4 ~]# which awk
/bin/awk
[root@attic4 ~]# which -a awk
/bin/awk
/usr/bin/awk
```

Here we find the `awk` command in `/bin` (which contains commands that may be used by both the system administrator and by users, but which are required when no other filesystems are mounted) and also in `/sbin` (which contains the binaries essential for booting, restoring, recovering, and/or repairing the system).

type

There are some commands that the `which` command will not find, such as shell builtins. The `type` builtin will tell you how a given command string will be evaluated for execution. Listing 68 shows an example using the `type` command itself.

Listing 68. Using type

```
[root@attic4 ~]# which type
/usr/bin/which: no type in (/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:
/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin)
[root@attic4 ~]# type type
type is a shell builtin
```

whereis

If you want to find more information than just the location of a program, then you can use the `whereis` command. For example, you can find the man pages or other information, as shown in Listing 69.

Listing 69. Using whereis

```
[root@attic4 ~]# whereis awk
awk: /bin/awk /usr/bin/awk /usr/libexec/awk /usr/share/awk
/usr/share/man/man1p/awk.1p.gz /usr/share/man/man1/awk.1.gz
```

Note that the copy of `awk` in `/sbin` was not found by `whereis`. The directories used by `whereis` are fixed, so the command may not always find what you are looking for. The `whereis` command can also search for source files, specify alternate search paths, and search for unusual entries. Consult the man pages to see how to override this behavior or change the fixed paths used by `whereis`.

find

In the tutorial for topic 103, "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)," you learned how to find files based on name (including wildcards), path, size, or timestamp. In the earlier section on [Hard and symbolic links](#), you learned how to find the links to a particular file or inode.

The `find` command is the Swiss army knife of file searching tools on Linux systems. Two other capabilities that you may find useful are the ability to find files based on

user or group name and the ability to find files based on permissions.

Listing 70 shows a directory listing for our sample workgroup directory `~greg/xml101`, along with an example of how to find all the files owned by user `ian`, and all the ones that do not have the `xml-101` group. Note how the exclamation point, `!`, negates the sense of a test when using `find`.

Listing 70. Finding files by user and group

```
ian@pinguino:~$ ls -l ~greg/xml101/*
-rw-r--r--  1 greg xml-101 0 2005-12-27 07:38 /home/greg/xml101/file1.c
-rw-r-----  1 greg xml-101 0 2005-12-27 07:39 /home/greg/xml101/file2.c
-rw-r--r--  1 tom  xml-101 0 2005-12-27 07:41 /home/greg/xml101/file3.c
-rw-r--r--  1 ian  ian    0 2005-12-27 07:40 /home/greg/xml101/file4.c
-rw-r--r--  1 tom  xml-101 0 2005-12-27 07:41 /home/greg/xml101/file5.c
-rw-r--r--  1 ian  xml-101 0 2005-12-27 07:40 /home/greg/xml101/file6.c
-rw-r--r--  1 tom  xml-101 0 2005-12-27 07:43 /home/greg/xml101/file7.c
-rwxr-xr-x  1 tom  xml-101 0 2005-12-27 07:42 /home/greg/xml101/myprogram
ian@pinguino:~$ find ~greg/xml101 -user ian
/home/greg/xml101/file4.c
/home/greg/xml101/file6.c
ian@pinguino:~$ find ~greg/xml101 ! -group xml-101
/home/greg/xml101/file4.c
```

To find files by permission, you can use the `-perm` test along with symbolic expressions similar to those used with the `chmod` or `umask` commands. You can search for exact permissions, but it is often more useful to prefix the permission expression with a hyphen to indicate that you want files with those permissions set, but you don't care about other permissions. Using the files of Listing 70, Listing 71 illustrates how to find files that are executable by user and group, and two different ways of finding files that are not readable by others.

Listing 71. Finding files by permission

```
ian@pinguino:~$ find ~greg/xml101 -perm -ug=x
/home/greg/xml101
/home/greg/xml101/myprogram
ian@pinguino:~$ find ~greg/xml101 ! -perm -o=r
/home/greg/xml101/file2.c
ian@pinguino:~$ find ~greg/xml101 ! -perm -0004
/home/greg/xml101/file2.c
```

We have covered several major types of search that you can do with the `find` command. To further narrow your output, you can combine multiple expressions and you can add regular expressions to the mix. To learn more about this versatile command, use the man page, or better, use `info find` if you have the info system installed.

Listing 72 shows a final example of searching with `find`. This example does a `cd` to `/usr/include` to keep the listing length manageable, then finds all files containing `xt` in their path name without regard to case. The second example further restricts this

output to files that are not directories and that are at least 2 kilobytes in size. Actual output on your system may differ depending on what packages you have installed.

Listing 72. A final example of find

```
ian@pinguino:/usr/include$ find . -iregex ".*xt.*"
./X11/xterm
./X11/xterm/ptyx.h
./irssi/src/fe-common/core/printtext.h
./irssi/src/fe-common/core/highlight-text.h
ian@pinguino:/usr/include$ find . -iregex ".*xt.*" ! -type d -size +2k
./X11/xterm/ptyx.h
./irssi/src/fe-common/core/printtext.h
```

Note that the regular expression must match the full path returned by `find`, and remember the difference between regular expressions and wildcards.

locate and updatedb

The `find` command searches all the directories you specify, every time you run it. To speed things up, you can use another command, `locate`, which uses a database of stored path information rather than searching the filesystem every time.

locate and slocate

The `locate` command searches for matching files in a database that is usually updated daily by a cron job. On modern Linux systems, this command is usually replaced by the `slocate` command, which stores permissions as well as paths and thus prevents users from prying into directories that they could not otherwise see. On these systems you will usually find that `locate` is a symbolic link to `slocate`, so you can use either command.

The `locate` command matches against any part of a pathname, not just the file itself. Listing 73 shows that `locate` is a symlink to `slocate` and then shows how to find paths containing the string `bin/ls`.

Listing 73. Using locate

```
[ian@attic4 ~]$ ls -l $(which locate)
lrwxrwxrwx 1 root slocate 7 Aug 24 23:04 /usr/bin/locate -> slocate
[ian@attic4 ~]$ locate bin/ls
/bin/ls
/usr/bin/lsb_release
/usr/bin/lscatproc
/usr/bin/lspgpot
/usr/bin/lsattr
/usr/bin/lscat
/usr/bin/lshal
/usr/bin/lstdiff
/usr/sbin/lsof
```

```
/sbin/lsmmod  
/sbin/lsub  
/sbin/lspci
```

updatedb

The database used by `slocate` is stored in the `/var` filesystem, in a location such as `/var/lib/slocate/slocate.db`. If you see output such as Listing 74, then your system is not running this job.

Listing 74. No database for slocate

```
[ian@attic4 ~]$ locate bin/ls  
warning: locate: could not open database: /var/lib/slocate/slocate.db: No such file or  
directory  
warning: You need to run the 'updatedb' command (as root) to create the database.  
Please edit /etc/updatedb.conf to enable the daily cron job.
```

The database is created or updated using the `updatedb` command. This is usually run daily as a cron job. The file `/etc/updatedb.conf` is the configuration file for `updatedb`. To enable daily updates, the root user needs to edit `/etc/updatedb.conf` and set `DAILY_UPDATE=yes`. To create the database immediately, run the `updatedb` command as root.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- In the [Proceedings of the Linux Symposium, Volume One](#) (July 2005, Ottawa, Canada), learn more about ext3 in "State of the Art: Where we are with the Ext3 filesystem," a paper by M. Cao, et. al.
- [XFS: A high-performance journaling filesystem](#) is the home page of the XFS project at SGI.
- [Reiser4](#) is the next version of the ReiserFS filesystem.
- [qtparted](#) is a graphical partitioning tool that uses the Qt toolkit.
- [gparted](#) is a graphical partitioning tool designed for the GNOME desktop; it uses the GTK+GUI library.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- The [Quota mini-HOWTO](#) can help answer questions on quotas.
- Visit the home of the [Filesystem Hierarchy Standard](#) (FHS).
- "[Advanced filesystem implementor's guide, Part 3](#)" tells you more about the tmpfs virtual memory filesystem.
- At the [LSB home page](#), learn about The Linux Standard Base (LSB), a Free Standards Group (FSG) project to develop a standard binary operating environment.
- [LPI Linux Certification in a Nutshell](#) (O'Reilly, 2001) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are references for those who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content.](#)
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep: Kernel

Junior Level Administration (LPIC-1) topic 105

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

21 Mar 2006

In this tutorial, Ian Shields begins preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this first in a [series of nine tutorials](#), Ian introduces you to the kernel on Linux®. By the end of this tutorial, you will know how to build, install, and query a Linux kernel and its kernel modules.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics

LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	(This tutorial). Learn how to install and maintain Linux kernels and kernel modules. See detailed objectives below.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Coming soon.
Topic 107	LPI exam 102 prep: Printing	Coming soon.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Kernel," the first of nine tutorials designed to prepare you for LPI exam

102. In this tutorial, you learn how to build, install, and query a Linux kernel and its kernel modules.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.105.1 Manage and query kernel and kernel modules at runtime	Weight 4	Learn to query and manage a kernel and kernel-loadable modules.
1.105.2 Reconfigure, build, and install a custom kernel and kernel modules	Weight 3	Learn to customize, build, and install a kernel and kernel-loadable modules from source.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be very familiar with the material from [LPI exam 101 prep: Hardware and architecture](#) tutorial.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Runtime kernel management

This section covers material for topic 1.105.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use command-line utilities to get information about the currently running kernel and kernel modules
- Manually load and unload kernel modules

- Determine when modules can be unloaded
- Configure the system to load modules by names other than their file name

Technically, Linux is the kernel of your system. The kernel provides a framework for applications to run and use various hardware devices. It is low-level code that deals with hardware interfaces, scheduling, and memory management among other things. Many people refer to a whole system as GNU/Linux because many of the tools that make most distributions usable come from the GNU project of the Free Software Foundation. Nevertheless, you will often just see "Linux" instead of "GNU/Linux."

uname

The `uname` command prints information about your system and its kernel. Listing 1 shows the various options for `uname` and the resulting information; each option is defined in Table 3.

Listing 1. The `uname` command

```

ian@pinguino:~$ uname
Linux
ian@pinguino:~$ uname -s
Linux
ian@pinguino:~$ uname -n
pinguino
ian@pinguino:~$ uname -r
2.6.12-10-386
ian@pinguino:~$ uname -v
#1 Mon Jan 16 17:18:08 UTC 2006
ian@pinguino:~$ uname -m
i686
ian@pinguino:~$ uname -o
GNU/Linux
ian@pinguino:~$ uname -a
Linux pinguino 2.6.12-10-386 #1 Mon Jan 16 17:18:08 UTC 2006 i686 GNU/Linux
    
```

Table 3. Options for <code>uname</code>	
Option	Description
-s	Print the kernel name. This is the default if no option is specified.
-n	Print the nodename or hostname.
-r	Print the release of the kernel. This option is often used with module-handling commands.

-v	Print the version of the kernel.
-m	Print the machine's hardware (CPU) name.
-o	Print the operating system name.
-a	Print all of the above information.

Listing 1 is from an Ubuntu system running on an Intel® CPU. The `uname` command is available on most UNIX® and UNIX-like systems as well as Linux. The information printed will vary by Linux distribution and version as well as by the type of machine you are running on. Listing 2 shows the output from an AMD Athlon 64 system running Fedora Core 4 and, for comparison, an Apple PowerBook.

Listing 2. Using `uname` with another system

```
Linux attic4 2.6.14-1.1656_FC4 #1 Thu Jan 5 22:13:55 EST 2006 x86_64
x86_64 x86_64 GNU/Linuxfilesystem

Darwin Ian-Shields-Computer.local 7.9.0 Darwin Kernel Version 7.9.0:
Wed Mar 30 20:11:17 PST 2005; root:xnu/xnu-517.12.7.obj~1/RELEASE_PPC
Power Macintosh powerpc
```

Kernel modules

The kernel manages many of the low-level aspects of your system, including hardware and interfaces. With a large variety of possible hardware and several different file systems, a kernel that supported everything would be rather large. Fortunately, *kernel modules* allow you to load support software such as hardware drivers or file systems when needed, so you can start your system with a small kernel and then load other modules as needed. Often the loading is automatic, such as when USB devices are plugged in.

The remainder of this section looks at the commands and configuration for kernel modules.

The commands for tasks such as loading or unloading modules require root authority. The commands for displaying information about modules can usually be run by general users. However, since they reside in `/sbin`, they are not usually on a non-root user's path, so you will probably have to use full path names if you are not root.

lsmod

Use the `lsmod` command to display the modules that are currently loaded on your system, as shown in Listing 3. Your output is likely to be different, although you should see some common entries.

Listing 3. Displaying kernel modules with `lsmod`

```
[ian@attic4 ~]$ /sbin/lsmod
Module              Size Used by
nvnet               74148 0
nvidia             4092336 12
forcedeth          24129 0
md5                 4161 1
ipv6                268737 12
parport_pc         29189 1
lp                 13129 0
parport            40969 2 parport_pc,lp
autofs4            29637 1
sunrpc             168453 1
ipt_REJECT         5825 1
ipt_state          1985 3
ip_conntrack       42009 1 ipt_state
iptables_filter    3137 1
ip_tables          19521 3 ipt_REJECT,ipt_state,iptables_filter
dm_mod             58613 0
video              16069 0
button             4161 0
battery            9541 0
ac                 4933 0
ohci_hcd           26977 0
ehci_hcd           41165 0
i2c_nforce2        7105 0
i2c_core           21825 1 i2c_nforce2
shpchp             94661 0
snd_intel8x0       34945 1
snd_ac97_codec     76217 1 snd_intel8x0
snd_seq_dummy      3781 0
snd_seq_oss        37569 0
snd_seq_midi_event 9409 1 snd_seq_oss
snd_seq            62801 5 snd_seq_dummy,snd_seq_oss,snd_seq_midi_event
snd_seq_device     9037 3 snd_seq_dummy,snd_seq_oss,snd_seq
snd_pcm_oss        51569 0
snd_mixer_oss     18113 1 snd_pcm_oss
snd_pcm            100553 3 snd_intel8x0,snd_ac97_codec,snd_pcm_oss
snd_timer         33733 2 snd_seq,snd_pcm
snd                57669 11 snd_intel8x0,snd_ac97_codec,snd_seq_oss,snd_seq,
snd_seq_device,snd_pcm_oss,snd_mixer_oss,snd_pcm,snd_timer
soundcore          11169 1 snd
snd_page_alloc     9925 2 snd_intel8x0,snd_pcm
floppy             65397 0
ext3               132681 3
jbd                86233 1 ext3
sata_nv            9541 0
libata             47301 1 sata_nv
sd_mod             20545 0
scsi_mod           147977 2 libata,sd_mod
[ian@attic4 ~]$
```

You can see that this system has many loaded modules. Most of these are supplied with the kernel. However, some, such as the `nvnet`, `nvidia`, and `sata_nv` modules from NVIDIA Corporation include proprietary code and are not supplied as part of a standard kernel. In this way, the modular approach allows proprietary code to be plugged in to an open source kernel. Assuming the vendor license permits it, a Linux

distributor may add proprietary modules to a distribution, saving you the effort of getting them directly from a vendor and helping to ensure that you have the appropriate levels.

From Listing 3, you can also see that modular support extends to devices such as video, SATA and SCSI hard drives, floppy disks, and sound cards, as well as to networking features such as IPV6, file system support such as ext3, and Sun remote procedure call (RPC).

In addition to the module name, `lsmod` also shows the module size and the number of users of the module. If the module is used by any other modules, these are listed. So, for example, the `soundcore` module is used by the `snd` module, which in turn is used by several other sound modules.

modinfo

The `modinfo` command displays information about one or more modules. As shown in Listing 4, the information includes the full path to the file, the author, license, any parameters that the module might accept, version, dependencies, and other information.

Listing 4. Basic module information

```
[ian@attic4 ~]$ /sbin/modinfo floppy
filename:      /lib/modules/2.6.12-1.1456_FC4/kernel/drivers/block/floppy.ko
author:       Alain L. Knaff
license:      GPL
alias:        block-major-2-*
vermagic:     2.6.12-1.1456_FC4 686 REGPARM 4KSTACKS gcc-4.0
depends:
srcversion:   2633BC999A0747D8D215F1F
parm:         FLOPPY_DMA:int
parm:         FLOPPY_IRQ:int
parm:         floppy:charp
[ian@attic4 ~]$ /sbin/modinfo sata_nv
filename:      /lib/modules/2.6.12-1.1456_FC4/kernel/drivers/scsi/sata_nv.ko
author:       NVIDIA
description:   low-level driver for NVIDIA nForce SATA controller
license:      GPL
version:      0.6
vermagic:     2.6.12-1.1456_FC4 686 REGPARM 4KSTACKS gcc-4.0
depends:       libata
alias:        pci:v000010DEd00000008Esv*sd*bc*sc*i*
alias:        pci:v000010DEd000000E3sv*sd*bc*sc*i*
alias:        pci:v000010DEd000000EEsv*sd*bc*sc*i*
alias:        pci:v000010DEd00000054sv*sd*bc*sc*i*
alias:        pci:v000010DEd00000055sv*sd*bc*sc*i*
alias:        pci:v000010DEd00000036sv*sd*bc*sc*i*
alias:        pci:v000010DEd0000003Esv*sd*bc*sc*i*
alias:        pci:v000010DEd*sv*sd*bc01sc01i*
srcversion:   3094AD48C1B869BCC301E9F
```

In Listing 4, notice in the lines giving the module filenames that these filenames end in a `.ko` suffix. This distinguishes modules for 2.6 kernels from other object files and

from modules for 2.4 and earlier kernels, which used the same .o suffix as other object files.

You will also notice that the module path include the kernel version. For example, `/lib/modules/2.6.12-1.1456_FC4/kernel/drivers/block/floppy.ko` includes `2.6.12-1.1456_FC4` as a path element. This is the same value emitted by `uname -r`. Kernel modules are specific to a given kernel, and this structure manages that relationship.

On 2.6 kernels you can also use `modinfo` to limit requests to specific information about a module. Use the `-F` option to extract a single information type, such as `parm`, `description`, `license`, `filename`, or `alias`. Use the command multiple times with different options if you need different types of information. On 2.4 kernels, parameters such as `-p` extracted parameter information. The current `modinfo` command also supports the older parameters. Listing 5 shows some examples.

Listing 5. Specific module information

```
[ian@attic4 ~]$ /sbin/modinfo -F parm snd
cards_limit:Count of auto-loadable soundcards.
major:Major # for sound driver.
[ian@attic4 ~]$ /sbin/modinfo -F license nvidia floppy
NVIDIA
GPL
[ian@attic4 ~]$ /sbin/modinfo -p snd
major:Major # for sound driver.
cards_limit:Count of auto-loadable soundcards.
```

Using your Linux skills

You may use some of the techniques covered in the tutorial "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)" to extract information such as the number of parameters accepted by any module that accepts parameters. Listing 6 shows an example.

Listing 6. Number of parameters per module

```
[ian@attic4 ~]$ for n in `/sbin/lsmmod | tail +2 | cut -d " " -f1`;
> do echo "$n $(/sbin/modinfo -p $n | wc -l )" | grep -v " 0$"; done
nvnet 12
forcedeth 1
parport_pc 5
dm_mod 1
ohci_hcd 2
ehci_hcd 2
shpchp 3
snd_intel8x0 7
snd_ac97_codec 1
snd_seq_dummy 2
snd_seq_oss 2
snd_seq 7
snd_pcm_oss 3
snd_pcm 2
snd_timer 1
snd 2
```

```
snd_page_alloc 1
scsi_mod 6
```

rmmod

If a module's use count is 0, you may safely remove it. For example, you might do this in preparation for loading an updated version. This is a great feature of a modular kernel because you do not always have to reboot just to update support for one or another particular device. To remove a mod, use the `rmmod` command along with the module name as shown in Listing 7.

Listing 7. Removing a module for a running system

```
[root@attic4 ~]# rmmod floppy
```

Consult the man pages for other options available with `rmmod`.

insmod and modprobe

Once you have removed a module, you may need to reload it. You can do this using the `insmod` command, which takes the full path name of the module to be reloaded, along with any module options that may be required. If you use this command, you will probably want to use command substitution for generating the filename. Two ways of doing this are shown in Listing 8.

Listing 8. Loading a module using insmod

```
[root@attic4 ~]# insmod /lib/modules/`uname
-r`/kernel/drivers/block/floppy.ko
[root@attic4 ~]# rmmod floppy
[root@attic4 ~]# insmod $(modinfo -F filename floppy)
```

The second form above saves you the need to remember which subdirectory (drivers/block in this case) a module is located in, but there is an even better way to load a module. The `modprobe` command provides a higher-level interface that operates with the module name instead of file path. It also handles loading additional modules upon which a module depends, and can remove modules as well as load them.

Listing 9 shows how to use `modprobe` to remove the `vfat` module, along with the `fat` module that uses it. It then shows what the system would do if the module were reloaded, and finally the result of reloading the module. Note that the `-v` option is specified to obtain verbose output; otherwise, `modprobe` (and the underlying `insmod` command) will display only error messages from the module itself. Between

each step, the output of `lsmod` is piped through `grep` to show whether either the `vfat` or `fat` module is loaded or not.

Listing 9. Loading a module using `modprobe`

```
[root@lyrebird root]# modprobe -r vfat
vfat: Device or resource busy
[root@lyrebird root]# lsmod | grep fat
vfat          13132    1
fat           38744    0 [vfat]
[root@lyrebird root]# umount /windows/D
[root@lyrebird root]# modprobe -r vfat
[root@lyrebird root]# modprobe -v --show vfat
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
[root@lyrebird root]# lsmod | grep fat
[root@lyrebird root]# modprobe -v vfat
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
Using /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
Symbol version prefix ''
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
Using /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
[root@lyrebird root]# lsmod | grep fat
vfat          13132    0 (unused)
fat           38744    0 [vfat]
```

depmod

You have just seen that `modprobe` can handle the automatic loading of multiple modules when some are dependent on others. The dependencies are kept in the `modules.dep` file in the `/lib/modules` subdirectory for the appropriate kernel, as given by the `uname -r` command. This file, along with several map files, is generated by the `depmod` command. The `-a` (for *all*) is now optional.

The `depmod` command scans the modules in the subdirectories of `/lib/modules` for the kernel you are working on and freshens the dependency information. An example, along with the resulting changed files, is shown in Listing 10.

Listing 10. Using `depmod` to rebuild `modules.dep`

```
[root@lyrebird root]# date
Thu Mar 16 10:41:05 EST 2006
[root@lyrebird root]# depmod
[root@lyrebird root]# cd /lib/modules/`uname -r`
[root@lyrebird 2.4.21-37.0.1.EL]# ls -l mod*
-rw-rw-r-- 1 root root 54194 Mar 16 10:41 modules.dep
-rw-rw-r-- 1 root root 31 Mar 16 10:41 modules.generic_string
-rw-rw-r-- 1 root root 73 Mar 16 10:41 modules.ieee1394map
-rw-rw-r-- 1 root root 1614 Mar 16 10:41 modules.isapnpmap
-rw-rw-r-- 1 root root 29 Mar 16 10:41 modules.parpportmap
-rw-rw-r-- 1 root root 65171 Mar 16 10:41 modules.pcimmap
-rw-rw-r-- 1 root root 24 Mar 16 10:41 modules.pnpbiosmap
-rw-rw-r-- 1 root root 122953 Mar 16 10:41 modules.usbmap
[root@lyrebird 2.4.21-37.0.1.EL]# cd -
/root
```

You can customize the behavior of `modprobe` and `depmod` using the configuration file `/etc/modules.conf`. This is commonly used to alias module names and to specify commands that should be run after a module is loaded or before it is unloaded. However, an extensive range of other configuration can be done. Listing 11 shows an example of `/etc/modules.conf`. Consult the man page for `modules.conf` for more details.

Listing 11. Example `/etc/modules` file

```
[root@lyrebird root]# cat /etc/modules.conf
alias eth0 e100
alias usb-controller usb-uhci
alias usb-controller1 ehci-hcd
alias sound-slot-0 i810_audio
post-install sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -L >/dev/null 2>&1 || :
pre-remove sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -S >/dev/null 2>&1 || :
```

You should also be aware that some systems use another configuration file called `modprobe.conf`, while others store module configuration information in the `/etc/modules.d` directory. You may also find a file called `/etc/modules` on some systems; this file contains the names of kernel modules that should be loaded at boot time.

USB modules

When you hot plug a USB device into your Linux system, the kernel must determine which modules to load to handle the device. This is usually done for you by a hot plug script that uses the `usbmodules` command to find the appropriate module. You can also run `usbmodules` (as root) to see for yourself. Listing 12 shows an example.

Listing 12. USB modules

```
root@pinguino:~# lsusb
Bus 005 Device 004: ID 1058:0401 Western Digital Technologies, Inc.
Bus 005 Device 003: ID 054c:0220 Sony Corp.
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 003: ID 04b3:310b IBM Corp. Red Wheel Mouse
Bus 001 Device 001: ID 0000:0000
root@pinguino:~# usbmodules --device /proc/bus/usb/005/003
usb-storage
root@pinguino:~# usbmodules --device /proc/bus/usb/001/003
usbmouse
usbhid
```

The next section shows you how to build and configure a custom kernel.

Section 3. Customize and build kernels and kernel modules

This section covers material for topic 1.105.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Customize the current kernel configuration
- Build a new kernel and appropriate kernel modules
- Install a new kernel and any modules
- Ensure that the boot manager can locate the new kernel and associated files

As you learned in the previous section, [Runtime kernel management](#), the kernel provides the low-level support for your system hardware and file systems. A modern kernel image usually contains only essential functions, but is configured to support additional functions that you might need through the use of *kernel modules*. The additional support is loaded only when needed, for example when a device is plugged in or otherwise enabled.

The modular code becomes an integral part of the kernel, dynamically extending the kernel functions. If the functions of a loaded kernel module have not been used for several minutes, the kernel can voluntarily disassociate it from the rest of the kernel and unload it from memory through a process known as *autocleaning*.

Without kernel modules, your running kernel, which is loaded from disk as a single binary file, would have to contain all the functionality you might possibly ever need. You would also need to build a completely new kernel every time you wanted to add functionality to your system.

You cannot put *everything* in a module, however. At a bare minimum, the kernel image that is loaded must be able to mount your root file system. But, as you learned in the tutorial "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)," your boot loader can load an *initial RAM disk* (or *initrd*), which may contain the modules necessary to mount the root file system. Nevertheless, the kernel image must at least contain support for the RAM file system used in the initial RAM disk. If it does not, your system will not boot.

Once your system has bootstrapped itself this far, it proceeds to mount the root file system and then start the other initialization processes. After a few seconds, the

system is up and ready for you to use. The kernel, however, remains in control awaiting requests to perform work for user processes and scheduling the system resources among the tasks that require them.

Modular kernels work well in modern systems with plenty of RAM and disk space. However, you may have a new piece of hardware, such as a video card or storage system, that is not supported by the kernel that came with your distribution. Indeed, some drivers contain proprietary code that is said to *taint* a pure Linux kernel, so some distributors will not include it, even if the vendor license terms permit it to be distributed by your chosen distributor. In this case, you will need to at least build new modules, and possibly even build a new kernel.

Linux can be used in many environments, from embedded systems such as mobile phones, to networking devices such as routers, to set-top boxes as well as more traditional computing environments. Some of these devices use a kernel that is customized to support only those functions that the system is intended to support. For example, a system intended to be a diskless firewall probably does not need support for any file system other than the read-only file system from which it loaded, yet it may need support for advanced networking hardware that is not part of a standard kernel. Again, a custom kernel will be required.

Source packages

The ultimate source for the Linux kernel is the Linux Kernel Archives (see [Resources](#) for a link). Unless you already know what you are doing, you should use a kernel package from your Linux distribution, because your distributor may have added custom patches. If you are already familiar with obtaining and extracting source packages, review the tutorial "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)." As with anything that may change your system, make backups first so that you can recover if things go wrong.

If you download source from the public kernel archives, you will download a compressed file, and you will need to decompress it using `gzip` or `bzip2`, according to whether you download the `.gz` or the `.bz2` version of the kernel source. The `pub/linux/kernel/` directory on the download server has a directory for each kernel version, such as 2.4, 2.5, or 2.6. At the date of this writing, the latest `bzip2` version of the 2.6 kernel is `linux-2.6.15.tar.bz2`.

In that kernel directory, you will also see a corresponding `ChangeLog-2.6.15.6` file that describes changes in this version, and a `patch-2.6.15.bz2`, which is a smaller file that allows you to patch the prior version of source to bring it up to 2.6.15 level. You will also notice signature files that you may use to verify that your downloaded file was not corrupted, either accidentally or maliciously.

The compressed source is normally uncompressed in `/usr/src`, and it creates a new

subdirectory for the kernel version, such as `linux-2.6.15`, containing the tree of files needed to build your kernel. If you already have such a directory, you may want to back it up or rename it before unpacking the new kernel source. This will ensure that you can go back if you need to, and also that you will not have stray files that should not be in your kernel source tree. You need about 40MB of disk space for the tarball and about 350MB for the expanded source code.

Some distributors, notably Red Hat, now distribute the kernel headers and source necessary for building kernel modules as a kernel development package. Documentation may be in a separate kernel documentation package. These are designed for and sufficient for building modules, such as a proprietary vendor graphics card module, but they are not sufficient for rebuilding a custom kernel. Your distribution should have information about how to rebuild a kernel and how the source can be obtained. Check for documentation such as release notes.

Suppose you use FTP or HTTP to download the `kernel-2.6.15-1.1833_FC4.src.rpm` source RPM from the `pub/fedora/linux/core/updates/4/SRPMS/` at `download.fedora.redhat.com`, and the file is in the `/root` directory. Note that version numbers used here will probably be different for your system, so make sure you get the updated version of source corresponding to your installed kernel. Now, for Fedora, you must install the source RPM, then switch to the `/usr/src/redhat/SPECS` directory, and finally build the source RPM in order to create the Linux kernel source tree as shown in Listing 13.

Listing 13. Creating the kernel source tree for Fedora Core

```
[root@attic4 ~]# uname -r
2.6.15-1.1833_FC4
[root@attic4 ~]# rpm -Uvh kernel-2.6.15-1.1833_FC4.src.rpm
 1:kernel ##### [100%]
[root@attic4 ~]# cd /usr/src/redhat/SPECS
[root@attic4 SPECS]# rpmbuild -bp --target $(arch) kernel-2.6.spec
Building target platforms: x86_64
Building for target x86_64
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.23188
+ umask 022
+ cd /usr/src/redhat/BUILD
+ LANG=C
+ export LANG
+ unset DISPLAY
+ '[' '!' -d kernel-2.6.15/vanilla ']'
+ cd /usr/src/redhat/BUILD
+ rm -rf kernel-2.6.15
+ /bin/mkdir -p kernel-2.6.15
+ cd kernel-2.6.15
+ /usr/bin/bzip2 -dc /usr/src/redhat/SOURCES/linux-2.6.15.tar.bz2
+ tar -xf -
+ . . .
+ echo '# x86_64'
+ cat .config
+ perl -p -i -e 's/^SUBLEVEL.*/SUBLEVEL = 15/' Makefile
+ perl -p -i -e 's/^EXTRAVERSION.*/EXTRAVERSION = -prep/' Makefile
+ find . -name '*.orig' -o -name '*~' -exec rm -f '{}' ';'
+ exit 0
```

The Linux kernel source for Fedora is now located in `/usr/src/redhat/BUILD/kernel-2.6.15/linux-2.6.15`. By convention, the `/linux-2.6.15` tree is often moved to `/usr/src` and symbolically linked to `/usr/src/linux`, as shown in Listing 14. This is not strictly necessary, but it's easier to follow along with references that assume the kernel source tree will be in `.usr./src/linux`.

Listing 14. Moving the source tree to `/usr/src`

```
[root@attic4 SPECS]# mv ../BUILD/kernel-2.6.15/linux-2.6.15 /usr/src
[root@attic4 SPECS]# cd /usr/src
[root@attic4 src]# ln -s linux-2.6.15 linux
[root@attic4 src]# ls -ld lin*
lrwxrwxrwx  1 root root  12 Mar 20 18:23 linux -> linux-2.6.15
drwxr-xr-x 20 root root 4096 Mar 20 18:13 linux-2.6.15
```

Before you attempt to build anything, review the `Changes` file that is located in the `Documentation` directory. Among other things, it lists the minimum levels of various software packages that you need to build a kernel. Make sure that you have these packages installed.

You may notice `Makefile` and `.config` among the files shown in Listing 13. The `make` file contains several `make` targets for tasks such as configuring the kernel options, building the kernel and its modules, and installing the modules and building RPM or deb packages. More recent kernel sources allow you to use `make help` for brief help on each target. For older systems, you may need to consult the documentation or examine the `make` file. Listing 15 shows partial output for `make help`.

Listing 15. Help for kernel building `make` file

```
[ian@attic4 linux-2.6.15]$ make help
Cleaning targets:
  clean          - remove most generated files but keep the config
  mrproper       - remove all generated files + config + various backup files

Configuration targets:
  config         - Update current config utilising a line-oriented program
  menuconfig     - Update current config utilising a menu based program
  xconfig        - Update current config utilising a QT based front-end
  gconfig        - Update current config utilising a GTK based front-end
  oldconfig      - Update current config utilising a provided .config as base
  randconfig     - New config with random answer to all options
  defconfig      - New config with default answer to all options
  allmodconfig   - New config selecting modules when possible
  allyesconfig   - New config where all options are accepted with yes
  allnoconfig    - New minimal config

Other generic targets:
  all            - Build all targets marked with [*]
  * vmlinux      - Build the bare kernel
  * modules      - Build all modules
  modules_install - Install all modules
  dir/           - Build all files in dir and below
  dir/file.[ois] - Build specified target only
  ...
```

Configuration

The `.config` file in your kernel build directory contains configuration information for your kernel, including the target machine environment, the features to be included, and whether a feature should be compiled into the kernel or built as a module. Creating a `.config` file is the first step to building or rebuilding a kernel. You create it using one of the configuration targets in the make file.

The main configuration options are:

config

The `config` target uses a command-line interface to obtain answers to many questions to either build or update your `.config` file. With the advent of the menu-based configuration targets, this command-line interface is rarely used today.

menuconfig

The `menuconfig` target uses an ncurses-based, menu-based program to create or update your `.config` file. You need only answer questions for items you want to change. This approach has superseded the older `config` target. You run this in a terminal window either remotely or locally.

xconfig

The `xconfig` target uses a graphical menu system based on a QT front-end, like the one used with the KDE desktop.

gconfig

The `gconfig` target uses a graphical menu system based on a GTK front-end, like the one used with the GNOME desktop.

oldconfig

The `oldconfig` target allows you to build a configuration using an existing `.config` file, such as you might have from a previous build or another system. For example, if you installed the kernel source for Fedora as described above, you may copy the configuration file for your running system from `/lib/modules/$(uname -r)/build/.config` to `/usr/src/linux`. Once you've built it, you may use one of the menu configuration targets to modify it if necessary.

Figure 1 shows what you might see if you run `make menuconfig` for a 2.4 series kernel. Press **Enter** to descend into lower-level menus, and press **Esc** to return. Help is available for most items. Either tab to the **< Help >** button and press **Enter**, or simply type **h**. Press **Esc** to return to configuring.

Figure 1. Running make menuconfig on a 2.4 kernel

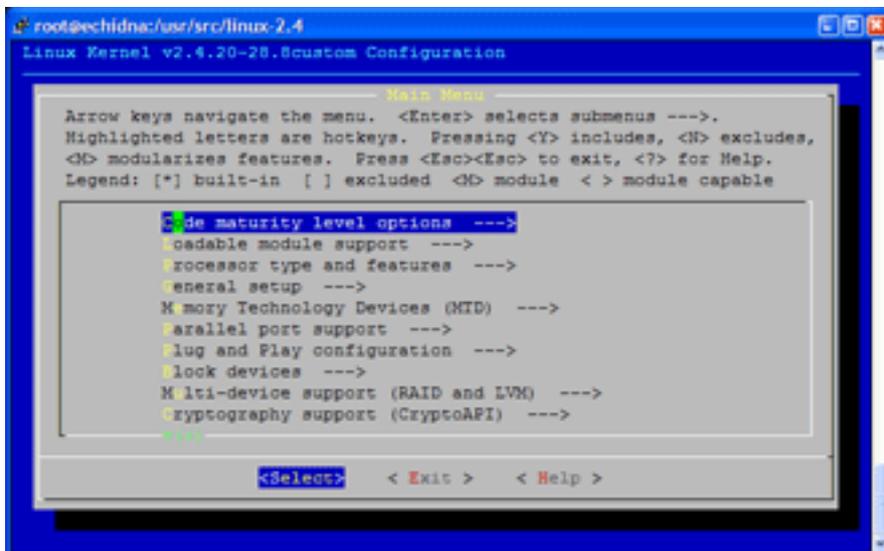
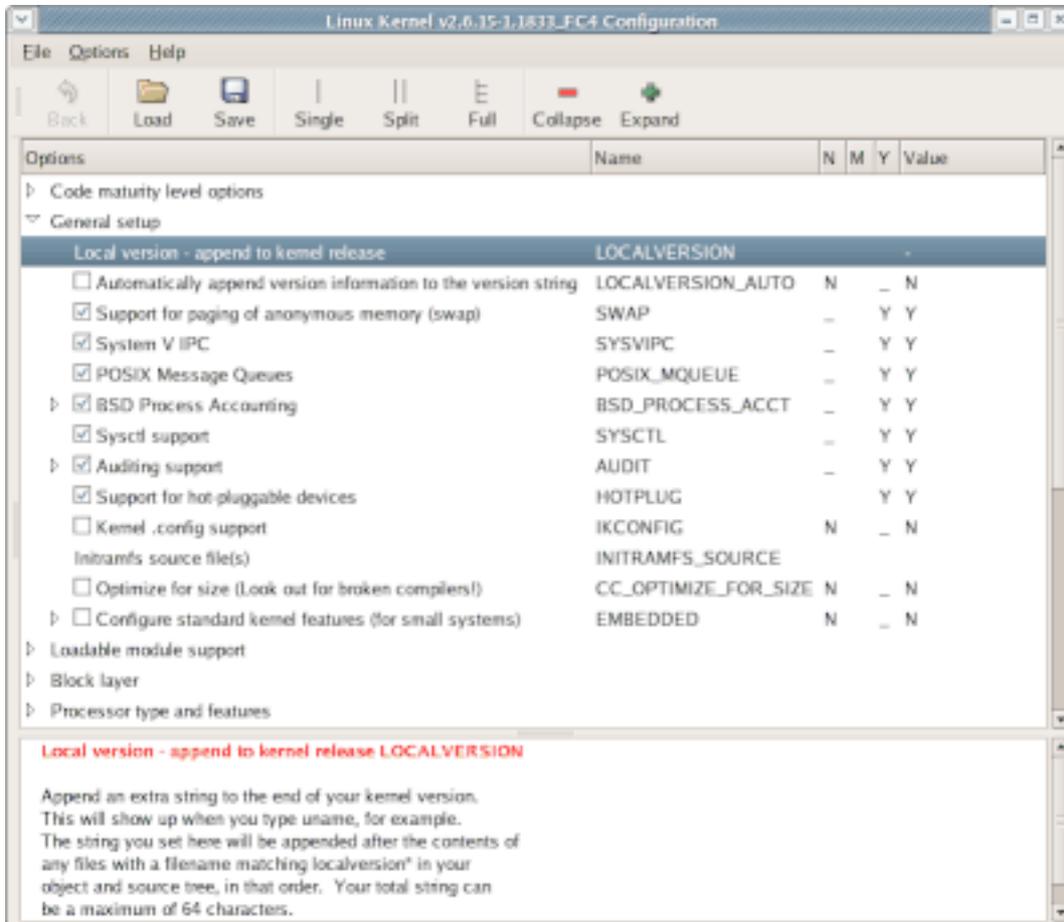


Table 4 shows the various options for including features in the kernel, either built in or as modules. When an option is highlighted, press the space bar to toggle between the allowable choices for that feature. You can also press **y** to enable an option, **n** to disable it, or **m** to have it compiled as a module if possible.

Table 4. Options for menuconfig	
Option	Description
[*]	Feature will be built into the kernel.
[]	Feature will not be included in the kernel.
<M>	Feature will be built as a kernel module.
< >	Feature will not be included in the kernel but is capable of being built as a module.

Figure 2 shows what you might see if you run `make gconfig` for a 2.6 series kernel. Click the arrows to expand or collapse menu items. Help is displayed in a lower pane.

Figure 2. Running make gconfig on a 2.6 kernel



The major configuration sections for a 2.6 kernel are described below. You may not find all of these with 2.4 and earlier kernels, but this list gives you an overview of where to find what.

Code maturity level options

This section contains an option that determines whether remaining options give you a choice for code that is considered experimental. If you do not select this option, then you will be able to select only options that are considered stable. Be warned that functions you choose may or may not work at the current code level on your system, so you might have a chance to help with debugging.

General setup

This section lets you include an identification string with your new kernel, along with options for several kernel attributes that do not belong elsewhere but that you must specify.

Loadable module support

This section contains an option that determines whether your kernel will support modules and whether they may be automatically loaded and unloaded. You should enable module support.

Block layer

This section contains support for disks larger than 2TB, and allows you to choose the type of disk scheduling that you would like.

Processor type and features

This section contains CPU-specific configuration options. Here you choose the processor or processor family that your kernel will support, as well as whether or not to enable access to various processor features. Be sure to enable symmetric multi-processing support if you have more than one CPU or a hyperthreaded CPU. Generally, you should enable the MTRR option to allow better graphic performance with AGP or PCI video cards.

Power management options

This section contains several power management options. These are particularly useful on laptops. Besides controlling power states, you will find options here to control and monitor such things as temperatures or fan states.

Bus options (PCI etc.)

This section contains options for buses supported by your system, such as PCI, PCI Express, and PC Card buses. You can also enable the `/proc/pci` file system here, although you should generally use `lspci` instead.

Executable file formats / Emulations

This section contains options for supporting various binary file formats. You should enable ELF binary support. You may also enable support for DOS binaries to run under DOSEMU, as well as wrapper-driven binaries such as Java™, Python, Emacs-Lisp, and so on. Finally, for a 64-bit system that supports 32-bit emulation, you probably want to enable 32-bit binary support.

Networking

The networking section is large. Here you can enable basic sockets and TCP/IP networking, as well as packet filtering, bridging, routing, and support for a variety of protocols such as IPV6, IPX, Appletalk, and X.25. You can also enable wireless, infrared, and amateur radio support here.

Device drivers

This section is also very large. Here you enable support for most of your hardware devices, including IDE/ATAPI or SCSI hard drives and flash memory devices. Enable DMA for your IDE devices; otherwise, they will work in the slower PIO mode. If you want support for multiple devices such as RAID or LVM, this is where you enable it. You can also configure parallel port support here if you want parallel printer support. This is also where you configure a vast range of possible networking devices to support the networking protocols you configured above. You will also find support here for audio and video capture devices, USB and IEEE 1384 (Firewire) devices, as well as a variety of hardware monitoring devices. Under the character devices subsection, you will

probably want to enable parallel print support and direct rendering support.

Firmware drivers

This section contains a few options related to BIOS setting and updating, such as using the Dell System Management functions on certain Dell systems.

File systems

This section is for configuring the file systems that you want your kernel to support, either compiled in or as modules. You will also find file systems here for removable media such as diskettes and CD or DVD devices, along with support for networked file systems such as NFS, SMB, or CIFS. Support for a variety of partitions and Native Language Support is found here too.

Instrumentation support

This section allows you to enable experimental profiling support for profiling your system's activity.

Kernel hacking

This section allows you to enable kernel debugging and choose which features will be enabled.

Security options

This section allows you to configure several security options and to enable and configure SELinux (Security Enhanced Linux).

Cryptographic options

This section allows you to configure several cryptographic algorithms, such as MD4, DES, and SHA256.

Library routines

This section allows you to decide whether certain CRC algorithms should be compiled in or built as modules.

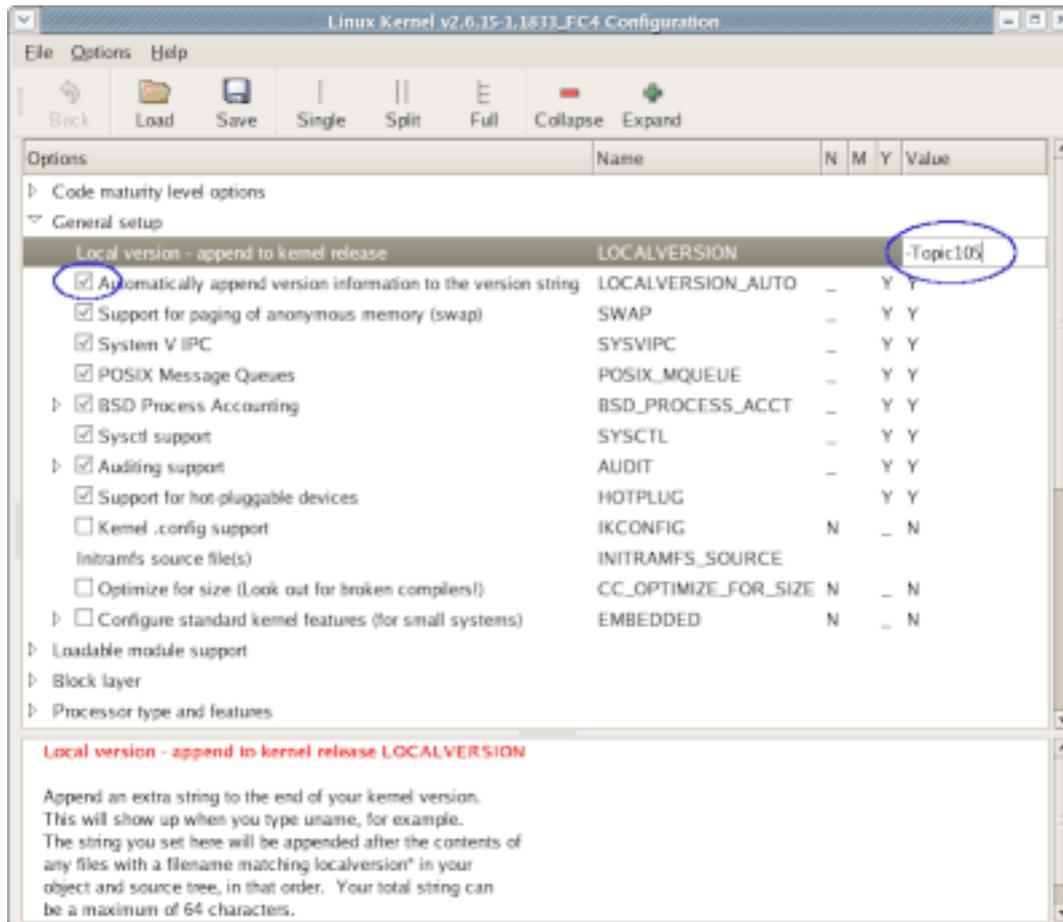
Building

Now that you've seen the major aspects of configuring a kernel, you're ready to build one. If you are not sure of the state of your build tree, run `make clean` before configuring your new kernel. For an even more extreme cleanup target, run `make mrproper`; this will remove your `.config` file as well as some other files used by the build process. If you do this and then need to restore a backed up `.config` file, you will need to run `make oldconfig` before configuring.

While you are experimenting, you should give your new kernel a custom name so you can easily identify it. Do this by setting a local version value and enabling the option to automatically append version information to the version string under the

General setup section as shown in Figure 3.

Figure 3. Configuring a custom kernel



In the spirit of taking small steps, the examples in the remainder of this tutorial are based on building a kernel with just the two changes shown in Figure 3.

In principle, the kernel does not require root authority to build, although you will need root authority to install your new kernel. However, if you are using the package installed by your distribution, you will probably have to run as root because of the file and directory permissions that have been set up. You can practice in user mode by downloading a kernel source tarball from the Linux kernel archives and unpacking it in your home directory, or by making a copy of your kernel build tree and changing the permissions to your userid.

To start building a 2.6 kernel, type `make`.

To start building a 2.4 kernel, run these three commands:

```
make dep
make bzImage
make modules
```

The first makes necessary dependency files. The second builds the kernel. And the last builds your modules.

Running `make` on my AMD Athlon 3500+ system takes about a half hour to complete the build from a clean start. Slower systems may take a couple of hours to complete the job, so take a break or do something else while you wait. You will see progress messages such as those in Listing 16 while the build is running.

Listing 16. Running make

```
[root@attic4 linux]# make
CHK      include/linux/version.h
HOSTCC   scripts/basic/fixdep
HOSTCC   scripts/basic/split-include
HOSTCC   scripts/basic/docproc
SPLIT    include/linux/autoconf.h -> include/config/*
CC       arch/x86_64/kernel/asm-offsets.s
GEN      include/asm-x86_64/asm-offsets.h
...
LD [M]   sound/usb/snd-usb-lib.ko
CC       sound/usb/usx2y/snd-usb-usx2y.mod.o
LD [M]   sound/usb/usx2y/snd-usb-usx2y.ko
```

Installing

When you have completed building your kernel, you still have a couple of steps to go. First, you need to run `make modules_install` to install your kernel modules in a new subdirectory of `./lib/modules`.

If you need proprietary modules for a video card or network driver, as I need for my nVidia graphics card and nForce 4 motherboard chipset, now is a good time to build those modules using the vendor-supplied tools.

Finally, you need to run `make install` to install the new kernel and initial RAM disk in `/boot` and update your boot loader configuration. These steps are illustrated in Listing 17.

Listing 17. Installing the kernel and modules

```
[root@attic4 linux]# make modules_install
INSTALL arch/x86_64/crypto/aes-x86_64.ko
INSTALL arch/x86_64/kernel/cpufreq/acpi-cpufreq.ko
INSTALL arch/x86_64/kernel/microcode.ko
INSTALL arch/x86_64/oprofile/oprofile.ko
INSTALL crypto/aes.ko
INSTALL crypto/anubis.ko
INSTALL crypto/arc4.ko
...
[root@attic4 linux]# ls -lrt /lib/modules | tail -n 3
drwxr-xr-x  5 root root 4096 Mar  4 14:48 2.6.15-1.1831_FC4
drwxr-xr-x  5 root root 4096 Mar 20 18:52 2.6.15-1.1833_FC4
drwxr-xr-x  3 root root 4096 Mar 20 21:38 2.6.15-prep-Topic105
[root@attic4 linux]# sh /root/NFORCE-Linux-x86_64-1.0-0310-pkg1.run -a \
> -n -K -k 2.6.15-prep-Topic105
```

```

Verifying archive integrity...OK
Uncompressing NVIDIA nForce drivers for Linux-x86_64 1.0-0310.....
[root@attic4 linux]# sh /root/NVIDIA-Linux-x86_64-1.0-8178-pkg2.run -a \
> -n -K -k 2.6.15-prep-Topic105
Verifying archive integrity... OK
Uncompressing NVIDIA Accelerated Graphics Driver for Linux-x86_64 1.0-8178.....
[root@attic4 linux]# make install
CHK include/linux/version.h
CHK include/linux/compile.h
CHK usr/initramfs_list
Kernel: arch/x86_64/boot/bzImage is ready (#2)
sh /usr/src/linux-2.6.15/arch/x86_64/boot/install.sh 2.6.15-prep-Topic105
arch/x86_64/boot/bzImage System.map "/boot"
[root@attic4 linux]# ls -lrt /boot | tail -n 6
-rw-r--r-- 1 root root 1743149 Mar 20 21:45 vmlinuz-2.6.15-prep-Topic105
lrwxrwxrwx 1 root root 28 Mar 20 21:45 vmlinuz -> vmlinuz-2.6.15-prep-Topic105
-rw-r--r-- 1 root root 980796 Mar 20 21:45 System.map-2.6.15-prep-Topic105
lrwxrwxrwx 1 root root 31 Mar 20 21:45 System.map -> System.map-2.6.15-prep-Topic105
-rw-r--r-- 1 root root 1318741 Mar 20 21:45 initrd-2.6.15-prep-Topic105.img
drwxr-xr-x 2 root root 4096 Mar 20 21:45 grub

```

Initial RAM disk

Notice that the build process automatically created the necessary initial RAM disk (initrd) for you. If you ever need to create one manually, you do so using the `mkinitrd` command. See the man pages for details.

Boot loaders

If everything worked correctly, the `make install` step should have also updated your boot loader configuration. Some lines from mine are shown in Listing 18.

Listing 18. Updated GRUB configuration file

```

                                default=1
timeout=10
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
password --md5 $1$y.uQRs1W$Sqs30hDB3GtE957PoidWO.
title Fedora Core (2.6.15-prep-Topic105)
    root (hd0,11)
    kernel /boot/vmlinuz-2.6.15-prep-Topic105 ro root=LABEL=FC4-64 rhgb quiet
    initrd /boot/initrd-2.6.15-prep-Topic105.img
title Fedora Core -x86-64 (2.6.15-1.1833_FC4)

```

The entry for the newly built kernel has been placed at the top, but the default entry has been adjusted to remain as the previous default. If you use LILO instead, then the `grubby` command that is used in the build script should have updated your LILO configuration. If the configuration was not updated correctly for any reason, refer to the tutorial "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)," where you will find full instructions on setting up your boot loader.

One final note. You may wonder why the sample configuration added `-Topic105`, yet the created files all had `-prep-Topic105` instead. This is a Fedora safety measure to prevent you from inadvertently destroying your live kernel. This is

controlled by the `EXTRAVERSION` variable set near the top of the main make file, as shown in Listing 19. Edit the file if you need to remove this.

Listing 19. Updated GRUB configuration file

```
[root@attic4 linux]# head -n 6 Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 15
EXTRAVERSION = -prep
NAME=Sliding Snow Leopard
```

Rebooting

If all is well, you should now be able to boot your new system. You will need to select the configuration entry for the new kernel because it is not (yet) the default. After you are happy with it, you can make it the default. When you reboot, use the `uname` command to check your system's kernel as shown in Listing 20.

Listing 20. Checking your new system

```
[ian@attic4 ~]$ uname -rv
2.6.15-prep-Topic105 #2 Mon Mar 20 21:13:20 EST 2006
```

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- [The Linux Kernel Archives](#) is the ultimate resource for the Linux kernel. Check for your nearest mirror before you download.
- The [kernelnewbies project](#) has lots of information for those new to kernels and building them.
- The [Kernel Rebuild Guide](#) shows you how to configure, build, and install a new kernel.
- The [Linux Kernel Module Programming Guide](#) from [Linuxtopia](#) is an online book about kernel modules for Linux.
- [LPI Linux Certification in a Nutshell](#) (O'Reilly, 2001) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep: Kernel

Junior Level Administration (LPIC-1) topic 105

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

21 Mar 2006

In this tutorial, Ian Shields begins preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this first in a [series of nine tutorials](#), Ian introduces you to the kernel on Linux®. By the end of this tutorial, you will know how to build, install, and query a Linux kernel and its kernel modules.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics

LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	(This tutorial). Learn how to install and maintain Linux kernels and kernel modules. See detailed objectives below.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Coming soon.
Topic 107	LPI exam 102 prep: Printing	Coming soon.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Kernel," the first of nine tutorials designed to prepare you for LPI exam

102. In this tutorial, you learn how to build, install, and query a Linux kernel and its kernel modules.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.105.1 Manage and query kernel and kernel modules at runtime	Weight 4	Learn to query and manage a kernel and kernel-loadable modules.
1.105.2 Reconfigure, build, and install a custom kernel and kernel modules	Weight 3	Learn to customize, build, and install a kernel and kernel-loadable modules from source.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be very familiar with the material from [LPI exam 101 prep: Hardware and architecture](#) tutorial.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Runtime kernel management

This section covers material for topic 1.105.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use command-line utilities to get information about the currently running kernel and kernel modules
- Manually load and unload kernel modules

- Determine when modules can be unloaded
- Configure the system to load modules by names other than their file name

Technically, Linux is the kernel of your system. The kernel provides a framework for applications to run and use various hardware devices. It is low-level code that deals with hardware interfaces, scheduling, and memory management among other things. Many people refer to a whole system as GNU/Linux because many of the tools that make most distributions usable come from the GNU project of the Free Software Foundation. Nevertheless, you will often just see "Linux" instead of "GNU/Linux."

uname

The `uname` command prints information about your system and its kernel. Listing 1 shows the various options for `uname` and the resulting information; each option is defined in Table 3.

Listing 1. The `uname` command

```

ian@pinguino:~$ uname
Linux
ian@pinguino:~$ uname -s
Linux
ian@pinguino:~$ uname -n
pinguino
ian@pinguino:~$ uname -r
2.6.12-10-386
ian@pinguino:~$ uname -v
#1 Mon Jan 16 17:18:08 UTC 2006
ian@pinguino:~$ uname -m
i686
ian@pinguino:~$ uname -o
GNU/Linux
ian@pinguino:~$ uname -a
Linux pinguino 2.6.12-10-386 #1 Mon Jan 16 17:18:08 UTC 2006 i686 GNU/Linux
    
```

Table 3. Options for <code>uname</code>	
Option	Description
-s	Print the kernel name. This is the default if no option is specified.
-n	Print the nodename or hostname.
-r	Print the release of the kernel. This option is often used with module-handling commands.

-v	Print the version of the kernel.
-m	Print the machine's hardware (CPU) name.
-o	Print the operating system name.
-a	Print all of the above information.

Listing 1 is from an Ubuntu system running on an Intel® CPU. The `uname` command is available on most UNIX® and UNIX-like systems as well as Linux. The information printed will vary by Linux distribution and version as well as by the type of machine you are running on. Listing 2 shows the output from an AMD Athlon 64 system running Fedora Core 4 and, for comparison, an Apple PowerBook.

Listing 2. Using `uname` with another system

```
Linux attic4 2.6.14-1.1656_FC4 #1 Thu Jan 5 22:13:55 EST 2006 x86_64
x86_64 x86_64 GNU/Linuxfilesystem

Darwin Ian-Shields-Computer.local 7.9.0 Darwin Kernel Version 7.9.0:
Wed Mar 30 20:11:17 PST 2005; root:xnu/xnu-517.12.7.obj~1/RELEASE_PPC
Power Macintosh powerpc
```

Kernel modules

The kernel manages many of the low-level aspects of your system, including hardware and interfaces. With a large variety of possible hardware and several different file systems, a kernel that supported everything would be rather large. Fortunately, *kernel modules* allow you to load support software such as hardware drivers or file systems when needed, so you can start your system with a small kernel and then load other modules as needed. Often the loading is automatic, such as when USB devices are plugged in.

The remainder of this section looks at the commands and configuration for kernel modules.

The commands for tasks such as loading or unloading modules require root authority. The commands for displaying information about modules can usually be run by general users. However, since they reside in `/sbin`, they are not usually on a non-root user's path, so you will probably have to use full path names if you are not root.

lsmod

Use the `lsmod` command to display the modules that are currently loaded on your system, as shown in Listing 3. Your output is likely to be different, although you should see some common entries.

Listing 3. Displaying kernel modules with `lsmod`

```
[ian@attic4 ~]$ /sbin/lsmod
Module              Size Used by
nvnet               74148 0
nvidia             4092336 12
forcedeth          24129 0
md5                 4161 1
ipv6                268737 12
parport_pc         29189 1
lp                 13129 0
parport            40969 2 parport_pc,lp
autofs4            29637 1
sunrpc             168453 1
ipt_REJECT         5825 1
ipt_state          1985 3
ip_conntrack       42009 1 ipt_state
iptables_filter    3137 1
ip_tables          19521 3 ipt_REJECT,ipt_state,iptables_filter
dm_mod             58613 0
video              16069 0
button             4161 0
battery            9541 0
ac                 4933 0
ohci_hcd           26977 0
ehci_hcd           41165 0
i2c_nforce2        7105 0
i2c_core           21825 1 i2c_nforce2
shpchp             94661 0
snd_intel8x0       34945 1
snd_ac97_codec     76217 1 snd_intel8x0
snd_seq_dummy      3781 0
snd_seq_oss        37569 0
snd_seq_midi_event 9409 1 snd_seq_oss
snd_seq            62801 5 snd_seq_dummy,snd_seq_oss,snd_seq_midi_event
snd_seq_device     9037 3 snd_seq_dummy,snd_seq_oss,snd_seq
snd_pcm_oss        51569 0
snd_mixer_oss      18113 1 snd_pcm_oss
snd_pcm            100553 3 snd_intel8x0,snd_ac97_codec,snd_pcm_oss
snd_timer          33733 2 snd_seq,snd_pcm
snd                 57669 11 snd_intel8x0,snd_ac97_codec,snd_seq_oss,snd_seq,
snd_seq_device,snd_pcm_oss,snd_mixer_oss,snd_pcm,snd_timer
soundcore          11169 1 snd
snd_page_alloc     9925 2 snd_intel8x0,snd_pcm
floppy             65397 0
ext3               132681 3
jbd                86233 1 ext3
sata_nv            9541 0
libata             47301 1 sata_nv
sd_mod             20545 0
scsi_mod           147977 2 libata,sd_mod
[ian@attic4 ~]$
```

You can see that this system has many loaded modules. Most of these are supplied with the kernel. However, some, such as the `nvnet`, `nvidia`, and `sata_nv` modules from NVIDIA Corporation include proprietary code and are not supplied as part of a standard kernel. In this way, the modular approach allows proprietary code to be plugged in to an open source kernel. Assuming the vendor license permits it, a Linux

distributor may add proprietary modules to a distribution, saving you the effort of getting them directly from a vendor and helping to ensure that you have the appropriate levels.

From Listing 3, you can also see that modular support extends to devices such as video, SATA and SCSI hard drives, floppy disks, and sound cards, as well as to networking features such as IPV6, file system support such as ext3, and Sun remote procedure call (RPC).

In addition to the module name, `lsmod` also shows the module size and the number of users of the module. If the module is used by any other modules, these are listed. So, for example, the `soundcore` module is used by the `snd` module, which in turn is used by several other sound modules.

modinfo

The `modinfo` command displays information about one or more modules. As shown in Listing 4, the information includes the full path to the file, the author, license, any parameters that the module might accept, version, dependencies, and other information.

Listing 4. Basic module information

```
[ian@attic4 ~]$ /sbin/modinfo floppy
filename:      /lib/modules/2.6.12-1.1456_FC4/kernel/drivers/block/floppy.ko
author:       Alain L. Knaff
license:      GPL
alias:        block-major-2-*
vermagic:     2.6.12-1.1456_FC4 686 REGPARM 4KSTACKS gcc-4.0
depends:
srcversion:   2633BC999A0747D8D215F1F
parm:         FLOPPY_DMA:int
parm:         FLOPPY_IRQ:int
parm:         floppy:charp
[ian@attic4 ~]$ /sbin/modinfo sata_nv
filename:      /lib/modules/2.6.12-1.1456_FC4/kernel/drivers/scsi/sata_nv.ko
author:       NVIDIA
description:  low-level driver for NVIDIA nForce SATA controller
license:      GPL
version:      0.6
vermagic:     2.6.12-1.1456_FC4 686 REGPARM 4KSTACKS gcc-4.0
depends:       libata
alias:        pci:v000010DEd00000008Esv*sd*bc*sc*i*
alias:        pci:v000010DEd000000E3sv*sd*bc*sc*i*
alias:        pci:v000010DEd000000EEsv*sd*bc*sc*i*
alias:        pci:v000010DEd00000054sv*sd*bc*sc*i*
alias:        pci:v000010DEd00000055sv*sd*bc*sc*i*
alias:        pci:v000010DEd00000036sv*sd*bc*sc*i*
alias:        pci:v000010DEd0000003Esv*sd*bc*sc*i*
alias:        pci:v000010DEd*sv*sd*bc01sc01i*
srcversion:   3094AD48C1B869BCC301E9F
```

In Listing 4, notice in the lines giving the module filenames that these filenames end in a `.ko` suffix. This distinguishes modules for 2.6 kernels from other object files and

from modules for 2.4 and earlier kernels, which used the same .o suffix as other object files.

You will also notice that the module path include the kernel version. For example, `/lib/modules/2.6.12-1.1456_FC4/kernel/drivers/block/floppy.ko` includes `2.6.12-1.1456_FC4` as a path element. This is the same value emitted by `uname -r`. Kernel modules are specific to a given kernel, and this structure manages that relationship.

On 2.6 kernels you can also use `modinfo` to limit requests to specific information about a module. Use the `-F` option to extract a single information type, such as `parm`, `description`, `license`, `filename`, or `alias`. Use the command multiple times with different options if you need different types of information. On 2.4 kernels, parameters such as `-p` extracted parameter information. The current `modinfo` command also supports the older parameters. Listing 5 shows some examples.

Listing 5. Specific module information

```
[ian@attic4 ~]$ /sbin/modinfo -F parm snd
cards_limit:Count of auto-loadable soundcards.
major:Major # for sound driver.
[ian@attic4 ~]$ /sbin/modinfo -F license nvidia floppy
NVIDIA
GPL
[ian@attic4 ~]$ /sbin/modinfo -p snd
major:Major # for sound driver.
cards_limit:Count of auto-loadable soundcards.
```

Using your Linux skills

You may use some of the techniques covered in the tutorial "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)" to extract information such as the number of parameters accepted by any module that accepts parameters. Listing 6 shows an example.

Listing 6. Number of parameters per module

```
[ian@attic4 ~]$ for n in `/sbin/lsmmod | tail +2 | cut -d " " -f1`;
> do echo "$n $(/sbin/modinfo -p $n | wc -l )" | grep -v " 0$"; done
nvnet 12
forcedeth 1
parport_pc 5
dm_mod 1
ohci_hcd 2
ehci_hcd 2
shpchp 3
snd_intel8x0 7
snd_ac97_codec 1
snd_seq_dummy 2
snd_seq_oss 2
snd_seq 7
snd_pcm_oss 3
snd_pcm 2
snd_timer 1
snd 2
```

```
snd_page_alloc 1
scsi_mod 6
```

rmmod

If a module's use count is 0, you may safely remove it. For example, you might do this in preparation for loading an updated version. This is a great feature of a modular kernel because you do not always have to reboot just to update support for one or another particular device. To remove a mod, use the `rmmod` command along with the module name as shown in Listing 7.

Listing 7. Removing a module for a running system

```
[root@attic4 ~]# rmmod floppy
```

Consult the man pages for other options available with `rmmod`.

insmod and modprobe

Once you have removed a module, you may need to reload it. You can do this using the `insmod` command, which takes the full path name of the module to be reloaded, along with any module options that may be required. If you use this command, you will probably want to use command substitution for generating the filename. Two ways of doing this are shown in Listing 8.

Listing 8. Loading a module using insmod

```
[root@attic4 ~]# insmod /lib/modules/`uname
-r`/kernel/drivers/block/floppy.ko
[root@attic4 ~]# rmmod floppy
[root@attic4 ~]# insmod $(modinfo -F filename floppy)
```

The second form above saves you the need to remember which subdirectory (drivers/block in this case) a module is located in, but there is an even better way to load a module. The `modprobe` command provides a higher-level interface that operates with the module name instead of file path. It also handles loading additional modules upon which a module depends, and can remove modules as well as load them.

Listing 9 shows how to use `modprobe` to remove the `vfat` module, along with the `fat` module that uses it. It then shows what the system would do if the module were reloaded, and finally the result of reloading the module. Note that the `-v` option is specified to obtain verbose output; otherwise, `modprobe` (and the underlying `insmod` command) will display only error messages from the module itself. Between

each step, the output of `lsmod` is piped through `grep` to show whether either the `vfat` or `fat` module is loaded or not.

Listing 9. Loading a module using `modprobe`

```
[root@lyrebird root]# modprobe -r vfat
vfat: Device or resource busy
[root@lyrebird root]# lsmod | grep fat
vfat          13132    1
fat           38744    0 [vfat]
[root@lyrebird root]# umount /windows/D
[root@lyrebird root]# modprobe -r vfat
[root@lyrebird root]# modprobe -v --show vfat
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
[root@lyrebird root]# lsmod | grep fat
[root@lyrebird root]# modprobe -v vfat
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
Using /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
Symbol version prefix ''
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
Using /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
[root@lyrebird root]# lsmod | grep fat
vfat          13132    0 (unused)
fat           38744    0 [vfat]
```

depmod

You have just seen that `modprobe` can handle the automatic loading of multiple modules when some are dependent on others. The dependencies are kept in the `modules.dep` file in the `/lib/modules` subdirectory for the appropriate kernel, as given by the `uname -r` command. This file, along with several map files, is generated by the `depmod` command. The `-a` (for *all*) is now optional.

The `depmod` command scans the modules in the subdirectories of `/lib/modules` for the kernel you are working on and freshens the dependency information. An example, along with the resulting changed files, is shown in Listing 10.

Listing 10. Using `depmod` to rebuild `modules.dep`

```
[root@lyrebird root]# date
Thu Mar 16 10:41:05 EST 2006
[root@lyrebird root]# depmod
[root@lyrebird root]# cd /lib/modules/`uname -r`
[root@lyrebird 2.4.21-37.0.1.EL]# ls -l mod*
-rw-rw-r-- 1 root root 54194 Mar 16 10:41 modules.dep
-rw-rw-r-- 1 root root 31 Mar 16 10:41 modules.generic_string
-rw-rw-r-- 1 root root 73 Mar 16 10:41 modules.ieee1394map
-rw-rw-r-- 1 root root 1614 Mar 16 10:41 modules.isapnpmap
-rw-rw-r-- 1 root root 29 Mar 16 10:41 modules.parpportmap
-rw-rw-r-- 1 root root 65171 Mar 16 10:41 modules.pcimmap
-rw-rw-r-- 1 root root 24 Mar 16 10:41 modules.pnpbiosmap
-rw-rw-r-- 1 root root 122953 Mar 16 10:41 modules.usbmap
[root@lyrebird 2.4.21-37.0.1.EL]# cd -
/root
```

You can customize the behavior of `modprobe` and `depmod` using the configuration file `/etc/modules.conf`. This is commonly used to alias module names and to specify commands that should be run after a module is loaded or before it is unloaded. However, an extensive range of other configuration can be done. Listing 11 shows an example of `/etc/modules.conf`. Consult the man page for `modules.conf` for more details.

Listing 11. Example `/etc/modules` file

```
[root@lyrebird root]# cat /etc/modules.conf
alias eth0 e100
alias usb-controller usb-uhci
alias usb-controller1 ehci-hcd
alias sound-slot-0 i810_audio
post-install sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -L >/dev/null 2>&1 || :
pre-remove sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -S >/dev/null 2>&1 || :
```

You should also be aware that some systems use another configuration file called `modprobe.conf`, while others store module configuration information in the `/etc/modules.d` directory. You may also find a file called `/etc/modules` on some systems; this file contains the names of kernel modules that should be loaded at boot time.

USB modules

When you hot plug a USB device into your Linux system, the kernel must determine which modules to load to handle the device. This is usually done for you by a hot plug script that uses the `usbmodules` command to find the appropriate module. You can also run `usbmodules` (as root) to see for yourself. Listing 12 shows an example.

Listing 12. USB modules

```
root@pinguino:~# lsusb
Bus 005 Device 004: ID 1058:0401 Western Digital Technologies, Inc.
Bus 005 Device 003: ID 054c:0220 Sony Corp.
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 003: ID 04b3:310b IBM Corp. Red Wheel Mouse
Bus 001 Device 001: ID 0000:0000
root@pinguino:~# usbmodules --device /proc/bus/usb/005/003
usb-storage
root@pinguino:~# usbmodules --device /proc/bus/usb/001/003
usbmouse
usbhid
```

The next section shows you how to build and configure a custom kernel.

Section 3. Customize and build kernels and kernel modules

This section covers material for topic 1.105.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Customize the current kernel configuration
- Build a new kernel and appropriate kernel modules
- Install a new kernel and any modules
- Ensure that the boot manager can locate the new kernel and associated files

As you learned in the previous section, [Runtime kernel management](#), the kernel provides the low-level support for your system hardware and file systems. A modern kernel image usually contains only essential functions, but is configured to support additional functions that you might need through the use of *kernel modules*. The additional support is loaded only when needed, for example when a device is plugged in or otherwise enabled.

The modular code becomes an integral part of the kernel, dynamically extending the kernel functions. If the functions of a loaded kernel module have not been used for several minutes, the kernel can voluntarily disassociate it from the rest of the kernel and unload it from memory through a process known as *autocleaning*.

Without kernel modules, your running kernel, which is loaded from disk as a single binary file, would have to contain all the functionality you might possibly ever need. You would also need to build a completely new kernel every time you wanted to add functionality to your system.

You cannot put *everything* in a module, however. At a bare minimum, the kernel image that is loaded must be able to mount your root file system. But, as you learned in the tutorial "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)," your boot loader can load an *initial RAM disk* (or *initrd*), which may contain the modules necessary to mount the root file system. Nevertheless, the kernel image must at least contain support for the RAM file system used in the initial RAM disk. If it does not, your system will not boot.

Once your system has bootstrapped itself this far, it proceeds to mount the root file system and then start the other initialization processes. After a few seconds, the

system is up and ready for you to use. The kernel, however, remains in control awaiting requests to perform work for user processes and scheduling the system resources among the tasks that require them.

Modular kernels work well in modern systems with plenty of RAM and disk space. However, you may have a new piece of hardware, such as a video card or storage system, that is not supported by the kernel that came with your distribution. Indeed, some drivers contain proprietary code that is said to *taint* a pure Linux kernel, so some distributors will not include it, even if the vendor license terms permit it to be distributed by your chosen distributor. In this case, you will need to at least build new modules, and possibly even build a new kernel.

Linux can be used in many environments, from embedded systems such as mobile phones, to networking devices such as routers, to set-top boxes as well as more traditional computing environments. Some of these devices use a kernel that is customized to support only those functions that the system is intended to support. For example, a system intended to be a diskless firewall probably does not need support for any file system other than the read-only file system from which it loaded, yet it may need support for advanced networking hardware that is not part of a standard kernel. Again, a custom kernel will be required.

Source packages

The ultimate source for the Linux kernel is the Linux Kernel Archives (see [Resources](#) for a link). Unless you already know what you are doing, you should use a kernel package from your Linux distribution, because your distributor may have added custom patches. If you are already familiar with obtaining and extracting source packages, review the tutorial "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)." As with anything that may change your system, make backups first so that you can recover if things go wrong.

If you download source from the public kernel archives, you will download a compressed file, and you will need to decompress it using `gzip` or `bzip2`, according to whether you download the `.gz` or the `.bz2` version of the kernel source. The `pub/linux/kernel/` directory on the download server has a directory for each kernel version, such as 2.4, 2.5, or 2.6. At the date of this writing, the latest `bzip2` version of the 2.6 kernel is `linux-2.6.15.tar.bz2`.

In that kernel directory, you will also see a corresponding `ChangeLog-2.6.15.6` file that describes changes in this version, and a `patch-2.6.15.bz2`, which is a smaller file that allows you to patch the prior version of source to bring it up to 2.6.15 level. You will also notice signature files that you may use to verify that your downloaded file was not corrupted, either accidentally or maliciously.

The compressed source is normally uncompressed in `/usr/src`, and it creates a new

subdirectory for the kernel version, such as `linux-2.6.15`, containing the tree of files needed to build your kernel. If you already have such a directory, you may want to back it up or rename it before unpacking the new kernel source. This will ensure that you can go back if you need to, and also that you will not have stray files that should not be in your kernel source tree. You need about 40MB of disk space for the tarball and about 350MB for the expanded source code.

Some distributors, notably Red Hat, now distribute the kernel headers and source necessary for building kernel modules as a kernel development package. Documentation may be in a separate kernel documentation package. These are designed for and sufficient for building modules, such as a proprietary vendor graphics card module, but they are not sufficient for rebuilding a custom kernel. Your distribution should have information about how to rebuild a kernel and how the source can be obtained. Check for documentation such as release notes.

Suppose you use FTP or HTTP to download the `kernel-2.6.15-1.1833_FC4.src.rpm` source RPM from the `pub/fedora/linux/core/updates/4/SRPMS/` at `download.fedora.redhat.com`, and the file is in the `/root` directory. Note that version numbers used here will probably be different for your system, so make sure you get the updated version of source corresponding to your installed kernel. Now, for Fedora, you must install the source RPM, then switch to the `/usr/src/redhat/SPECS` directory, and finally build the source RPM in order to create the Linux kernel source tree as shown in Listing 13.

Listing 13. Creating the kernel source tree for Fedora Core

```
[root@attic4 ~]# uname -r
2.6.15-1.1833_FC4
[root@attic4 ~]# rpm -Uvh kernel-2.6.15-1.1833_FC4.src.rpm
 1:kernel ##### [100%]
[root@attic4 ~]# cd /usr/src/redhat/SPECS
[root@attic4 SPECS]# rpmbuild -bp --target $(arch) kernel-2.6.spec
Building target platforms: x86_64
Building for target x86_64
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.23188
+ umask 022
+ cd /usr/src/redhat/BUILD
+ LANG=C
+ export LANG
+ unset DISPLAY
+ '[' '!' -d kernel-2.6.15/vanilla ']'
+ cd /usr/src/redhat/BUILD
+ rm -rf kernel-2.6.15
+ /bin/mkdir -p kernel-2.6.15
+ cd kernel-2.6.15
+ /usr/bin/bzip2 -dc /usr/src/redhat/SOURCES/linux-2.6.15.tar.bz2
+ tar -xf -
+ . . .
+ echo '# x86_64'
+ cat .config
+ perl -p -i -e 's/^\SUBLEVEL.*\/SUBLEVEL = 15/' Makefile
+ perl -p -i -e 's/^\EXTRAVERSION.*\/EXTRAVERSION = -prep/' Makefile
+ find . -name '*.orig' -o -name '*~' -exec rm -f '{}' ';'
+ exit 0
```

The Linux kernel source for Fedora is now located in `/usr/src/redhat/BUILD/kernel-2.6.15/linux-2.6.15`. By convention, the `/linux-2.6.15` tree is often moved to `/usr/src` and symbolically linked to `/usr/src/linux`, as shown in Listing 14. This is not strictly necessary, but it's easier to follow along with references that assume the kernel source tree will be in `.usr./src/linux`.

Listing 14. Moving the source tree to `/usr/src`

```
[root@attic4 SPECS]# mv ../BUILD/kernel-2.6.15/linux-2.6.15 /usr/src
[root@attic4 SPECS]# cd /usr/src
[root@attic4 src]# ln -s linux-2.6.15 linux
[root@attic4 src]# ls -ld lin*
lrwxrwxrwx  1 root root   12 Mar 20 18:23 linux -> linux-2.6.15
drwxr-xr-x 20 root root 4096 Mar 20 18:13 linux-2.6.15
```

Before you attempt to build anything, review the `Changes` file that is located in the `Documentation` directory. Among other things, it lists the minimum levels of various software packages that you need to build a kernel. Make sure that you have these packages installed.

You may notice `Makefile` and `.config` among the files shown in Listing 13. The `make` file contains several `make` targets for tasks such as configuring the kernel options, building the kernel and its modules, and installing the modules and building RPM or deb packages. More recent kernel sources allow you to use `make help` for brief help on each target. For older systems, you may need to consult the documentation or examine the `make` file. Listing 15 shows partial output for `make help`.

Listing 15. Help for kernel building `make` file

```
[ian@attic4 linux-2.6.15]$ make help
Cleaning targets:
  clean          - remove most generated files but keep the config
  mrproper       - remove all generated files + config + various backup files

Configuration targets:
  config         - Update current config utilising a line-oriented program
  menuconfig     - Update current config utilising a menu based program
  xconfig        - Update current config utilising a QT based front-end
  gconfig        - Update current config utilising a GTK based front-end
  oldconfig      - Update current config utilising a provided .config as base
  randconfig     - New config with random answer to all options
  defconfig      - New config with default answer to all options
  allmodconfig   - New config selecting modules when possible
  allyesconfig   - New config where all options are accepted with yes
  allnoconfig    - New minimal config

Other generic targets:
  all            - Build all targets marked with [*]
  * vmlinux      - Build the bare kernel
  * modules      - Build all modules
  modules_install - Install all modules
  dir/           - Build all files in dir and below
  dir/file.[ois] - Build specified target only
  ...
```

Configuration

The `.config` file in your kernel build directory contains configuration information for your kernel, including the target machine environment, the features to be included, and whether a feature should be compiled into the kernel or built as a module. Creating a `.config` file is the first step to building or rebuilding a kernel. You create it using one of the configuration targets in the make file.

The main configuration options are:

config

The `config` target uses a command-line interface to obtain answers to many questions to either build or update your `.config` file. With the advent of the menu-based configuration targets, this command-line interface is rarely used today.

menuconfig

The `menuconfig` target uses an ncurses-based, menu-based program to create or update your `.config` file. You need only answer questions for items you want to change. This approach has superseded the older `config` target. You run this in a terminal window either remotely or locally.

xconfig

The `xconfig` target uses a graphical menu system based on a QT front-end, like the one used with the KDE desktop.

gconfig

The `gconfig` target uses a graphical menu system based on a GTK front-end, like the one used with the GNOME desktop.

oldconfig

The `oldconfig` target allows you to build a configuration using an existing `.config` file, such as you might have from a previous build or another system. For example, if you installed the kernel source for Fedora as described above, you may copy the configuration file for your running system from `/lib/modules/$(uname -r)/build/.config` to `/usr/src/linux`. Once you've built it, you may use one of the menu configuration targets to modify it if necessary.

Figure 1 shows what you might see if you run `make menuconfig` for a 2.4 series kernel. Press **Enter** to descend into lower-level menus, and press **Esc** to return. Help is available for most items. Either tab to the **< Help >** button and press **Enter**, or simply type **h**. Press **Esc** to return to configuring.

Figure 1. Running make menuconfig on a 2.4 kernel

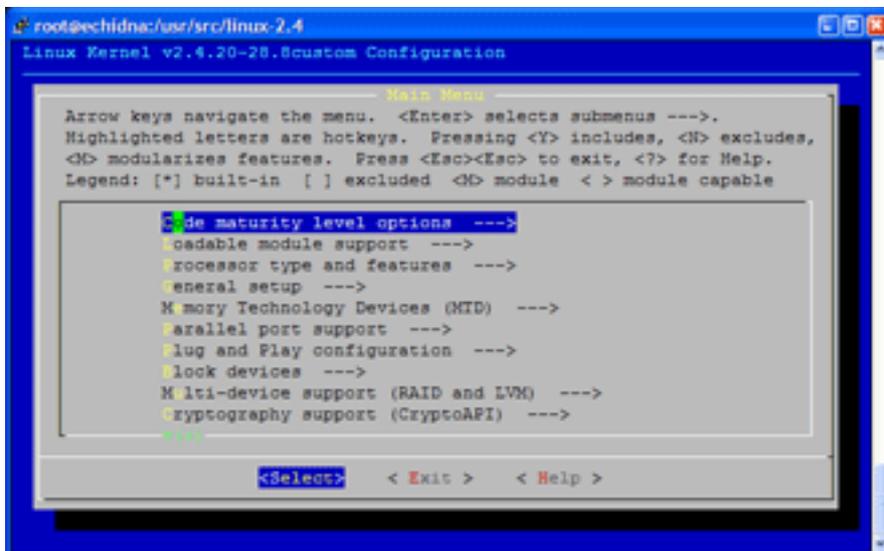
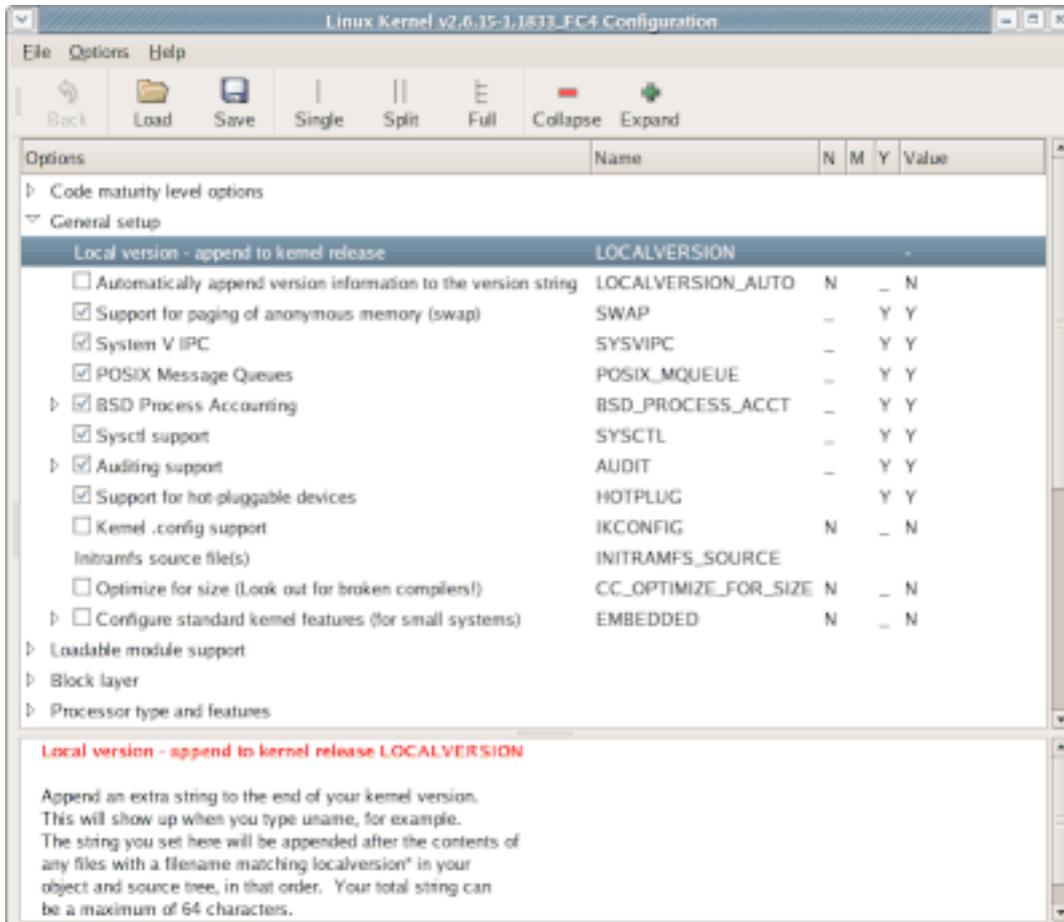


Table 4 shows the various options for including features in the kernel, either built in or as modules. When an option is highlighted, press the space bar to toggle between the allowable choices for that feature. You can also press **y** to enable an option, **n** to disable it, or **m** to have it compiled as a module if possible.

Table 4. Options for menuconfig	
Option	Description
[*]	Feature will be built into the kernel.
[]	Feature will not be included in the kernel.
<M>	Feature will be built as a kernel module.
< >	Feature will not be included in the kernel but is capable of being built as a module.

Figure 2 shows what you might see if you run `make gconfig` for a 2.6 series kernel. Click the arrows to expand or collapse menu items. Help is displayed in a lower pane.

Figure 2. Running make gconfig on a 2.6 kernel



The major configuration sections for a 2.6 kernel are described below. You may not find all of these with 2.4 and earlier kernels, but this list gives you an overview of where to find what.

Code maturity level options

This section contains an option that determines whether remaining options give you a choice for code that is considered experimental. If you do not select this option, then you will be able to select only options that are considered stable. Be warned that functions you choose may or may not work at the current code level on your system, so you might have a chance to help with debugging.

General setup

This section lets you include an identification string with your new kernel, along with options for several kernel attributes that do not belong elsewhere but that you must specify.

Loadable module support

This section contains an option that determines whether your kernel will support modules and whether they may be automatically loaded and unloaded. You should enable module support.

Block layer

This section contains support for disks larger than 2TB, and allows you to choose the type of disk scheduling that you would like.

Processor type and features

This section contains CPU-specific configuration options. Here you choose the processor or processor family that your kernel will support, as well as whether or not to enable access to various processor features. Be sure to enable symmetric multi-processing support if you have more than one CPU or a hyperthreaded CPU. Generally, you should enable the MTRR option to allow better graphic performance with AGP or PCI video cards.

Power management options

This section contains several power management options. These are particularly useful on laptops. Besides controlling power states, you will find options here to control and monitor such things as temperatures or fan states.

Bus options (PCI etc.)

This section contains options for buses supported by your system, such as PCI, PCI Express, and PC Card buses. You can also enable the `/proc/pci` file system here, although you should generally use `lspci` instead.

Executable file formats / Emulations

This section contains options for supporting various binary file formats. You should enable ELF binary support. You may also enable support for DOS binaries to run under DOSEMU, as well as wrapper-driven binaries such as Java™, Python, Emacs-Lisp, and so on. Finally, for a 64-bit system that supports 32-bit emulation, you probably want to enable 32-bit binary support.

Networking

The networking section is large. Here you can enable basic sockets and TCP/IP networking, as well as packet filtering, bridging, routing, and support for a variety of protocols such as IPV6, IPX, Appletalk, and X.25. You can also enable wireless, infrared, and amateur radio support here.

Device drivers

This section is also very large. Here you enable support for most of your hardware devices, including IDE/ATAPI or SCSI hard drives and flash memory devices. Enable DMA for your IDE devices; otherwise, they will work in the slower PIO mode. If you want support for multiple devices such as RAID or LVM, this is where you enable it. You can also configure parallel port support here if you want parallel printer support. This is also where you configure a vast range of possible networking devices to support the networking protocols you configured above. You will also find support here for audio and video capture devices, USB and IEEE 1384 (Firewire) devices, as well as a variety of hardware monitoring devices. Under the character devices subsection, you will

probably want to enable parallel print support and direct rendering support.

Firmware drivers

This section contains a few options related to BIOS setting and updating, such as using the Dell System Management functions on certain Dell systems.

File systems

This section is for configuring the file systems that you want your kernel to support, either compiled in or as modules. You will also find file systems here for removable media such as diskettes and CD or DVD devices, along with support for networked file systems such as NFS, SMB, or CIFS. Support for a variety of partitions and Native Language Support is found here too.

Instrumentation support

This section allows you to enable experimental profiling support for profiling your system's activity.

Kernel hacking

This section allows you to enable kernel debugging and choose which features will be enabled.

Security options

This section allows you to configure several security options and to enable and configure SELinux (Security Enhanced Linux).

Cryptographic options

This section allows you to configure several cryptographic algorithms, such as MD4, DES, and SHA256.

Library routines

This section allows you to decide whether certain CRC algorithms should be compiled in or built as modules.

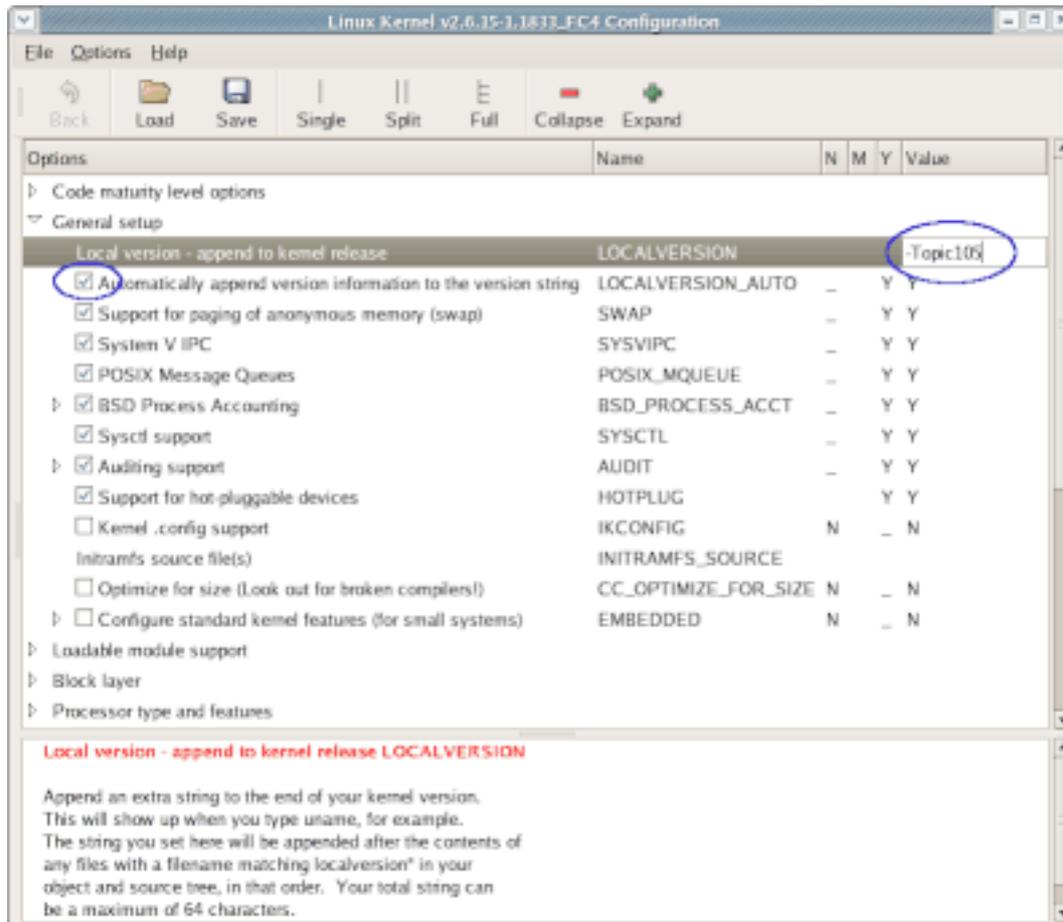
Building

Now that you've seen the major aspects of configuring a kernel, you're ready to build one. If you are not sure of the state of your build tree, run `make clean` before configuring your new kernel. For an even more extreme cleanup target, run `make mrproper`; this will remove your `.config` file as well as some other files used by the build process. If you do this and then need to restore a backed up `.config` file, you will need to run `make oldconfig` before configuring.

While you are experimenting, you should give your new kernel a custom name so you can easily identify it. Do this by setting a local version value and enabling the option to automatically append version information to the version string under the

General setup section as shown in Figure 3.

Figure 3. Configuring a custom kernel



In the spirit of taking small steps, the examples in the remainder of this tutorial are based on building a kernel with just the two changes shown in Figure 3.

In principle, the kernel does not require root authority to build, although you will need root authority to install your new kernel. However, if you are using the package installed by your distribution, you will probably have to run as root because of the file and directory permissions that have been set up. You can practice in user mode by downloading a kernel source tarball from the Linux kernel archives and unpacking it in your home directory, or by making a copy of your kernel build tree and changing the permissions to your userid.

To start building a 2.6 kernel, type `make`.

To start building a 2.4 kernel, run these three commands:

```
make dep
make bzImage
make modules
```

The first makes necessary dependency files. The second builds the kernel. And the last builds your modules.

Running `make` on my AMD Athlon 3500+ system takes about a half hour to complete the build from a clean start. Slower systems may take a couple of hours to complete the job, so take a break or do something else while you wait. You will see progress messages such as those in Listing 16 while the build is running.

Listing 16. Running make

```
[root@attic4 linux]# make
CHK      include/linux/version.h
HOSTCC   scripts/basic/fixdep
HOSTCC   scripts/basic/split-include
HOSTCC   scripts/basic/docproc
SPLIT    include/linux/autoconf.h -> include/config/*
CC       arch/x86_64/kernel/asm-offsets.s
GEN       include/asm-x86_64/asm-offsets.h
...
LD [M]   sound/usb/snd-usb-lib.ko
CC       sound/usb/usx2y/snd-usb-usx2y.mod.o
LD [M]   sound/usb/usx2y/snd-usb-usx2y.ko
```

Installing

When you have completed building your kernel, you still have a couple of steps to go. First, you need to run `make modules_install` to install your kernel modules in a new subdirectory of `./lib/modules`.

If you need proprietary modules for a video card or network driver, as I need for my nVidia graphics card and nForce 4 motherboard chipset, now is a good time to build those modules using the vendor-supplied tools.

Finally, you need to run `make install` to install the new kernel and initial RAM disk in `/boot` and update your boot loader configuration. These steps are illustrated in Listing 17.

Listing 17. Installing the kernel and modules

```
[root@attic4 linux]# make modules_install
INSTALL arch/x86_64/crypto/aes-x86_64.ko
INSTALL arch/x86_64/kernel/cpufreq/acpi-cpufreq.ko
INSTALL arch/x86_64/kernel/microcode.ko
INSTALL arch/x86_64/oprofile/oprofile.ko
INSTALL crypto/aes.ko
INSTALL crypto/anubis.ko
INSTALL crypto/arc4.ko
...
[root@attic4 linux]# ls -lrt /lib/modules | tail -n 3
drwxr-xr-x  5 root root 4096 Mar  4 14:48 2.6.15-1.1831_FC4
drwxr-xr-x  5 root root 4096 Mar 20 18:52 2.6.15-1.1833_FC4
drwxr-xr-x  3 root root 4096 Mar 20 21:38 2.6.15-prep-Topic105
[root@attic4 linux]# sh /root/NFORCE-Linux-x86_64-1.0-0310-pkg1.run -a \
> -n -K -k 2.6.15-prep-Topic105
```

```

Verifying archive integrity...OK
Uncompressing NVIDIA nForce drivers for Linux-x86_64 1.0-0310.....
[root@attic4 linux]# sh /root/NVIDIA-Linux-x86_64-1.0-8178-pkg2.run -a \
> -n -K -k 2.6.15-prep-Topic105
Verifying archive integrity... OK
Uncompressing NVIDIA Accelerated Graphics Driver for Linux-x86_64 1.0-8178.....
[root@attic4 linux]# make install
CHK include/linux/version.h
CHK include/linux/compile.h
CHK usr/initramfs_list
Kernel: arch/x86_64/boot/bzImage is ready (#2)
sh /usr/src/linux-2.6.15/arch/x86_64/boot/install.sh 2.6.15-prep-Topic105
arch/x86_64/boot/bzImage System.map "/boot"
[root@attic4 linux]# ls -lrt /boot | tail -n 6
-rw-r--r-- 1 root root 1743149 Mar 20 21:45 vmlinuz-2.6.15-prep-Topic105
lrwxrwxrwx 1 root root 28 Mar 20 21:45 vmlinuz -> vmlinuz-2.6.15-prep-Topic105
-rw-r--r-- 1 root root 980796 Mar 20 21:45 System.map-2.6.15-prep-Topic105
lrwxrwxrwx 1 root root 31 Mar 20 21:45 System.map -> System.map-2.6.15-prep-Topic105
-rw-r--r-- 1 root root 1318741 Mar 20 21:45 initrd-2.6.15-prep-Topic105.img
drwxr-xr-x 2 root root 4096 Mar 20 21:45 grub

```

Initial RAM disk

Notice that the build process automatically created the necessary initial RAM disk (initrd) for you. If you ever need to create one manually, you do so using the `mkinitrd` command. See the man pages for details.

Boot loaders

If everything worked correctly, the `make install` step should have also updated your boot loader configuration. Some lines from mine are shown in Listing 18.

Listing 18. Updated GRUB configuration file

```

                                default=1
timeout=10
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
password --md5 $1$y.uQRs1W$Sqs30hDB3GtE957PoidWO.
title Fedora Core (2.6.15-prep-Topic105)
    root (hd0,11)
    kernel /boot/vmlinuz-2.6.15-prep-Topic105 ro root=LABEL=FC4-64 rhgb quiet
    initrd /boot/initrd-2.6.15-prep-Topic105.img
title Fedora Core -x86-64 (2.6.15-1.1833_FC4)

```

The entry for the newly built kernel has been placed at the top, but the default entry has been adjusted to remain as the previous default. If you use LILO instead, then the `grubby` command that is used in the build script should have updated your LILO configuration. If the configuration was not updated correctly for any reason, refer to the tutorial "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)," where you will find full instructions on setting up your boot loader.

One final note. You may wonder why the sample configuration added `-Topic105`, yet the created files all had `-prep-Topic105` instead. This is a Fedora safety measure to prevent you from inadvertently destroying your live kernel. This is

controlled by the `EXTRAVERSION` variable set near the top of the main make file, as shown in Listing 19. Edit the file if you need to remove this.

Listing 19. Updated GRUB configuration file

```
[root@attic4 linux]# head -n 6 Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 15
EXTRAVERSION = -prep
NAME=Sliding Snow Leopard
```

Rebooting

If all is well, you should now be able to boot your new system. You will need to select the configuration entry for the new kernel because it is not (yet) the default. After you are happy with it, you can make it the default. When you reboot, use the `uname` command to check your system's kernel as shown in Listing 20.

Listing 20. Checking your new system

```
[ian@attic4 ~]$ uname -rv
2.6.15-prep-Topic105 #2 Mon Mar 20 21:13:20 EST 2006
```

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- [The Linux Kernel Archives](#) is the ultimate resource for the Linux kernel. Check for your nearest mirror before you download.
- The [kernelnewbies project](#) has lots of information for those new to kernels and building them.
- The [Kernel Rebuild Guide](#) shows you how to configure, build, and install a new kernel.
- The [Linux Kernel Module Programming Guide](#) from [Linuxtopia](#) is an online book about kernel modules for Linux.
- [LPI Linux Certification in a Nutshell](#) (O'Reilly, 2001) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 106: Boot, initialization, shutdown, and runlevels

Junior Level Administration (LPIC-1) topic 106

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

04 Apr 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this second in a [series of nine tutorials](#), Ian introduces you to startup and shutdown on Linux®. By the end of this tutorial, you will know guide a system through booting, set kernel parameters, and shut down or reboot a system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding

developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	(This tutorial). Learn how to boot a system, set kernel parameters, and shut down or reboot a system. See detailed objectives below.
Topic 107	LPI exam 102 prep: Printing	Coming soon.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Boot, initialization, shutdown, and runlevels," the second of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to boot a system, set kernel parameters, and shut down or reboot a system.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.106.1 Boot the system	Weight 3	Guide the system through the booting process, including giving commands to the boot loader and setting kernel options at boot time. Learn how to check boot events in log files.
1.106.2 Change runlevels, and shut down or reboot system	Weight 3	Manage the runlevel of the system, and set the default runlevel, plus change to single-user mode, shut down and reboot the system. Learn how to alert users before switching runlevel, and how to properly terminate processes.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be very familiar with the material from the "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)" tutorial.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Boot the system

This section covers material for topic 1.106.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Guide the system through the booting process
- Give commands to the boot loader at boot time
- Give options to the kernel at boot time
- Check boot events in the log files

Boot overview

To summarize the boot process for PCs:

1. When a PC is turned on, the BIOS (*Basic Input Output Service*) performs a self test.
2. When the machine passes its self test, the BIOS loads the *Master Boot Record* (or *MBR*, usually from the first 512-byte sector of the boot drive). This is usually the first hard drive on the system, but may also be a diskette, CD, or USB key.
3. For a hard drive, the MBR loads a stage 1 boot loader, which is typically either the LILO or GRUB stage1 boot loader on a Linux system. This is another 512-byte, single-sector record.
4. The stage 1 boot loader usually loads a sequence of records called the stage 2 boot loader (or sometimes the stage 1.5 loader).
5. The stage 2 loader loads the operating system. For Linux, this is the kernel, and possibly an initial RAM disk (initrd).

At this point, your system should be able to install either of two popular boot loaders, LILO (the LInux LOader) or GRUB (the GRand Unified Boot loader). You should be able to use your chosen boot loader to boot normally as described above. Refer back to the "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)" tutorial if you need to review boot loader installation or basic booting.

To influence your system's boot process, you can:

1. Change the device from which you boot. You may normally boot from a

hard drive, but you may sometimes need to boot from a floppy disk, a USB memory key, a CD or DVD, or a network. Setting up such alternate boot devices requires your BIOS to be appropriately configured. The method for doing this is specific to your system and its BIOS. This is beyond the scope of this tutorial or the requirements of this LPI objective, so consult your system documentation.

2. You can interact with the boot loader to select which of several possible configurations you may wish to boot. You will learn how to do this for both LILO and GRUB in this tutorial.
3. You can use GRUB or LILO to pass parameters to the kernel to control the way that your kernel starts the system once it has been loaded by the boot loader.

LILO

The LILO configuration file defaults to `/etc/lilo.conf`. Listing 1 shows an example from a system that is currently running Red Hat Enterprise Linux 3. The system has a Red Hat 9 partition whose root filesystem is mounted on `/mnt/hda7` and a Windows® XP system on `/dev/hda1`.

Listing 1. Sample LILO configuration

```
[root@lyrebird root]# cat /etc/lilo.conf
prompt
timeout=50
compact
default=latest-EL
boot=/dev/fd0
map=/boot/map
install=/boot/boot.b
message=/boot/message2
lba32
password=mypassword
restricted

image=/mnt/hda7/boot/vmlinuz-2.4.20-31.9
    label=redhat9
    alias=shrike
    initrd=/mnt/hda7/boot/initrd-2.4.20-31.9.img
    read-only
    append="hdd=ide-scsi root=LABEL=RH9"

image=/boot/vmlinuz-2.4.21-40.EL
    label=2.4.21-40.EL
    alias=latest-EL
    initrd=/boot/initrd-2.4.21-40.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-37.0.1.EL
    label=2.4.21-37a.EL
    initrd=/boot/initrd-2.4.21-37.0.1.EL.img
```

```
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-37.EL
    label=2.4.21-37.EL
    initrd=/boot/initrd-2.4.21-37.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-32.0.1.EL
    label=2.4.21-32.EL
    alias=early
    initrd=/boot/initrd-2.4.21-32.0.1.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

other=/dev/hda1
    loader=/boot/chain.b
    label=WIN-XP
    alias=xp
```

Remember that whenever you make changes to `/etc/lilo.conf`, or whenever you install a new kernel, you **must** run `lilo`. The `lilo` program rewrites the MBR or the partition boot record to reflect your changes, including recording the absolute disk location of the kernel. If your configuration file includes Linux images from multiple partitions, you must mount the partitions because the `lilo` command needs to access the partition to locate the image.

The `message` parameter in the LILO configuration file may refer to a text file or a specially created file in PCX format. The Red Hat Enterprise Linux distribution includes a graphical `/boot/message` file that shows a Red Hat logo. For the purposes of illustration, the configuration in Listing 1 uses a text file that is shown in Listing 2. Customizing graphical LILO messages is beyond the scope of this tutorial. Be warned that it is also not very well documented.

Listing 2. LILO text boot message

```
[root@lyrebird root]# cat /boot/message2
Booting lyrebird
```

If your configuration file does not include the `message` parameter, then you will see a very terse prompt, **LIL0 boot:**. Otherwise you will see either a text prompt or a graphical background and a menu. You may need to hold down the Shift key during boot to see the prompt, as a system may be configured so that it bypasses the prompt.

If you have the default prompt, or a custom text prompt, you may press the Tab key to display a list of available images to boot. You may either type the name of an image as shown in Listing 3, or press **Enter** to select the first entry. If you have a graphical menu, use the cursor movement keys to highlight the entry you wish to boot.

Listing 3. Sample LILO prompt

```
LILO

Booting lyrebird

boot:
latest-EL      shrike          redhat9          2.4.21-40.EL
2.4.21-37a.EL  2.4.21-37.EL    early           2.4.21-32.EL
xp             WIN-XP
boot: latest-EL
```

In addition to displaying the LILO configuration file, you may use the `-q` option of the `lilo` command to display information about the LILO boot choices. Add `-v` options for more verbose output. Two examples using the configuration file of Listing 1 are shown in Listing 4.

Listing 4. Displaying LILO configuration

```
[root@lyrebird root]# lilo -q
latest-EL      *
shrike
redhat9
2.4.21-40.EL
2.4.21-37a.EL
2.4.21-37.EL
early
2.4.21-32.EL
xp
WIN-XP
[root@lyrebird root]# lilo -q -v | tail +22 | head -n 9
shrike
  Password is required for specifying options
  Boot command-line won't be locked
  No single-key activation
  VGA mode is taken from boot image
  Kernel is loaded "high", at 0x00100000
  Initial RAM disk is 149789 bytes
  No fallback
  Options: "ro BOOT_FILE=/mnt/hda7/boot/vmlinuz-2.4.20-31.9 hdd=ide-scsi root=LABEL=RH9"
```

GRUB

The GRUB configuration file defaults to `/boot/grub/grub.conf` or `/boot/grub/menu.lst`. If both are present, one will usually be a symbolic link to the other. Listing 5 shows an example from the same system that you saw above for LILO, although only a few of the entries are illustrated here.

Listing 5. Sample GRUB configuration

```
default=1
timeout=10
splashimage=(hd0,2)/boot/grub/fig1x.xpm.gz
```

```
foreground=23334c
background=82a6bc
password --md5 $1$H8LlM1$cI0Lfs5.C06xFJYPQ8Ixz/
title Red Hat Linux (2.4.20-31.9)
    root (hd0,6)
    kernel /boot/vmlinuz-2.4.20-31.9 ro root=LABEL=RH9 hdd=ide-scsi
    initrd /boot/initrd-2.4.20-31.9.img
    savedefault
    boot

title Red Hat Enterprise Linux WS A (2.4.21-40.EL)
    root (hd0,10)
    kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi
    initrd /boot/initrd-2.4.21-40.EL.img

title Win/XP
    rootnoverify (hd0,0)
    chainloader +1
```

GRUB provides a menu interface instead of LILO's prompt. It can also use a password encrypted with the MD5 algorithm as opposed to the plain text password of LILO. And, perhaps most importantly, changes made to the GRUB configuration file do not require GRUB to be reinstalled in the MBR. Note that many distributions will automatically update the GRUB (or LILO) configuration file when updating to a new kernel level, but if you install a new kernel yourself or create a new initial RAM disk, you may need to edit the configuration file.

GRUB also does not require a partition to be mounted in order to configure a boot entry for it. You will notice entries such as `root (hd0,6)` and `splashimage=(hd0,2)/boot/grub/fig1x.xpm.gz`. GRUB refers to your hard disks as `hdn`, where `n` is an integer starting from 0. Similarly, partitions on a disk are similarly numbered starting from 0.

So, on this system, `(hd0,2)` represents the primary partition `/dev/hda3`, while `(hd0,6)` represents the logical partition `/dev/hda7`. A floppy drive is usually `(fd0)`. Remember to quote these if you are invoking GRUB with parameters from a bash shell, for example, when installing GRUB onto a floppy or your MBR.

If you want a different background image for GRUB, you are limited to 14 colors. Your favorite JPEG image may look a little different when reduced to 14 colors. You can see the effect in Figure 1, which shows the image from the above configuration file using a photo I took in Glacier Bay, Alaska. You may also want to pick suitable foreground and background colors for your textual prompts from the colors in the image; Figure 2 illustrates custom foreground and background colors.

Figure 1. Photo reduced to 14 colors for a GRUB splash image



Figure 2. Text and text background colors chosen from photo colors

Red Hat Enterprise Linux WS A (2.4.21-40.EL)

When the GRUB menu is displayed, you select a boot image using the cursor movement keys to move up and down the list.

Unlike LILO, GRUB behaves as a small shell, with several commands that allow you to do things such as edit the commands before they are executed or do things such as find and load a configuration file, or display files using a `cat` command. From the menu, you may press **e** on an entry to edit it, **c** to switch to a GRUB command line, **b** to boot the system, **p** to enter a password, and **Esc** to return to the menu or to the previous step. There is also a `grub` command, which creates a simulated shell in which you may test your GRUB configuration or your GRUB command skills. Within the GRUB shell, the `help` command provides a list of commands. Using `help commandname` provides help for the command named *commandname*. Listing 6 illustrates the help and the available commands.

Listing 6. Using the GRUB shell

```
[root@lyrebird root]# grub
Probing devices to guess BIOS drives. This may take a long time.
find FILENAME          geometry DRIVE [CYLINDER HEAD SECTOR [
halt [--no-apm]        help [--all] [PATTERN ...]
hide PARTITION         initrd FILE [ARG ...]
kernel [--no-mem-option] [--type=TYPE] makeactive
map TO_DRIVE FROM_DRIVE md5crypt
module FILE [ARG ...]  modulenounzip FILE [ARG ...]
pager [FLAG]           partnew PART TYPE START LEN
parttype PART TYPE     quit
reboot                root [DEVICE [HDBIAS]]
rootnoverify [DEVICE [HDBIAS]] serial [--unit=UNIT] [--port=PORT] [--
setkey [TO_KEY FROM_KEY] setup [--prefix=DIR] [--stage2=STAGE2_
terminal [--dumb] [--no-echo] [--no-ed terminfo [--name=NAME --cursor-address
testvbe MODE           unhide PARTITION
uppermem KBYTES       vbeprobe [MODE]
```

```
grub> help rootnoverify
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
    Similar to `root', but don't attempt to mount the partition. This
    is useful for when an OS is outside of the area of the disk that
```

```
GRUB can read, but setting the correct root device is still
desired. Note that the items mentioned in `root' which derived
from attempting the mount will NOT work correctly.
```

```
grub>
```

As a practical example, you might continue with the previous example and use GRUB's `find` command to find configuration files. Next, you might load the configuration file from `(hd0,2)` which is `/dev/hda3`, as illustrated in Listing 7.

Listing 7. Using GRUB to find and load a GRUB configuration file

```
grub> find /boot/grub/menu.lst
(hd0,2)
(hd0,6)
(hd0,7)
(hd0,8)
(hd0,9)
(hd0,10)

grub> configfile (hd0,2)/boot/grub/menu.lst
```

When you load the configuration file, you might see a menu similar to that in Listing 8. Remember that this was done under the GRUB shell, which simulates the real GRUB environment and does not display the splash image. However, this is essentially the same as you would see superimposed on your splash image when you really boot the system using GRUB.

Listing 8. The GRUB menu

```
GRUB version 0.93 (640K lower / 3072K upper memory)

+-----+
| Red Hat Linux (2.4.20-31.9)
| Red Hat Linux (2.4.20-6)
| Red Hat Enterprise Linux WS A (2.4.21-40.EL)
| Red Hat Enterprise Linux WS A (2.4.21-37.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-37.EL)
| Red Hat Enterprise Linux WS A (2.4.21-32.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.4.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.2.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-20.EL)
| Red Hat Enterprise Linux WS (2.4.21-27.0.1.EL)
| Red Hat Enterprise Linux WS (2.4.21-15.0.2.EL)
+-----+
v

Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS or 'p' to enter a
password to unlock the next set of features.
```

The highlighted entry will be booted automatically in 5 seconds.

Suppose you highlighted the third entry for Red Hat Enterprise Linux WS A (2.4.21-40.EL) and the pressed `e` to edit it. You would see something similar to

Listing 9.

Listing 9. Editing a GRUB configuration entry

```
GRUB version 0.93 (640K lower / 3072K upper memory)

+-----+
| root (hd0,10)                                     |
| kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi |
| initrd /boot/initrd-2.4.21-40.EL.img           |
+-----+

Use the ^ and v keys to select which entry is highlighted.
Press 'b' to boot, 'e' to edit the selected command in the
boot sequence, 'c' for a command-line, 'o' to open a new line
after ('O' for before) the selected line, 'd' to remove the
selected line, or escape to go back to the main menu.
```

Again, you use the arrow keys to select the line to be edited, and then press **e** to edit it. For example, if you had deleted the partition `/dev/hda5`, then your root partition should now be `/dev/hda10` or `(hd0,9)` instead of `/dev/hda11` or `(hd0,10)`. Back up and edit the value. When done, press **Enter** to accept your change, or press **Esc** to cancel. Finally, press **b** to boot the system.

GRUB has enough capability to display files on your filesystem, and it is run as if it were the root user, so you really need to protect your system with GRUB passwords. Remember, however, that if a user can boot from removable media, that user can provide his or her own GRUB configuration. See the security section in the GRUB manual (see [Resources](#)) for more information on GRUB security and other aspects of GRUB. You may also view it on your system using the `info grub` command.

Kernel parameters

Kernel parameters (sometimes called boot parameters) supply the kernel with information about hardware parameters that it might not determine on its own, to override values that it might otherwise detect or to avoid detection of inappropriate values. For example, you might want to boot an SMP system in uniprocessor mode, or you may want to specify an alternate root filesystem. Some kernel levels require a parameter to enable large memory support on systems with more than a certain amount of RAM.

If you use LILO, you specify additional (or overriding) parameters after you type the name of the kernel to be booted. For example, if you had just built a new kernel called `/boot/vmlinuz-2.4.21-40.EL-prep`, you might enter a command to tell LILO to use it, as shown in Listing 10.

Listing 10. Specifying boot parameters with LILO

```
boot: latest-EL image=/boot/vmlinuz-2.4.21-40.EL-prep
```

With GRUB, you could type in another set of commands for the kernel and `initrd` statements, or, preferably, you could use the edit facility that you just learned about to edit an existing entry by adding `-prep` to the end of the existing kernel image name.

When the kernel finishes loading, it usually starts `/sbin/init`. This program remains running until the system is shut down. It is always assigned process ID 1, as you can see in Listing 11.

Listing 11. The init process

```
[root@lyrebird root]# ps --pid 1
  PID TTY          TIME CMD
    1 ?            00:00:04 init
```

The `init` program boots the rest of your system by running a series of scripts. These scripts typically live in `/etc/rc.d/init.d` or `/etc/init.d`, and they perform services such as setting the system's hostname, checking the filesystem for errors, mounting additional filesystems, enabling networking, starting print services, and so on. When the scripts complete, `init` starts a program called `getty`, which displays the login prompt on consoles. Graphical login screens are handled differently as you learned in the tutorial "[LPI exam 102 prep, topic 105: Kernel](#)."

If your system will load a kernel, but cannot run `init` successfully, you may try to recover by specifying an alternate initialization program. For example, specifying `init=/bin/sh` will boot your system into a shell prompt with root authority, from which you might be able to repair the system.

You can find out more about the available boot parameters using the man pages for `bootparam`, or by browsing `/usr/src/linux/Documentation/ramdisk.txt`, which may be called `/usr/src/linux-$(uname -r)/Documentation/kernel-parameters.txt` on some systems.

Needless to say, if you have to apply the same set of additional parameters every time you boot, you should add them to the configuration file. Remember to rerun `lilo` if you are using LILO.

Boot events

During the Linux boot process, a large number of messages are emitted to the console, describing the kernel being booted, the hardware of your system, and other

things related to the kernel. These messages usually flash by quickly and you probably won't be able to read them, unless there is a delay while the boot process waits for something, such as inability to reach a time server, or a filesystem that must be checked. With the advent of the Linux Bootsplash project (see [Resources](#)), these messages may be superimposed on a graphical background, or they may be hidden and replaced by a simple status bar. If your distribution supports the hidden mode, you will usually be able to switch back to displaying boot messages by pressing a key such as F2.

dmesg

It's nice to be able to go back and review the kernel messages. Since standard output is related to a process, and the kernel does not have a process identifier, it keeps kernel (and module) output messages in the *kernel ring buffer*. You may display the kernel ring buffer using the `dmesg` command, which displays these messages on standard output. Of course, you may redirect this output to a file for later analysis or forward it to a kernel developer for debugging purposes. Listing 12 illustrates some of the output that you might see.

Listing 12. Partial dmesg output

```
[root@lyrebird root]# dmesg | head -n 30
Linux version 2.4.21-40.EL (bhcompile@hs20-bc1-7.build.redhat.com) (gcc version 3.2.3
20030502 (Red Hat Linux 3.2.3-54)) #1 Thu Feb 2 22:32:00 EST 2006
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000e0000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000005f6f000 (usable)
 BIOS-e820: 0000000005f6f000 - 0000000005f6fb00 (ACPI data)
 BIOS-e820: 0000000005f6fb00 - 0000000005f70000 (ACPI NVS)
 BIOS-e820: 0000000005f70000 - 0000000005f78000 (usable)
 BIOS-e820: 0000000005f78000 - 0000000006000000 (reserved)
 BIOS-e820: 00000000fec00000 - 00000000fec10000 (reserved)
 BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
 BIOS-e820: 00000000fff80000 - 00000000ffc00000 (reserved)
 BIOS-e820: 00000000fffffc00 - 0000000100000000 (reserved)
631MB HIGHMEM available.
896MB LOWMEM available.
NX protection not present; using segment protection
On node 0 totalpages: 391040
zone(0): 4096 pages.
zone(1): 225280 pages.
zone(2): 161664 pages.
IBM machine detected. Enabling interrupts during APM calls.
Kernel command line: ro root=LABEL=RHEL3 hdd=ide-scsi
ide_setup: hdd=ide-scsi
Initializing CPU#0
Detected 2392.059 MHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 4771.02 BogoMIPS
Page-cache hash table entries: 524288 (order: 9, 2048 KB)
Page-pin hash table entries: 131072 (order: 7, 512 KB)
```

The kernel ring buffer is also used for some events after the system is booted. These include certain program failures and hot-plug events. Listing 13 shows an

entry for a program that failed with a segmentation fault and several entries related to plugging in a USB memory key.

Listing 13. Later events in kernel ring buffer

```
[root@attic4 ~]# dmesg |tail -n 19
main[15961]: segfault at 0000000000529000 rip 000000000403b5d rsp 00007fffffd15d00
error 6
usb 1-4.3: new high speed USB device using ehci_hcd and address 4
scsi5 : SCSI emulation for USB Mass Storage devices
usb-storage: device found at 4
usb-storage: waiting for device to settle before scanning
Vendor: Sony          Model: Storage Media    Rev: 0100
Type:   Direct-Access   ANSI SCSI revision: 00
SCSI device sdb: 1014784 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 43 00 00 00
sdb: assuming drive cache: write through
SCSI device sdb: 1014784 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 43 00 00 00
sdb: assuming drive cache: write through
sdb: sdb1
sd 5:0:0:0: Attached scsi removable disk sdb
usb-storage: device scan complete
SELinux: initialized (dev sdb1, type vfat), uses genfs_contexts
```

/var/log/messages

Once your system has started to the point of running `/sbin/init`, the kernel still logs events in the ring buffer as you just saw, but processes use the `syslog` daemon to log messages, usually in `/var/log/messages`. In contrast to the ring buffer, each `syslog` line has a timestamp, and the file persists between system restarts. This file is where you should first look for errors that occurred during the `init` scripts stage of booting.

Most daemons have names that end in `'d'`. Listing 14 shows how to see the last few daemon status messages after a reboot.

Listing 14. Daemon messages form /var/log/messages

```
[root@lyrebird root]# grep "^Apr.*d\:" /var/log/messages|tail -n 14
Apr  2 15:36:50 lyrebird kernel: hdd: attached ide-scsi driver.
Apr  2 15:36:52 lyrebird apmd: apmd startup succeeded
Apr  2 15:36:26 lyrebird rc.sysinit: Setting hostname lyrebird: succeeded
Apr  2 15:36:26 lyrebird rc.sysinit: Initializing USB keyboard: succeeded
Apr  2 15:36:55 lyrebird sshd: succeeded
Apr  2 15:36:55 lyrebird xinetd: xinetd startup succeeded
Apr  2 15:36:56 lyrebird ntpd: succeeded
Apr  2 15:36:56 lyrebird ntpd: succeeded
Apr  2 15:36:56 lyrebird ntpd: ntpd startup succeeded
Apr  2 15:36:57 lyrebird crond: crond startup succeeded
Apr  2 15:36:58 lyrebird atd: atd startup succeeded
Apr  2 15:36:58 lyrebird snastart: insmod: streams: no module by that name found
Apr  2 15:36:58 lyrebird rhnsd: rhnsd startup succeeded
```

You will also find logs for many other system programs in /var/log. For example, you can see the startup log for your X Window system that you learned about in the tutorial "[LPI exam 101 prep, Topic 110: The X Window System](#) ."

Section 3. Runlevels , shutdown, and reboot

This section covers material for topic 1.106.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Manage the runlevel of the system
- Change to single-user mode
- Set the default runlevel
- Shut down or reboot the system
- Alert users before switching runlevel
- Terminate processes properly

Runlevels

Runlevels define what tasks can be accomplished in the current state (or runlevel) of a Linux system. Every Linux system supports three basic runlevels, plus one or more runlevels for normal operation. The basic runlevels are shown in Table 3.

Level	Purpose
0	Shut down (or halt) the system
1	Single-user mode; usually aliased as s or S
6	Reboot the system

Beyond the basics, runlevel usage differs among distributions. One common usage set is shown in Table 4.

Table 4. Other common Linux runlevels

Level	Purpose
2	Multi-user mode without networking
3	Multi-user mode with networking
5	Multi-user mode with networking and the X Window System

The Slackware distribution uses runlevel 4 instead of 5 for a full system running the X Window system. Debian uses a single runlevel for any multi-user mode, typically runlevel 2. Be sure to consult the documentation for your distribution.

Default runlevel

When a Linux system starts, the default runlevel is determined from the `id:` entry in `/etc/inittab`. Listing 15 illustrates a typical entry for a system such as Red Hat Enterprise Linux, which uses runlevel 5 for the X Window System.

Listing 15. Default runlevel in `/etc/inittab`

```
[root@lyrebird root]# grep "^id:" /etc/inittab
id:5:initdefault:
```

Changing runlevels

There are several ways to change runlevels. To make a permanent change, you can edit `/etc/inittab` and change the default level that you just saw above.

If you just need to bring the system up in a different runlevel, you have a couple of ways to do this. For example, suppose you just installed a new kernel and need to build some kernel modules after the system booted with the new kernel, but before you start the X Window System. You might want to bring up the system in runlevel 3 to accomplish this. You do this at boot time by editing the kernel line (GRUB) or adding a parameter after the selected system name (LILO). Use a single digit to specify the desired runlevel (3, in this case). For example, you might edit a line from Listing 5 that you saw in the previous section to read as shown in Listing 16.

Listing 16. Setting default runlevel at boot time

```
kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi 3
```

Once you have finished your setup work in runlevel 3, you might want to switch to

runlevel 5. Fortunately, you do not need to reboot the system. You can use the `telinit` command to tell the `init` process what runlevel it should switch to.

You can determine the current runlevel using the `runlevel` command, which shows the previous runlevel as well as the current one. If the first output character is 'N', the runlevel has not been changed since the system was booted. Listing 17 illustrates verifying and changing the runlevel.

Listing17. Verifying and changing the runlevel

```
[root@lyrebird root]# runlevel
N 3
[root@lyrebird root]# telinit 5
[root@lyrebird root]# runlevel
3 5
```

If you use the `ls` command to display a long listing of the `telinit` command, you will see that it really is a symbolic link to the `init` command. The `init` executable knows which way it was called and behaves accordingly. Since `init` normally runs as PID 1, it is also smart enough to know if you invoke it using `init` rather than `telinit`. If you do, it will assume you want it to behave as if you had called `telinit` instead. For example, you may use `int 5` instead of `telinit 5` to switch to runlevel 5.

Single-user mode

In contrast to personal computer operating systems such as DOS or Windows, Linux is inherently a multiuser system. However, there are times when that can be a problem, such as when you need to recover a major filesystem or database, or install and test some new hardware. Runlevel 1, or *single-user mode*, is your answer for these situations. The actual implementation varies by distribution, but you will usually start in a shell with only a minimal system. Usually there will be no networking and no (or very few) daemons running. On some systems, you must authenticate by logging in, but on others you go straight into a shell prompt as `root`. Single-user mode can be a lifesaver, but you can also destroy your system, so always be very careful whenever you are running with root authority.

As with switching to regular multiuser runlevels, you can also switch to single-user mode using `telinit 1`. As noted in Table 3, 's' and 'S' are aliases for runlevel 1, so you could, for example, use `telinit s` instead.

Shutdown commands

While you can use `telinit` or `init` to stop multiuser activity and switch to single-user mode, you may also use the `shutdown` command. The `shutdown`

command sends a warning message to all logged on users and blocks further logins. It then signals `init` to switch runlevels. The `init` process then sends all running processes a `SIGTERM` signal, giving them a chance to save data or otherwise properly terminate. After 5 seconds, or another delay if specified, `init` sends a `SIGKILL` signal to forcibly end each remaining process.

By default, `shutdown` switches to runlevel 1 (single-user mode). You may specify the `-h` option to halt the system, or the `-r` option to reboot. A standard message is issued in addition to any message you specify. The time may be specified as an absolute time in `hh:mm` format, or as a relative time, `n`, where `n` is the number of minutes until shutdown. For immediate shutdown, use `now`, which is equivalent to `+0`.

If you have issued a delayed shutdown and the time has not yet expired, you may cancel the shutdown by pressing **Ctrl-c** if the command is running in the foreground, or by issuing `shutdown` with the `-c` option to cancel a pending shutdown. Listing 18 shows several examples of the use of `shutdown`, along with ways to cancel the command.

Listing 18. Shutdown examples

```
[root@lyrebird root]# shutdown 5 File system recovery needed
Broadcast message from root (pts/0) (Mon Apr  3 22:44:29 2006):

File system recovery needed
The system is going DOWN to maintenance mode in 5 minutes!

Shutdown cancelled.
[root@lyrebird root]# shutdown -r 10 Reloading updated kernel&
[1] 5388

Broadcast message from root (pts/0) (Mon Apr  3 22:45:15 2006):

Reloading updated kernel
The system is going DOWN for reboot in 10 minutes!
[root@lyrebird root]# fg
shutdown -r 10 Reloading updated kernel

Shutdown cancelled.
[root@lyrebird root]# shutdown -h 23:59&
[1] 5390
[root@lyrebird root]# shutdown -c

Shutdown cancelled.
[1]+  Done                  shutdown -h 23:59
```

You may have noticed that our last example did not cause a warning message to be sent. If the time till shutdown exceeds 15 minutes, then the message is not sent until 15 minutes before the event as shown in Listing 19. Listing 19 also shows the use of the `-t` option to increase the default delay between `SIGTERM` and `SIGKILL` signals from 5 seconds to 60 seconds.

Listing 19. Another shutdown example

```
[root@lyrebird root]# date;shutdown -t60 17 Time to do backups
Mon Apr  3 22:51:45 EDT 2006

Broadcast message from root (pts/0) (Mon Apr  3 22:53:45 2006):

Time to do backups
The system is going DOWN to maintenance mode in 15 minutes!
```

Listing 20. Rebooting the system

```
[root@lyrebird root]# reboot

Broadcast message from root (pts/0) (Mon Apr  3 22:58:27 2006):

The system is going down for reboot NOW!
```

And sure enough, the system did reboot back to runlevel 3, as is shown by the use of the `runlevel` and `uptime` commands in Listing 21.

Listing 21. Another example of rebooting the system

```
[ian@lyrebird ian]$ /sbin/runlevel
N 3
[ian@lyrebird ian]$ uptime
23:05:51 up 6 min,  1 user,  load average: 0.00, 0.06, 0.03
```

It is also possible to use `telinit` (or `init`) to shut down or reboot the system. As with other uses of `telinit`, no warning is sent to users, and the command takes effect immediately, although there is still a delay between `SIGTERM` and `SIGKILL` signals. For additional options of `telinit`, `init`, and `shutdown`, consult the appropriate man pages.

Halt, reboot, and poweroff

You should know about a few more commands related to shutdown and reboot.

- The `halt` command halts the system.
- The `poweroff` command is a symbolic link to the `halt` command, which halts the system and then attempts to power it off.
- The `reboot` command is another symbolic link to the `halt` command, which halts the system and then reboots it.

If any of these are called when the system is not in runlevel 0 or 6, then the corresponding `shutdown` command will be invoked instead.

For additional options that you may use with these commands, as well as more detailed information on their operation, consult the man page.

Runlevel configuration

By now, you may be wondering why pressing **Ctrl-Alt-Delete** on some systems causes a reboot, or how all this runlevel stuff is configured. Remember the

```
id
```

field in `/etc/inittab`? Well, there are several other fields in `/etc/inittab`, along with a set of init scripts in directories such as `rc1.d` or `rc5.d`, where the digit identifies the runlevel to which the scripts in that directory apply. Listing 22 shows the entry for **Ctrl-Alt-Delete**, so you see why it causes the system to be rebooted.

Listing 22. Trapping ctrl-alt-delete

```
[root@lyrebird root]# grep -i ctrl /etc/inittab
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

The scripts used by `init` when starting the system, changing runlevels, or shutting down are typically stored in the `/etc/init.d` or `/etc/rc.d/init.d` directory. A series of symbolic links in the `rcn.d` directories, one directory for each runlevel n , control whether a script is started when entering a runlevel or stopped when leaving it. These links start with either a `K` or an `S`, followed by a two-digit number and then the name of the service, as shown in Listing 23.

Listing 23. Init scripts

```
[root@lyrebird root]# find /etc -path "*rc[0-9]*.d/???au*"
/etc/rc.d/rc0.d/K95audit
/etc/rc.d/rc0.d/K72autofs
/etc/rc.d/rc1.d/K95audit
/etc/rc.d/rc1.d/K72autofs
/etc/rc.d/rc2.d/S20audit
/etc/rc.d/rc2.d/K72autofs
/etc/rc.d/rc3.d/S20audit
/etc/rc.d/rc3.d/S28autofs
/etc/rc.d/rc4.d/K95audit
/etc/rc.d/rc4.d/S28autofs
/etc/rc.d/rc5.d/S20audit
/etc/rc.d/rc5.d/S28autofs
/etc/rc.d/rc6.d/K95audit
/etc/rc.d/rc6.d/K72autofs
[root@lyrebird root]# cd /etc/rc.d/rc5.d
[root@lyrebird rc5.d]# ls -l ???a*
lrwxr-xr-x 1 root root 15 Jan 11 2005 S20audit -> ../init.d/audit
lrwxr-xr-x 1 root root 14 Jan 11 2005 S26apmd -> ../init.d/apmd
lrwxr-xr-x 1 root root 16 Jan 11 2005 S28autofs -> ../init.d/autofs
lrwxr-xr-x 1 root root 13 Jan 11 2005 S95atd -> ../init.d/atd
```

Here you see that the `audit` and `autofs` services have `Knn` entries in all runlevels and `Snn` entries for both in runlevels 3 and 5. The `S` indicates that the service is started when that runlevel is entered, while the `K` entry indicates that it should be stopped. The `nn` component of the link name indicates the priority order in which the service should be started or stopped. In this example, `audit` is started before `autofs`, and it is stopped later.

Consult the man pages for `init` and `inittab` for more information.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- [The Linux System Administrator's Guide](#) introduces novices to Linux system administration.
- "[Boot loader showdown: Getting to know LILO and GRUB](#)" (developerWorks, August 2005) helps you contrast and compare these two contenders.
- The [GRUB Manual](#) has a wealth of information on GRUB.
- The [GRUB Splash Image Howto](#) will help you create your own GRUB splash image.
- Check out the [Bootsplash project](#) for a graphical boot process for the Linux kernel.
- The [LILO Mini-HOWTO](#) shows you how to use the Linux Loader.
- "[Automate OS switching on a dual-boot Linux system](#)" (developerWorks, March 2006) shows you how to switch between Linux and Windows on the same machine without manual intervention.
- [LPI Linux Certification in a Nutshell](#) (O'Reilly, 2001) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content.](#)
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 106: Boot, initialization, shutdown, and runlevels

Junior Level Administration (LPIC-1) topic 106

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

04 Apr 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this second in a [series of nine tutorials](#), Ian introduces you to startup and shutdown on Linux®. By the end of this tutorial, you will know guide a system through booting, set kernel parameters, and shut down or reboot a system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding

developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	(This tutorial). Learn how to boot a system, set kernel parameters, and shut down or reboot a system. See detailed objectives below.
Topic 107	LPI exam 102 prep: Printing	Coming soon.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Boot, initialization, shutdown, and runlevels," the second of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to boot a system, set kernel parameters, and shut down or reboot a system.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.106.1 Boot the system	Weight 3	Guide the system through the booting process, including giving commands to the boot loader and setting kernel options at boot time. Learn how to check boot events in log files.
1.106.2 Change runlevels, and shut down or reboot system	Weight 3	Manage the runlevel of the system, and set the default runlevel, plus change to single-user mode, shut down and reboot the system. Learn how to alert users before switching runlevel, and how to properly terminate processes.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be very familiar with the material from the "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)" tutorial.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Boot the system

This section covers material for topic 1.106.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Guide the system through the booting process
- Give commands to the boot loader at boot time
- Give options to the kernel at boot time
- Check boot events in the log files

Boot overview

To summarize the boot process for PCs:

1. When a PC is turned on, the BIOS (*Basic Input Output Service*) performs a self test.
2. When the machine passes its self test, the BIOS loads the *Master Boot Record* (or *MBR*, usually from the first 512-byte sector of the boot drive). This is usually the first hard drive on the system, but may also be a diskette, CD, or USB key.
3. For a hard drive, the MBR loads a stage 1 boot loader, which is typically either the LILO or GRUB stage1 boot loader on a Linux system. This is another 512-byte, single-sector record.
4. The stage 1 boot loader usually loads a sequence of records called the stage 2 boot loader (or sometimes the stage 1.5 loader).
5. The stage 2 loader loads the operating system. For Linux, this is the kernel, and possibly an initial RAM disk (initrd).

At this point, your system should be able to install either of two popular boot loaders, LILO (the LInux LOader) or GRUB (the GRand Unified Boot loader). You should be able to use your chosen boot loader to boot normally as described above. Refer back to the "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)" tutorial if you need to review boot loader installation or basic booting.

To influence your system's boot process, you can:

1. Change the device from which you boot. You may normally boot from a

hard drive, but you may sometimes need to boot from a floppy disk, a USB memory key, a CD or DVD, or a network. Setting up such alternate boot devices requires your BIOS to be appropriately configured. The method for doing this is specific to your system and its BIOS. This is beyond the scope of this tutorial or the requirements of this LPI objective, so consult your system documentation.

2. You can interact with the boot loader to select which of several possible configurations you may wish to boot. You will learn how to do this for both LILO and GRUB in this tutorial.
3. You can use GRUB or LILO to pass parameters to the kernel to control the way that your kernel starts the system once it has been loaded by the boot loader.

LILO

The LILO configuration file defaults to `/etc/lilo.conf`. Listing 1 shows an example from a system that is currently running Red Hat Enterprise Linux 3. The system has a Red Hat 9 partition whose root filesystem is mounted on `/mnt/hda7` and a Windows® XP system on `/dev/hda1`.

Listing 1. Sample LILO configuration

```
[root@lyrebird root]# cat /etc/lilo.conf
prompt
timeout=50
compact
default=latest-EL
boot=/dev/fd0
map=/boot/map
install=/boot/boot.b
message=/boot/message2
lba32
password=mypassword
restricted

image=/mnt/hda7/boot/vmlinuz-2.4.20-31.9
    label=redhat9
    alias=shrike
    initrd=/mnt/hda7/boot/initrd-2.4.20-31.9.img
    read-only
    append="hdd=ide-scsi root=LABEL=RH9"

image=/boot/vmlinuz-2.4.21-40.EL
    label=2.4.21-40.EL
    alias=latest-EL
    initrd=/boot/initrd-2.4.21-40.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-37.0.1.EL
    label=2.4.21-37a.EL
    initrd=/boot/initrd-2.4.21-37.0.1.EL.img
```

```
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-37.EL
    label=2.4.21-37.EL
    initrd=/boot/initrd-2.4.21-37.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-32.0.1.EL
    label=2.4.21-32.EL
    alias=early
    initrd=/boot/initrd-2.4.21-32.0.1.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

other=/dev/hda1
    loader=/boot/chain.b
    label=WIN-XP
    alias=xp
```

Remember that whenever you make changes to `/etc/lilo.conf`, or whenever you install a new kernel, you **must** run `lilo`. The `lilo` program rewrites the MBR or the partition boot record to reflect your changes, including recording the absolute disk location of the kernel. If your configuration file includes Linux images from multiple partitions, you must mount the partitions because the `lilo` command needs to access the partition to locate the image.

The `message` parameter in the LILO configuration file may refer to a text file or a specially created file in PCX format. The Red Hat Enterprise Linux distribution includes a graphical `/boot/message` file that shows a Red Hat logo. For the purposes of illustration, the configuration in Listing 1 uses a text file that is shown in Listing 2. Customizing graphical LILO messages is beyond the scope of this tutorial. Be warned that it is also not very well documented.

Listing 2. LILO text boot message

```
[root@lyrebird root]# cat /boot/message2
Booting lyrebird
```

If your configuration file does not include the `message` parameter, then you will see a very terse prompt, **LIL0 boot:**. Otherwise you will see either a text prompt or a graphical background and a menu. You may need to hold down the Shift key during boot to see the prompt, as a system may be configured so that it bypasses the prompt.

If you have the default prompt, or a custom text prompt, you may press the Tab key to display a list of available images to boot. You may either type the name of an image as shown in Listing 3, or press **Enter** to select the first entry. If you have a graphical menu, use the cursor movement keys to highlight the entry you wish to boot.

Listing 3. Sample LILO prompt

```
LILO

Booting lyrebird

boot:
latest-EL      shrike          redhat9          2.4.21-40.EL
2.4.21-37a.EL  2.4.21-37.EL    early           2.4.21-32.EL
xp             WIN-XP
boot: latest-EL
```

In addition to displaying the LILO configuration file, you may use the `-q` option of the `lilo` command to display information about the LILO boot choices. Add `-v` options for more verbose output. Two examples using the configuration file of Listing 1 are shown in Listing 4.

Listing 4. Displaying LILO configuration

```
[root@lyrebird root]# lilo -q
latest-EL      *
shrike
redhat9
2.4.21-40.EL
2.4.21-37a.EL
2.4.21-37.EL
early
2.4.21-32.EL
xp
WIN-XP
[root@lyrebird root]# lilo -q -v | tail +22 | head -n 9
shrike
  Password is required for specifying options
  Boot command-line won't be locked
  No single-key activation
  VGA mode is taken from boot image
  Kernel is loaded "high", at 0x00100000
  Initial RAM disk is 149789 bytes
  No fallback
  Options: "ro BOOT_FILE=/mnt/hda7/boot/vmlinuz-2.4.20-31.9 hdd=ide-scsi root=LABEL=RH9"
```

GRUB

The GRUB configuration file defaults to `/boot/grub/grub.conf` or `/boot/grub/menu.lst`. If both are present, one will usually be a symbolic link to the other. Listing 5 shows an example from the same system that you saw above for LILO, although only a few of the entries are illustrated here.

Listing 5. Sample GRUB configuration

```
default=1
timeout=10
splashimage=(hd0,2)/boot/grub/fig1x.xpm.gz
```

```
foreground=23334c
background=82a6bc
password --md5 $1$H8LlM1$cI0Lfs5.C06xFJYPQ8Ixz/
title Red Hat Linux (2.4.20-31.9)
    root (hd0,6)
    kernel /boot/vmlinuz-2.4.20-31.9 ro root=LABEL=RH9 hdd=ide-scsi
    initrd /boot/initrd-2.4.20-31.9.img
    savedefault
    boot

title Red Hat Enterprise Linux WS A (2.4.21-40.EL)
    root (hd0,10)
    kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi
    initrd /boot/initrd-2.4.21-40.EL.img

title Win/XP
    rootnoverify (hd0,0)
    chainloader +1
```

GRUB provides a menu interface instead of LILO's prompt. It can also use a password encrypted with the MD5 algorithm as opposed to the plain text password of LILO. And, perhaps most importantly, changes made to the GRUB configuration file do not require GRUB to be reinstalled in the MBR. Note that many distributions will automatically update the GRUB (or LILO) configuration file when updating to a new kernel level, but if you install a new kernel yourself or create a new initial RAM disk, you may need to edit the configuration file.

GRUB also does not require a partition to be mounted in order to configure a boot entry for it. You will notice entries such as `root (hd0,6)` and `splashimage=(hd0,2)/boot/grub/fig1x.xpm.gz`. GRUB refers to your hard disks as `hdn`, where `n` is an integer starting from 0. Similarly, partitions on a disk are similarly numbered starting from 0.

So, on this system, `(hd0,2)` represents the primary partition `/dev/hda3`, while `(hd0,6)` represents the logical partition `/dev/hda7`. A floppy drive is usually `(fd0)`. Remember to quote these if you are invoking GRUB with parameters from a bash shell, for example, when installing GRUB onto a floppy or your MBR.

If you want a different background image for GRUB, you are limited to 14 colors. Your favorite JPEG image may look a little different when reduced to 14 colors. You can see the effect in Figure 1, which shows the image from the above configuration file using a photo I took in Glacier Bay, Alaska. You may also want to pick suitable foreground and background colors for your textual prompts from the colors in the image; Figure 2 illustrates custom foreground and background colors.

Figure 1. Photo reduced to 14 colors for a GRUB splash image



Figure 2. Text and text background colors chosen from photo colors

Red Hat Enterprise Linux WS A (2.4.21-40.EL)

When the GRUB menu is displayed, you select a boot image using the cursor movement keys to move up and down the list.

Unlike LILO, GRUB behaves as a small shell, with several commands that allow you to do things such as edit the commands before they are executed or do things such as find and load a configuration file, or display files using a `cat` command. From the menu, you may press **e** on an entry to edit it, **c** to switch to a GRUB command line, **b** to boot the system, **p** to enter a password, and **Esc** to return to the menu or to the previous step. There is also a `grub` command, which creates a simulated shell in which you may test your GRUB configuration or your GRUB command skills. Within the GRUB shell, the `help` command provides a list of commands. Using `help commandname` provides help for the command named *commandname*. Listing 6 illustrates the help and the available commands.

Listing 6. Using the GRUB shell

```
[root@lyrebird root]# grub
Probing devices to guess BIOS drives. This may take a long time.
find FILENAME          geometry DRIVE [CYLINDER HEAD SECTOR [
halt [--no-apm]        help [--all] [PATTERN ...]
hide PARTITION         initrd FILE [ARG ...]
kernel [--no-mem-option] [--type=TYPE] makeactive
map TO_DRIVE FROM_DRIVE md5crypt
module FILE [ARG ...]  modulenounzip FILE [ARG ...]
pager [FLAG]           partnew PART TYPE START LEN
parttype PART TYPE     quit
reboot                root [DEVICE [HDBIAS]]
rootnoverify [DEVICE [HDBIAS]] serial [--unit=UNIT] [--port=PORT] [--
setkey [TO_KEY FROM_KEY] setup [--prefix=DIR] [--stage2=STAGE2_
terminal [--dumb] [--no-echo] [--no-ed terminfo [--name=NAME --cursor-address
testvbe MODE           unhide PARTITION
uppermem KBYTES       vbeprobe [MODE]
```

```
grub> help rootnoverify
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
    Similar to `root', but don't attempt to mount the partition. This
    is useful for when an OS is outside of the area of the disk that
```

```
GRUB can read, but setting the correct root device is still
desired. Note that the items mentioned in `root' which derived
from attempting the mount will NOT work correctly.
```

```
grub>
```

As a practical example, you might continue with the previous example and use GRUB's `find` command to find configuration files. Next, you might load the configuration file from `(hd0,2)` which is `/dev/hda3`, as illustrated in Listing 7.

Listing 7. Using GRUB to find and load a GRUB configuration file

```
grub> find /boot/grub/menu.lst
(hd0,2)
(hd0,6)
(hd0,7)
(hd0,8)
(hd0,9)
(hd0,10)

grub> configfile (hd0,2)/boot/grub/menu.lst
```

When you load the configuration file, you might see a menu similar to that in Listing 8. Remember that this was done under the GRUB shell, which simulates the real GRUB environment and does not display the splash image. However, this is essentially the same as you would see superimposed on your splash image when you really boot the system using GRUB.

Listing 8. The GRUB menu

```
GRUB version 0.93 (640K lower / 3072K upper memory)

+-----+
| Red Hat Linux (2.4.20-31.9)
| Red Hat Linux (2.4.20-6)
| Red Hat Enterprise Linux WS A (2.4.21-40.EL)
| Red Hat Enterprise Linux WS A (2.4.21-37.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-37.EL)
| Red Hat Enterprise Linux WS A (2.4.21-32.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.4.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.2.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-20.EL)
| Red Hat Enterprise Linux WS (2.4.21-27.0.1.EL)
| Red Hat Enterprise Linux WS (2.4.21-15.0.2.EL)
+-----+
v

Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS or 'p' to enter a
password to unlock the next set of features.
```

The highlighted entry will be booted automatically in 5 seconds.

Suppose you highlighted the third entry for Red Hat Enterprise Linux WS A (2.4.21-40.EL) and the pressed `e` to edit it. You would see something similar to

Listing 9.

Listing 9. Editing a GRUB configuration entry

```
GRUB version 0.93 (640K lower / 3072K upper memory)

+-----+
| root (hd0,10)                               |
| kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi |
| initrd /boot/initrd-2.4.21-40.EL.img       |
+-----+

Use the ^ and v keys to select which entry is highlighted.
Press 'b' to boot, 'e' to edit the selected command in the
boot sequence, 'c' for a command-line, 'o' to open a new line
after ('O' for before) the selected line, 'd' to remove the
selected line, or escape to go back to the main menu.
```

Again, you use the arrow keys to select the line to be edited, and then press **e** to edit it. For example, if you had deleted the partition `/dev/hda5`, then your root partition should now be `/dev/hda10` or `(hd0,9)` instead of `/dev/hda11` or `(hd0,10)`. Back up and edit the value. When done, press **Enter** to accept your change, or press **Esc** to cancel. Finally, press **b** to boot the system.

GRUB has enough capability to display files on your filesystem, and it is run as if it were the root user, so you really need to protect your system with GRUB passwords. Remember, however, that if a user can boot from removable media, that user can provide his or her own GRUB configuration. See the security section in the GRUB manual (see [Resources](#)) for more information on GRUB security and other aspects of GRUB. You may also view it on your system using the `info grub` command.

Kernel parameters

Kernel parameters (sometimes called boot parameters) supply the kernel with information about hardware parameters that it might not determine on its own, to override values that it might otherwise detect or to avoid detection of inappropriate values. For example, you might want to boot an SMP system in uniprocessor mode, or you may want to specify an alternate root filesystem. Some kernel levels require a parameter to enable large memory support on systems with more than a certain amount of RAM.

If you use LILO, you specify additional (or overriding) parameters after you type the name of the kernel to be booted. For example, if you had just built a new kernel called `/boot/vmlinuz-2.4.21-40.EL-prep`, you might enter a command to tell LILO to use it, as shown in Listing 10.

Listing 10. Specifying boot parameters with LILO

```
boot: latest-EL image=/boot/vmlinuz-2.4.21-40.EL-prep
```

With GRUB, you could type in another set of commands for the kernel and `initrd` statements, or, preferably, you could use the edit facility that you just learned about to edit an existing entry by adding `-prep` to the end of the existing kernel image name.

When the kernel finishes loading, it usually starts `/sbin/init`. This program remains running until the system is shut down. It is always assigned process ID 1, as you can see in Listing 11.

Listing 11. The init process

```
[root@lyrebird root]# ps --pid 1
  PID TTY          TIME CMD
    1 ?            00:00:04 init
```

The `init` program boots the rest of your system by running a series of scripts. These scripts typically live in `/etc/rc.d/init.d` or `/etc/init.d`, and they perform services such as setting the system's hostname, checking the filesystem for errors, mounting additional filesystems, enabling networking, starting print services, and so on. When the scripts complete, `init` starts a program called `getty`, which displays the login prompt on consoles. Graphical login screens are handled differently as you learned in the tutorial "[LPI exam 102 prep, topic 105: Kernel](#)."

If your system will load a kernel, but cannot run `init` successfully, you may try to recover by specifying an alternate initialization program. For example, specifying `init=/bin/sh` will boot your system into a shell prompt with root authority, from which you might be able to repair the system.

You can find out more about the available boot parameters using the man pages for `bootparam`, or by browsing `/usr/src/linux/Documentation/ramdisk.txt`, which may be called `/usr/src/linux-$(uname -r)/Documentation/kernel-parameters.txt` on some systems.

Needless to say, if you have to apply the same set of additional parameters every time you boot, you should add them to the configuration file. Remember to rerun `lilo` if you are using LILO.

Boot events

During the Linux boot process, a large number of messages are emitted to the console, describing the kernel being booted, the hardware of your system, and other

things related to the kernel. These messages usually flash by quickly and you probably won't be able to read them, unless there is a delay while the boot process waits for something, such as inability to reach a time server, or a filesystem that must be checked. With the advent of the Linux Bootsplash project (see [Resources](#)), these messages may be superimposed on a graphical background, or they may be hidden and replaced by a simple status bar. If your distribution supports the hidden mode, you will usually be able to switch back to displaying boot messages by pressing a key such as F2.

dmesg

It's nice to be able to go back and review the kernel messages. Since standard output is related to a process, and the kernel does not have a process identifier, it keeps kernel (and module) output messages in the *kernel ring buffer*. You may display the kernel ring buffer using the `dmesg` command, which displays these messages on standard output. Of course, you may redirect this output to a file for later analysis or forward it to a kernel developer for debugging purposes. Listing 12 illustrates some of the output that you might see.

Listing 12. Partial dmesg output

```
[root@lyrebird root]# dmesg | head -n 30
Linux version 2.4.21-40.EL (bhcompile@hs20-bc1-7.build.redhat.com) (gcc version 3.2.3
20030502 (Red Hat Linux 3.2.3-54)) #1 Thu Feb 2 22:32:00 EST 2006
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000e0000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000005f6f000 (usable)
 BIOS-e820: 0000000005f6f000 - 0000000005f6fb00 (ACPI data)
 BIOS-e820: 0000000005f6fb00 - 0000000005f70000 (ACPI NVS)
 BIOS-e820: 0000000005f70000 - 0000000005f78000 (usable)
 BIOS-e820: 0000000005f78000 - 0000000006000000 (reserved)
 BIOS-e820: 00000000fec00000 - 00000000fec10000 (reserved)
 BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
 BIOS-e820: 00000000ff800000 - 00000000ffc00000 (reserved)
 BIOS-e820: 00000000fffffc00 - 0000000100000000 (reserved)
631MB HIGHMEM available.
896MB LOWMEM available.
NX protection not present; using segment protection
On node 0 totalpages: 391040
zone(0): 4096 pages.
zone(1): 225280 pages.
zone(2): 161664 pages.
IBM machine detected. Enabling interrupts during APM calls.
Kernel command line: ro root=LABEL=RHEL3 hdd=ide-scsi
ide_setup: hdd=ide-scsi
Initializing CPU#0
Detected 2392.059 MHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 4771.02 BogoMIPS
Page-cache hash table entries: 524288 (order: 9, 2048 KB)
Page-pin hash table entries: 131072 (order: 7, 512 KB)
```

The kernel ring buffer is also used for some events after the system is booted. These include certain program failures and hot-plug events. Listing 13 shows an

entry for a program that failed with a segmentation fault and several entries related to plugging in a USB memory key.

Listing 13. Later events in kernel ring buffer

```
[root@attic4 ~]# dmesg |tail -n 19
main[15961]: segfault at 0000000000529000 rip 000000000403b5d rsp 00007fffffd15d00
error 6
usb 1-4.3: new high speed USB device using ehci_hcd and address 4
scsi5 : SCSI emulation for USB Mass Storage devices
usb-storage: device found at 4
usb-storage: waiting for device to settle before scanning
Vendor: Sony          Model: Storage Media    Rev: 0100
Type:   Direct-Access      ANSI SCSI revision: 00
SCSI device sdb: 1014784 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 43 00 00 00
sdb: assuming drive cache: write through
SCSI device sdb: 1014784 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 43 00 00 00
sdb: assuming drive cache: write through
sdb: sdb1
sd 5:0:0:0: Attached scsi removable disk sdb
usb-storage: device scan complete
SELinux: initialized (dev sdb1, type vfat), uses genfs_contexts
```

/var/log/messages

Once your system has started to the point of running `/sbin/init`, the kernel still logs events in the ring buffer as you just saw, but processes use the `syslog` daemon to log messages, usually in `/var/log/messages`. In contrast to the ring buffer, each `syslog` line has a timestamp, and the file persists between system restarts. This file is where you should first look for errors that occurred during the `init` scripts stage of booting.

Most daemons have names that end in `'d'`. Listing 14 shows how to see the last few daemon status messages after a reboot.

Listing 14. Daemon messages form /var/log/messages

```
[root@lyrebird root]# grep "^Apr.*d\:" /var/log/messages|tail -n 14
Apr  2 15:36:50 lyrebird kernel: hdd: attached ide-scsi driver.
Apr  2 15:36:52 lyrebird apmd: apmd startup succeeded
Apr  2 15:36:26 lyrebird rc.sysinit: Setting hostname lyrebird: succeeded
Apr  2 15:36:26 lyrebird rc.sysinit: Initializing USB keyboard: succeeded
Apr  2 15:36:55 lyrebird sshd: succeeded
Apr  2 15:36:55 lyrebird xinetd: xinetd startup succeeded
Apr  2 15:36:56 lyrebird ntpd: succeeded
Apr  2 15:36:56 lyrebird ntpd: succeeded
Apr  2 15:36:56 lyrebird ntpd: ntpd startup succeeded
Apr  2 15:36:57 lyrebird crond: crond startup succeeded
Apr  2 15:36:58 lyrebird atd: atd startup succeeded
Apr  2 15:36:58 lyrebird snastart: insmod: streams: no module by that name found
Apr  2 15:36:58 lyrebird rhnsd: rhnsd startup succeeded
```

You will also find logs for many other system programs in /var/log. For example, you can see the startup log for your X Window system that you learned about in the tutorial "[LPI exam 101 prep, Topic 110: The X Window System](#) ."

Section 3. Runlevels , shutdown, and reboot

This section covers material for topic 1.106.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Manage the runlevel of the system
- Change to single-user mode
- Set the default runlevel
- Shut down or reboot the system
- Alert users before switching runlevel
- Terminate processes properly

Runlevels

Runlevels define what tasks can be accomplished in the current state (or runlevel) of a Linux system. Every Linux system supports three basic runlevels, plus one or more runlevels for normal operation. The basic runlevels are shown in Table 3.

Level	Purpose
0	Shut down (or halt) the system
1	Single-user mode; usually aliased as s or S
6	Reboot the system

Beyond the basics, runlevel usage differs among distributions. One common usage set is shown in Table 4.

Table 4. Other common Linux runlevels

Level	Purpose
2	Multi-user mode without networking
3	Multi-user mode with networking
5	Multi-user mode with networking and the X Window System

The Slackware distribution uses runlevel 4 instead of 5 for a full system running the X Window system. Debian uses a single runlevel for any multi-user mode, typically runlevel 2. Be sure to consult the documentation for your distribution.

Default runlevel

When a Linux system starts, the default runlevel is determined from the `id:` entry in `/etc/inittab`. Listing 15 illustrates a typical entry for a system such as Red Hat Enterprise Linux, which uses runlevel 5 for the X Window System.

Listing 15. Default runlevel in `/etc/inittab`

```
[root@lyrebird root]# grep "^id:" /etc/inittab
id:5:initdefault:
```

Changing runlevels

There are several ways to change runlevels. To make a permanent change, you can edit `/etc/inittab` and change the default level that you just saw above.

If you just need to bring the system up in a different runlevel, you have a couple of ways to do this. For example, suppose you just installed a new kernel and need to build some kernel modules after the system booted with the new kernel, but before you start the X Window System. You might want to bring up the system in runlevel 3 to accomplish this. You do this at boot time by editing the kernel line (GRUB) or adding a parameter after the selected system name (LILO). Use a single digit to specify the desired runlevel (3, in this case). For example, you might edit a line from Listing 5 that you saw in the previous section to read as shown in Listing 16.

Listing 16. Setting default runlevel at boot time

```
kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi 3
```

Once you have finished your setup work in runlevel 3, you might want to switch to

runlevel 5. Fortunately, you do not need to reboot the system. You can use the `telinit` command to tell the `init` process what runlevel it should switch to.

You can determine the current runlevel using the `runlevel` command, which shows the previous runlevel as well as the current one. If the first output character is 'N', the runlevel has not been changed since the system was booted. Listing 17 illustrates verifying and changing the runlevel.

Listing17. Verifying and changing the runlevel

```
[root@lyrebird root]# runlevel
N 3
[root@lyrebird root]# telinit 5
[root@lyrebird root]# runlevel
3 5
```

If you use the `ls` command to display a long listing of the `telinit` command, you will see that it really is a symbolic link to the `init` command. The `init` executable knows which way it was called and behaves accordingly. Since `init` normally runs as PID 1, it is also smart enough to know if you invoke it using `init` rather than `telinit`. If you do, it will assume you want it to behave as if you had called `telinit` instead. For example, you may use `int 5` instead of `telinit 5` to switch to runlevel 5.

Single-user mode

In contrast to personal computer operating systems such as DOS or Windows, Linux is inherently a multiuser system. However, there are times when that can be a problem, such as when you need to recover a major filesystem or database, or install and test some new hardware. Runlevel 1, or *single-user mode*, is your answer for these situations. The actual implementation varies by distribution, but you will usually start in a shell with only a minimal system. Usually there will be no networking and no (or very few) daemons running. On some systems, you must authenticate by logging in, but on others you go straight into a shell prompt as `root`. Single-user mode can be a lifesaver, but you can also destroy your system, so always be very careful whenever you are running with root authority.

As with switching to regular multiuser runlevels, you can also switch to single-user mode using `telinit 1`. As noted in Table 3, 's' and 'S' are aliases for runlevel 1, so you could, for example, use `telinit s` instead.

Shutdown commands

While you can use `telinit` or `init` to stop multiuser activity and switch to single-user mode, you may also use the `shutdown` command. The `shutdown`

command sends a warning message to all logged on users and blocks further logins. It then signals `init` to switch runlevels. The `init` process then sends all running processes a `SIGTERM` signal, giving them a chance to save data or otherwise properly terminate. After 5 seconds, or another delay if specified, `init` sends a `SIGKILL` signal to forcibly end each remaining process.

By default, `shutdown` switches to runlevel 1 (single-user mode). You may specify the `-h` option to halt the system, or the `-r` option to reboot. A standard message is issued in addition to any message you specify. The time may be specified as an absolute time in `hh:mm` format, or as a relative time, `n`, where `n` is the number of minutes until shutdown. For immediate shutdown, use `now`, which is equivalent to `+0`.

If you have issued a delayed shutdown and the time has not yet expired, you may cancel the shutdown by pressing **Ctrl-c** if the command is running in the foreground, or by issuing `shutdown` with the `-c` option to cancel a pending shutdown. Listing 18 shows several examples of the use of `shutdown`, along with ways to cancel the command.

Listing 18. Shutdown examples

```
[root@lyrebird root]# shutdown 5 File system recovery needed
Broadcast message from root (pts/0) (Mon Apr  3 22:44:29 2006):

File system recovery needed
The system is going DOWN to maintenance mode in 5 minutes!

Shutdown cancelled.
[root@lyrebird root]# shutdown -r 10 Reloading updated kernel&
[1] 5388

Broadcast message from root (pts/0) (Mon Apr  3 22:45:15 2006):

Reloading updated kernel
The system is going DOWN for reboot in 10 minutes!
[root@lyrebird root]# fg
shutdown -r 10 Reloading updated kernel

Shutdown cancelled.
[root@lyrebird root]# shutdown -h 23:59&
[1] 5390
[root@lyrebird root]# shutdown -c

Shutdown cancelled.
[1]+  Done                  shutdown -h 23:59
```

You may have noticed that our last example did not cause a warning message to be sent. If the time till shutdown exceeds 15 minutes, then the message is not sent until 15 minutes before the event as shown in Listing 19. Listing 19 also shows the use of the `-t` option to increase the default delay between `SIGTERM` and `SIGKILL` signals from 5 seconds to 60 seconds.

Listing 19. Another shutdown example

```
[root@lyrebird root]# date;shutdown -t60 17 Time to do backups
Mon Apr  3 22:51:45 EDT 2006

Broadcast message from root (pts/0) (Mon Apr  3 22:53:45 2006):

Time to do backups
The system is going DOWN to maintenance mode in 15 minutes!
```

Listing 20. Rebooting the system

```
[root@lyrebird root]# reboot

Broadcast message from root (pts/0) (Mon Apr  3 22:58:27 2006):

The system is going down for reboot NOW!
```

And sure enough, the system did reboot back to runlevel 3, as is shown by the use of the `runlevel` and `uptime` commands in Listing 21.

Listing 21. Another example of rebooting the system

```
[ian@lyrebird ian]$ /sbin/runlevel
N 3
[ian@lyrebird ian]$ uptime
23:05:51 up 6 min,  1 user,  load average: 0.00, 0.06, 0.03
```

It is also possible to use `telinit` (or `init`) to shut down or reboot the system. As with other uses of `telinit`, no warning is sent to users, and the command takes effect immediately, although there is still a delay between `SIGTERM` and `SIGKILL` signals. For additional options of `telinit`, `init`, and `shutdown`, consult the appropriate man pages.

Halt, reboot, and poweroff

You should know about a few more commands related to shutdown and reboot.

- The `halt` command halts the system.
- The `poweroff` command is a symbolic link to the `halt` command, which halts the system and then attempts to power it off.
- The `reboot` command is another symbolic link to the `halt` command, which halts the system and then reboots it.

If any of these are called when the system is not in runlevel 0 or 6, then the corresponding `shutdown` command will be invoked instead.

For additional options that you may use with these commands, as well as more detailed information on their operation, consult the man page.

Runlevel configuration

By now, you may be wondering why pressing **Ctrl-Alt-Delete** on some systems causes a reboot, or how all this runlevel stuff is configured. Remember the

```
id
```

field in `/etc/inittab`? Well, there are several other fields in `/etc/inittab`, along with a set of init scripts in directories such as `rc1.d` or `rc5.d`, where the digit identifies the runlevel to which the scripts in that directory apply. Listing 22 shows the entry for **Ctrl-Alt-Delete**, so you see why it causes the system to be rebooted.

Listing 22. Trapping ctrl-alt-delete

```
[root@lyrebird root]# grep -i ctrl /etc/inittab
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

The scripts used by `init` when starting the system, changing runlevels, or shutting down are typically stored in the `/etc/init.d` or `/etc/rc.d/init.d` directory. A series of symbolic links in the `rcn.d` directories, one directory for each runlevel n , control whether a script is started when entering a runlevel or stopped when leaving it. These links start with either a `K` or an `S`, followed by a two-digit number and then the name of the service, as shown in Listing 23.

Listing 23. Init scripts

```
[root@lyrebird root]# find /etc -path "*rc[0-9]*.d/???au*"
/etc/rc.d/rc0.d/K95audit
/etc/rc.d/rc0.d/K72autofs
/etc/rc.d/rc1.d/K95audit
/etc/rc.d/rc1.d/K72autofs
/etc/rc.d/rc2.d/S20audit
/etc/rc.d/rc2.d/K72autofs
/etc/rc.d/rc3.d/S20audit
/etc/rc.d/rc3.d/S28autofs
/etc/rc.d/rc4.d/K95audit
/etc/rc.d/rc4.d/S28autofs
/etc/rc.d/rc5.d/S20audit
/etc/rc.d/rc5.d/S28autofs
/etc/rc.d/rc6.d/K95audit
/etc/rc.d/rc6.d/K72autofs
[root@lyrebird root]# cd /etc/rc.d/rc5.d
[root@lyrebird rc5.d]# ls -l ???a*
lrwxr-xr-x 1 root root 15 Jan 11 2005 S20audit -> ../init.d/audit
lrwxr-xr-x 1 root root 14 Jan 11 2005 S26apmd -> ../init.d/apmd
lrwxr-xr-x 1 root root 16 Jan 11 2005 S28autofs -> ../init.d/autofs
lrwxr-xr-x 1 root root 13 Jan 11 2005 S95atd -> ../init.d/atd
```

Here you see that the `audit` and `autofs` services have `Knn` entries in all runlevels and `Snn` entries for both in runlevels 3 and 5. The `S` indicates that the service is started when that runlevel is entered, while the `K` entry indicates that it should be stopped. The `nn` component of the link name indicates the priority order in which the service should be started or stopped. In this example, `audit` is started before `autofs`, and it is stopped later.

Consult the man pages for `init` and `inittab` for more information.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- [The Linux System Administrator's Guide](#) introduces novices to Linux system administration.
- "[Boot loader showdown: Getting to know LILO and GRUB](#)" (developerWorks, August 2005) helps you contrast and compare these two contenders.
- The [GRUB Manual](#) has a wealth of information on GRUB.
- The [GRUB Splash Image Howto](#) will help you create your own GRUB splash image.
- Check out the [Bootsplash project](#) for a graphical boot process for the Linux kernel.
- The [LILO Mini-HOWTO](#) shows you how to use the Linux Loader.
- "[Automate OS switching on a dual-boot Linux system](#)" (developerWorks, March 2006) shows you how to switch between Linux and Windows on the same machine without manual intervention.
- [LPI Linux Certification in a Nutshell](#) (O'Reilly, 2001) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content.](#)
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 106: Boot, initialization, shutdown, and runlevels

Junior Level Administration (LPIC-1) topic 106

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

04 Apr 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this second in a [series of nine tutorials](#), Ian introduces you to startup and shutdown on Linux®. By the end of this tutorial, you will know guide a system through booting, set kernel parameters, and shut down or reboot a system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding

developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	(This tutorial). Learn how to boot a system, set kernel parameters, and shut down or reboot a system. See detailed objectives below.
Topic 107	LPI exam 102 prep: Printing	Coming soon.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Boot, initialization, shutdown, and runlevels," the second of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to boot a system, set kernel parameters, and shut down or reboot a system.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.106.1 Boot the system	Weight 3	Guide the system through the booting process, including giving commands to the boot loader and setting kernel options at boot time. Learn how to check boot events in log files.
1.106.2 Change runlevels, and shut down or reboot system	Weight 3	Manage the runlevel of the system, and set the default runlevel, plus change to single-user mode, shut down and reboot the system. Learn how to alert users before switching runlevel, and how to properly terminate processes.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be very familiar with the material from the "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)" tutorial.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Boot the system

This section covers material for topic 1.106.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Guide the system through the booting process
- Give commands to the boot loader at boot time
- Give options to the kernel at boot time
- Check boot events in the log files

Boot overview

To summarize the boot process for PCs:

1. When a PC is turned on, the BIOS (*Basic Input Output Service*) performs a self test.
2. When the machine passes its self test, the BIOS loads the *Master Boot Record* (or *MBR*, usually from the first 512-byte sector of the boot drive). This is usually the first hard drive on the system, but may also be a diskette, CD, or USB key.
3. For a hard drive, the MBR loads a stage 1 boot loader, which is typically either the LILO or GRUB stage1 boot loader on a Linux system. This is another 512-byte, single-sector record.
4. The stage 1 boot loader usually loads a sequence of records called the stage 2 boot loader (or sometimes the stage 1.5 loader).
5. The stage 2 loader loads the operating system. For Linux, this is the kernel, and possibly an initial RAM disk (initrd).

At this point, your system should be able to install either of two popular boot loaders, LILO (the LInux LOader) or GRUB (the GRand Unified Boot loader). You should be able to use your chosen boot loader to boot normally as described above. Refer back to the "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)" tutorial if you need to review boot loader installation or basic booting.

To influence your system's boot process, you can:

1. Change the device from which you boot. You may normally boot from a

hard drive, but you may sometimes need to boot from a floppy disk, a USB memory key, a CD or DVD, or a network. Setting up such alternate boot devices requires your BIOS to be appropriately configured. The method for doing this is specific to your system and its BIOS. This is beyond the scope of this tutorial or the requirements of this LPI objective, so consult your system documentation.

2. You can interact with the boot loader to select which of several possible configurations you may wish to boot. You will learn how to do this for both LILO and GRUB in this tutorial.
3. You can use GRUB or LILO to pass parameters to the kernel to control the way that your kernel starts the system once it has been loaded by the boot loader.

LILO

The LILO configuration file defaults to `/etc/lilo.conf`. Listing 1 shows an example from a system that is currently running Red Hat Enterprise Linux 3. The system has a Red Hat 9 partition whose root filesystem is mounted on `/mnt/hda7` and a Windows® XP system on `/dev/hda1`.

Listing 1. Sample LILO configuration

```
[root@lyrebird root]# cat /etc/lilo.conf
prompt
timeout=50
compact
default=latest-EL
boot=/dev/fd0
map=/boot/map
install=/boot/boot.b
message=/boot/message2
lba32
password=mypassword
restricted

image=/mnt/hda7/boot/vmlinuz-2.4.20-31.9
    label=redhat9
    alias=shrike
    initrd=/mnt/hda7/boot/initrd-2.4.20-31.9.img
    read-only
    append="hdd=ide-scsi root=LABEL=RH9"

image=/boot/vmlinuz-2.4.21-40.EL
    label=2.4.21-40.EL
    alias=latest-EL
    initrd=/boot/initrd-2.4.21-40.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-37.0.1.EL
    label=2.4.21-37a.EL
    initrd=/boot/initrd-2.4.21-37.0.1.EL.img
```

```
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-37.EL
    label=2.4.21-37.EL
    initrd=/boot/initrd-2.4.21-37.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-32.0.1.EL
    label=2.4.21-32.EL
    alias=early
    initrd=/boot/initrd-2.4.21-32.0.1.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

other=/dev/hda1
    loader=/boot/chain.b
    label=WIN-XP
    alias=xp
```

Remember that whenever you make changes to `/etc/lilo.conf`, or whenever you install a new kernel, you **must** run `lilo`. The `lilo` program rewrites the MBR or the partition boot record to reflect your changes, including recording the absolute disk location of the kernel. If your configuration file includes Linux images from multiple partitions, you must mount the partitions because the `lilo` command needs to access the partition to locate the image.

The `message` parameter in the LILO configuration file may refer to a text file or a specially created file in PCX format. The Red Hat Enterprise Linux distribution includes a graphical `/boot/message` file that shows a Red Hat logo. For the purposes of illustration, the configuration in Listing 1 uses a text file that is shown in Listing 2. Customizing graphical LILO messages is beyond the scope of this tutorial. Be warned that it is also not very well documented.

Listing 2. LILO text boot message

```
[root@lyrebird root]# cat /boot/message2
Booting lyrebird
```

If your configuration file does not include the `message` parameter, then you will see a very terse prompt, **LILO boot:**. Otherwise you will see either a text prompt or a graphical background and a menu. You may need to hold down the Shift key during boot to see the prompt, as a system may be configured so that it bypasses the prompt.

If you have the default prompt, or a custom text prompt, you may press the Tab key to display a list of available images to boot. You may either type the name of an image as shown in Listing 3, or press **Enter** to select the first entry. If you have a graphical menu, use the cursor movement keys to highlight the entry you wish to boot.

Listing 3. Sample LILO prompt

```
LILO

Booting lyrebird

boot:
latest-EL      shrike          redhat9          2.4.21-40.EL
2.4.21-37a.EL  2.4.21-37.EL    early           2.4.21-32.EL
xp             WIN-XP
boot: latest-EL
```

In addition to displaying the LILO configuration file, you may use the `-q` option of the `lilo` command to display information about the LILO boot choices. Add `-v` options for more verbose output. Two examples using the configuration file of Listing 1 are shown in Listing 4.

Listing 4. Displaying LILO configuration

```
[root@lyrebird root]# lilo -q
latest-EL      *
shrike
redhat9
2.4.21-40.EL
2.4.21-37a.EL
2.4.21-37.EL
early
2.4.21-32.EL
xp
WIN-XP
[root@lyrebird root]# lilo -q -v | tail +22 | head -n 9
shrike
  Password is required for specifying options
  Boot command-line won't be locked
  No single-key activation
  VGA mode is taken from boot image
  Kernel is loaded "high", at 0x00100000
  Initial RAM disk is 149789 bytes
  No fallback
  Options: "ro BOOT_FILE=/mnt/hda7/boot/vmlinuz-2.4.20-31.9 hdd=ide-scsi root=LABEL=RH9"
```

GRUB

The GRUB configuration file defaults to `/boot/grub/grub.conf` or `/boot/grub/menu.lst`. If both are present, one will usually be a symbolic link to the other. Listing 5 shows an example from the same system that you saw above for LILO, although only a few of the entries are illustrated here.

Listing 5. Sample GRUB configuration

```
default=1
timeout=10
splashimage=(hd0,2)/boot/grub/fig1x.xpm.gz
```

```
foreground=23334c
background=82a6bc
password --md5 $1$H8LlM1$cI0Lfs5.C06xFJYPQ8Ixz/
title Red Hat Linux (2.4.20-31.9)
    root (hd0,6)
    kernel /boot/vmlinuz-2.4.20-31.9 ro root=LABEL=RH9 hdd=ide-scsi
    initrd /boot/initrd-2.4.20-31.9.img
    savedefault
    boot

title Red Hat Enterprise Linux WS A (2.4.21-40.EL)
    root (hd0,10)
    kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi
    initrd /boot/initrd-2.4.21-40.EL.img

title Win/XP
    rootnoverify (hd0,0)
    chainloader +1
```

GRUB provides a menu interface instead of LILO's prompt. It can also use a password encrypted with the MD5 algorithm as opposed to the plain text password of LILO. And, perhaps most importantly, changes made to the GRUB configuration file do not require GRUB to be reinstalled in the MBR. Note that many distributions will automatically update the GRUB (or LILO) configuration file when updating to a new kernel level, but if you install a new kernel yourself or create a new initial RAM disk, you may need to edit the configuration file.

GRUB also does not require a partition to be mounted in order to configure a boot entry for it. You will notice entries such as `root (hd0,6)` and `splashimage=(hd0,2)/boot/grub/fig1x.xpm.gz`. GRUB refers to your hard disks as `hdn`, where `n` is an integer starting from 0. Similarly, partitions on a disk are similarly numbered starting from 0.

So, on this system, `(hd0,2)` represents the primary partition `/dev/hda3`, while `(hd0,6)` represents the logical partition `/dev/hda7`. A floppy drive is usually `(fd0)`. Remember to quote these if you are invoking GRUB with parameters from a bash shell, for example, when installing GRUB onto a floppy or your MBR.

If you want a different background image for GRUB, you are limited to 14 colors. Your favorite JPEG image may look a little different when reduced to 14 colors. You can see the effect in Figure 1, which shows the image from the above configuration file using a photo I took in Glacier Bay, Alaska. You may also want to pick suitable foreground and background colors for your textual prompts from the colors in the image; Figure 2 illustrates custom foreground and background colors.

Figure 1. Photo reduced to 14 colors for a GRUB splash image



Figure 2. Text and text background colors chosen from photo colors

Red Hat Enterprise Linux WS A (2.4.21-40.EL)

When the GRUB menu is displayed, you select a boot image using the cursor movement keys to move up and down the list.

Unlike LILO, GRUB behaves as a small shell, with several commands that allow you to do things such as edit the commands before they are executed or do things such as find and load a configuration file, or display files using a `cat` command. From the menu, you may press **e** on an entry to edit it, **c** to switch to a GRUB command line, **b** to boot the system, **p** to enter a password, and **Esc** to return to the menu or to the previous step. There is also a `grub` command, which creates a simulated shell in which you may test your GRUB configuration or your GRUB command skills. Within the GRUB shell, the `help` command provides a list of commands. Using `help commandname` provides help for the command named *commandname*. Listing 6 illustrates the help and the available commands.

Listing 6. Using the GRUB shell

```
[root@lyrebird root]# grub
Probing devices to guess BIOS drives. This may take a long time.
find FILENAME          geometry DRIVE [CYLINDER HEAD SECTOR [
halt [--no-apm]        help [--all] [PATTERN ...]
hide PARTITION         initrd FILE [ARG ...]
kernel [--no-mem-option] [--type=TYPE] makeactive
map TO_DRIVE FROM_DRIVE md5crypt
module FILE [ARG ...]  modulenounzip FILE [ARG ...]
pager [FLAG]           partnew PART TYPE START LEN
parttype PART TYPE     quit
reboot                 root [DEVICE [HDBIAS]]
rootnoverify [DEVICE [HDBIAS]] serial [--unit=UNIT] [--port=PORT] [--
setkey [TO_KEY FROM_KEY] setup [--prefix=DIR] [--stage2=STAGE2_
terminal [--dumb] [--no-echo] [--no-ed terminfo [--name=NAME --cursor-address
testvbe MODE           unhide PARTITION
uppermem KBYTES       vbeprobe [MODE]
```

```
grub> help rootnoverify
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
    Similar to `root', but don't attempt to mount the partition. This
    is useful for when an OS is outside of the area of the disk that
```

```
GRUB can read, but setting the correct root device is still
desired. Note that the items mentioned in `root' which derived
from attempting the mount will NOT work correctly.
```

```
grub>
```

As a practical example, you might continue with the previous example and use GRUB's `find` command to find configuration files. Next, you might load the configuration file from `(hd0,2)` which is `/dev/hda3`, as illustrated in Listing 7.

Listing 7. Using GRUB to find and load a GRUB configuration file

```
grub> find /boot/grub/menu.lst
(hd0,2)
(hd0,6)
(hd0,7)
(hd0,8)
(hd0,9)
(hd0,10)

grub> configfile (hd0,2)/boot/grub/menu.lst
```

When you load the configuration file, you might see a menu similar to that in Listing 8. Remember that this was done under the GRUB shell, which simulates the real GRUB environment and does not display the splash image. However, this is essentially the same as you would see superimposed on your splash image when you really boot the system using GRUB.

Listing 8. The GRUB menu

```
GRUB version 0.93 (640K lower / 3072K upper memory)

+-----+
| Red Hat Linux (2.4.20-31.9)
| Red Hat Linux (2.4.20-6)
| Red Hat Enterprise Linux WS A (2.4.21-40.EL)
| Red Hat Enterprise Linux WS A (2.4.21-37.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-37.EL)
| Red Hat Enterprise Linux WS A (2.4.21-32.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.4.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.2.EL)
| Red Hat Enterprise Linux WS A (2.4.21-27.0.1.EL)
| Red Hat Enterprise Linux WS A (2.4.21-20.EL)
| Red Hat Enterprise Linux WS (2.4.21-27.0.1.EL)
| Red Hat Enterprise Linux WS (2.4.21-15.0.2.EL)
+-----+ v

Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS or 'p' to enter a
password to unlock the next set of features.
```

The highlighted entry will be booted automatically in 5 seconds.

Suppose you highlighted the third entry for Red Hat Enterprise Linux WS A (2.4.21-40.EL) and the pressed `e` to edit it. You would see something similar to

Listing 9.

Listing 9. Editing a GRUB configuration entry

```
GRUB version 0.93 (640K lower / 3072K upper memory)

+-----+
| root (hd0,10)                               |
| kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi |
| initrd /boot/initrd-2.4.21-40.EL.img      |
+-----+

Use the ^ and v keys to select which entry is highlighted.
Press 'b' to boot, 'e' to edit the selected command in the
boot sequence, 'c' for a command-line, 'o' to open a new line
after ('O' for before) the selected line, 'd' to remove the
selected line, or escape to go back to the main menu.
```

Again, you use the arrow keys to select the line to be edited, and then press **e** to edit it. For example, if you had deleted the partition `/dev/hda5`, then your root partition should now be `/dev/hda10` or `(hd0,9)` instead of `/dev/hda11` or `(hd0,10)`. Back up and edit the value. When done, press **Enter** to accept your change, or press **Esc** to cancel. Finally, press **b** to boot the system.

GRUB has enough capability to display files on your filesystem, and it is run as if it were the root user, so you really need to protect your system with GRUB passwords. Remember, however, that if a user can boot from removable media, that user can provide his or her own GRUB configuration. See the security section in the GRUB manual (see [Resources](#)) for more information on GRUB security and other aspects of GRUB. You may also view it on your system using the `info grub` command.

Kernel parameters

Kernel parameters (sometimes called boot parameters) supply the kernel with information about hardware parameters that it might not determine on its own, to override values that it might otherwise detect or to avoid detection of inappropriate values. For example, you might want to boot an SMP system in uniprocessor mode, or you may want to specify an alternate root filesystem. Some kernel levels require a parameter to enable large memory support on systems with more than a certain amount of RAM.

If you use LILO, you specify additional (or overriding) parameters after you type the name of the kernel to be booted. For example, if you had just built a new kernel called `/boot/vmlinuz-2.4.21-40.EL-prep`, you might enter a command to tell LILO to use it, as shown in Listing 10.

Listing 10. Specifying boot parameters with LILO

```
boot: latest-EL image=/boot/vmlinuz-2.4.21-40.EL-prep
```

With GRUB, you could type in another set of commands for the kernel and `initrd` statements, or, preferably, you could use the edit facility that you just learned about to edit an existing entry by adding `-prep` to the end of the existing kernel image name.

When the kernel finishes loading, it usually starts `/sbin/init`. This program remains running until the system is shut down. It is always assigned process ID 1, as you can see in Listing 11.

Listing 11. The `init` process

```
[root@lyrebird root]# ps --pid 1
  PID TTY          TIME CMD
    1 ?            00:00:04 init
```

The `init` program boots the rest of your system by running a series of scripts. These scripts typically live in `/etc/rc.d/init.d` or `/etc/init.d`, and they perform services such as setting the system's hostname, checking the filesystem for errors, mounting additional filesystems, enabling networking, starting print services, and so on. When the scripts complete, `init` starts a program called `getty`, which displays the login prompt on consoles. Graphical login screens are handled differently as you learned in the tutorial "[LPI exam 102 prep, topic 105: Kernel](#)."

If your system will load a kernel, but cannot run `init` successfully, you may try to recover by specifying an alternate initialization program. For example, specifying `init=/bin/sh` will boot your system into a shell prompt with root authority, from which you might be able to repair the system.

You can find out more about the available boot parameters using the man pages for `bootparam`, or by browsing `/usr/src/linux/Documentation/ramdisk.txt`, which may be called `/usr/src/linux-$(uname -r)/Documentation/kernel-parameters.txt` on some systems.

Needless to say, if you have to apply the same set of additional parameters every time you boot, you should add them to the configuration file. Remember to rerun `lilo` if you are using LILO.

Boot events

During the Linux boot process, a large number of messages are emitted to the console, describing the kernel being booted, the hardware of your system, and other

things related to the kernel. These messages usually flash by quickly and you probably won't be able to read them, unless there is a delay while the boot process waits for something, such as inability to reach a time server, or a filesystem that must be checked. With the advent of the Linux Bootsplash project (see [Resources](#)), these messages may be superimposed on a graphical background, or they may be hidden and replaced by a simple status bar. If your distribution supports the hidden mode, you will usually be able to switch back to displaying boot messages by pressing a key such as F2.

dmesg

It's nice to be able to go back and review the kernel messages. Since standard output is related to a process, and the kernel does not have a process identifier, it keeps kernel (and module) output messages in the *kernel ring buffer*. You may display the kernel ring buffer using the `dmesg` command, which displays these messages on standard output. Of course, you may redirect this output to a file for later analysis or forward it to a kernel developer for debugging purposes. Listing 12 illustrates some of the output that you might see.

Listing 12. Partial dmesg output

```
[root@lyrebird root]# dmesg | head -n 30
Linux version 2.4.21-40.EL (bhcompile@hs20-bc1-7.build.redhat.com) (gcc version 3.2.3
20030502 (Red Hat Linux 3.2.3-54)) #1 Thu Feb 2 22:32:00 EST 2006
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000e0000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000005f6f000 (usable)
 BIOS-e820: 0000000005f6f000 - 0000000005f6fb00 (ACPI data)
 BIOS-e820: 0000000005f6fb00 - 0000000005f70000 (ACPI NVS)
 BIOS-e820: 0000000005f70000 - 0000000005f78000 (usable)
 BIOS-e820: 0000000005f78000 - 0000000006000000 (reserved)
 BIOS-e820: 00000000fec00000 - 00000000fec10000 (reserved)
 BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
 BIOS-e820: 00000000fff80000 - 00000000ffc00000 (reserved)
 BIOS-e820: 00000000fffffc00 - 0000000100000000 (reserved)
631MB HIGHMEM available.
896MB LOWMEM available.
NX protection not present; using segment protection
On node 0 totalpages: 391040
zone(0): 4096 pages.
zone(1): 225280 pages.
zone(2): 161664 pages.
IBM machine detected. Enabling interrupts during APM calls.
Kernel command line: ro root=LABEL=RHEL3 hdd=ide-scsi
ide_setup: hdd=ide-scsi
Initializing CPU#0
Detected 2392.059 MHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 4771.02 BogoMIPS
Page-cache hash table entries: 524288 (order: 9, 2048 KB)
Page-pin hash table entries: 131072 (order: 7, 512 KB)
```

The kernel ring buffer is also used for some events after the system is booted. These include certain program failures and hot-plug events. Listing 13 shows an

entry for a program that failed with a segmentation fault and several entries related to plugging in a USB memory key.

Listing 13. Later events in kernel ring buffer

```
[root@attic4 ~]# dmesg |tail -n 19
main[15961]: segfault at 0000000000529000 rip 000000000403b5d rsp 00007fffffd15d00
error 6
usb 1-4.3: new high speed USB device using ehci_hcd and address 4
scsi5 : SCSI emulation for USB Mass Storage devices
usb-storage: device found at 4
usb-storage: waiting for device to settle before scanning
Vendor: Sony          Model: Storage Media    Rev: 0100
Type:   Direct-Access      ANSI SCSI revision: 00
SCSI device sdb: 1014784 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 43 00 00 00
sdb: assuming drive cache: write through
SCSI device sdb: 1014784 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 43 00 00 00
sdb: assuming drive cache: write through
sdb: sdb1
sd 5:0:0:0: Attached scsi removable disk sdb
usb-storage: device scan complete
SELinux: initialized (dev sdb1, type vfat), uses genfs_contexts
```

/var/log/messages

Once your system has started to the point of running `/sbin/init`, the kernel still logs events in the ring buffer as you just saw, but processes use the `syslog` daemon to log messages, usually in `/var/log/messages`. In contrast to the ring buffer, each `syslog` line has a timestamp, and the file persists between system restarts. This file is where you should first look for errors that occurred during the `init` scripts stage of booting.

Most daemons have names that end in `'d'`. Listing 14 shows how to see the last few daemon status messages after a reboot.

Listing 14. Daemon messages form /var/log/messages

```
[root@lyrebird root]# grep "^Apr.*d\:" /var/log/messages|tail -n 14
Apr  2 15:36:50 lyrebird kernel: hdd: attached ide-scsi driver.
Apr  2 15:36:52 lyrebird apmd: apmd startup succeeded
Apr  2 15:36:26 lyrebird rc.sysinit: Setting hostname lyrebird: succeeded
Apr  2 15:36:26 lyrebird rc.sysinit: Initializing USB keyboard: succeeded
Apr  2 15:36:55 lyrebird sshd: succeeded
Apr  2 15:36:55 lyrebird xinetd: xinetd startup succeeded
Apr  2 15:36:56 lyrebird ntpd: succeeded
Apr  2 15:36:56 lyrebird ntpd: succeeded
Apr  2 15:36:56 lyrebird ntpd: ntpd startup succeeded
Apr  2 15:36:57 lyrebird crond: crond startup succeeded
Apr  2 15:36:58 lyrebird atd: atd startup succeeded
Apr  2 15:36:58 lyrebird snastart: insmod: streams: no module by that name found
Apr  2 15:36:58 lyrebird rhnsd: rhnsd startup succeeded
```

You will also find logs for many other system programs in /var/log. For example, you can see the startup log for your X Window system that you learned about in the tutorial "[LPI exam 101 prep, Topic 110: The X Window System](#) ."

Section 3. Runlevels , shutdown, and reboot

This section covers material for topic 1.106.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Manage the runlevel of the system
- Change to single-user mode
- Set the default runlevel
- Shut down or reboot the system
- Alert users before switching runlevel
- Terminate processes properly

Runlevels

Runlevels define what tasks can be accomplished in the current state (or runlevel) of a Linux system. Every Linux system supports three basic runlevels, plus one or more runlevels for normal operation. The basic runlevels are shown in Table 3.

Level	Purpose
0	Shut down (or halt) the system
1	Single-user mode; usually aliased as s or S
6	Reboot the system

Beyond the basics, runlevel usage differs among distributions. One common usage set is shown in Table 4.

Table 4. Other common Linux runlevels

Level	Purpose
2	Multi-user mode without networking
3	Multi-user mode with networking
5	Multi-user mode with networking and the X Window System

The Slackware distribution uses runlevel 4 instead of 5 for a full system running the X Window system. Debian uses a single runlevel for any multi-user mode, typically runlevel 2. Be sure to consult the documentation for your distribution.

Default runlevel

When a Linux system starts, the default runlevel is determined from the `id:` entry in `/etc/inittab`. Listing 15 illustrates a typical entry for a system such as Red Hat Enterprise Linux, which uses runlevel 5 for the X Window System.

Listing 15. Default runlevel in `/etc/inittab`

```
[root@lyrebird root]# grep "^id:" /etc/inittab
id:5:initdefault:
```

Changing runlevels

There are several ways to change runlevels. To make a permanent change, you can edit `/etc/inittab` and change the default level that you just saw above.

If you just need to bring the system up in a different runlevel, you have a couple of ways to do this. For example, suppose you just installed a new kernel and need to build some kernel modules after the system booted with the new kernel, but before you start the X Window System. You might want to bring up the system in runlevel 3 to accomplish this. You do this at boot time by editing the kernel line (GRUB) or adding a parameter after the selected system name (LILO). Use a single digit to specify the desired runlevel (3, in this case). For example, you might edit a line from Listing 5 that you saw in the previous section to read as shown in Listing 16.

Listing 16. Setting default runlevel at boot time

```
kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi 3
```

Once you have finished your setup work in runlevel 3, you might want to switch to

runlevel 5. Fortunately, you do not need to reboot the system. You can use the `telinit` command to tell the `init` process what runlevel it should switch to.

You can determine the current runlevel using the `runlevel` command, which shows the previous runlevel as well as the current one. If the first output character is 'N', the runlevel has not been changed since the system was booted. Listing 17 illustrates verifying and changing the runlevel.

Listing17. Verifying and changing the runlevel

```
[root@lyrebird root]# runlevel
N 3
[root@lyrebird root]# telinit 5
[root@lyrebird root]# runlevel
3 5
```

If you use the `ls` command to display a long listing of the `telinit` command, you will see that it really is a symbolic link to the `init` command. The `init` executable knows which way it was called and behaves accordingly. Since `init` normally runs as PID 1, it is also smart enough to know if you invoke it using `init` rather than `telinit`. If you do, it will assume you want it to behave as if you had called `telinit` instead. For example, you may use `int 5` instead of `telinit 5` to switch to runlevel 5.

Single-user mode

In contrast to personal computer operating systems such as DOS or Windows, Linux is inherently a multiuser system. However, there are times when that can be a problem, such as when you need to recover a major filesystem or database, or install and test some new hardware. Runlevel 1, or *single-user mode*, is your answer for these situations. The actual implementation varies by distribution, but you will usually start in a shell with only a minimal system. Usually there will be no networking and no (or very few) daemons running. On some systems, you must authenticate by logging in, but on others you go straight into a shell prompt as `root`. Single-user mode can be a lifesaver, but you can also destroy your system, so always be very careful whenever you are running with root authority.

As with switching to regular multiuser runlevels, you can also switch to single-user mode using `telinit 1`. As noted in Table 3, 's' and 'S' are aliases for runlevel 1, so you could, for example, use `telinit s` instead.

Shutdown commands

While you can use `telinit` or `init` to stop multiuser activity and switch to single-user mode, you may also use the `shutdown` command. The `shutdown`

command sends a warning message to all logged on users and blocks further logins. It then signals `init` to switch runlevels. The `init` process then sends all running processes a `SIGTERM` signal, giving them a chance to save data or otherwise properly terminate. After 5 seconds, or another delay if specified, `init` sends a `SIGKILL` signal to forcibly end each remaining process.

By default, `shutdown` switches to runlevel 1 (single-user mode). You may specify the `-h` option to halt the system, or the `-r` option to reboot. A standard message is issued in addition to any message you specify. The time may be specified as an absolute time in `hh:mm` format, or as a relative time, `n`, where `n` is the number of minutes until shutdown. For immediate shutdown, use `now`, which is equivalent to `+0`.

If you have issued a delayed shutdown and the time has not yet expired, you may cancel the shutdown by pressing **Ctrl-c** if the command is running in the foreground, or by issuing `shutdown` with the `-c` option to cancel a pending shutdown. Listing 18 shows several examples of the use of `shutdown`, along with ways to cancel the command.

Listing 18. Shutdown examples

```
[root@lyrebird root]# shutdown 5 File system recovery needed
Broadcast message from root (pts/0) (Mon Apr  3 22:44:29 2006):

File system recovery needed
The system is going DOWN to maintenance mode in 5 minutes!

Shutdown cancelled.
[root@lyrebird root]# shutdown -r 10 Reloading updated kernel&
[1] 5388

Broadcast message from root (pts/0) (Mon Apr  3 22:45:15 2006):

Reloading updated kernel
The system is going DOWN for reboot in 10 minutes!
[root@lyrebird root]# fg
shutdown -r 10 Reloading updated kernel

Shutdown cancelled.
[root@lyrebird root]# shutdown -h 23:59&
[1] 5390
[root@lyrebird root]# shutdown -c

Shutdown cancelled.
[1]+  Done                  shutdown -h 23:59
```

You may have noticed that our last example did not cause a warning message to be sent. If the time till shutdown exceeds 15 minutes, then the message is not sent until 15 minutes before the event as shown in Listing 19. Listing 19 also shows the use of the `-t` option to increase the default delay between `SIGTERM` and `SIGKILL` signals from 5 seconds to 60 seconds.

Listing 19. Another shutdown example

```
[root@lyrebird root]# date;shutdown -t60 17 Time to do backups
Mon Apr  3 22:51:45 EDT 2006

Broadcast message from root (pts/0) (Mon Apr  3 22:53:45 2006):

Time to do backups
The system is going DOWN to maintenance mode in 15 minutes!
```

Listing 20. Rebooting the system

```
[root@lyrebird root]# reboot

Broadcast message from root (pts/0) (Mon Apr  3 22:58:27 2006):

The system is going down for reboot NOW!
```

And sure enough, the system did reboot back to runlevel 3, as is shown by the use of the `runlevel` and `uptime` commands in Listing 21.

Listing 21. Another example of rebooting the system

```
[ian@lyrebird ian]$ /sbin/runlevel
N 3
[ian@lyrebird ian]$ uptime
23:05:51 up 6 min,  1 user,  load average: 0.00, 0.06, 0.03
```

It is also possible to use `telinit` (or `init`) to shut down or reboot the system. As with other uses of `telinit`, no warning is sent to users, and the command takes effect immediately, although there is still a delay between `SIGTERM` and `SIGKILL` signals. For additional options of `telinit`, `init`, and `shutdown`, consult the appropriate man pages.

Halt, reboot, and poweroff

You should know about a few more commands related to shutdown and reboot.

- The `halt` command halts the system.
- The `poweroff` command is a symbolic link to the `halt` command, which halts the system and then attempts to power it off.
- The `reboot` command is another symbolic link to the `halt` command, which halts the system and then reboots it.

If any of these are called when the system is not in runlevel 0 or 6, then the corresponding `shutdown` command will be invoked instead.

For additional options that you may use with these commands, as well as more detailed information on their operation, consult the man page.

Runlevel configuration

By now, you may be wondering why pressing **Ctrl-Alt-Delete** on some systems causes a reboot, or how all this runlevel stuff is configured. Remember the

```
id
```

field in `/etc/inittab`? Well, there are several other fields in `/etc/inittab`, along with a set of init scripts in directories such as `rc1.d` or `rc5.d`, where the digit identifies the runlevel to which the scripts in that directory apply. Listing 22 shows the entry for **Ctrl-Alt-Delete**, so you see why it causes the system to be rebooted.

Listing 22. Trapping ctrl-alt-delete

```
[root@lyrebird root]# grep -i ctrl /etc/inittab
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

The scripts used by `init` when starting the system, changing runlevels, or shutting down are typically stored in the `/etc/init.d` or `/etc/rc.d/init.d` directory. A series of symbolic links in the `rcn.d` directories, one directory for each runlevel n , control whether a script is started when entering a runlevel or stopped when leaving it. These links start with either a `K` or an `S`, followed by a two-digit number and then the name of the service, as shown in Listing 23.

Listing 23. Init scripts

```
[root@lyrebird root]# find /etc -path "*rc[0-9]*.d/???au*"
/etc/rc.d/rc0.d/K95audit
/etc/rc.d/rc0.d/K72autofs
/etc/rc.d/rc1.d/K95audit
/etc/rc.d/rc1.d/K72autofs
/etc/rc.d/rc2.d/S20audit
/etc/rc.d/rc2.d/K72autofs
/etc/rc.d/rc3.d/S20audit
/etc/rc.d/rc3.d/S28autofs
/etc/rc.d/rc4.d/K95audit
/etc/rc.d/rc4.d/S28autofs
/etc/rc.d/rc5.d/S20audit
/etc/rc.d/rc5.d/S28autofs
/etc/rc.d/rc6.d/K95audit
/etc/rc.d/rc6.d/K72autofs
[root@lyrebird root]# cd /etc/rc.d/rc5.d
[root@lyrebird rc5.d]# ls -l ???a*
lrwxr-xr-x 1 root root 15 Jan 11 2005 S20audit -> ../init.d/audit
lrwxr-xr-x 1 root root 14 Jan 11 2005 S26apmd -> ../init.d/apmd
lrwxr-xr-x 1 root root 16 Jan 11 2005 S28autofs -> ../init.d/autofs
lrwxr-xr-x 1 root root 13 Jan 11 2005 S95atd -> ../init.d/atd
```

Here you see that the `audit` and `autofs` services have `Knn` entries in all runlevels and `Snn` entries for both in runlevels 3 and 5. The `S` indicates that the service is started when that runlevel is entered, while the `K` entry indicates that it should be stopped. The `nn` component of the link name indicates the priority order in which the service should be started or stopped. In this example, `audit` is started before `autofs`, and it is stopped later.

Consult the man pages for `init` and `inittab` for more information.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- [The Linux System Administrator's Guide](#) introduces novices to Linux system administration.
- "[Boot loader showdown: Getting to know LILO and GRUB](#)" (developerWorks, August 2005) helps you contrast and compare these two contenders.
- The [GRUB Manual](#) has a wealth of information on GRUB.
- The [GRUB Splash Image Howto](#) will help you create your own GRUB splash image.
- Check out the [Bootsplash project](#) for a graphical boot process for the Linux kernel.
- The [LILO Mini-HOWTO](#) shows you how to use the Linux Loader.
- "[Automate OS switching on a dual-boot Linux system](#)" (developerWorks, March 2006) shows you how to switch between Linux and Windows on the same machine without manual intervention.
- [LPI Linux Certification in a Nutshell](#) (O'Reilly, 2001) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content.](#)
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 107: Printing

Junior Level Administration (LPIC-1) topic 107

Skill Level: Intermediate

[Ian Shields](#)

Senior Programmer
IBM developerWorks

22 Aug 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this third in a [series of nine tutorials](#), Ian introduces you to printing in Linux®. By the end of this tutorial, you will know how to manage printers, print queues, and user print jobs on a Linux system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics

LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	(This tutorial). Learn how to manage printers, print queues, and user print jobs on a Linux system. See detailed objectives below.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Printing in Linux," the third of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to configure printers and manage print jobs in Linux.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.107.2 Manage printers and print queues	Weight 1	Configure and monitor print servers. Manage print queues and troubleshoot general printing problems.
1.107.3 Print files	Weight 1	Add and remove jobs from configured printer queues. Convert text files to PostScript for printing.
1.107.4 Printer installation and configuration	Weight 1	Install and configure local and remote printers, including printer daemons and print filters. Use local and remote printers, including PostScript, non-PostScript and Samba printers.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

At the time of writing, the published LPI objectives for this topic are largely directed toward the Common UNIX Printing System (CUPS), with vestiges of the earlier LPD (line printer daemon) and LPRng (the next generation line printer, or LPR) printing systems. Accordingly, this tutorial is directed mainly toward CUPS, with only minor mention of the earlier technologies. For more complete preparation, you should also review other material on LPD and LPRng printing technologies.

Section 2. Manage printers and print queues

This section covers material for topic 1.107.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Configure and monitor a print server
- Manage user print queues
- Troubleshoot general printing problems

Introduction

In the early days of computers, printing was done by *line printers*, which printed a line of text at a time using fixed-pitch characters and a single font. To speed up overall system performance, early mainframe computers interleaved work for slow peripherals such as card readers, card punches, and line printers with other work. Thus was born *Simultaneous Peripheral Operation On Line* or *spooling*, a term that is still commonly used when talking about computer printing.

In UNIX® and Linux systems, printing initially used the Berkeley Software Distribution (BSD) printing subsystem, consisting of a line printer daemon (lpd) running as a server, and client commands such as `lpr` to submit jobs for printing. This protocol was later standardized by the IETF as RFC 1179, "Line Printer Daemon Protocol".

System V UNIX also had a printing daemon. It was functionally similar to the Berkeley LPD, but had a different command set. You will frequently see two commands with different options that accomplish the same task. For example, `lpr` from the Berkeley implementation and `lp` from the System V implementation are both use to print files.

With advances in printer technology, it became possible to mix different fonts on a page and to print images as well as words. Variable pitch fonts, and more advanced printing techniques such as kerning and ligatures, all became possible. Several improvements to the basic lpd/lpr approach to printing were devised, such as LPRng, the next generation LPR, and CUPS, the Common UNIX Printing System.

Many printers capable of graphical printing use the Adobe PostScript language. A

PostScript printer has an engine that interprets the commands in a print job and produces finished pages from these commands. PostScript is often used as an intermediate form between an original file, such as a text or image file, and a final form suitable for a particular printer that does not have PostScript capability. Conversion of a print job, such as an ASCII text file or a JPEG image to PostScript, and conversion from PostScript to the final raster form required for a non-PostScript printer is done using *filters*.

The rest of this tutorial focuses on the Common UNIX Printing System (CUPS), which supports the traditional commands as well as newer graphical interfaces to printing functions. CUPS 1.1 is assumed; it includes several features not in earlier versions, such as digest passwords for increased security. Many distributions and desktops provide front-end graphical programs for CUPS, so the material covered here is not exhaustive. This tutorial covers the major concepts, but a particular implementation may differ. Note also that physical installation of your printer is beyond the scope of this tutorial.

Print servers

The CUPS server runs as a daemon process, `cupsd` under control of a configuration file normally located in `/etc/cups/cupsd.conf`. The `/etc/cups` directory also contains other configuration files related to CUPS. It is usually started during system initialization, but may be controlled by the `cups` script located in `/etc/rc.d/init.d` or `/etc/init.d`, according to your distribution. As with most such scripts, you can stop, start, or restart the daemon as shown in Listing 1.

Listing 1. Starting and stopping the cups daemon

```
[root@attic4 ~]# /etc/rc.d/init.d/cups
Usage: cups {start|stop|restart|condrestart|reload|status}
[root@attic4 ~]# /etc/rc.d/init.d/cups stop
Stopping cups:                                [ OK ]
[root@attic4 ~]# /etc/rc.d/init.d/cups start
Starting cups:                                [ OK ]
[root@attic4 ~]# /etc/rc.d/init.d/cups restart
Stopping cups:                                [ OK ]
Starting cups:                                [ OK ]
```

The configuration file, `/etc/cups/cupsd.conf`, contains parameters that you may set to control such things as access to the printing system, whether remote printing is allowed, the location of spool files, and so on. On some systems, a second part describes individual print queues and is usually generated automatically by configuration tools. Listing 2 shows some entries for a default `cupsd.conf` file. Note that comments start with a `#` character, so entries that were changed from default would have the leading `#` character removed. Note also that the spool files are stored by default in the `/var/spool` filesystem as you would expect from the Filesystem Hierarchy Standard (FHS).

Listing 2. Parts of a default /etc/cups/cupsd.conf

```
#
# RequestRoot: the directory where request files are stored.
# By default "/var/spool/cups".
#
#RequestRoot /var/spool/cups

#
# RemoteRoot: the name of the user assigned to unauthenticated accesses
# from remote systems. By default "remroot".
#
#RemoteRoot remroot

#
# ServerBin: the root directory for the scheduler executables.
# By default "/usr/lib64/cups".
#
#ServerBin /usr/lib64/cups

#
# ServerRoot: the root directory for the scheduler.
# By default "/etc/cups".
#
#ServerRoot /etc/cups
```

You should also be aware of the /etc/printcap file. This was the name of the configuration file for LPD print servers, and many applications still use it to determine available printers and their properties. It is usually generated automatically in a CUPS system, so you will probably not modify it yourself. However, you may need to check it if you are diagnosing user printing problems. An example is shown in Listing 3.

Listing 3. Automatically generated /etc/printcap

```
# This file was automatically generated by cupsd(8) from the
# /etc/cups/printers.conf file. All changes to this file
# will be lost.
xerox|Xerox Docuprint C20:rm=localhost.localdomain:rp=xerox:
anyprint|Pick any printer:rm=localhost.localdomain:rp=anyprint:
r220|Epson R220:rm=localhost.localdomain:rp=r220:
```

Each line here has a printer name and printer description as well as the name of the remote machine (rm) and remote printer (rp) on that machine. A traditional /etc/printcap file also describes the printer capabilities.

Finally, CUPS 1.1 introduced a passwd.md5 file. This allows CUPS users to be defined using the `lppasswd` command. The CUPS user ids do not have to be system userids.

Print queues

A *print queue* is a logical entity to which users direct print jobs. Frequently, particularly in single-user systems, a print queue is synonymous with a printer. However, CUPS allows a system without an attached printer to queue print jobs for eventual printing on a remote system, and also, through the use of *classes* to allow a print job directed at a class to be printed on the first available printer of that class. These are discussed more in the final section of this tutorial.

Several commands permit inspection and manipulation of print queues. Some of these have their roots in LPD commands, although the currently supported options are frequently a limited subset of those supported by the original LPD printing system. Other commands are new for CUPS. In general, a user can manipulate his or her own print jobs, but root or another authorized user is usually required to manipulate the jobs of others. Most CUPS commands support a `-E` option for encrypted communication between the CUPS client command and CUPS server.

You can check the queues known to the system using the CUPS `lpstat` command. Some common options are shown in Table 3.

Table 3. Options for lpstat	
Option	Purpose
-a	Display accepting status of printers.
-c	Display print classes.
-p	Display print status: enabled or disabled.
-s	Display default printer, printers, and classes. Equivalent to <code>-d -c -v</code> . Note that multiple options must be separated as values can be specified for many.
-s	Display printers and their devices.

You may also use the LPD `lpc` command, found in `/usr/sbin`, with the `status` option. If you do not specify a printer name, all queues are listed. Listing 4 shows some examples of both commands.

Listing 4. Displaying available print queues

```
[ian@attic4 ~]$ lpstat -d
system default destination: xerox
[ian@attic4 ~]$ lpstat -v xerox
device for xerox: lpd://192.168.0.10/PS-66D975-P1
[ian@attic4 ~]$ lpstat -s
system default destination: xerox
members of class anyprint:
    r220
    xerox
device for anyprint: ///dev/null
device for r220: smb://MSHOME/DEN/EPSON220
device for xerox: lpd://192.168.0.10/PS-66D975-P1
[ian@attic4 ~]$ lpstat -a r220
r220 accepting requests since Sat 12 Aug 2006 04:01:38 PM EDT
[ian@attic4 ~]$ /usr/sbin/lpc status xerox
xerox:
    printer is on device 'lpd' speed -1
    queuing is disabled
    printing is enabled
    no entries
    daemon present
```

This example shows two printers, xerox and r220, and a class, anyprint, which allows print jobs to be directed to the first available of these two printers.

In the previous example, queuing of print jobs to xerox is currently disabled, although printing is enabled, as might be done in order to drain the queue before taking the printer offline for maintenance. Whether queuing is enabled or disabled is controlled by the `accept` and `reject` commands. Whether printing is enabled or disabled is controlled by the `cupsenable` and `cupsdisable` commands. In earlier versions of CUPS, these were called `enable` and `disable`, which allowed confusion with the bash shell builtin `enable`. Listing 5 shows how to enable queuing on printer xerox while disabling printing. Note that an authorized user must perform these tasks. This may be root or another authorized user. See the `cupsd.conf` file and the man pages for the `lppasswd` command for more information on authorizing users.

Listing 5. Enabling queuing and disabling printing

```
[root@attic4 ~]# lpc status xerox
xerox:
    printer is on device 'lpd' speed -1
    queuing is enabled
    printing is enabled
    no entries
    daemon present
[root@attic4 ~]# cupsdisable xerox
[[root@attic4 ~]# lpstat -p -a
printer anyprint is idle.  enabled since Sat 12 Aug 2006 04:07:46 PM EDT
printer r220 is idle.  enabled since Sat 12 Aug 2006 04:01:38 PM EDT
printer xerox disabled since Sat 12 Aug 2006 06:43:09 PM EDT -
    Paused
anyprint accepting requests since Sat 12 Aug 2006 04:07:46 PM EDT
r220 accepting requests since Sat 12 Aug 2006 04:01:38 PM EDT
xerox accepting requests since Sat 12 Aug 2006 06:43:09 PM EDT
```

Managing print jobs on print queues

Now that you have seen a little of how to check on print queues and classes, let's look at how to manage print jobs on printer queues. The first thing you might want to do is find out whether any jobs are queued for a particular printer or for all printers. You do this with the `lpq` command. If no option is specified, `lpq` displays the queue for the default printer. Use the `-P` option with a printer name to specify a particular printer or the `-a` option to specify all printers, as shown in Listing 6.

Listing 6. Checking print queues with `lpq`

```
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)          Total Size
1st   brendan 14      RobotPlayer.java 1024 bytes
2nd   ian     16      .bashrc          1024 bytes
3rd   ian     17      .bashrc          1024 bytes
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)          Total Size
1st   brendan 14      RobotPlayer.java 1024 bytes
2nd   ian     16      .bashrc          1024 bytes
3rd   ian     17      .bashrc          1024 bytes
[ian@attic4 ~]$ lpq -P r220
r220 is ready
no entries
[ian@attic4 ~]$ lpq -a
Rank  Owner  Job      File(s)          Total Size
1st   brendan 14      RobotPlayer.java 1024 bytes
2nd   ian     16      .bashrc          1024 bytes
3rd   ian     17      .bashrc          1024 bytes
```

In this example, three jobs, 14, 16, and 17, are queued for the printer named xerox. Note that when the `-P` option is included, the output indicates that the printer is not ready. Note also that user ian submitted a job twice, a common user action when a job does not print the first time. You can avoid printing the extra copy by removing a job from the queue with the `lprm` command. The usual authorization setup allows a user to remove his or her own jobs, but not those of other users. The root user or another authorized user can remove jobs for other users. With no options, the current job is removed. With the `-` option, all jobs are removed. Otherwise, you can give a list of jobs to be removed as shown in Listing 7.

Listing 7. Deleting print jobs with `lprm`

```
[[ian@attic4 ~]$ lprm
Password for ian on localhost?
lprm: Unauthorized
[ian@attic4 ~]$ lprm 17
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)          Total Size
1st   brendan 14      RobotPlayer.java 1024 bytes
2nd   ian     16      .bashrc          1024 bytes
```

Note that user ian was not able to remove the first job on the queue, because it was for user brendan. However, he was able to remove his own job number 17.

Another command that will help you manipulate jobs on print queues is the `lp` command. You may use it to alter attributes of jobs, such as priority or number of copies. Suppose user ian wants his job to print before that of user brendan, and he really did want two copies of it. The job priority ranges from a lowest priority of 1 to a highest priority of 100 with a default of 50. User ian could use the `-i`, `-n`, and `-q` options to specify a job to alter and a new number of copies and priority as shown in Listing 8. Note the use of the `-l` option of the `lpq` command, which provides more verbose output.

Listing 8. Changing the number of copies and priority with lp

```
[ian@attic4 ~]$ lp -i 16 -n 2
[ian@attic4 ~]$ lpq -l
xerox is not ready

ian: 1st                               [job 16 localhost]
      2 copies of .bashrc                1024 bytes

brendan: 2nd                           [job 14 localhost]
      RobotPlayer.java                   1024 bytes
```

Finally, the `lpmove` command allows jobs to be moved from one queue to another. For example, we might want to do this because printer xerox is not currently printing. This command requires an authorized user. Listing 9 shows how to move these jobs to another queue, specifying first by printer and job id, then all jobs for a given printer. By the time we check the queues again, two of the jobs have already printed.

Listing 9. Moving jobs to another print queue with lpmove

```
[root@attic4 ~]# lpmove xerox-16 anyprint
[root@attic4 ~]# lpmove xerox r220
[root@attic4 ~]# lpq
xerox is not ready
no entries
[root@attic4 ~]# lpq -a
Rank  Owner   Job      File(s)      Total Size
----  -
active ian      18      fig1.gif     26624 bytes
```

If you happen to use a print server that is not CUPS, such as LPD or LPRng, you will find that many of the queue administration functions that we have just looked at are handled as subcommands of the `lpc` command. For example, you might use `lpc topq` to move a job to the top of a queue. Other `lpc` commands may include `disable`, `down`, `enable`, `hold`, `move`, `redirect`, `release`, and `start`.

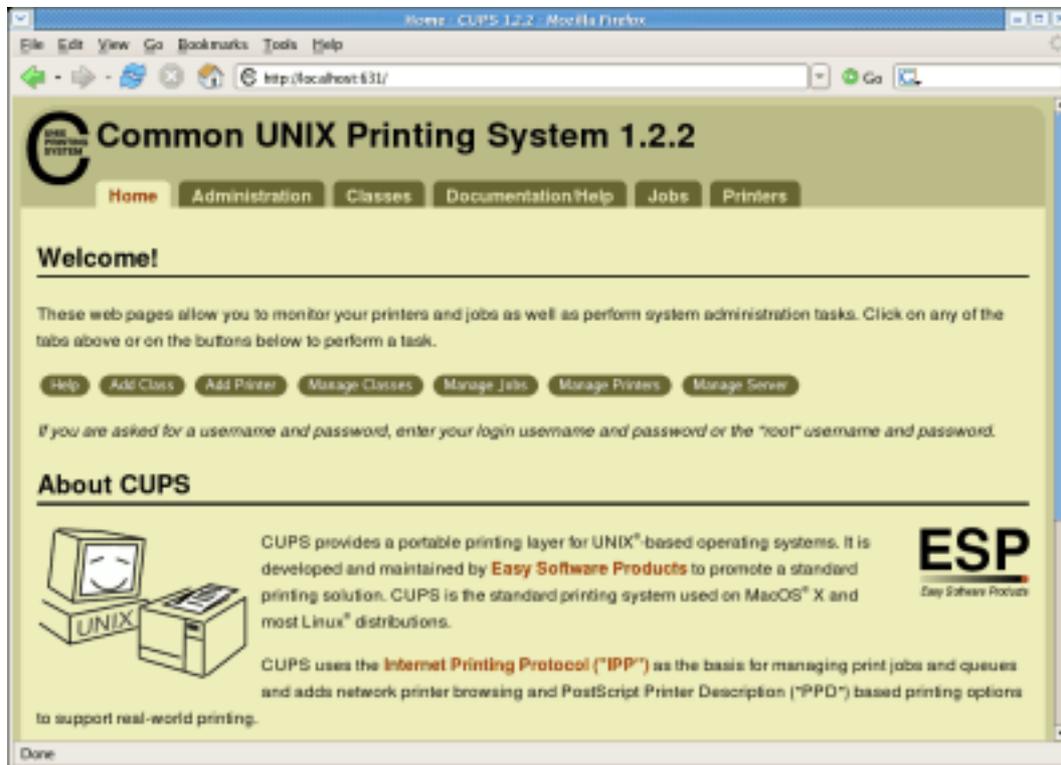
Troubleshooting

If you are having trouble printing, try these tips:

- Ensure that the cups server is running. You can use the `lpstat` command, which will report an error if it is unable to connect to the `cupsd` daemon. Alternatively, you might use the `ps -ef` command and check for `cupsd` in the output.
- If you try to queue a job for printing and get an error message indicating that the printer is not accepting jobs results, use `lpstat -a` or `lpc status` to check that the printer is accepting jobs.
- If a queued job does not print, use `lpstat -p` or `lpc status` to check that the printer is accepting jobs. You may need to move the job to another printer as discussed in the next section.
- If the printer is remote, you may need to check that it still exists on the remote system and that it is operational.
- You may need to update the configuration file to allow a particular user or remote system to print on your printer.
- You may need to ensure that your firewall allows remote printing requests, either from another system to your system, or from your system to another, as appropriate.
- You may need to verify that you have the right driver (as discussed in the final section of this tutorial).

As you can see, printing involves the correct functioning of several components of your system and possibly network. In a tutorial of this length, we can only give you starting points for diagnosis. Most CUPS systems also have a graphical interface to the command-line functions that we discuss here. Generally, this interface is accessible from the local host using a browser pointed to port 631 (<http://localhost:631> or <http://127.0.0.1:631>), as shown in Figure 1.

Figure 1. CUPS home page on port 631



Section 3. Print files

This section covers material for topic 1.107.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Add and remove jobs from configured printer queues
- Convert text files to PostScript for printing

Print

You learned how to remove files from print queues in [the previous section](#). Here you learn how to print files and change job options.

Many graphical programs provide a method of printing, usually under the **File** menu option. These programs provide graphical tools for choosing a printer, margin sizes, color or black-and-white printing, number of copies, selecting 2-up printing (which is

2 pages per sheet, often used for handouts), and so on. This section shows you the command-line tools for controlling such features, and then a graphical implementation for comparison.

The simplest way to print any file is to use the `lpr` command and provide the file name. This prints the file on the default printer. Listing 10 shows a simple example plus a more complex example. The more complex command is explained below.

Listing 10. Printing with `lpr`

```
[ian@attic4 ~]$ echo abc>abc.txt
[ian@attic4 ~]$ lpr abc.txt
[ian@attic4 ~]$ lpr -Pxerox -J "Ian's text file" -#2 -m -p -q -r abc.txt
[ian@attic4 ~]$ lpq -l
xerox is ready

ian: 1st                               [job 25 localhost]
      2 copies of Ian's text file       1024 bytes
[ian@attic4 ~]$ ls abc.txt
ls: abc.txt: No such file or directory
```

Table 4 explains the options used on the more complex command above, along with other options that you may use with `lpr`.

Table 4. Options for <code>lpr</code>	
Option	Purpose
-C, -J, or -T	Set a job name.
-P	Select a particular printer.
-#	Specify number of copies. Note this is different to the <code>-n</code> option you saw with the <code>lp</code> command.
-m	Send email upon job completion.
-l	The print file is already formatted for printing. Equivalent to <code>-o raw</code> .
-o	Set a job option.
-p	Format a text file with a shaded header. Equivalent to <code>-o prettyprint</code> .
-q	Hold (or queue) the job for later printing.
-r	Remove the file after it

```
has been spooled for
printing.
```

So, in our complex example: `lpr -Pxerox -J "Ian's text file" -#2 -m -p -q -r abc.txt`, user ian is requesting a specific printer, giving a name to the job, requesting 2 copies, requesting an email confirmation after printing, holding the job, and having the file `abc.txt` removed after it has been spooled. The subsequent commands show the held job and the fact that the file has indeed been removed.

In addition to the `lpr` command, the `lp` command covered in the previous section can also be used to print jobs, as well as modify them. Both `lp` and `lpr` also accept a file from stdin if no file name is given on the command line. In contrast to `lpr`, which quietly spools the job, the `lp` default is to display the job number of the spooled job as shown in Listing 11. Note that not all the equivalent options on `lp` and `lpr` have the same name; for example, `-n` on `lp` is equivalent to `-#` on `lpr`.

Listing 11. Printing from stdin with lp

```
[ian@attic4 ~]$ lp
abc
request id is xerox-27 (1 file(s))
```

So we now have a held job in the xerox print queue. What to do? The `lp` command has options to hold and release jobs, using various values with the `-H` option. Listing 12 shows how to release the held job. Check the man page of `lp` for information on other options.

Listing 12. Resuming printing of a held print job

```
[ian@attic4 ~]$ lp -i 25 -H resume
```

Many different printers are available today, but not all of them support the same set of options. You can find out what general options are set for a printer using the `lpoptions` command. Add the `-l` option to display printer-specific options; Listing 13 shows an example. The man page for the `lp` command also lists several common options, particularly relating to portrait/landscape printing, page dimensions, and placement of the output on the pages.

Listing 13. Checking printer options

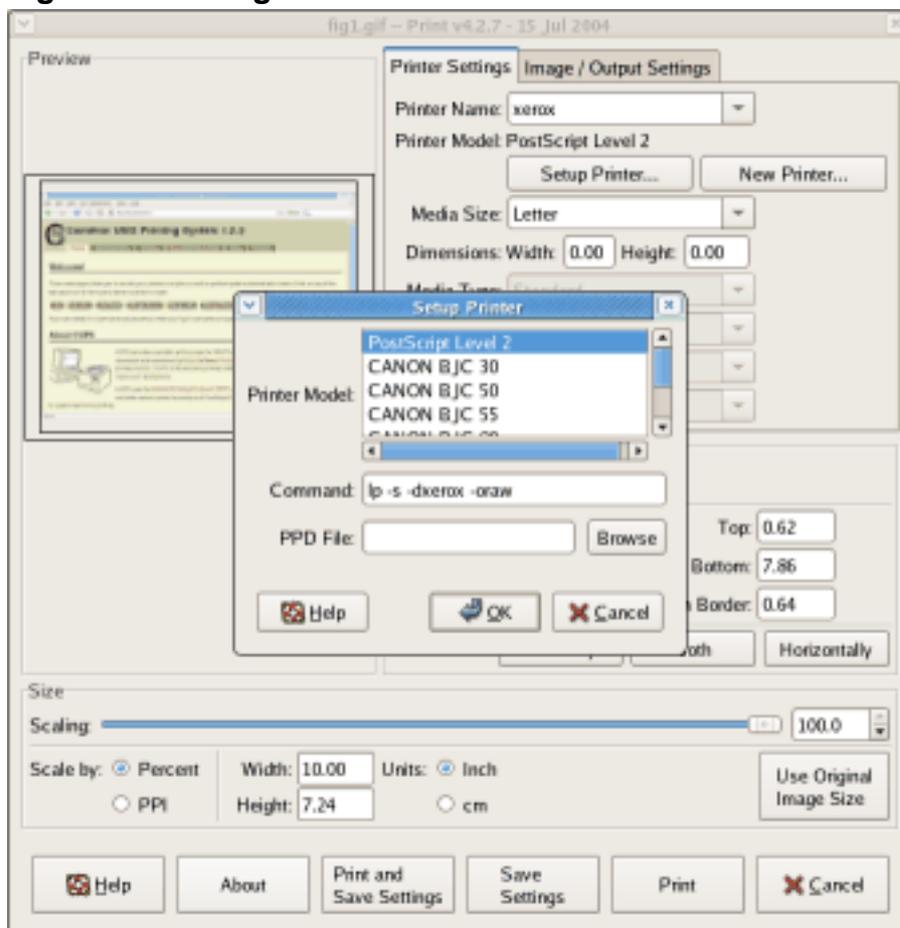
```
[ian@attic4 ~]$ lpoptions -p xerox
job-sheets=none,none printer-info='Xerox Docuprint C20' printer-is-accepting-
jobs=1 printer-is-shared=1 printer-make-and-model='Xerox DocuPrint C20 Foomat
ic/Postscript (recommended)' printer-state=3 printer-state-change-time=115550
6374 printer-state-reasons=none printer-type=143388 cpi=12 scp-fc5=true lpi=7
page-bottom=86 page-left=57 page-right=57 page-top=72 scaling=100 wrap=true
```

```
[ian@attic4 ~]$ lpoptions -l
PageSize/Page Size: *Letter A4 11x17 A3 A5 B5 Env10 EnvC5 EnvDL EnvISOB5 EnvM
onarch Executive Legal
PageRegion/PageRegion: Letter A4 11x17 A3 A5 B5 Env10 EnvC5 EnvDL EnvISOB5 En
vMonarch Executive Legal
Duplex/Double-Sided Printing: DuplexNoTumble DuplexTumble *None
Resolution/Resolution: *default 150x150dpi 300x300dpi 600x600dpi
PreFilter/GhostScript pre-filtering: EmbedFonts Level1 Level2 *No
```

So far, all our commands have been directed to the local CUPS server. You can also direct most commands to the server on another system, by specifying the `-h` option along with a port number if it is not the CUPS default of 631.

Before moving on to filters, let's look at how all this magic avails itself in a GUI application. Figure 2 shows the illustration of Figure 1 in the GIMP, an image manipulation program. Using the **File > Print** option, you have many choices about how to print the image. In this application, you can also click the **Setup Printer** button to choose a printer and see the command that will be used to print the file, in this case, `lp -s -dxerox -oraw`.

Figure 2. Printing from the GIMP



Convert files

You may have noticed that we were able to print text files above, even though the r220 printer happens to be an Epson photo printer, while the xerox printer is a PostScript Xerox Docuprint C20. This magic feat is accomplished through the use of *filters*. Indeed, a popular filter for many years was named *magicfilter*.

The number of filters included with most CUPS packages allows almost any kind of file to be printed. Additional filters are available commercially, from companies including Easy Software Products, the developers of CUPS.

CUPS uses MIME (Multipurpose Internet Mail Extensions) types to determine the appropriate conversion filter when printing a file. The section on [filter installation](#), later in this tutorial, goes into detail. Other printing packages may use the *magic number* mechanism as used by the `file` command. See the man pages for `file` or `magic` for more details.

The general print flow is to convert the input file to a PostScript format using the appropriate filter for the file type, such as `texttops`, `imagetops`, or `pdftops`. The PostScript format is then filtered through a `pstoraster` filter to create an intermediate raster format for non-PostScript printers before being filtered through a printer backend, which prepares it for printing on a particular printer. Ghostscript is a popular program that can print PostScript files on many different printers. A companion viewer allows display of the file on a monitor. Many printer backends are derived from Ghostscript printer drivers.

Before all of this was handled so automatically, it was necessary to convert input to PostScript format. Images could be handled with a program such as the GIMP that we saw earlier. ASCII text files were usually converted to PostScript using the `a2ps` command. The default for plain text files is to print 2-up with a header and direct output to the default printer as shown in Listing 14.

Listing 14. Printing text files with `a2ps`

```
[ian@attic4 ~]$ a2ps -4 abc.txt -o abc.ps
[abc.txt (plain): 1 page on 1 sheet]
[Total: 1 page on 1 sheet] saved into the file `abc.ps'
```

The `a2ps` command can handle a wide range of text file types and make intelligent decisions about the best way to format them. For example, the default for LaTeX files is to first format the file and then print 2-up. Listing 15 uses `a2ps` to print a copy of the `sample2e.tex` file that is distributed with LaTeX, and then shows a copy renamed to `sample2e.txt`, and printed 4-up with headers. Both are saved to an output PostScript format file. Figure 3 shows how the output of the second command is formatted.

Listing 15. Saving output from a2ps as a PostScript file

```
[ian@attic4 ~]$ a2ps -4 -E -o fig3.ps sample2e.tex
[sample2e.tex (tex, delegated to texi2dvi): 1 page on 1 sheet]
[Total: 4 pages on 1 sheet] saved into the file `fig3.ps'
[ian@attic4 ~]$ a2ps -4 -E -o fig3.ps sample2e.txt
[sample2e.txt (plain): 4 pages on 1 sheet]
[Total: 4 pages on 1 sheet] saved into the file `fig3.ps'
```

Figure 3. Pretty printed output from a2ps



There are many other filters that you can use to format files for printing in special ways. Most have a range of options. Check the man pages for more details. Some examples are:

mpage

Formats test files for printing multiple pages on a single page.

psnup

Performs similar functions for PostScript files as mpage does for text files.

psbook

Rearranges the pages of a PostScript document for printing as a book or booklet, taking into account the number of pages per sheet and how the sheet is folded.

Section 4. Printer installation and configuration

This section covers material for topic 1.107.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Install a printer daemon
- Install and configure a print filter
- Access local and remote printers of various kinds

Printer daemons

Install a printer daemon by first installing the printer package, either CUPS, or another such as LPRng, which is usually shipped with a distribution. If you require one that is not shipped with your distribution, you may find a package prebuilt for your distribution, or you may build it yourself from source. Refer to the tutorial for Exam 101 topic 102, "[LPI exam 101 prep: Linux installation and package management](#)," if you need help with this task.

Once the printer package is installed, ensure that the printer daemon starts when your system starts. This is covered in the tutorial for Exam 102 topic 106 "[LPI exam 102 prep: Boot, initialization, shutdown, and runlevels](#)."

Use the `cups` script located in `/etc/rc.d/init.d` or `/etc/init.d`, according to your distribution with the `status` command. You can also use the `lpstat` or `lpc status` command to check whether your daemon is running. If you are using a different printer daemon, use the appropriate script for your package. An example is shown in Listing 16.

Listing 16. Checking CUPS daemon status

```
[root@attic4 ~]# /etc/init.d/cups stop  
Stopping cups: [ OK ]  
[root@attic4 ~]# /etc/init.d/cups status
```

```

cupsd is stopped
[root@attic4 ~]# lpstat -d
lpstat: Unable to connect to server
[root@attic4 ~]# /etc/init.d/cups start
Starting cups: [ OK ]

```

If you ever need to debug CUPS, you can run it in the foreground rather than as a daemon process. You can also test alternate configuration files if necessary. Run `cupsd -h` for more information, or see the man pages.

Listing 17. Running cupsd from the command line

```

[root@attic4 ~]# cupsd -h
Usage: cupsd [-c config-file] [-f] [-F] [-h] [-l] [--ppdsdat]

-c config-file      Load alternate configuration file
-f                  Run in the foreground
-F                  Run in the foreground but detach
-h                  Show this usage message
-l                  Run cupsd from launchd(8)
--ppdsdat           Just build ppds.dat

```

CUPS also maintains an access log and an error log. You can change the level of logging using the `LogLevel` statement in `/etc/cups/cupsd.conf`. By default, logs are stored in the `/var/log/cups` directory. They may be viewed from the **Administration** tab on the Web interface (<http://localhost:631>).

Print filters

So how does CUPS determine the filter to use for formatting a particular file type? CUPS uses MIME (Multipurpose Internet Mail Extensions) types to determine the appropriate conversion filter when printing a file. Note that other printing packages may use the *magic number* mechanism as used by the `file` command. See the man pages for `file` or `magic` for more details.

MIME types are used for transmitting various files as mail attachments. They consist of a type, such as `text` or `image`, and a subtype, such as `html`, `postscript` `gif`, or `jpeg`. The type and subtype are separated by a semicolon (;). Optional parameters may include information such as character set encoding, or language. CUPS uses rules from `/etc/cups/mime.types` to determine the type of a file and then uses a suitable filter chosen from those listed in `/etc/cups/conv.types` for the given MIME type. MIME types are registered with IANA, the Internet Assigned Numbers Authority. If you need a type that is not registered, prefix the subtype with 'x-'. Some image type examples are shown in Listing 18.

Listing 18. Some MIME type entries from /etc/cups/mime.types

```

image/gif          gif string(0,GIF87a) string(0,GIF89a)
image/png          png string(0,<89>PNG)
image/jpeg         jpeg jpg jpe string(0,<FFD8FF>) &&\
                  (char(3,0xe0) char(3,0xe1) char(3,0xe2) char(3,0xe3)\
                  char(3,0xe4) char(3,0xe5) char(3,0xe6) char(3,0xe7)\
                  char(3,0xe8) char(3,0xe9) char(3,0xea) char(3,0xeb)\
                  char(3,0xec) char(3,0xed) char(3,0xee) char(3,0xef))
image/tiff         tiff tif string(0,MM) string(0,II)
image/x-photocd   pcd string(2048,PCD_IPI)
image/x-portable-anymap  pnm

```

The format of the entries is beyond the scope of this tutorial. Check the files `/usr/share/mime/magic` or `/usr/share/file/magic` for some insight on how the *magic numbers* are used to identify files.

Once the MIME type of a file has been determined, the correct filter is found using the `/etc/cups/mime.convs` file. Lines in this file have four entries, a source and destination MIME type, a *cost*, and the name of the filter. The least-cost filter is used. Some examples are shown in Listing 19.

Listing 19. Filter entries from `/etc/cups/mime.convs`

```

text/plain          application/postscript  33      texttops
text/html          application/postscript  33      texttops
image/gif          application/vnd.cups-postscript  66      imagetops
image/png          application/vnd.cups-postscript  66      imagetops
image/jpeg         application/vnd.cups-postscript  66      imagetops
image/tiff         application/vnd.cups-postscript  66      imagetops
image/x-bitmap     application/vnd.cups-postscript  66      imagetops

```

If a suitable filter cannot be found, your attempt to print a file will result in an error message. If you are using a printer daemon other than CUPS, you may get unexpected output instead. Listing 20 shows how this works with a DVI file (the normal output from TeX and LaTeX).

Listing 20. Printing an unsupported file type

```

[ian@attic4 ~]$ lpr sampl.dvi
lpr: Unsupported format 'application/octet-stream'!

```

Fortunately, the `tetex` package that provides TeX and LaTeX also provides a conversion utility, `dvips` to convert from DVI to PostScript. Unfortunately, it won't work as a filter because it does not know how to handle the arguments that a CUPS filter must handle, namely, job id, user, job title, number of copies, and job options. The first filter in a filter pipeline will also have an additional parameter, the filename, if the input comes from a file.

The solution is to create a wrapper script that will be the filter. The `dvips` command does not accept input from stdin, so the script may need to create a temporary file and copy stdin to that file before calling `dvips`. A possible script is shown in Listing

21.

Listing 21. A CUPS DVI to PostScript filter script

```
#!/bin/bash
# CUPS filter to process DVI files using dvips
# Create a sandbox for working if input on stdin
if [ $# -lt 6 ]; then
    sandbox=${TMPDIR-/tmp}/cups-dvitops.$$
    (umask 077 && mkdir $sandbox) || {
        echo "Cannot create temporary directory! Exiting." 1>&2
        exit 1
    }
    fn="$sandbox/cups-dvitops.$$"
    cat > "$fn"
else
    fn="$6"
fi
# Call dvips quietly, securely and with output to stdout
dvips -R -q -o - "$fn"
# Erase sandbox if we created one
if [ $# -lt 6 ]; then
    rm -rf "$sandbox"
fi
```

Recall that CUPS uses two files in `/etc/cups` to determine the MIME type and filter to use. These files will be overwritten whenever you reinstall or upgrade CUPS. Fortunately, CUPS will read **all** files with an extension of `.types` or `.convs` whenever it starts or restarts. So you should create a pair of files for your new filter, for example `/etc/cups/dvitops.types` and `/etc/cups/dvitops.convs` as shown in Listing 22 shows a partial output listing for Docuprint drivers.

Listing 22. Configuration files for CUPS dvitops filter

```
[ian@attic4 ~]$ cat /etc/cups/dvitops.types
# Local MIME definition for DVI files
application/x-dvi dvi string(0,<F702>)
[ian@attic4 ~]$ cat /etc/cups/dvitops.convs
# Local DVI to PostScript filter for CUPS
application/x-dvi application/postscript 50 dvitops
```

This says that DVI files are identified by having the hexadecimal digits F7 and 02 in the first two positions and that such files should be processed by the `dvitops` filter.

Next, as root, copy the script above in `/usr/lib/cups/filter/dvitops` and make sure it is world readable and executable (`-rwxr-xr-x`). The name you give the script must match that in the `/etc/cups/dvitops.convs` file above. If you are running SELinux in enforcing mode, you should also run `restorecon` in the `/usr/lib/cups/filter` directory to update the security contexts. Otherwise, your `lpr` command will appear to work, but your file will not print.

Finally, use the restart option with the cups script located in `/etc/rc.d/init.d` or `/etc/init.d`, to restart CUPS and use your new filter.

If you are using an older print spooler, you will probably use either the `magicfilter` or `apsfilter` as input filters to convert various input files to PostScript format for printing to a PostScript printer or, using Ghostscript, to a non-PostScript printer.

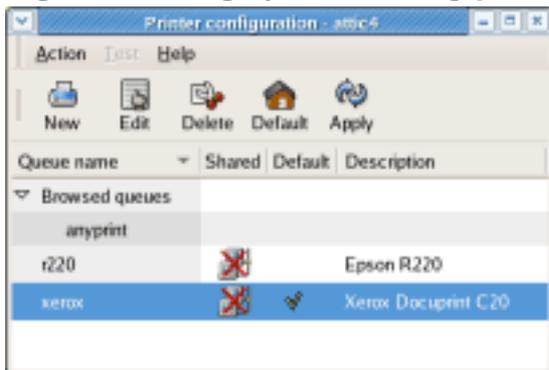
Accessing printers

CUPS supports a variety of printers, including:

- Locally attached parallel and USB printers
- IPP (Internet Printing Protocol) printers
- Remote LPD printers
- Windows® printers using SAMBA
- Novell printers using NCP
- HP JetDirect attached printers

Most systems now attempt to autodetect and autoconfigure local hardware when the system starts or when the device is attached. Similarly, many network printers can be autodetected. You can use the CUPS Web administration tool (<http://localhost:631> or <http://127.0.0.1:631>) to search for or add printers. Many distributions include their own configuration tools, for example YaST on SUSE systems. Figure 4 illustrates the system-config-printer tool on Fedora Core 5.

Figure 4. Using system-config-printer on Fedora Core 5



You can also configure printers from a command line. The rest of this tutorial shows you how. An understanding of this material will help you answer exam questions on the GUI interfaces.

Before you configure a printer, you need some basic information about the printer and about how it is connected. For illustration, we will use a Xerox Docuprint C20 attached through a D-Link print server. The print server provides LPD printing function. To configure it, we will need the IP address (192.168.0.10, in this case) and

a printer queue name on the LPD server. This is set in the print server and is PS-66D975-P1 in this case. If a remote system needs a user id or password, you will also need that information.

You will also need to know what driver to use for your printer. Check at LinuxPrinting.org (see [Resources](#) later in this tutorial for a link) to see if there is a driver for your particular printer. The `lpinfo` command can also help you identify available device types and drivers. Use the `-v` option to list supported devices and the `-m` option to list drivers, as shown in Listing 23.

Listing 23. Available printer drivers

```
lyrebird:~ # lpinfo -m | grep -i "docuprint.c"
Xerox/DocuPrint_C6-cdj550.ppd.gz Xerox DocuPrint C6 Foomatic/cdj550 (recommended)
Xerox/DocuPrint_C8-cdj550.ppd.gz Xerox DocuPrint C8 Foomatic/cdj550 (recommended)
Xerox/DocuPrint_C11-cdj500.ppd.gz Xerox DocuPrint C11 Foomatic/cdj500
Xerox/DocuPrint_C11-hpdj.ppd.gz Xerox DocuPrint C11 Foomatic/hpdj
Xerox/DocuPrint_C11-pcl3.ppd.gz Xerox DocuPrint C11 Foomatic/pcl3 (recommended)
Xerox/DocuPrint_C20-cljet5.ppd.gz Xerox DocuPrint C20 Foomatic/cljet5
Xerox/DocuPrint_C20-hpijs.ppd.gz Xerox DocuPrint C20 Foomatic/hpijs
Xerox/DocuPrint_C20-Postscript.ppd.gz Xerox DocuPrint C20 Foomatic/Postscript
(recommended)
Xerox/DocuPrint_C55-Postscript.ppd.gz Xerox DocuPrint C55 Foomatic/Postscript
(recommended)
```

Several choices are shown here for the Docuprint C20. The recommended driver is the PostScript driver, which is not surprising, since this printer supports PostScript. Again, if you can't find your printer listed, check at LinuxPrinting.org (see [Resources](#)) for the appropriate driver. Drivers come in the form of PPD (PostScript Printer Description) files.

Now that you have the basic information, you can configure a printer using the `lpadmin` command as shown in Listing 24. Note that this system did not list the specific Xerox Docuprint driver, so we use the generic PostScript driver instead.

Listing 24. Configuring a printer

```
[root@attic4 ~]# lpinfo -m | grep -i generic
textonly.ppd Generic text-only printer
postscript.ppd.gz Generic postscript printer
[root@attic4 ~]# lpadmin -p xerox1 -E -m "postscript.ppd.gz" \
> -v "lpd:192.168.0.1/PS-66D975-P1" -D "Xerox 1"
[root@attic4 ~]# lpstat -a
anyprint accepting requests since Sat 12 Aug 2006 04:07:46 PM EDT
r220 accepting requests since Tue 22 Aug 2006 11:13:40 AM EDT
xerox accepting requests since Tue 22 Aug 2006 11:13:40 AM EDT
xerox1 accepting requests since Tue 22 Aug 2006 11:17:59 AM EDT
```

If you need to remove a printer, use `lpadmin` with the `-x` option as shown in Listing 25.

Listing 25. Removing a printer

```
[root@attic4 ~]# lpadmin -x xerox1
```

You can also set various printer options using the `lpadmin` or `lpoptions` commands.

Spool files

CUPS uses the `/var/spool/cups` directory for spooling. This will usually be set up correctly when you install CUPS. If you are using the LPD daemon, spool files are stored in directories such as `/var/spool/lpd/xerox`, for our printer 'xerox'. Spool directories and files should have permissions set to protect them from being read or written by users other than the printing system.

Other input filters

If you are using LPD, LPRng, or another printing system, you will probably use either `magicfilter` or `apsfilter` for converting input files to PostScript format, and you will probably use Ghostscript as the printer driver for non-PostScript printers. Configuration for printers and filters will be in `/etc/printcap`. If you are using these, consult the man pages or online documentation such as the *Apsfilter handbook* listed in [Resources](#) later in this tutorial.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents for system administrators, especially its HOWTOs.
- [LinuxPrinting.org](#) provides information on drivers and printer support as well as a CUPS Quick Start.
- The [Apsfilter handbook](#) can help you configure printers and filters if you are using LPD, LPRng, or another printing system.
- [The Printer Working Group](#) of the IEEE Industry Standards and Technology Organization (IEEE-ISTO) develops standards that make printers -- and the applications and operating systems supporting them -- work better together.
- *CUPS: Common UNIX Printing System* (Sams, 2001) is a detailed reference for CUPS.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 107: Printing

Junior Level Administration (LPIC-1) topic 107

Skill Level: Intermediate

[Ian Shields](#)

Senior Programmer
IBM developerWorks

22 Aug 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this third in a [series of nine tutorials](#), Ian introduces you to printing in Linux®. By the end of this tutorial, you will know how to manage printers, print queues, and user print jobs on a Linux system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	(This tutorial). Learn how to manage printers, print queues, and user print jobs on a Linux system. See detailed objectives below.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Printing in Linux," the third of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to configure printers and manage print jobs in Linux.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.107.2 Manage printers and print queues	Weight 1	Configure and monitor print servers. Manage print queues and troubleshoot general printing problems.
1.107.3 Print files	Weight 1	Add and remove jobs from configured printer queues. Convert text files to PostScript for printing.
1.107.4 Printer installation and configuration	Weight 1	Install and configure local and remote printers, including printer daemons and print filters. Use local and remote printers, including PostScript, non-PostScript and Samba printers.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

At the time of writing, the published LPI objectives for this topic are largely directed toward the Common UNIX Printing System (CUPS), with vestiges of the earlier LPD (line printer daemon) and LPRng (the next generation line printer, or LPR) printing systems. Accordingly, this tutorial is directed mainly toward CUPS, with only minor mention of the earlier technologies. For more complete preparation, you should also review other material on LPD and LPRng printing technologies.

Section 2. Manage printers and print queues

This section covers material for topic 1.107.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Configure and monitor a print server
- Manage user print queues
- Troubleshoot general printing problems

Introduction

In the early days of computers, printing was done by *line printers*, which printed a line of text at a time using fixed-pitch characters and a single font. To speed up overall system performance, early mainframe computers interleaved work for slow peripherals such as card readers, card punches, and line printers with other work. Thus was born *Simultaneous Peripheral Operation On Line* or *spooling*, a term that is still commonly used when talking about computer printing.

In UNIX® and Linux systems, printing initially used the Berkeley Software Distribution (BSD) printing subsystem, consisting of a line printer daemon (`lpd`) running as a server, and client commands such as `lpr` to submit jobs for printing. This protocol was later standardized by the IETF as RFC 1179, "Line Printer Daemon Protocol".

System V UNIX also had a printing daemon. It was functionally similar to the Berkeley LPD, but had a different command set. You will frequently see two commands with different options that accomplish the same task. For example, `lpr` from the Berkeley implementation and `lp` from the System V implementation are both used to print files.

With advances in printer technology, it became possible to mix different fonts on a page and to print images as well as words. Variable pitch fonts, and more advanced printing techniques such as kerning and ligatures, all became possible. Several improvements to the basic `lpd/lpr` approach to printing were devised, such as LPRng, the next generation LPR, and CUPS, the Common UNIX Printing System.

Many printers capable of graphical printing use the Adobe PostScript language. A

PostScript printer has an engine that interprets the commands in a print job and produces finished pages from these commands. PostScript is often used as an intermediate form between an original file, such as a text or image file, and a final form suitable for a particular printer that does not have PostScript capability. Conversion of a print job, such as an ASCII text file or a JPEG image to PostScript, and conversion from PostScript to the final raster form required for a non-PostScript printer is done using *filters*.

The rest of this tutorial focuses on the Common UNIX Printing System (CUPS), which supports the traditional commands as well as newer graphical interfaces to printing functions. CUPS 1.1 is assumed; it includes several features not in earlier versions, such as digest passwords for increased security. Many distributions and desktops provide front-end graphical programs for CUPS, so the material covered here is not exhaustive. This tutorial covers the major concepts, but a particular implementation may differ. Note also that physical installation of your printer is beyond the scope of this tutorial.

Print servers

The CUPS server runs as a daemon process, `cupsd` under control of a configuration file normally located in `/etc/cups/cupsd.conf`. The `/etc/cups` directory also contains other configuration files related to CUPS. It is usually started during system initialization, but may be controlled by the `cups` script located in `/etc/rc.d/init.d` or `/etc/init.d`, according to your distribution. As with most such scripts, you can stop, start, or restart the daemon as shown in Listing 1.

Listing 1. Starting and stopping the cups daemon

```
[root@attic4 ~]# /etc/rc.d/init.d/cups
Usage: cups {start|stop|restart|condrestart|reload|status}
[root@attic4 ~]# /etc/rc.d/init.d/cups stop
Stopping cups:                                [ OK ]
[root@attic4 ~]# /etc/rc.d/init.d/cups start
Starting cups:                                [ OK ]
[root@attic4 ~]# /etc/rc.d/init.d/cups restart
Stopping cups:                                [ OK ]
Starting cups:                                [ OK ]
```

The configuration file, `/etc/cups/cupsd.conf`, contains parameters that you may set to control such things as access to the printing system, whether remote printing is allowed, the location of spool files, and so on. On some systems, a second part describes individual print queues and is usually generated automatically by configuration tools. Listing 2 shows some entries for a default `cupsd.conf` file. Note that comments start with a `#` character, so entries that were changed from default would have the leading `#` character removed. Note also that the spool files are stored by default in the `/var/spool` filesystem as you would expect from the Filesystem Hierarchy Standard (FHS).

Listing 2. Parts of a default /etc/cups/cupsd.conf

```
#
# RequestRoot: the directory where request files are stored.
# By default "/var/spool/cups".
#
#RequestRoot /var/spool/cups

#
# RemoteRoot: the name of the user assigned to unauthenticated accesses
# from remote systems. By default "remroot".
#
#RemoteRoot remroot

#
# ServerBin: the root directory for the scheduler executables.
# By default "/usr/lib64/cups".
#
#ServerBin /usr/lib64/cups

#
# ServerRoot: the root directory for the scheduler.
# By default "/etc/cups".
#
#ServerRoot /etc/cups
```

You should also be aware of the /etc/printcap file. This was the name of the configuration file for LPD print servers, and many applications still use it to determine available printers and their properties. It is usually generated automatically in a CUPS system, so you will probably not modify it yourself. However, you may need to check it if you are diagnosing user printing problems. An example is shown in Listing 3.

Listing 3. Automatically generated /etc/printcap

```
# This file was automatically generated by cupsd(8) from the
# /etc/cups/printers.conf file. All changes to this file
# will be lost.
xerox|Xerox Docuprint C20:rm=localhost.localdomain:rp=xerox:
anyprint|Pick any printer:rm=localhost.localdomain:rp=anyprint:
r220|Epson R220:rm=localhost.localdomain:rp=r220:
```

Each line here has a printer name and printer description as well as the name of the remote machine (rm) and remote printer (rp) on that machine. A traditional /etc/printcap file also describes the printer capabilities.

Finally, CUPS 1.1 introduced a passwd.md5 file. This allows CUPS users to be defined using the `lppasswd` command. The CUPS user ids do not have to be system userids.

Print queues

A *print queue* is a logical entity to which users direct print jobs. Frequently, particularly in single-user systems, a print queue is synonymous with a printer. However, CUPS allows a system without an attached printer to queue print jobs for eventual printing on a remote system, and also, through the use of *classes* to allow a print job directed at a class to be printed on the first available printer of that class. These are discussed more in the final section of this tutorial.

Several commands permit inspection and manipulation of print queues. Some of these have their roots in LPD commands, although the currently supported options are frequently a limited subset of those supported by the original LPD printing system. Other commands are new for CUPS. In general, a user can manipulate his or her own print jobs, but root or another authorized user is usually required to manipulate the jobs of others. Most CUPS commands support a `-E` option for encrypted communication between the CUPS client command and CUPS server.

You can check the queues known to the system using the CUPS `lpstat` command. Some common options are shown in Table 3.

Table 3. Options for lpstat	
Option	Purpose
-a	Display accepting status of printers.
-c	Display print classes.
-p	Display print status: enabled or disabled.
-s	Display default printer, printers, and classes. Equivalent to <code>-d -c -v</code> . Note that multiple options must be separated as values can be specified for many.
-s	Display printers and their devices.

You may also use the LPD `lpc` command, found in `/usr/sbin`, with the `status` option. If you do not specify a printer name, all queues are listed. Listing 4 shows some examples of both commands.

Listing 4. Displaying available print queues

```
[ian@attic4 ~]$ lpstat -d
system default destination: xerox
[ian@attic4 ~]$ lpstat -v xerox
device for xerox: lpd://192.168.0.10/PS-66D975-P1
[ian@attic4 ~]$ lpstat -s
system default destination: xerox
members of class anyprint:
    r220
    xerox
device for anyprint: ///dev/null
device for r220: smb://MSHOME/DEN/EPSON220
device for xerox: lpd://192.168.0.10/PS-66D975-P1
[ian@attic4 ~]$ lpstat -a r220
r220 accepting requests since Sat 12 Aug 2006 04:01:38 PM EDT
[ian@attic4 ~]$ /usr/sbin/lpc status xerox
xerox:
    printer is on device 'lpd' speed -1
    queuing is disabled
    printing is enabled
    no entries
    daemon present
```

This example shows two printers, xerox and r220, and a class, anyprint, which allows print jobs to be directed to the first available of these two printers.

In the previous example, queuing of print jobs to xerox is currently disabled, although printing is enabled, as might be done in order to drain the queue before taking the printer offline for maintenance. Whether queuing is enabled or disabled is controlled by the `accept` and `reject` commands. Whether printing is enabled or disabled is controlled by the `cupsenable` and `cupsdisable` commands. In earlier versions of CUPS, these were called `enable` and `disable`, which allowed confusion with the bash shell builtin `enable`. Listing 5 shows how to enable queuing on printer xerox while disabling printing. Note that an authorized user must perform these tasks. This may be root or another authorized user. See the `cupsd.conf` file and the man pages for the `lppasswd` command for more information on authorizing users.

Listing 5. Enabling queuing and disabling printing

```
[root@attic4 ~]# lpc status xerox
xerox:
    printer is on device 'lpd' speed -1
    queuing is enabled
    printing is enabled
    no entries
    daemon present
[root@attic4 ~]# cupsdisable xerox
[[root@attic4 ~]# lpstat -p -a
printer anyprint is idle.  enabled since Sat 12 Aug 2006 04:07:46 PM EDT
printer r220 is idle.  enabled since Sat 12 Aug 2006 04:01:38 PM EDT
printer xerox disabled since Sat 12 Aug 2006 06:43:09 PM EDT -
    Paused
anyprint accepting requests since Sat 12 Aug 2006 04:07:46 PM EDT
r220 accepting requests since Sat 12 Aug 2006 04:01:38 PM EDT
xerox accepting requests since Sat 12 Aug 2006 06:43:09 PM EDT
```

Managing print jobs on print queues

Now that you have seen a little of how to check on print queues and classes, let's look at how to manage print jobs on printer queues. The first thing you might want to do is find out whether any jobs are queued for a particular printer or for all printers. You do this with the `lpq` command. If no option is specified, `lpq` displays the queue for the default printer. Use the `-P` option with a printer name to specify a particular printer or the `-a` option to specify all printers, as shown in Listing 6.

Listing 6. Checking print queues with `lpq`

```
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)                Total Size
1st   brendan 14      RobotPlayer.java      1024 bytes
2nd   ian     16      .bashrc                1024 bytes
3rd   ian     17      .bashrc                1024 bytes
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)                Total Size
1st   brendan 14      RobotPlayer.java      1024 bytes
2nd   ian     16      .bashrc                1024 bytes
3rd   ian     17      .bashrc                1024 bytes
[ian@attic4 ~]$ lpq -P r220
r220 is ready
no entries
[ian@attic4 ~]$ lpq -a
Rank  Owner  Job      File(s)                Total Size
1st   brendan 14      RobotPlayer.java      1024 bytes
2nd   ian     16      .bashrc                1024 bytes
3rd   ian     17      .bashrc                1024 bytes
```

In this example, three jobs, 14, 16, and 17, are queued for the printer named xerox. Note that when the `-P` option is included, the output indicates that the printer is not ready. Note also that user ian submitted a job twice, a common user action when a job does not print the first time. You can avoid printing the extra copy by removing a job from the queue with the `lprm` command. The usual authorization setup allows a user to remove his or her own jobs, but not those of other users. The root user or another authorized user can remove jobs for other users. With no options, the current job is removed. With the `-` option, all jobs are removed. Otherwise, you can give a list of jobs to be removed as shown in Listing 7.

Listing 7. Deleting print jobs with `lprm`

```
[[ian@attic4 ~]$ lprm
Password for ian on localhost?
lprm: Unauthorized
[ian@attic4 ~]$ lprm 17
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)                Total Size
1st   brendan 14      RobotPlayer.java      1024 bytes
2nd   ian     16      .bashrc                1024 bytes
```

Note that user ian was not able to remove the first job on the queue, because it was for user brendan. However, he was able to remove his own job number 17.

Another command that will help you manipulate jobs on print queues is the `lp` command. You may use it to alter attributes of jobs, such as priority or number of copies. Suppose user ian wants his job to print before that of user brendan, and he really did want two copies of it. The job priority ranges from a lowest priority of 1 to a highest priority of 100 with a default of 50. User ian could use the `-i`, `-n`, and `-q` options to specify a job to alter and a new number of copies and priority as shown in Listing 8. Note the use of the `-l` option of the `lpq` command, which provides more verbose output.

Listing 8. Changing the number of copies and priority with lp

```
[ian@attic4 ~]$ lp -i 16 -n 2
[ian@attic4 ~]$ lpq -l
xerox is not ready

ian: 1st                               [job 16 localhost]
      2 copies of .bashrc                1024 bytes

brendan: 2nd                            [job 14 localhost]
      RobotPlayer.java                   1024 bytes
```

Finally, the `lpmove` command allows jobs to be moved from one queue to another. For example, we might want to do this because printer xerox is not currently printing. This command requires an authorized user. Listing 9 shows how to move these jobs to another queue, specifying first by printer and job id, then all jobs for a given printer. By the time we check the queues again, two of the jobs have already printed.

Listing 9. Moving jobs to another print queue with lpmove

```
[root@attic4 ~]# lpmove xerox-16 anyprint
[root@attic4 ~]# lpmove xerox r220
[root@attic4 ~]# lpq
xerox is not ready
no entries
[root@attic4 ~]# lpq -a
Rank  Owner  Job      File(s)      Total Size
----  -
active ian     18      fig1.gif     26624 bytes
```

If you happen to use a print server that is not CUPS, such as LPD or LPRng, you will find that many of the queue administration functions that we have just looked at are handled as subcommands of the `lpc` command. For example, you might use `lpc topq` to move a job to the top of a queue. Other `lpc` commands may include `disable`, `down`, `enable`, `hold`, `move`, `redirect`, `release`, and `start`.

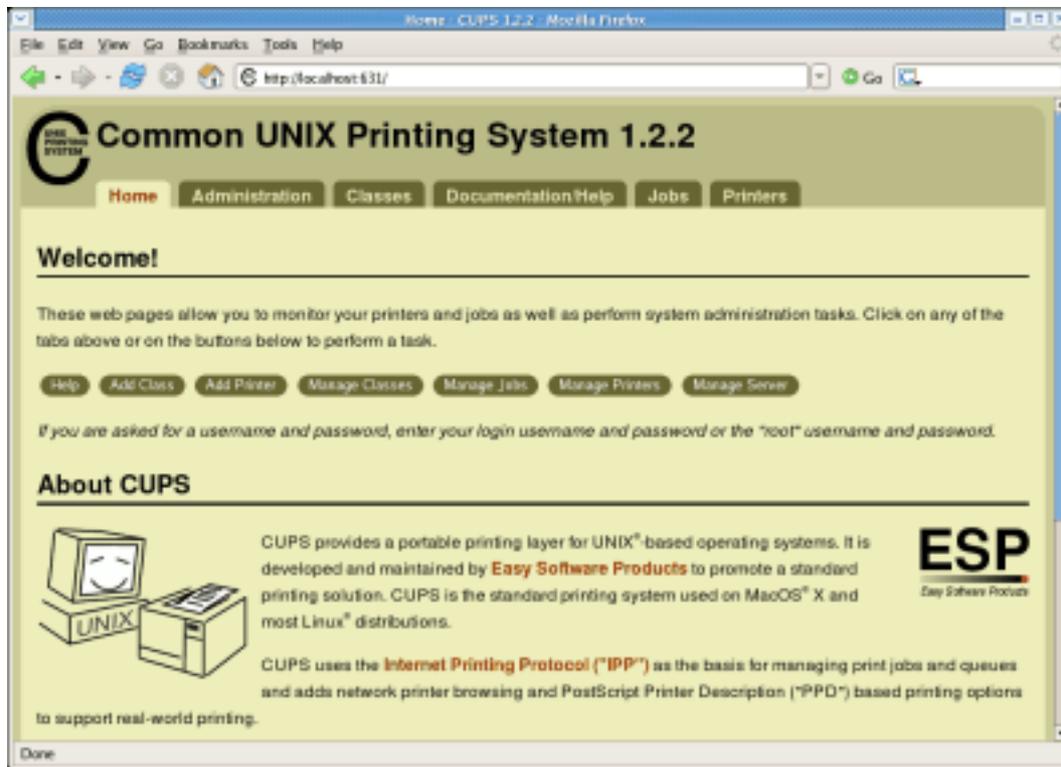
Troubleshooting

If you are having trouble printing, try these tips:

- Ensure that the cups server is running. You can use the `lpstat` command, which will report an error if it is unable to connect to the `cupsd` daemon. Alternatively, you might use the `ps -ef` command and check for `cupsd` in the output.
- If you try to queue a job for printing and get an error message indicating that the printer is not accepting jobs results, use `lpstat -a` or `lpc status` to check that the printer is accepting jobs.
- If a queued job does not print, use `lpstat -p` or `lpc status` to check that the printer is accepting jobs. You may need to move the job to another printer as discussed in the next section.
- If the printer is remote, you may need to check that it still exists on the remote system and that it is operational.
- You may need to update the configuration file to allow a particular user or remote system to print on your printer.
- You may need to ensure that your firewall allows remote printing requests, either from another system to your system, or from your system to another, as appropriate.
- You may need to verify that you have the right driver (as discussed in the final section of this tutorial).

As you can see, printing involves the correct functioning of several components of your system and possibly network. In a tutorial of this length, we can only give you starting points for diagnosis. Most CUPS systems also have a graphical interface to the command-line functions that we discuss here. Generally, this interface is accessible from the local host using a browser pointed to port 631 (<http://localhost:631> or <http://127.0.0.1:631>), as shown in Figure 1.

Figure 1. CUPS home page on port 631



Section 3. Print files

This section covers material for topic 1.107.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Add and remove jobs from configured printer queues
- Convert text files to PostScript for printing

Print

You learned how to remove files from print queues in [the previous section](#). Here you learn how to print files and change job options.

Many graphical programs provide a method of printing, usually under the **File** menu option. These programs provide graphical tools for choosing a printer, margin sizes, color or black-and-white printing, number of copies, selecting 2-up printing (which is

2 pages per sheet, often used for handouts), and so on. This section shows you the command-line tools for controlling such features, and then a graphical implementation for comparison.

The simplest way to print any file is to use the `lpr` command and provide the file name. This prints the file on the default printer. Listing 10 shows a simple example plus a more complex example. The more complex command is explained below.

Listing 10. Printing with lpr

```
[ian@attic4 ~]$ echo abc>abc.txt
[ian@attic4 ~]$ lpr abc.txt
[ian@attic4 ~]$ lpr -Pxerox -J "Ian's text file" -#2 -m -p -q -r abc.txt
[ian@attic4 ~]$ lpq -l
xerox is ready

ian: 1st                               [job 25 localhost]
      2 copies of Ian's text file       1024 bytes
[ian@attic4 ~]$ ls abc.txt
ls: abc.txt: No such file or directory
```

Table 4 explains the options used on the more complex command above, along with other options that you may use with `lpr`.

Table 4. Options for lpr	
Option	Purpose
-C, -J, or -T	Set a job name.
-P	Select a particular printer.
-#	Specify number of copies. Note this is different to the <code>-n</code> option you saw with the <code>lp</code> command.
-m	Send email upon job completion.
-l	The print file is already formatted for printing. Equivalent to <code>-o raw</code> .
-o	Set a job option.
-p	Format a text file with a shaded header. Equivalent to <code>-o prettyprint</code> .
-q	Hold (or queue) the job for later printing.
-r	Remove the file after it

```
has been spooled for
printing.
```

So, in our complex example: `lpr -Pxerox -J "Ian's text file" -#2 -m -p -q -r abc.txt`, user ian is requesting a specific printer, giving a name to the job, requesting 2 copies, requesting an email confirmation after printing, holding the job, and having the file `abc.txt` removed after it has been spooled. The subsequent commands show the held job and the fact that the file has indeed been removed.

In addition to the `lpr` command, the `lp` command covered in the previous section can also be used to print jobs, as well as modify them. Both `lp` and `lpr` also accept a file from stdin if no file name is given on the command line. In contrast to `lpr`, which quietly spools the job, the `lp` default is to display the job number of the spooled job as shown in Listing 11. Note that not all the equivalent options on `lp` and `lpr` have the same name; for example, `-n` on `lp` is equivalent to `-#` on `lpr`.

Listing 11. Printing from stdin with lp

```
[ian@attic4 ~]$ lp
abc
request id is xerox-27 (1 file(s))
```

So we now have a held job in the xerox print queue. What to do? The `lp` command has options to hold and release jobs, using various values with the `-H` option. Listing 12 shows how to release the held job. Check the man page of `lp` for information on other options.

Listing 12. Resuming printing of a held print job

```
[ian@attic4 ~]$ lp -i 25 -H resume
```

Many different printers are available today, but not all of them support the same set of options. You can find out what general options are set for a printer using the `lpoptions` command. Add the `-l` option to display printer-specific options; Listing 13 shows an example. The man page for the `lp` command also lists several common options, particularly relating to portrait/landscape printing, page dimensions, and placement of the output on the pages.

Listing 13. Checking printer options

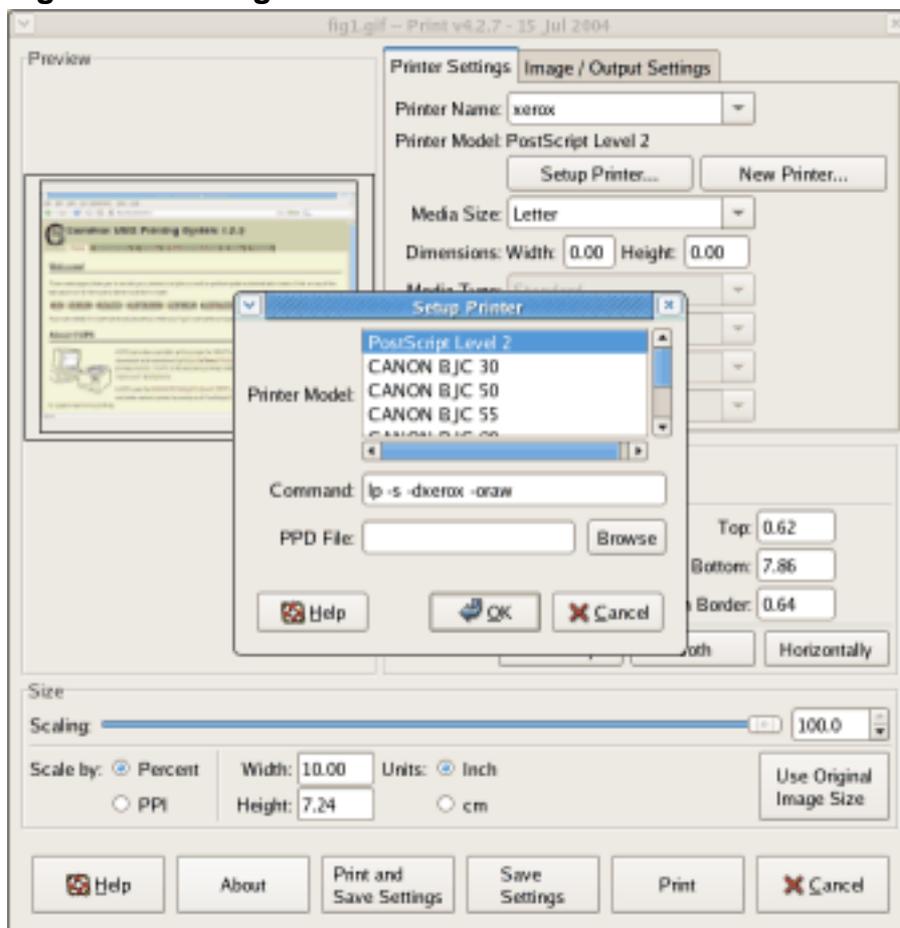
```
[ian@attic4 ~]$ lpoptions -p xerox
job-sheets=none,none printer-info='Xerox Docuprint C20' printer-is-accepting-
jobs=1 printer-is-shared=1 printer-make-and-model='Xerox DocuPrint C20 Foomat
ic/Postscript (recommended)' printer-state=3 printer-state-change-time=115550
6374 printer-state-reasons=none printer-type=143388 cpi=12 scp-fc5=true lpi=7
page-bottom=86 page-left=57 page-right=57 page-top=72 scaling=100 wrap=true
```

```
[ian@attic4 ~]$ lpoptions -l
PageSize/Page Size: *Letter A4 11x17 A3 A5 B5 Env10 EnvC5 EnvDL EnvISOB5 EnvM
onarch Executive Legal
PageRegion/PageRegion: Letter A4 11x17 A3 A5 B5 Env10 EnvC5 EnvDL EnvISOB5 En
vMonarch Executive Legal
Duplex/Double-Sided Printing: DuplexNoTumble DuplexTumble *None
Resolution/Resolution: *default 150x150dpi 300x300dpi 600x600dpi
PreFilter/GhostScript pre-filtering: EmbedFonts Level1 Level2 *No
```

So far, all our commands have been directed to the local CUPS server. You can also direct most commands to the server on another system, by specifying the `-h` option along with a port number if it is not the CUPS default of 631.

Before moving on to filters, let's look at how all this magic avails itself in a GUI application. Figure 2 shows the illustration of Figure 1 in the GIMP, an image manipulation program. Using the **File > Print** option, you have many choices about how to print the image. In this application, you can also click the **Setup Printer** button to choose a printer and see the command that will be used to print the file, in this case, `lp -s -dxerox -oraw`.

Figure 2. Printing from the GIMP



Convert files

You may have noticed that we were able to print text files above, even though the r220 printer happens to be an Epson photo printer, while the xerox printer is a PostScript Xerox Docuprint C20. This magic feat is accomplished through the use of *filters*. Indeed, a popular filter for many years was named *magicfilter*.

The number of filters included with most CUPS packages allows almost any kind of file to be printed. Additional filters are available commercially, from companies including Easy Software Products, the developers of CUPS.

CUPS uses MIME (Multipurpose Internet Mail Extensions) types to determine the appropriate conversion filter when printing a file. The section on [filter installation](#), later in this tutorial, goes into detail. Other printing packages may use the *magic number* mechanism as used by the `file` command. See the man pages for `file` or `magic` for more details.

The general print flow is to convert the input file to a PostScript format using the appropriate filter for the file type, such as `texttops`, `imagetops`, or `pdftops`. The PostScript format is then filtered through a `pstoraster` filter to create an intermediate raster format for non-PostScript printers before being filtered through a printer backend, which prepares it for printing on a particular printer. Ghostscript is a popular program that can print PostScript files on many different printers. A companion viewer allows display of the file on a monitor. Many printer backends are derived from Ghostscript printer drivers.

Before all of this was handled so automatically, it was necessary to convert input to PostScript format. Images could be handled with a program such as the GIMP that we saw earlier. ASCII text files were usually converted to PostScript using the `a2ps` command. The default for plain text files is to print 2-up with a header and direct output to the default printer as shown in Listing 14.

Listing 14. Printing text files with `a2ps`

```
[ian@attic4 ~]$ a2ps -4 abc.txt -o abc.ps
[abc.txt (plain): 1 page on 1 sheet]
[Total: 1 page on 1 sheet] saved into the file `abc.ps'
```

The `a2ps` command can handle a wide range of text file types and make intelligent decisions about the best way to format them. For example, the default for LaTeX files is to first format the file and then print 2-up. Listing 15 uses `a2ps` to print a copy of the `sample2e.tex` file that is distributed with LaTeX, and then shows a copy renamed to `sample2e.txt`, and printed 4-up with headers. Both are saved to an output PostScript format file. Figure 3 shows how the output of the second command is formatted.

Listing 15. Saving output from a2ps as a PostScript file

```
[ian@attic4 ~]$ a2ps -4 -E -o fig3.ps sample2e.tex
[sample2e.tex (tex, delegated to texi2dvi): 1 page on 1 sheet]
[Total: 4 pages on 1 sheet] saved into the file `fig3.ps'
[ian@attic4 ~]$ a2ps -4 -E -o fig3.ps sample2e.txt
[sample2e.txt (plain): 4 pages on 1 sheet]
[Total: 4 pages on 1 sheet] saved into the file `fig3.ps'
```

Figure 3. Pretty printed output from a2ps



There are many other filters that you can use to format files for printing in special ways. Most have a range of options. Check the man pages for more details. Some examples are:

mpage

Formats test files for printing multiple pages on a single page.

psnup

Performs similar functions for PostScript files as mpage does for text files.

psbook

Rearranges the pages of a PostScript document for printing as a book or booklet, taking into account the number of pages per sheet and how the sheet is folded.

Section 4. Printer installation and configuration

This section covers material for topic 1.107.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Install a printer daemon
- Install and configure a print filter
- Access local and remote printers of various kinds

Printer daemons

Install a printer daemon by first installing the printer package, either CUPS, or another such as LPRng, which is usually shipped with a distribution. If you require one that is not shipped with your distribution, you may find a package prebuilt for your distribution, or you may build it yourself from source. Refer to the tutorial for Exam 101 topic 102, "[LPI exam 101 prep: Linux installation and package management](#)," if you need help with this task.

Once the printer package is installed, ensure that the printer daemon starts when your system starts. This is covered in the tutorial for Exam 102 topic 106 "[LPI exam 102 prep: Boot, initialization, shutdown, and runlevels](#)."

Use the `cups` script located in `/etc/rc.d/init.d` or `/etc/init.d`, according to your distribution with the `status` command. You can also use the `lpstat` or `lpc status` command to check whether your daemon is running. If you are using a different printer daemon, use the appropriate script for your package. An example is shown in Listing 16.

Listing 16. Checking CUPS daemon status

```
[root@attic4 ~]# /etc/init.d/cups stop
Stopping cups:                                     [ OK ]
[root@attic4 ~]# /etc/init.d/cups status
```

```

cupsd is stopped
[root@attic4 ~]# lpstat -d
lpstat: Unable to connect to server
[root@attic4 ~]# /etc/init.d/cups start
Starting cups: [ OK ]

```

If you ever need to debug CUPS, you can run it in the foreground rather than as a daemon process. You can also test alternate configuration files if necessary. Run `cupsd -h` for more information, or see the man pages.

Listing 17. Running cupsd from the command line

```

[root@attic4 ~]# cupsd -h
Usage: cupsd [-c config-file] [-f] [-F] [-h] [-l] [--ppdsdat]

-c config-file      Load alternate configuration file
-f                 Run in the foreground
-F                 Run in the foreground but detach
-h                 Show this usage message
-l                 Run cupsd from launchd(8)
--ppdsdat          Just build ppds.dat

```

CUPS also maintains an access log and an error log. You can change the level of logging using the `LogLevel` statement in `/etc/cups/cupsd.conf`. By default, logs are stored in the `/var/log/cups` directory. They may be viewed from the **Administration** tab on the Web interface (<http://localhost:631>).

Print filters

So how does CUPS determine the filter to use for formatting a particular file type? CUPS uses MIME (Multipurpose Internet Mail Extensions) types to determine the appropriate conversion filter when printing a file. Note that other printing packages may use the *magic number* mechanism as used by the `file` command. See the man pages for `file` or `magic` for more details.

MIME types are used for transmitting various files as mail attachments. They consist of a type, such as `text` or `image`, and a subtype, such as `html`, `postscript` `gif`, or `jpeg`. The type and subtype are separated by a semicolon (;). Optional parameters may include information such as character set encoding, or language. CUPS uses rules from `/etc/cups/mime.types` to determine the type of a file and then uses a suitable filter chosen from those listed in `/etc/cups/conv.types` for the given MIME type. MIME types are registered with IANA, the Internet Assigned Numbers Authority. If you need a type that is not registered, prefix the subtype with 'x-'. Some image type examples are shown in Listing 18.

Listing 18. Some MIME type entries from /etc/cups/mime.types

```

image/gif          gif string(0,GIF87a) string(0,GIF89a)
image/png          png string(0,<89>PNG)
image/jpeg         jpeg jpg jpe string(0,<FFD8FF>) &&\
                  (char(3,0xe0) char(3,0xe1) char(3,0xe2) char(3,0xe3)\
                  char(3,0xe4) char(3,0xe5) char(3,0xe6) char(3,0xe7)\
                  char(3,0xe8) char(3,0xe9) char(3,0xea) char(3,0xeb)\
                  char(3,0xec) char(3,0xed) char(3,0xee) char(3,0xef))
image/tiff         tiff tif string(0,MM) string(0,II)
image/x-photocd    pcd string(2048,PCD_IPI)
image/x-portable-anymap  pnm

```

The format of the entries is beyond the scope of this tutorial. Check the files `/usr/share/mime/magic` or `/usr/share/file/magic` for some insight on how the *magic numbers* are used to identify files.

Once the MIME type of a file has been determined, the correct filter is found using the `/etc/cups/mime.convs` file. Lines in this file have four entries, a source and destination MIME type, a *cost*, and the name of the filter. The least-cost filter is used. Some examples are shown in Listing 19.

Listing 19. Filter entries from `/etc/cups/mime.convs`

```

text/plain          application/postscript  33      texttops
text/html           application/postscript  33      texttops
image/gif           application/vnd.cups-postscript  66      imagetops
image/png           application/vnd.cups-postscript  66      imagetops
image/jpeg          application/vnd.cups-postscript  66      imagetops
image/tiff          application/vnd.cups-postscript  66      imagetops
image/x-bitmap      application/vnd.cups-postscript  66      imagetops

```

If a suitable filter cannot be found, your attempt to print a file will result in an error message. If you are using a printer daemon other than CUPS, you may get unexpected output instead. Listing 20 shows how this works with a DVI file (the normal output from TeX and LaTeX).

Listing 20. Printing an unsupported file type

```

[ian@attic4 ~]$ lpr sampl.dvi
lpr: Unsupported format 'application/octet-stream'!

```

Fortunately, the `tetex` package that provides TeX and LaTeX also provides a conversion utility, `dvips` to convert from DVI to PostScript. Unfortunately, it won't work as a filter because it does not know how to handle the arguments that a CUPS filter must handle, namely, job id, user, job title, number of copies, and job options. The first filter in a filter pipeline will also have an additional parameter, the filename, if the input comes from a file.

The solution is to create a wrapper script that will be the filter. The `dvips` command does not accept input from stdin, so the script may need to create a temporary file and copy stdin to that file before calling `dvips`. A possible script is shown in Listing

21.

Listing 21. A CUPS DVI to PostScript filter script

```
#!/bin/bash
# CUPS filter to process DVI files using dvips
# Create a sandbox for working if input on stdin
if [ $# -lt 6 ]; then
    sandbox=${TMPDIR-/tmp}/cups-dvitops.$$
    (umask 077 && mkdir $sandbox) || {
        echo "Cannot create temporary directory! Exiting." 1>&2
        exit 1
    }
    fn="$sandbox/cups-dvitops.$$"
    cat > "$fn"
else
    fn="$6"
fi
# Call dvips quietly, securely and with output to stdout
dvips -R -q -o - "$fn"
# Erase sandbox if we created one
if [ $# -lt 6 ]; then
    rm -rf "$sandbox"
fi
```

Recall that CUPS uses two files in `/etc/cups` to determine the MIME type and filter to use. These files will be overwritten whenever you reinstall or upgrade CUPS. Fortunately, CUPS will read **all** files with an extension of `.types` or `.convs` whenever it starts or restarts. So you should create a pair of files for your new filter, for example `/etc/cups/dvitops.types` and `/etc/cups/dvitops.convs` as shown in Listing 22 shows a partial output listing for Docuprint drivers.

Listing 22. Configuration files for CUPS dvitops filter

```
[ian@attic4 ~]$ cat /etc/cups/dvitops.types
# Local MIME definition for DVI files
application/x-dvi dvi string(0,<F702>)
[ian@attic4 ~]$ cat /etc/cups/dvitops.convs
# Local DVI to PostScript filter for CUPS
application/x-dvi application/postscript 50 dvitops
```

This says that DVI files are identified by having the hexadecimal digits F7 and 02 in the first two positions and that such files should be processed by the `dvitops` filter.

Next, as root, copy the script above in `/usr/lib/cups/filter/dvitops` and make sure it is world readable and executable (`-rwxr-xr-x`). The name you give the script must match that in the `/etc/cups/dvitops.convs` file above. If you are running SELinux in enforcing mode, you should also run `restorecon` in the `/usr/lib/cups/filter` directory to update the security contexts. Otherwise, your `lpr` command will appear to work, but your file will not print.

Finally, use the restart option with the cups script located in `/etc/rc.d/init.d` or `/etc/init.d`, to restart CUPS and use your new filter.

If you are using an older print spooler, you will probably use either the `magicfilter` or `apsfilter` as input filters to convert various input files to PostScript format for printing to a PostScript printer or, using Ghostscript, to a non-PostScript printer.

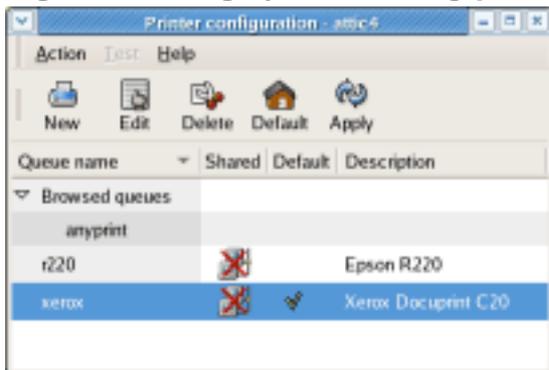
Accessing printers

CUPS supports a variety of printers, including:

- Locally attached parallel and USB printers
- IPP (Internet Printing Protocol) printers
- Remote LPD printers
- Windows® printers using SAMBA
- Novell printers using NCP
- HP JetDirect attached printers

Most systems now attempt to autodetect and autoconfigure local hardware when the system starts or when the device is attached. Similarly, many network printers can be autodetected. You can use the CUPS Web administration tool (<http://localhost:631> or <http://127.0.0.1:631>) to search for or add printers. Many distributions include their own configuration tools, for example YaST on SUSE systems. Figure 4 illustrates the system-config-printer tool on Fedora Core 5.

Figure 4. Using system-config-printer on Fedora Core 5



You can also configure printers from a command line. The rest of this tutorial shows you how. An understanding of this material will help you answer exam questions on the GUI interfaces.

Before you configure a printer, you need some basic information about the printer and about how it is connected. For illustration, we will use a Xerox Docuprint C20 attached through a D-Link print server. The print server provides LPD printing function. To configure it, we will need the IP address (192.168.0.10, in this case) and

a printer queue name on the LPD server. This is set in the print server and is PS-66D975-P1 in this case. If a remote system needs a user id or password, you will also need that information.

You will also need to know what driver to use for your printer. Check at LinuxPrinting.org (see [Resources](#) later in this tutorial for a link) to see if there is a driver for your particular printer. The `lpinfo` command can also help you identify available device types and drivers. Use the `-v` option to list supported devices and the `-m` option to list drivers, as shown in Listing 23.

Listing 23. Available printer drivers

```
lyrebird:~ # lpinfo -m | grep -i "docuprint.c"
Xerox/DocuPrint_C6-cdj550.ppd.gz Xerox DocuPrint C6 Foomatic/cdj550 (recommended)
Xerox/DocuPrint_C8-cdj550.ppd.gz Xerox DocuPrint C8 Foomatic/cdj550 (recommended)
Xerox/DocuPrint_C11-cdj500.ppd.gz Xerox DocuPrint C11 Foomatic/cdj500
Xerox/DocuPrint_C11-hpdj.ppd.gz Xerox DocuPrint C11 Foomatic/hpdj
Xerox/DocuPrint_C11-pcl3.ppd.gz Xerox DocuPrint C11 Foomatic/pcl3 (recommended)
Xerox/DocuPrint_C20-cljet5.ppd.gz Xerox DocuPrint C20 Foomatic/cljet5
Xerox/DocuPrint_C20-hpijs.ppd.gz Xerox DocuPrint C20 Foomatic/hpijs
Xerox/DocuPrint_C20-Postscript.ppd.gz Xerox DocuPrint C20 Foomatic/Postscript
(recommended)
Xerox/DocuPrint_C55-Postscript.ppd.gz Xerox DocuPrint C55 Foomatic/Postscript
(recommended)
```

Several choices are shown here for the Docuprint C20. The recommended driver is the PostScript driver, which is not surprising, since this printer supports PostScript. Again, if you can't find your printer listed, check at LinuxPrinting.org (see [Resources](#)) for the appropriate driver. Drivers come in the form of PPD (PostScript Printer Description) files.

Now that you have the basic information, you can configure a printer using the `lpadmin` command as shown in Listing 24. Note that this system did not list the specific Xerox Docuprint driver, so we use the generic PostScript driver instead.

Listing 24. Configuring a printer

```
[root@attic4 ~]# lpinfo -m | grep -i generic
textonly.ppd Generic text-only printer
postscript.ppd.gz Generic postscript printer
[root@attic4 ~]# lpadmin -p xerox1 -E -m "postscript.ppd.gz" \
> -v "lpd:192.168.0.1/PS-66D975-P1" -D "Xerox 1"
[root@attic4 ~]# lpstat -a
anyprint accepting requests since Sat 12 Aug 2006 04:07:46 PM EDT
r220 accepting requests since Tue 22 Aug 2006 11:13:40 AM EDT
xerox accepting requests since Tue 22 Aug 2006 11:13:40 AM EDT
xerox1 accepting requests since Tue 22 Aug 2006 11:17:59 AM EDT
```

If you need to remove a printer, use `lpadmin` with the `-x` option as shown in Listing 25.

Listing 25. Removing a printer

```
[root@attic4 ~]# lpadmin -x xerox1
```

You can also set various printer options using the `lpadmin` or `lpoptions` commands.

Spool files

CUPS uses the `/var/spool/cups` directory for spooling. This will usually be set up correctly when you install CUPS. If you are using the LPD daemon, spool files are stored in directories such as `/var/spool/lpd/xerox`, for our printer 'xerox'. Spool directories and files should have permissions set to protect them from being read or written by users other than the printing system.

Other input filters

If you are using LPD, LPRng, or another printing system, you will probably use either `magicfilter` or `apsfilter` for converting input files to PostScript format, and you will probably use Ghostscript as the printer driver for non-PostScript printers. Configuration for printers and filters will be in `/etc/printcap`. If you are using these, consult the man pages or online documentation such as the *Apsfilter handbook* listed in [Resources](#) later in this tutorial.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents for system administrators, especially its HOWTOs.
- [LinuxPrinting.org](#) provides information on drivers and printer support as well as a CUPS Quick Start.
- The [Apsfilter handbook](#) can help you configure printers and filters if you are using LPD, LPRng, or another printing system.
- [The Printer Working Group](#) of the IEEE Industry Standards and Technology Organization (IEEE-ISTO) develops standards that make printers -- and the applications and operating systems supporting them -- work better together.
- *CUPS: Common UNIX Printing System* (Sams, 2001) is a detailed reference for CUPS.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 107: Printing

Junior Level Administration (LPIC-1) topic 107

Skill Level: Intermediate

[Ian Shields](#)

Senior Programmer
IBM developerWorks

22 Aug 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this third in a [series of nine tutorials](#), Ian introduces you to printing in Linux®. By the end of this tutorial, you will know how to manage printers, print queues, and user print jobs on a Linux system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics

LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	(This tutorial). Learn how to manage printers, print queues, and user print jobs on a Linux system. See detailed objectives below.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Printing in Linux," the third of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to configure printers and manage print jobs in Linux.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.107.2 Manage printers and print queues	Weight 1	Configure and monitor print servers. Manage print queues and troubleshoot general printing problems.
1.107.3 Print files	Weight 1	Add and remove jobs from configured printer queues. Convert text files to PostScript for printing.
1.107.4 Printer installation and configuration	Weight 1	Install and configure local and remote printers, including printer daemons and print filters. Use local and remote printers, including PostScript, non-PostScript and Samba printers.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

At the time of writing, the published LPI objectives for this topic are largely directed toward the Common UNIX Printing System (CUPS), with vestiges of the earlier LPD (line printer daemon) and LPRng (the next generation line printer, or LPR) printing systems. Accordingly, this tutorial is directed mainly toward CUPS, with only minor mention of the earlier technologies. For more complete preparation, you should also review other material on LPD and LPRng printing technologies.

Section 2. Manage printers and print queues

This section covers material for topic 1.107.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Configure and monitor a print server
- Manage user print queues
- Troubleshoot general printing problems

Introduction

In the early days of computers, printing was done by *line printers*, which printed a line of text at a time using fixed-pitch characters and a single font. To speed up overall system performance, early mainframe computers interleaved work for slow peripherals such as card readers, card punches, and line printers with other work. Thus was born *Simultaneous Peripheral Operation On Line* or *spooling*, a term that is still commonly used when talking about computer printing.

In UNIX® and Linux systems, printing initially used the Berkeley Software Distribution (BSD) printing subsystem, consisting of a line printer daemon (lpd) running as a server, and client commands such as `lpr` to submit jobs for printing. This protocol was later standardized by the IETF as RFC 1179, "Line Printer Daemon Protocol".

System V UNIX also had a printing daemon. It was functionally similar to the Berkeley LPD, but had a different command set. You will frequently see two commands with different options that accomplish the same task. For example, `lpr` from the Berkeley implementation and `lp` from the System V implementation are both use to print files.

With advances in printer technology, it became possible to mix different fonts on a page and to print images as well as words. Variable pitch fonts, and more advanced printing techniques such as kerning and ligatures, all became possible. Several improvements to the basic lpd/lpr approach to printing were devised, such as LPRng, the next generation LPR, and CUPS, the Common UNIX Printing System.

Many printers capable of graphical printing use the Adobe PostScript language. A

PostScript printer has an engine that interprets the commands in a print job and produces finished pages from these commands. PostScript is often used as an intermediate form between an original file, such as a text or image file, and a final form suitable for a particular printer that does not have PostScript capability. Conversion of a print job, such as an ASCII text file or a JPEG image to PostScript, and conversion from PostScript to the final raster form required for a non-PostScript printer is done using *filters*.

The rest of this tutorial focuses on the Common UNIX Printing System (CUPS), which supports the traditional commands as well as newer graphical interfaces to printing functions. CUPS 1.1 is assumed; it includes several features not in earlier versions, such as digest passwords for increased security. Many distributions and desktops provide front-end graphical programs for CUPS, so the material covered here is not exhaustive. This tutorial covers the major concepts, but a particular implementation may differ. Note also that physical installation of your printer is beyond the scope of this tutorial.

Print servers

The CUPS server runs as a daemon process, `cupsd` under control of a configuration file normally located in `/etc/cups/cupsd.conf`. The `/etc/cups` directory also contains other configuration files related to CUPS. It is usually started during system initialization, but may be controlled by the `cups` script located in `/etc/rc.d/init.d` or `/etc/init.d`, according to your distribution. As with most such scripts, you can stop, start, or restart the daemon as shown in Listing 1.

Listing 1. Starting and stopping the cups daemon

```
[root@attic4 ~]# /etc/rc.d/init.d/cups
Usage: cups {start|stop|restart|condrestart|reload|status}
[root@attic4 ~]# /etc/rc.d/init.d/cups stop
Stopping cups:                                [ OK ]
[root@attic4 ~]# /etc/rc.d/init.d/cups start
Starting cups:                                [ OK ]
[root@attic4 ~]# /etc/rc.d/init.d/cups restart
Stopping cups:                                [ OK ]
Starting cups:                                [ OK ]
```

The configuration file, `/etc/cups/cupsd.conf`, contains parameters that you may set to control such things as access to the printing system, whether remote printing is allowed, the location of spool files, and so on. On some systems, a second part describes individual print queues and is usually generated automatically by configuration tools. Listing 2 shows some entries for a default `cupsd.conf` file. Note that comments start with a `#` character, so entries that were changed from default would have the leading `#` character removed. Note also that the spool files are stored by default in the `/var/spool` filesystem as you would expect from the Filesystem Hierarchy Standard (FHS).

Listing 2. Parts of a default /etc/cups/cupsd.conf

```
#
# RequestRoot: the directory where request files are stored.
# By default "/var/spool/cups".
#
#RequestRoot /var/spool/cups

#
# RemoteRoot: the name of the user assigned to unauthenticated accesses
# from remote systems. By default "remroot".
#
#RemoteRoot remroot

#
# ServerBin: the root directory for the scheduler executables.
# By default "/usr/lib64/cups".
#
#ServerBin /usr/lib64/cups

#
# ServerRoot: the root directory for the scheduler.
# By default "/etc/cups".
#
#ServerRoot /etc/cups
```

You should also be aware of the /etc/printcap file. This was the name of the configuration file for LPD print servers, and many applications still use it to determine available printers and their properties. It is usually generated automatically in a CUPS system, so you will probably not modify it yourself. However, you may need to check it if you are diagnosing user printing problems. An example is shown in Listing 3.

Listing 3. Automatically generated /etc/printcap

```
# This file was automatically generated by cupsd(8) from the
# /etc/cups/printers.conf file. All changes to this file
# will be lost.
xerox|Xerox Docuprint C20:rm=localhost.localdomain:rp=xerox:
anyprint|Pick any printer:rm=localhost.localdomain:rp=anyprint:
r220|Epson R220:rm=localhost.localdomain:rp=r220:
```

Each line here has a printer name and printer description as well as the name of the remote machine (rm) and remote printer (rp) on that machine. A traditional /etc/printcap file also describes the printer capabilities.

Finally, CUPS 1.1 introduced a passwd.md5 file. This allows CUPS users to be defined using the `lppasswd` command. The CUPS user ids do not have to be system userids.

Print queues

A *print queue* is a logical entity to which users direct print jobs. Frequently, particularly in single-user systems, a print queue is synonymous with a printer. However, CUPS allows a system without an attached printer to queue print jobs for eventual printing on a remote system, and also, through the use of *classes* to allow a print job directed at a class to be printed on the first available printer of that class. These are discussed more in the final section of this tutorial.

Several commands permit inspection and manipulation of print queues. Some of these have their roots in LPD commands, although the currently supported options are frequently a limited subset of those supported by the original LPD printing system. Other commands are new for CUPS. In general, a user can manipulate his or her own print jobs, but root or another authorized user is usually required to manipulate the jobs of others. Most CUPS commands support a `-E` option for encrypted communication between the CUPS client command and CUPS server.

You can check the queues known to the system using the CUPS `lpstat` command. Some common options are shown in Table 3.

Table 3. Options for lpstat	
Option	Purpose
-a	Display accepting status of printers.
-c	Display print classes.
-p	Display print status: enabled or disabled.
-s	Display default printer, printers, and classes. Equivalent to <code>-d -c -v</code> . Note that multiple options must be separated as values can be specified for many.
-s	Display printers and their devices.

You may also use the LPD `lpc` command, found in `/usr/sbin`, with the `status` option. If you do not specify a printer name, all queues are listed. Listing 4 shows some examples of both commands.

Listing 4. Displaying available print queues

```
[ian@attic4 ~]$ lpstat -d
system default destination: xerox
[ian@attic4 ~]$ lpstat -v xerox
device for xerox: lpd://192.168.0.10/PS-66D975-P1
[ian@attic4 ~]$ lpstat -s
system default destination: xerox
members of class anyprint:
    r220
    xerox
device for anyprint: ///dev/null
device for r220: smb://MSHOME/DEN/EPSON220
device for xerox: lpd://192.168.0.10/PS-66D975-P1
[ian@attic4 ~]$ lpstat -a r220
r220 accepting requests since Sat 12 Aug 2006 04:01:38 PM EDT
[ian@attic4 ~]$ /usr/sbin/lpc status xerox
xerox:
    printer is on device 'lpd' speed -1
    queuing is disabled
    printing is enabled
    no entries
    daemon present
```

This example shows two printers, xerox and r220, and a class, anyprint, which allows print jobs to be directed to the first available of these two printers.

In the previous example, queuing of print jobs to xerox is currently disabled, although printing is enabled, as might be done in order to drain the queue before taking the printer offline for maintenance. Whether queuing is enabled or disabled is controlled by the `accept` and `reject` commands. Whether printing is enabled or disabled is controlled by the `cupsenable` and `cupsdisable` commands. In earlier versions of CUPS, these were called `enable` and `disable`, which allowed confusion with the bash shell builtin `enable`. Listing 5 shows how to enable queuing on printer xerox while disabling printing. Note that an authorized user must perform these tasks. This may be root or another authorized user. See the `cupsd.conf` file and the man pages for the `lppasswd` command for more information on authorizing users.

Listing 5. Enabling queuing and disabling printing

```
[root@attic4 ~]# lpc status xerox
xerox:
    printer is on device 'lpd' speed -1
    queuing is enabled
    printing is enabled
    no entries
    daemon present
[root@attic4 ~]# cupsdisable xerox
[[root@attic4 ~]# lpstat -p -a
printer anyprint is idle.  enabled since Sat 12 Aug 2006 04:07:46 PM EDT
printer r220 is idle.  enabled since Sat 12 Aug 2006 04:01:38 PM EDT
printer xerox disabled since Sat 12 Aug 2006 06:43:09 PM EDT -
    Paused
anyprint accepting requests since Sat 12 Aug 2006 04:07:46 PM EDT
r220 accepting requests since Sat 12 Aug 2006 04:01:38 PM EDT
xerox accepting requests since Sat 12 Aug 2006 06:43:09 PM EDT
```

Managing print jobs on print queues

Now that you have seen a little of how to check on print queues and classes, let's look at how to manage print jobs on printer queues. The first thing you might want to do is find out whether any jobs are queued for a particular printer or for all printers. You do this with the `lpq` command. If no option is specified, `lpq` displays the queue for the default printer. Use the `-P` option with a printer name to specify a particular printer or the `-a` option to specify all printers, as shown in Listing 6.

Listing 6. Checking print queues with `lpq`

```
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)          Total Size
1st   brendan 14      RobotPlayer.java 1024 bytes
2nd   ian     16      .bashrc          1024 bytes
3rd   ian     17      .bashrc          1024 bytes
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)          Total Size
1st   brendan 14      RobotPlayer.java 1024 bytes
2nd   ian     16      .bashrc          1024 bytes
3rd   ian     17      .bashrc          1024 bytes
[ian@attic4 ~]$ lpq -P r220
r220 is ready
no entries
[ian@attic4 ~]$ lpq -a
Rank  Owner  Job      File(s)          Total Size
1st   brendan 14      RobotPlayer.java 1024 bytes
2nd   ian     16      .bashrc          1024 bytes
3rd   ian     17      .bashrc          1024 bytes
```

In this example, three jobs, 14, 16, and 17, are queued for the printer named xerox. Note that when the `-P` option is included, the output indicates that the printer is not ready. Note also that user ian submitted a job twice, a common user action when a job does not print the first time. You can avoid printing the extra copy by removing a job from the queue with the `lprm` command. The usual authorization setup allows a user to remove his or her own jobs, but not those of other users. The root user or another authorized user can remove jobs for other users. With no options, the current job is removed. With the `-` option, all jobs are removed. Otherwise, you can give a list of jobs to be removed as shown in Listing 7.

Listing 7. Deleting print jobs with `lprm`

```
[[ian@attic4 ~]$ lprm
Password for ian on localhost?
lprm: Unauthorized
[ian@attic4 ~]$ lprm 17
[ian@attic4 ~]$ lpq
xerox is not ready
Rank  Owner  Job      File(s)          Total Size
1st   brendan 14      RobotPlayer.java 1024 bytes
2nd   ian     16      .bashrc          1024 bytes
```

Note that user ian was not able to remove the first job on the queue, because it was for user brendan. However, he was able to remove his own job number 17.

Another command that will help you manipulate jobs on print queues is the `lp` command. You may use it to alter attributes of jobs, such as priority or number of copies. Suppose user ian wants his job to print before that of user brendan, and he really did want two copies of it. The job priority ranges from a lowest priority of 1 to a highest priority of 100 with a default of 50. User ian could use the `-i`, `-n`, and `-q` options to specify a job to alter and a new number of copies and priority as shown in Listing 8. Note the use of the `-l` option of the `lpq` command, which provides more verbose output.

Listing 8. Changing the number of copies and priority with lp

```
[ian@attic4 ~]$ lp -i 16 -n 2
[ian@attic4 ~]$ lpq -l
xerox is not ready

ian: 1st                               [job 16 localhost]
      2 copies of .bashrc              1024 bytes

brendan: 2nd                           [job 14 localhost]
      RobotPlayer.java                 1024 bytes
```

Finally, the `lpmove` command allows jobs to be moved from one queue to another. For example, we might want to do this because printer xerox is not currently printing. This command requires an authorized user. Listing 9 shows how to move these jobs to another queue, specifying first by printer and job id, then all jobs for a given printer. By the time we check the queues again, two of the jobs have already printed.

Listing 9. Moving jobs to another print queue with lpmove

```
[root@attic4 ~]# lpmove xerox-16 anyprint
[root@attic4 ~]# lpmove xerox r220
[root@attic4 ~]# lpq
xerox is not ready
no entries
[root@attic4 ~]# lpq -a
Rank  Owner  Job      File(s)      Total Size
----  -
active ian     18      fig1.gif     26624 bytes
```

If you happen to use a print server that is not CUPS, such as LPD or LPRng, you will find that many of the queue administration functions that we have just looked at are handled as subcommands of the `lpc` command. For example, you might use `lpc topq` to move a job to the top of a queue. Other `lpc` commands may include `disable`, `down`, `enable`, `hold`, `move`, `redirect`, `release`, and `start`.

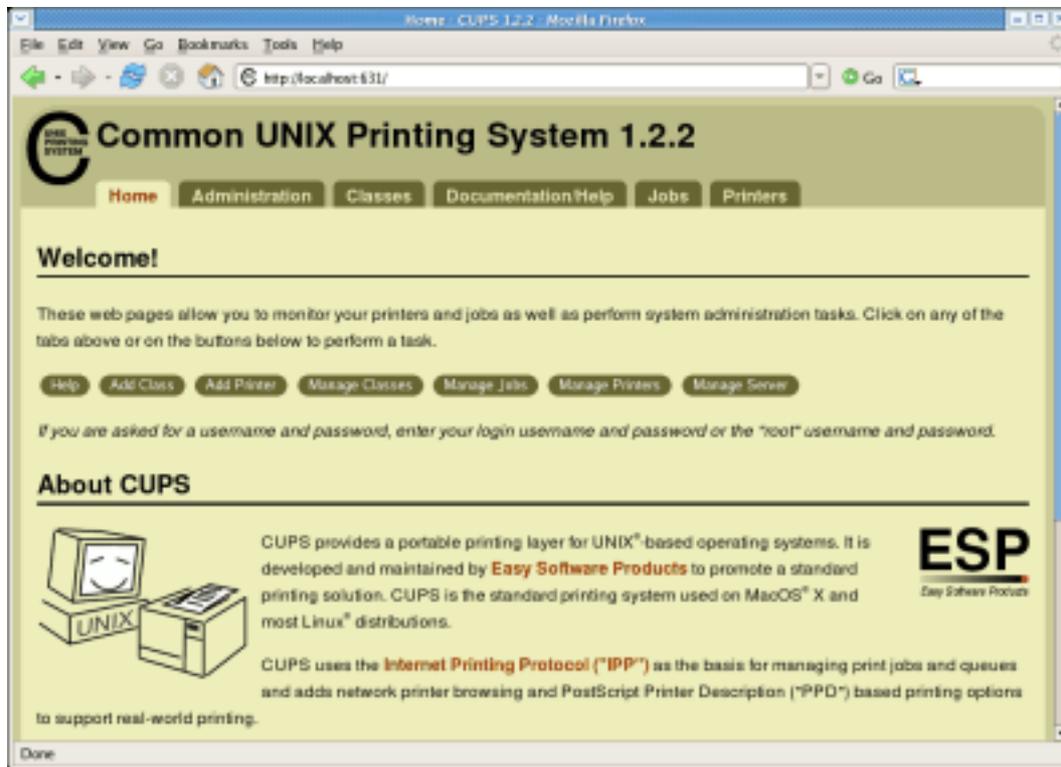
Troubleshooting

If you are having trouble printing, try these tips:

- Ensure that the cups server is running. You can use the `lpstat` command, which will report an error if it is unable to connect to the `cupsd` daemon. Alternatively, you might use the `ps -ef` command and check for `cupsd` in the output.
- If you try to queue a job for printing and get an error message indicating that the printer is not accepting jobs results, use `lpstat -a` or `lpc status` to check that the printer is accepting jobs.
- If a queued job does not print, use `lpstat -p` or `lpc status` to check that the printer is accepting jobs. You may need to move the job to another printer as discussed in the next section.
- If the printer is remote, you may need to check that it still exists on the remote system and that it is operational.
- You may need to update the configuration file to allow a particular user or remote system to print on your printer.
- You may need to ensure that your firewall allows remote printing requests, either from another system to your system, or from your system to another, as appropriate.
- You may need to verify that you have the right driver (as discussed in the final section of this tutorial).

As you can see, printing involves the correct functioning of several components of your system and possibly network. In a tutorial of this length, we can only give you starting points for diagnosis. Most CUPS systems also have a graphical interface to the command-line functions that we discuss here. Generally, this interface is accessible from the local host using a browser pointed to port 631 (<http://localhost:631> or <http://127.0.0.1:631>), as shown in Figure 1.

Figure 1. CUPS home page on port 631



Section 3. Print files

This section covers material for topic 1.107.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Add and remove jobs from configured printer queues
- Convert text files to PostScript for printing

Print

You learned how to remove files from print queues in [the previous section](#). Here you learn how to print files and change job options.

Many graphical programs provide a method of printing, usually under the **File** menu option. These programs provide graphical tools for choosing a printer, margin sizes, color or black-and-white printing, number of copies, selecting 2-up printing (which is

2 pages per sheet, often used for handouts), and so on. This section shows you the command-line tools for controlling such features, and then a graphical implementation for comparison.

The simplest way to print any file is to use the `lpr` command and provide the file name. This prints the file on the default printer. Listing 10 shows a simple example plus a more complex example. The more complex command is explained below.

Listing 10. Printing with lpr

```
[ian@attic4 ~]$ echo abc>abc.txt
[ian@attic4 ~]$ lpr abc.txt
[ian@attic4 ~]$ lpr -Pxerox -J "Ian's text file" -#2 -m -p -q -r abc.txt
[ian@attic4 ~]$ lpq -l
xerox is ready

ian: 1st                               [job 25 localhost]
      2 copies of Ian's text file      1024 bytes
[ian@attic4 ~]$ ls abc.txt
ls: abc.txt: No such file or directory
```

Table 4 explains the options used on the more complex command above, along with other options that you may use with `lpr`.

Table 4. Options for lpr	
Option	Purpose
-C, -J, or -T	Set a job name.
-P	Select a particular printer.
-#	Specify number of copies. Note this is different to the <code>-n</code> option you saw with the <code>lp</code> command.
-m	Send email upon job completion.
-l	The print file is already formatted for printing. Equivalent to <code>-o raw</code> .
-o	Set a job option.
-p	Format a text file with a shaded header. Equivalent to <code>-o prettyprint</code> .
-q	Hold (or queue) the job for later printing.
-r	Remove the file after it

```
has been spooled for
printing.
```

So, in our complex example: `lpr -Pxerox -J "Ian's text file" -#2 -m -p -q -r abc.txt`, user ian is requesting a specific printer, giving a name to the job, requesting 2 copies, requesting an email confirmation after printing, holding the job, and having the file `abc.txt` removed after it has been spooled. The subsequent commands show the held job and the fact that the file has indeed been removed.

In addition to the `lpr` command, the `lp` command covered in the previous section can also be used to print jobs, as well as modify them. Both `lp` and `lpr` also accept a file from stdin if no file name is given on the command line. In contrast to `lpr`, which quietly spools the job, the `lp` default is to display the job number of the spooled job as shown in Listing 11. Note that not all the equivalent options on `lp` and `lpr` have the same name; for example, `-n` on `lp` is equivalent to `-#` on `lpr`.

Listing 11. Printing from stdin with lp

```
[ian@attic4 ~]$ lp
abc
request id is xerox-27 (1 file(s))
```

So we now have a held job in the xerox print queue. What to do? The `lp` command has options to hold and release jobs, using various values with the `-H` option. Listing 12 shows how to release the held job. Check the man page of `lp` for information on other options.

Listing 12. Resuming printing of a held print job

```
[ian@attic4 ~]$ lp -i 25 -H resume
```

Many different printers are available today, but not all of them support the same set of options. You can find out what general options are set for a printer using the `lpoptions` command. Add the `-l` option to display printer-specific options; Listing 13 shows an example. The man page for the `lp` command also lists several common options, particularly relating to portrait/landscape printing, page dimensions, and placement of the output on the pages.

Listing 13. Checking printer options

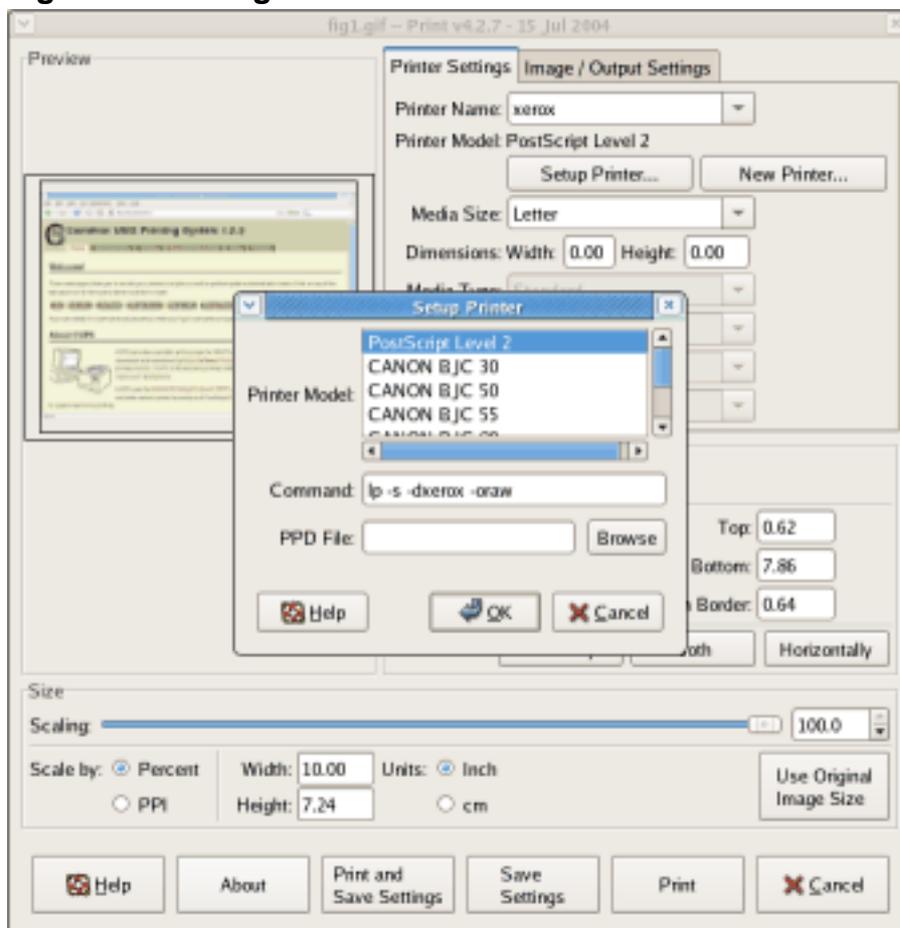
```
[ian@attic4 ~]$ lpoptions -p xerox
job-sheets=none,none printer-info='Xerox Docuprint C20' printer-is-accepting-
jobs=1 printer-is-shared=1 printer-make-and-model='Xerox DocuPrint C20 Foomat
ic/Postscript (recommended)' printer-state=3 printer-state-change-time=115550
6374 printer-state-reasons=none printer-type=143388 cpi=12 scp-fc5=true lpi=7
page-bottom=86 page-left=57 page-right=57 page-top=72 scaling=100 wrap=true
```

```
[ian@attic4 ~]$ lpoptions -l
PageSize/Page Size: *Letter A4 11x17 A3 A5 B5 Env10 EnvC5 EnvDL EnvISOB5 EnvM
onarch Executive Legal
PageRegion/PageRegion: Letter A4 11x17 A3 A5 B5 Env10 EnvC5 EnvDL EnvISOB5 En
vMonarch Executive Legal
Duplex/Double-Sided Printing: DuplexNoTumble DuplexTumble *None
Resolution/Resolution: *default 150x150dpi 300x300dpi 600x600dpi
PreFilter/GhostScript pre-filtering: EmbedFonts Level1 Level2 *No
```

So far, all our commands have been directed to the local CUPS server. You can also direct most commands to the server on another system, by specifying the `-h` option along with a port number if it is not the CUPS default of 631.

Before moving on to filters, let's look at how all this magic avails itself in a GUI application. Figure 2 shows the illustration of Figure 1 in the GIMP, an image manipulation program. Using the **File > Print** option, you have many choices about how to print the image. In this application, you can also click the **Setup Printer** button to choose a printer and see the command that will be used to print the file, in this case, `lp -s -dxerox -oraw`.

Figure 2. Printing from the GIMP



Convert files

You may have noticed that we were able to print text files above, even though the r220 printer happens to be an Epson photo printer, while the xerox printer is a PostScript Xerox Docuprint C20. This magic feat is accomplished through the use of *filters*. Indeed, a popular filter for many years was named *magicfilter*.

The number of filters included with most CUPS packages allows almost any kind of file to be printed. Additional filters are available commercially, from companies including Easy Software Products, the developers of CUPS.

CUPS uses MIME (Multipurpose Internet Mail Extensions) types to determine the appropriate conversion filter when printing a file. The section on [filter installation](#), later in this tutorial, goes into detail. Other printing packages may use the *magic number* mechanism as used by the `file` command. See the man pages for `file` or `magic` for more details.

The general print flow is to convert the input file to a PostScript format using the appropriate filter for the file type, such as `texttops`, `imagetops`, or `pdftops`. The PostScript format is then filtered through a `pstoraster` filter to create an intermediate raster format for non-PostScript printers before being filtered through a printer backend, which prepares it for printing on a particular printer. Ghostscript is a popular program that can print PostScript files on many different printers. A companion viewer allows display of the file on a monitor. Many printer backends are derived from Ghostscript printer drivers.

Before all of this was handled so automatically, it was necessary to convert input to PostScript format. Images could be handled with a program such as the GIMP that we saw earlier. ASCII text files were usually converted to PostScript using the `a2ps` command. The default for plain text files is to print 2-up with a header and direct output to the default printer as shown in Listing 14.

Listing 14. Printing text files with `a2ps`

```
[ian@attic4 ~]$ a2ps -4 abc.txt -o abc.ps
[abc.txt (plain): 1 page on 1 sheet]
[Total: 1 page on 1 sheet] saved into the file `abc.ps'
```

The `a2ps` command can handle a wide range of text file types and make intelligent decisions about the best way to format them. For example, the default for LaTeX files is to first format the file and then print 2-up. Listing 15 uses `a2ps` to print a copy of the `sample2e.tex` file that is distributed with LaTeX, and then shows a copy renamed to `sample2e.txt`, and printed 4-up with headers. Both are saved to an output PostScript format file. Figure 3 shows how the output of the second command is formatted.

Listing 15. Saving output from a2ps as a PostScript file

```
[ian@attic4 ~]$ a2ps -4 -E -o fig3.ps sample2e.tex
[sample2e.tex (tex, delegated to texi2dvi): 1 page on 1 sheet]
[Total: 4 pages on 1 sheet] saved into the file `fig3.ps'
[ian@attic4 ~]$ a2ps -4 -E -o fig3.ps sample2e.txt
[sample2e.txt (plain): 4 pages on 1 sheet]
[Total: 4 pages on 1 sheet] saved into the file `fig3.ps'
```

Figure 3. Pretty printed output from a2ps



There are many other filters that you can use to format files for printing in special ways. Most have a range of options. Check the man pages for more details. Some examples are:

mpage

Formats test files for printing multiple pages on a single page.

psnup

Performs similar functions for PostScript files as mpage does for text files.

psbook

Rearranges the pages of a PostScript document for printing as a book or booklet, taking into account the number of pages per sheet and how the sheet is folded.

Section 4. Printer installation and configuration

This section covers material for topic 1.107.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Install a printer daemon
- Install and configure a print filter
- Access local and remote printers of various kinds

Printer daemons

Install a printer daemon by first installing the printer package, either CUPS, or another such as LPRng, which is usually shipped with a distribution. If you require one that is not shipped with your distribution, you may find a package prebuilt for your distribution, or you may build it yourself from source. Refer to the tutorial for Exam 101 topic 102, "[LPI exam 101 prep: Linux installation and package management](#)," if you need help with this task.

Once the printer package is installed, ensure that the printer daemon starts when your system starts. This is covered in the tutorial for Exam 102 topic 106 "[LPI exam 102 prep: Boot, initialization, shutdown, and runlevels](#)."

Use the `cups` script located in `/etc/rc.d/init.d` or `/etc/init.d`, according to your distribution with the `status` command. You can also use the `lpstat` or `lpc status` command to check whether your daemon is running. If you are using a different printer daemon, use the appropriate script for your package. An example is shown in Listing 16.

Listing 16. Checking CUPS daemon status

```
[root@attic4 ~]# /etc/init.d/cups stop  
Stopping cups: [ OK ]  
[root@attic4 ~]# /etc/init.d/cups status
```

```

cupsd is stopped
[root@attic4 ~]# lpstat -d
lpstat: Unable to connect to server
[root@attic4 ~]# /etc/init.d/cups start
Starting cups: [ OK ]

```

If you ever need to debug CUPS, you can run it in the foreground rather than as a daemon process. You can also test alternate configuration files if necessary. Run `cupsd -h` for more information, or see the man pages.

Listing 17. Running cupsd from the command line

```

[root@attic4 ~]# cupsd -h
Usage: cupsd [-c config-file] [-f] [-F] [-h] [-l] [--ppdsdat]

-c config-file      Load alternate configuration file
-f                  Run in the foreground
-F                  Run in the foreground but detach
-h                  Show this usage message
-l                  Run cupsd from launchd(8)
--ppdsdat           Just build ppds.dat

```

CUPS also maintains an access log and an error log. You can change the level of logging using the `LogLevel` statement in `/etc/cups/cupsd.conf`. By default, logs are stored in the `/var/log/cups` directory. They may be viewed from the **Administration** tab on the Web interface (<http://localhost:631>).

Print filters

So how does CUPS determine the filter to use for formatting a particular file type? CUPS uses MIME (Multipurpose Internet Mail Extensions) types to determine the appropriate conversion filter when printing a file. Note that other printing packages may use the *magic number* mechanism as used by the `file` command. See the man pages for `file` or `magic` for more details.

MIME types are used for transmitting various files as mail attachments. They consist of a type, such as `text` or `image`, and a subtype, such as `html`, `postscript` `gif`, or `jpeg`. The type and subtype are separated by a semicolon (;). Optional parameters may include information such as character set encoding, or language. CUPS uses rules from `/etc/cups/mime.types` to determine the type of a file and then uses a suitable filter chosen from those listed in `/etc/cups/conv.types` for the given MIME type. MIME types are registered with IANA, the Internet Assigned Numbers Authority. If you need a type that is not registered, prefix the subtype with 'x-'. Some image type examples are shown in Listing 18.

Listing 18. Some MIME type entries from /etc/cups/mime.types

```

image/gif          gif string(0,GIF87a) string(0,GIF89a)
image/png          png string(0,<89>PNG)
image/jpeg         jpeg jpg jpe string(0,<FFD8FF>) &&\
                  (char(3,0xe0) char(3,0xe1) char(3,0xe2) char(3,0xe3)\
                  char(3,0xe4) char(3,0xe5) char(3,0xe6) char(3,0xe7)\
                  char(3,0xe8) char(3,0xe9) char(3,0xea) char(3,0xeb)\
                  char(3,0xec) char(3,0xed) char(3,0xee) char(3,0xef))
image/tiff         tiff tif string(0,MM) string(0,II)
image/x-photocd   pcd string(2048,PCD_IPI)
image/x-portable-anymap  pnm

```

The format of the entries is beyond the scope of this tutorial. Check the files `/usr/share/mime/magic` or `/usr/share/file/magic` for some insight on how the *magic numbers* are used to identify files.

Once the MIME type of a file has been determined, the correct filter is found using the `/etc/cups/mime.convs` file. Lines in this file have four entries, a source and destination MIME type, a *cost*, and the name of the filter. The least-cost filter is used. Some examples are shown in Listing 19.

Listing 19. Filter entries from `/etc/cups/mime.convs`

```

text/plain        application/postscript  33      texttops
text/html         application/postscript  33      texttops
image/gif         application/vnd.cups-postscript  66      imagetops
image/png         application/vnd.cups-postscript  66      imagetops
image/jpeg        application/vnd.cups-postscript  66      imagetops
image/tiff        application/vnd.cups-postscript  66      imagetops
image/x-bitmap    application/vnd.cups-postscript  66      imagetops

```

If a suitable filter cannot be found, your attempt to print a file will result in an error message. If you are using a printer daemon other than CUPS, you may get unexpected output instead. Listing 20 shows how this works with a DVI file (the normal output from TeX and LaTeX).

Listing 20. Printing an unsupported file type

```

[ian@attic4 ~]$ lpr sampl.dvi
lpr: Unsupported format 'application/octet-stream'!

```

Fortunately, the `tetex` package that provides TeX and LaTeX also provides a conversion utility, `dvips` to convert from DVI to PostScript. Unfortunately, it won't work as a filter because it does not know how to handle the arguments that a CUPS filter must handle, namely, job id, user, job title, number of copies, and job options. The first filter in a filter pipeline will also have an additional parameter, the filename, if the input comes from a file.

The solution is to create a wrapper script that will be the filter. The `dvips` command does not accept input from stdin, so the script may need to create a temporary file and copy stdin to that file before calling `dvips`. A possible script is shown in Listing

21.

Listing 21. A CUPS DVI to PostScript filter script

```
#!/bin/bash
# CUPS filter to process DVI files using dvips
# Create a sandbox for working if input on stdin
if [ $# -lt 6 ]; then
    sandbox=${TMPDIR-/tmp}/cups-dvitops.$$
    (umask 077 && mkdir $sandbox) || {
        echo "Cannot create temporary directory! Exiting." 1>&2
        exit 1
    }
    fn="$sandbox/cups-dvitops.$$"
    cat > "$fn"
else
    fn="$6"
fi
# Call dvips quietly, securely and with output to stdout
dvips -R -q -o - "$fn"
# Erase sandbox if we created one
if [ $# -lt 6 ]; then
    rm -rf "$sandbox"
fi
```

Recall that CUPS uses two files in `/etc/cups` to determine the MIME type and filter to use. These files will be overwritten whenever you reinstall or upgrade CUPS. Fortunately, CUPS will read **all** files with an extension of `.types` or `.convs` whenever it starts or restarts. So you should create a pair of files for your new filter, for example `/etc/cups/dvitops.types` and `/etc/cups/dvitops.convs` as shown in Listing 22 shows a partial output listing for Docuprint drivers.

Listing 22. Configuration files for CUPS dvitops filter

```
[ian@attic4 ~]$ cat /etc/cups/dvitops.types
# Local MIME definition for DVI files
application/x-dvi dvi string(0,<F702>)
[ian@attic4 ~]$ cat /etc/cups/dvitops.convs
# Local DVI to PostScript filter for CUPS
application/x-dvi application/postscript 50 dvitops
```

This says that DVI files are identified by having the hexadecimal digits F7 and 02 in the first two positions and that such files should be processed by the `dvitops` filter.

Next, as root, copy the script above in `/usr/lib/cups/filter/dvitops` and make sure it is world readable and executable (`-rwxr-xr-x`). The name you give the script must match that in the `/etc/cups/dvitops.convs` file above. If you are running SELinux in enforcing mode, you should also run `restorecon` in the `/usr/lib/cups/filter` directory to update the security contexts. Otherwise, your `lpr` command will appear to work, but your file will not print.

Finally, use the restart option with the cups script located in `/etc/rc.d/init.d` or `/etc/init.d`, to restart CUPS and use your new filter.

If you are using an older print spooler, you will probably use either the `magicfilter` or `apsfilter` as input filters to convert various input files to PostScript format for printing to a PostScript printer or, using Ghostscript, to a non-PostScript printer.

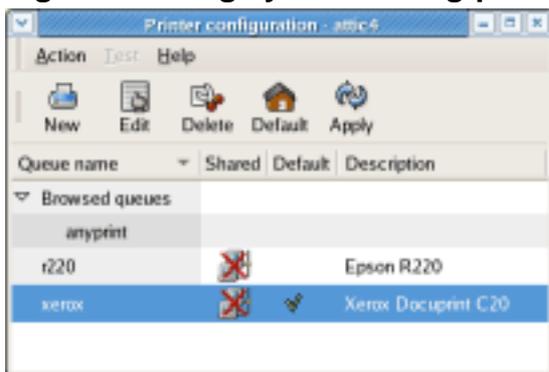
Accessing printers

CUPS supports a variety of printers, including:

- Locally attached parallel and USB printers
- IPP (Internet Printing Protocol) printers
- Remote LPD printers
- Windows® printers using SAMBA
- Novell printers using NCP
- HP JetDirect attached printers

Most systems now attempt to autodetect and autoconfigure local hardware when the system starts or when the device is attached. Similarly, many network printers can be autodetected. You can use the CUPS Web administration tool (<http://localhost:631> or <http://127.0.0.1:631>) to search for or add printers. Many distributions include their own configuration tools, for example YaST on SUSE systems. Figure 4 illustrates the system-config-printer tool on Fedora Core 5.

Figure 4. Using system-config-printer on Fedora Core 5



You can also configure printers from a command line. The rest of this tutorial shows you how. An understanding of this material will help you answer exam questions on the GUI interfaces.

Before you configure a printer, you need some basic information about the printer and about how it is connected. For illustration, we will use a Xerox Docuprint C20 attached through a D-Link print server. The print server provides LPD printing function. To configure it, we will need the IP address (192.168.0.10, in this case) and

a printer queue name on the LPD server. This is set in the print server and is PS-66D975-P1 in this case. If a remote system needs a user id or password, you will also need that information.

You will also need to know what driver to use for your printer. Check at LinuxPrinting.org (see [Resources](#) later in this tutorial for a link) to see if there is a driver for your particular printer. The `lpinfo` command can also help you identify available device types and drivers. Use the `-v` option to list supported devices and the `-m` option to list drivers, as shown in Listing 23.

Listing 23. Available printer drivers

```
lyrebird:~ # lpinfo -m | grep -i "docuprint.c"
Xerox/DocuPrint_C6-cdj550.ppd.gz Xerox DocuPrint C6 Foomatic/cdj550 (recommended)
Xerox/DocuPrint_C8-cdj550.ppd.gz Xerox DocuPrint C8 Foomatic/cdj550 (recommended)
Xerox/DocuPrint_C11-cdj500.ppd.gz Xerox DocuPrint C11 Foomatic/cdj500
Xerox/DocuPrint_C11-hpdj.ppd.gz Xerox DocuPrint C11 Foomatic/hpdj
Xerox/DocuPrint_C11-pcl3.ppd.gz Xerox DocuPrint C11 Foomatic/pcl3 (recommended)
Xerox/DocuPrint_C20-cljet5.ppd.gz Xerox DocuPrint C20 Foomatic/cljet5
Xerox/DocuPrint_C20-hpijs.ppd.gz Xerox DocuPrint C20 Foomatic/hpijs
Xerox/DocuPrint_C20-Postscript.ppd.gz Xerox DocuPrint C20 Foomatic/Postscript
(recommended)
Xerox/DocuPrint_C55-Postscript.ppd.gz Xerox DocuPrint C55 Foomatic/Postscript
(recommended)
```

Several choices are shown here for the Docuprint C20. The recommended driver is the PostScript driver, which is not surprising, since this printer supports PostScript. Again, if you can't find your printer listed, check at LinuxPrinting.org (see [Resources](#)) for the appropriate driver. Drivers come in the form of PPD (PostScript Printer Description) files.

Now that you have the basic information, you can configure a printer using the `lpadmin` command as shown in Listing 24. Note that this system did not list the specific Xerox Docuprint driver, so we use the generic PostScript driver instead.

Listing 24. Configuring a printer

```
[root@attic4 ~]# lpinfo -m | grep -i generic
textonly.ppd Generic text-only printer
postscript.ppd.gz Generic postscript printer
[root@attic4 ~]# lpadmin -p xerox1 -E -m "postscript.ppd.gz" \
> -v "lpd:192.168.0.1/PS-66D975-P1" -D "Xerox 1"
[root@attic4 ~]# lpstat -a
anyprint accepting requests since Sat 12 Aug 2006 04:07:46 PM EDT
r220 accepting requests since Tue 22 Aug 2006 11:13:40 AM EDT
xerox accepting requests since Tue 22 Aug 2006 11:13:40 AM EDT
xerox1 accepting requests since Tue 22 Aug 2006 11:17:59 AM EDT
```

If you need to remove a printer, use `lpadmin` with the `-x` option as shown in Listing 25.

Listing 25. Removing a printer

```
[root@attic4 ~]# lpadmin -x xerox1
```

You can also set various printer options using the `lpadmin` or `lpoptions` commands.

Spool files

CUPS uses the `/var/spool/cups` directory for spooling. This will usually be set up correctly when you install CUPS. If you are using the LPD daemon, spool files are stored in directories such as `/var/spool/lpd/xerox`, for our printer 'xerox'. Spool directories and files should have permissions set to protect them from being read or written by users other than the printing system.

Other input filters

If you are using LPD, LPRng, or another printing system, you will probably use either `magicfilter` or `apsfilter` for converting input files to PostScript format, and you will probably use Ghostscript as the printer driver for non-PostScript printers. Configuration for printers and filters will be in `/etc/printcap`. If you are using these, consult the man pages or online documentation such as the *Apsfilter handbook* listed in [Resources](#) later in this tutorial.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents for system administrators, especially its HOWTOs.
- [LinuxPrinting.org](#) provides information on drivers and printer support as well as a CUPS Quick Start.
- The [Apsfilter handbook](#) can help you configure printers and filters if you are using LPD, LPRng, or another printing system.
- [The Printer Working Group](#) of the IEEE Industry Standards and Technology Organization (IEEE-ISTO) develops standards that make printers -- and the applications and operating systems supporting them -- work better together.
- *CUPS: Common UNIX Printing System* (Sams, 2001) is a detailed reference for CUPS.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 108: Linux documentation

Junior Level Administration (LPIC-1) topic 108

Skill Level: Intermediate

[Ian Shields](#)

Senior Programmer
IBM developerWorks

20 Sep 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this fourth in a [series of nine tutorials](#), Ian introduces you to Linux® documentation. By the end of this tutorial, you will know how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding

developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	(This tutorial). Learn how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events. See detailed objectives below.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Linux documentation," the fourth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.108.1 Use and manage local system documentation	Weight 4	Find relevant man pages and search man page sections. Find commands and the man pages related to them. Configure access to man sources and the man system. Prepare man pages for printouts. Use the system documentation stored in /usr/share/doc/ and determine what documentation to keep in /usr/share/doc/.
1.108.2 Find Linux documentation on the Internet	Weight 3	Use Linux documentation at sources such as the Linux Documentation Project (LDP), vendor and third-party Web sites, newsgroups, newsgroup archives, and mailing lists.
1.108.5 Notify users of system-related issues	Weight 1	Notify the users about current issues related to the system through logon messages.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial. You will also need a connection to the Internet.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Local documentation

This section covers material for topic 1.108.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Find relevant man pages
- Search man page sections
- Find commands and man pages related to them
- Configure access to man sources and the man system
- Prepare man pages for printouts
- Use the system documentation stored in `/usr/share/doc/` and determine what documentation to keep in `/usr/share/doc/`

Find man pages

The primary (and traditional) source of documentation is the *manual pages*, which you can access using the `man` command. Ideally, you can look up the man page for any command, configuration file, or library routine. In practice, Linux is free software, and some pages haven't been written or are showing their age. Nonetheless, man pages are the first place to look when you need help. Figure 1 illustrates the manual page for the `man` command itself. Use the command `man man` to display this information.

Figure 1. Man page for the man command

```

ian@echidna:~
File Edit View Terminal Go Help
1 man(1) man(1)
2 NAME
  man - format and display the on-line manual pages
  manpath - determine user's search path for man pages
3 SYNOPSIS
  man [-acdfhkkTww] [--path] [-m system] [-p string] [-C config_file]
  [-M pathlist] [-P pager] [-S section_list] [section] name ...
4 DESCRIPTION
  man formats and displays the on-line manual pages.  If you specify sec-
  tion, man only looks in that section of the manual.  name is normally
  the name of the manual page, which is typically the name of a command,
  function, or file.  However, if name contains a slash (/) then man
  interprets it as a file specification, so that you can do man ./foo.5
  or even man /cd/foo/bar.1.gz.

  See below for a description of where man looks for the manual page
  files.
5 OPTIONS
  -C config_file
    Specify the configuration file to use; the default is
    /etc/man.config. (See man.conf(5).)

  -M path
    Specify the list of directories to search for man pages.  Sepa-
    rate the directories with colons.  An empty list is the same as
    not specifying -M at all.  See SEARCH PATH FOR MANUAL PAGES.

  -P pager
    Specify which pager to use.  This option overrides the MANPAGER
    environment variable, which in turn overrides the PAGER vari-
    able.  By default, man uses /usr/bin/less -isr.
:

```

Figure 1 shows some typical items in man pages:

1. A heading with the name of the command followed by its section number in parentheses
2. The name of the command and any related commands that are described on the same man page
3. A synopsis of the options and parameters applicable to the command
4. A short description of the command
5. Detailed information on each of the options

You may find other sections on usage, how to report bugs, author information, and a list of any related commands. For example, the man page for `man` tells us that related commands (and their manual sections) are: `apropos(1)`, `whatis(1)`, `less(1)`, `groff(1)`, and `man.conf(5)`.

Man pages are displayed using a *pager*, which is usually the `less` command on Linux systems. You can set this using the `$PAGER` environment variable, or by using the `-P` or `--pager` option, along with another pager name, on the `man` command. The pager will receive its input on `stdin`, so something like an editor that expects a file to manipulate does not work as a pager.

There are eight common manual page sections. Manual pages are usually installed when you install a package, so if you do not have a package installed, you probably won't have a manual page for it. Similarly, some of your manual sections may be empty or nearly empty. The common manual sections, with some example contents are:

1. User commands (`env`, `ls`, `echo`, `mkdir`, `tty`)
2. System calls or kernel functions (`link`, `sethostname`, `mkdir`)
3. Library routines (`acosh`, `asctime`, `btree`, `locale`, `XML::Parser`)
4. Device-related information (`isdn_audio`, `mouse`, `tty`, `zero`)
5. File format descriptions (`keymaps`, `motd`, `wvdial.conf`)
6. Games (note that many games are now graphical and have graphical help outside the man page system)
7. Miscellaneous (`arp`, `boot`, `regex`, `unix utf8`)
8. System administration (`debugfs`, `fdisk`, `fsck`, `mount`, `renice`, `rpm`)

Other man page sections that you might find include `9` for Linux kernel documentation, `n` for new documentation, `o` for old documentation, and `l` for local documentation.

Some entries appear in multiple sections. Our examples show `mkdir` in sections 1 and 2, and `tty` in sections 1 and 4.

The `info` command

In addition to the standard manual pages, the Free Software Foundation has created a number of *info* files that are processed with the `info` program. These provide extensive navigation facilities including the ability to jump to other sections. Try `man info` or `info info` for more information. Not all commands are documented with `info`, so you will find yourself using both `man` and `info` if you become an `info` user. You can also start at the top of the `info` tree by using `info` without parameters as shown in Listing 1.

Listing 1. The info command

```
File: dir,      Node: Top      This is the top of the INFO tree

This (the Directory node) gives a menu of major topics.
Typing "q" exits, "?" lists all Info commands, "d" returns here,
"h" gives a primer for first-timers,
"mEmacs<Return>" visits the Emacs manual, etc.

In Emacs, you can click mouse button 2 on a menu item or cross reference
to select it.

* Menu:

Utilities
* Bash: (bash).           The GNU Bourne-Again SHell.
* Enscript: (enscript).   GNU Enscript
* Gzip: (gzip).           The gzip command for compressing files.
* ZSH: (zsh).             The Z Shell Manual.

Libraries
* AA-lib: (aalib).        An ASCII-art graphics library
* History: (history).     The GNU history library API
* Libxmi: (libxmi).       The GNU libxmi 2-D rasterization library.
* Readline: (readline).   The GNU readline library API

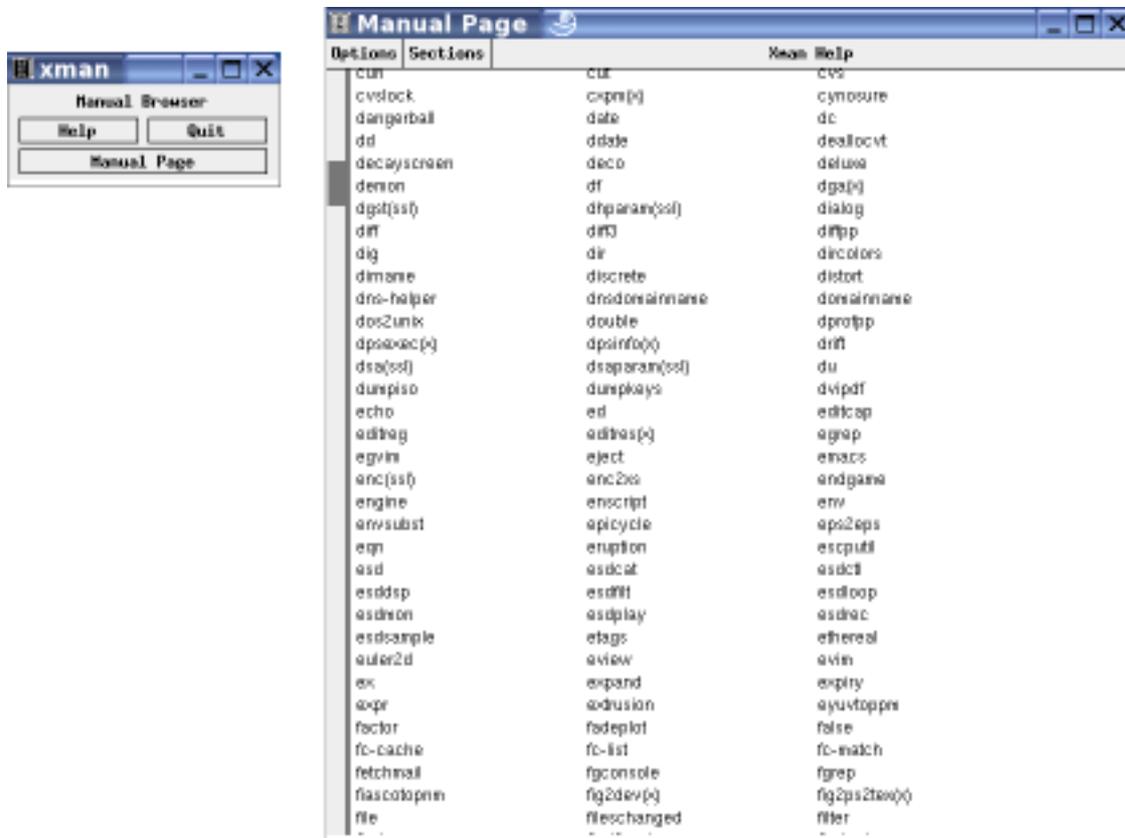
Texinfo documentem
* Info: (info).           Documentation browsing system.
-----Info: (dir)Top, 2104 lines --Top-----
Welcome to Info version 4.6. Type ? for help, m for menu item.
```

Graphical man page interfaces

In addition to the standard `man` command, which uses a terminal window and a pager, your system may also have one or more graphical interfaces to manual pages, such as `xman` (from the XFree86 Project) and `yelp` (the Gnome help browser).

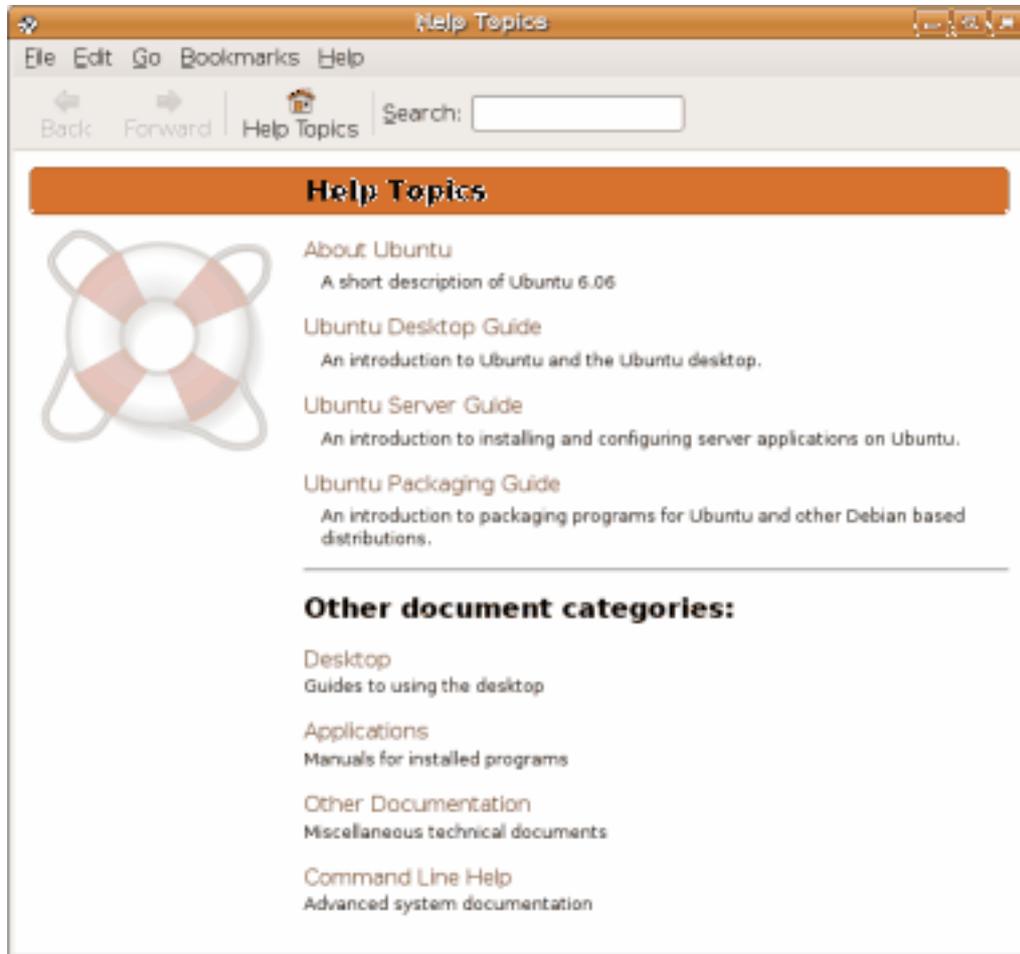
When you start `xman`, you will see a small window with three buttons. Click the **Manual Page** button to open a larger window where you can navigate through manual pages or search for information. Figure 2 shows an example of both windows.

Figure 2. Using xman



The `yelp` browser usually looks somewhat different from system to system. Figure 3 shows an example on Ubuntu 6.06. You can access either the man pages or the info pages using the **Command Line Help** item at the bottom of the display.

Figure 3. Using yelp on Ubuntu



Search man pages

If you know that a topic occurs in a particular section, you can specify the section. For example, `man 4 tty` or `man 2 mkdir`. An alternative is to use the `-a` option to display all applicable manual sections. If you specify `-a`, you will be prompted after quitting the page for each section. You may skip the next page, view it, or quit altogether.

As you saw earlier, some topics exist in more than one section. If you don't want to search through each section, you can use the `-aw` options of `man` to get a list of all available man pages for a topic. Listing 2 shows an example for `printf`. If you were writing a portable shell script, you might be interested in `man 1p printf` to learn about the POSIX version of the `printf` command. On the other hand, if you were writing a C or C++ program, you would be more interested in `man 3 printf`, which would show you the documentation for the `printf`, `fprintf`, `sprintf`, `snprintf`, `vprintf`, `vfprintf`, `vsprintf`, and `vsnprintf` library functions.

Listing 2. Available man pages for printf

```
ian@lyrebird:~> man -aw printf
/usr/share/man/man1/printf.1.gz
/usr/share/man/man1p/printf.1p.gz
/usr/share/man/man3/printf.3.gz
```

The `man` command pages output onto your display using a paging program. On most Linux systems, this is likely to be the `less` program. Another choice might be the older `more` program.

The `less` pager has several commands that help you search for strings within the displayed output. These are similar to `vi` editing commands. Use `man less` to find out more about `/` (search forwards), `?` (search backwards), and `n` (repeat last search), among many other commands.

The `info` command comes from the makers of `emacs`, so the searching commands are more like `emacs` commands. For example, `ctrl-s` searches forwards and `ctrl-r` searches backwards using an incremental search. You can also move around with the arrow keys, follow links (indicated with a star) using the Enter key, and quit using `q`. Use the `--vi-keys` option with `info` if you'd prefer similar key bindings to those used for `man`.

Find commands

Two important commands related to `man` are `whatis` and `apropos`. The `whatis` command searches man pages for the name you give and displays the name information from the appropriate manual pages. The `apropos` command does a keyword search of manual pages and lists ones containing your keyword. Listing 3 illustrates these commands.

Listing 3. Whatis and apropos examples

```
[ian@lyrebird ian]$ whatis man
man          (1) - format and display the on-line manual pages
man          (7) - macros to format man pages
man [manpath] (1) - format and display the on-line manual pages
man.conf [man] (5) - configuration data for man
[ian@lyrebird ian]$ whatis mkdir
mkdir        (1) - make directories
mkdir        (2) - create a directory
[ian@lyrebird ian]$ apropos mkdir
mkdir        (1) - make directories
mkdir        (2) - create a directory
mkdirhier    (1x) - makes a directory hierarchy
```

By the way, if you cannot find the manual page for `man.conf`, try running `man man.config` instead, which works on some systems.

The `apropos` command can produce a lot of output, so you may need to use more complex regular expressions rather than simple keywords. Alternatively, you may wish to filter the output through `grep` or another filter to reduce the output to something more of interest. As a practical example, you can use the `e2label` to display or change the label on an `ext2` or `ext3` filesystem, but you have to use another command to change the label on a ReiserFS filesystem. Suppose you run `mount` to display the mounted ReiserFS filesystems as shown in Listing 4.

Listing 4. Mounted ReiserFS filesystems

```
ian@lyrebird:~> mount -t reiserfs
LABEL=SLES9 on / type reiserfs (rw,acl,user_xattr)
```

Now you'd like to know what partition corresponds to the label `SLES9`, but you can't remember the command. Using `apropos label` might get you a couple of dozen responses, which isn't too bad to sift through. But wait. This command must have something to do with a filesystem of a volume. So you try the regular expressions shown in Listing 5.

Listing 5. Using `apropos` with regular expressions

```
ian@lyrebird:~> apropos "label.*file"
e2label (8)          - Change the label on an ext2/ext3 filesystem
ntfslabel (8)       - display/change the label on an ntfs file system
ian@lyrebird:~> apropos "label.*volume"
label.*volume: nothing appropriate.
```

Not exactly what you were looking for. You could try reversing the order of the terms in the regular expressions, or you could try filtering through `grep` or `egrep` as shown in Listing 6.

Listing 6. Filtering the output of `apropos`

```
ian@lyrebird:~> apropos label | grep -E "file|volume"
e2label (8)          - Change the label on an ext2/ext3 filesystem
mlabel (1)          - make an MSDOS volume label
ntfslabel (8)       - display/change the label on an ntfs file system
findfs (8)          - Find a filesystem by label or UUID
```

And there's the command that we need, `findfs`. Using it as shown in Listing 7 shows that the filesystem is on `/dev/hda10` on this particular system.

Listing 7. Finding the device for a mounted filesystem label

```
ian@lyrebird:~> /sbin/findfs LABEL=SLES9
/dev/hda10
```

Note that non-root users will usually have to give the full path to the `findfs` command.

As you can find out in the man page for the `man` command, you can also use `man -k` instead of `apropos` and `man -f` instead of `whatis`. Since these call the `apropos` or `whatis` command under the covers, there is probably little point in so doing.

Configuration

Manual pages may be in many locations on your system. You can determine the current search path using the `manpath` command. If the `MANPATH` environment variable is set, this will be used for searching for manual pages; otherwise, a path will be built automatically using information from a configuration file that we'll discuss in a moment. If the `MANPATH` environment variable is set, the `manpath` command will issue a warning message to this effect before displaying the path.

Listing 8. Displaying your MANPATH

```
[ian@echidna ian]$ manpath
/usr/local/share/man:/usr/share/man:/usr/X11R6/man:/usr/local/man

ian@lyrebird:~> manpath
manpath: warning: $MANPATH set, ignoring /etc/manpath.config
/usr/local/man:/usr/share/man:/usr/X11R6/man:/opt/gnome/share/man
```

Depending on your system, configuration information for the man system is stored in `/etc/man.config` or `/etc/manpath.config`. Older systems use `/etc/man.conf`. A current `man.config` file contains a list of directories (MANPATHs) that will be searched for manual pages, such as those shown in Listing 9.

Listing 9. MANPATH entries from /etc/man.config

```
MANPATH /usr/share/man
MANPATH /usr/man
MANPATH /usr/local/share/man
MANPATH /usr/local/man
MANPATH /usr/X11R6/man
```

In a `manpath.config` file, these entries will be `MANDATORY_MANPATH` entries, rather than `MANPATH` entries.

Besides these entries, you will also find entries giving a mapping between paths where executables may be found, and paths where the corresponding man pages might be, as shown in Listing 10.

Listing 10. MANPATH_MAP entries from /etc/man.config

```
MANPATH_MAP    /bin                /usr/share/man
MANPATH_MAP    /sbin              /usr/share/man
MANPATH_MAP    /usr/bin           /usr/share/man
MANPATH_MAP    /usr/sbin          /usr/share/man
MANPATH_MAP    /usr/local/bin     /usr/local/share/man
```

The `man` command uses a complicated method for searching for man pages, and setting these values will result in less wasted effort when searching for pages.

Another entry in the configuration file defines the search order for manual pages. Recall that the default is to display the first page found, so this ordering is important. Look near the bottom of `man.config` for a `MANSECT` line, or near the bottom of `manpath.config` for a `SECTION` line. Examine the configuration file on your system to see what other things can be configured.

You may have noticed that the `apropos` and `whatis` commands ran quickly. This is because they do not actually search the individual manual pages. Rather, they use a database created by the `makewhatis` command. This is usually run by the system either daily or weekly as a cron job.

Listing 11. Running `makewhatis`

```
[root@echidna root]# makewhatis
```

The command completes normally without any output message, but the `whatis` database is refreshed. This is usually stored in a location such as `/var/cache/man/whatis`. Note that some SUSE systems do not use the `whatis` database and therefore do not have a `makewhatis` command.

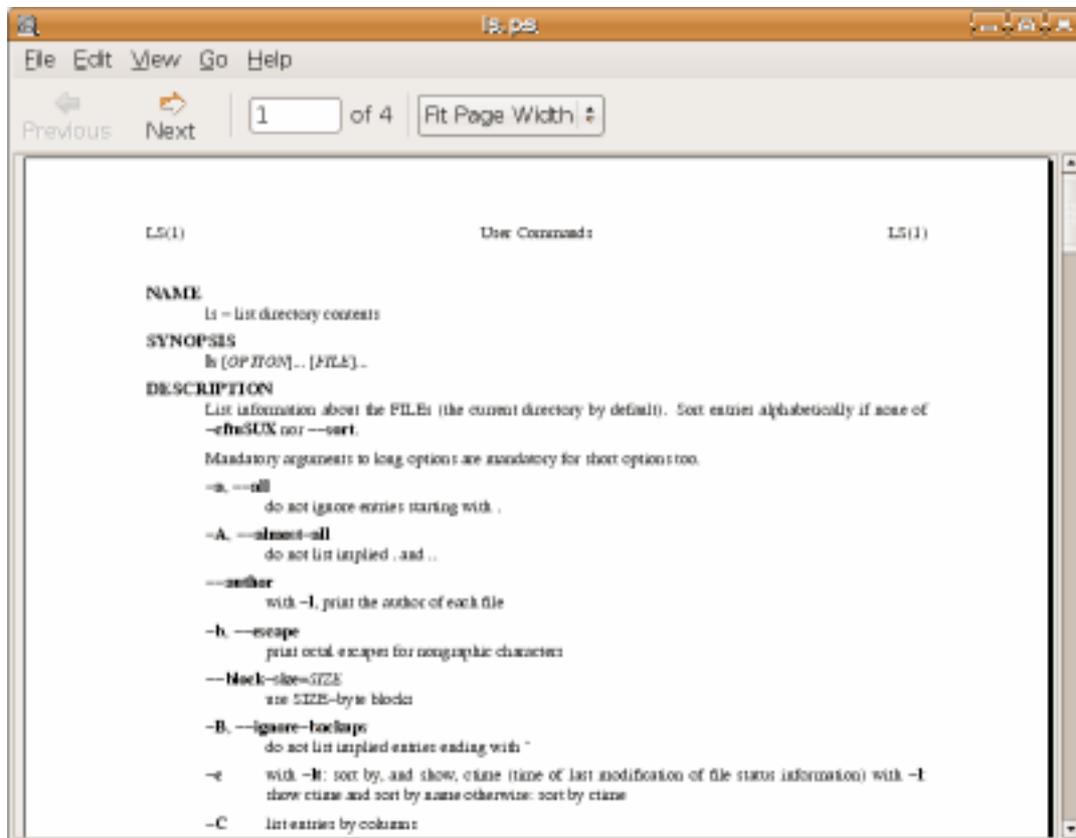
Printing man pages

If you wish to print the page, specify the `-t` option to format the page for printing using the `groff` or `troff` program. This will format the page for the default printer and send the output to `stdout`. Listing 12 shows how to format the man page for the `ls` command and save the output in a file, `ls.ps`. Figure 4 shows the formatted output.

Listing 12. Formatting the `ls` manpage for printing

```
ian@pinguino:~$ man -t ls > ls.ps
```

Figure 4. Formatted `ls` man page



If you need to format the page for a different device type, use the `-T` options with a device type, such as `dvi` or `ps`. See the man page for `man` for additional information.

/usr/share/doc/

In addition to the manual pages and info pages that you have already seen, your Linux system probably includes a lot more documentation. The customary place to store this is in `/usr/share/doc`, or `/usr/doc` on older systems. This additional documentation may be in any of several formats, such as text, PDF, PostScript, or HTML.

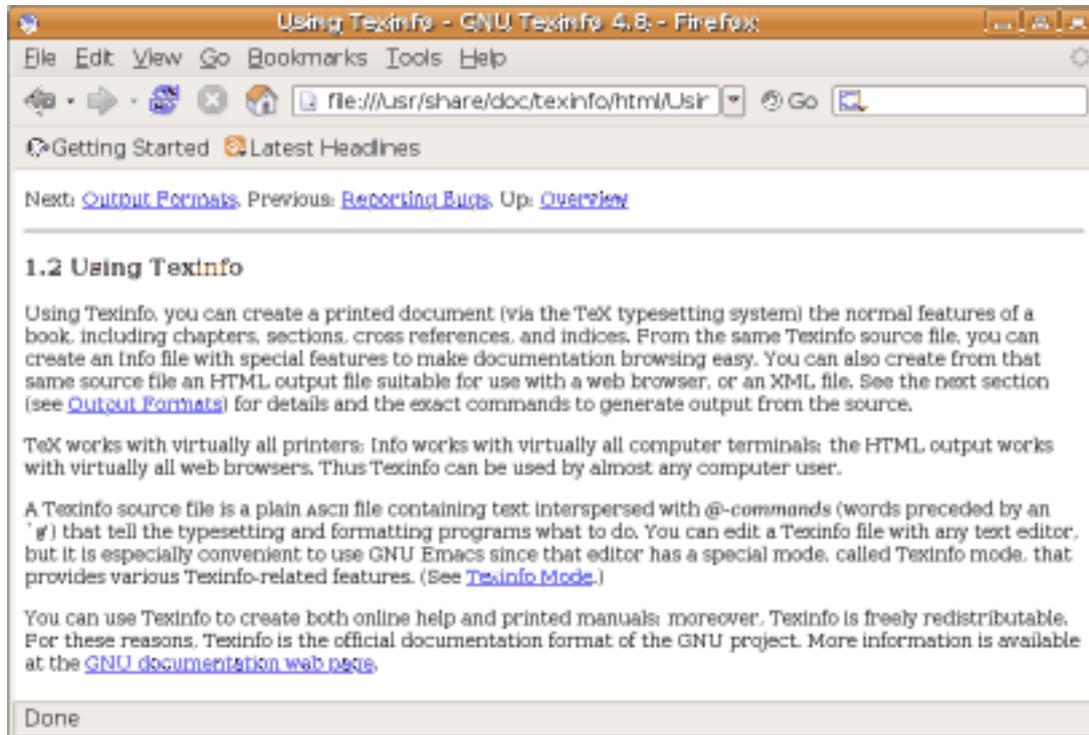
Searching through this documentation can often reveal gems that aren't available as man pages or info pages, such as tutorials or additional technical documentation. As Listing 13 shows, there can be a large number of files in `/usr/share/doc`, so you have plenty of reading resources.

Listing 13. Files in /usr/share/doc

```
ian@pinguino:~$ find /usr/share/doc -type f | wc -l
10144
```

Figure 5 shows an example of the HTML help for the Texinfo system that is used for the `info` command that you saw earlier.

Figure 5. Texinfo HTML help from `/usr/share/doc`



Sometimes, a man page will direct you to another source for documentation. For example, the man page for the `pngtopnm` command is shown in Listing 14. It directs you to a local copy in HTML format at `/usr/share/doc/packages/netpbm/doc/pngtopnm.html`, or to an online version if you do not have the local copy.

Listing 14. Pointer man page for `pngtopnm`

```
pngtopnm(1)           Netpbm pointer man pages           pngtopnm(1)

pngtopnm is part of the Netpbm package.  Netpbm documentation is
kept in HTML format.

Please refer to
<http://netpbm.sourceforge.net/doc/pngtopnm.html>.

If that doesn't work, also try <http://netpbm.sourceforge.net>
and emailing Bryan Henderson, bryanh@giraffe-data.com.

Local copy of the page is here:
/usr/share/doc/packages/netpbm/doc/pngtopnm.html
```

Other command help

Finally, if you can't find help for a command, try running the command with the `--help`, `--h`, or `--?` option. This may provide the command's help, or it may tell you how to get the help you need. Listing 15 shows an example for the `kdesu` command, which is usually present on systems with a KDE desktop.

Listing 15. Getting help for `kdesu` command

```
ian@lyrebird:~> man kdesu
No manual entry for kdesu
ian@lyrebird:~> kdesu --help
Usage: kdesu [Qt-options] [KDE-options] command

Runs a program with elevated privileges.

Generic options:
  --help                Show help about options
  --help-qt             Show Qt specific options
  --help-kde           Show KDE specific options
  --help-all          Show all options
  --author             Show author information
  -v, --version        Show version information
  --license            Show license information
  --                  End of options

Arguments:
  command              Specifies the command to run.

Options:
  -c <command>       Specifies the command to run. []
```

The next section covers online resources for help with Linux.

Section 3. Internet documentation

This section covers material for topic 1.108.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to find:

- Online documentation
- Newsgroups
- Mailing lists

Online documentation

In addition to the documentaiton on your system, there are many online sources of

documentation and help.

The Linux Documentation Project

The [Linux Documentation Project](#) is volunteer effort that is putting together the complete set of free Linux documentation. This project exists to consolidate various pieces of Linux documentation into a location that is easy to search and use.

The LDP is made up of the following areas:

HOWTOs

are subject-specific help, such as the [Linux IPv6 HOWTO](#).

Guides

are longer, in-depth books, such as [Introduction to Linux - A Hands on Guide](#).

FAQs

are Frequently Asked Questions, such as the [Linux Documentation Project \(LDP\) FAQ](#).

man pages

are help on individual commands, as you used in the previous section of this tutorial.

Linux Gazette

is an online magazine, currently available in English, French, German, Indonesian, Italian, Portuguese, Russian, and Spanish.

The examples here take you to the multiple-page HTML versions of the documentation. You will find most articles come in several formats, including single-page HTML, PDF, or plain text, among others.

The LDP also has links to [information in languages other than English](#).

The LDP site is well laid out with excellent navigation. If you aren't sure which section to peruse, you can take advantage of the search box, which helps you find things by topic.

If you'd like to help the LDP with Linux documentation, be sure to consult the [LDP Author Guide](#).

Distributor Web sites

Web sites for the various Linux distributions often provide updated documentation, installation instructions, hardware compatibility/incompatibility statements, and other support such as a knowledge base search tool. Some of these are:

- [Redhat Linux](#) is a large distributor of enterprise Linux products based in the United States.
- [SUSE Linux](#) was founded in Germany and is now owned by Novell.
- [Asianux](#) is an Asian Linux distributor, founded by Haansoft, Inc., Red Flag Software Co., Ltd., and Miracle Linux Corporation.
- [Turbolinux](#) is headquartered in Japan but distributes outside Asia as well.
- [Yellow Dog Linux](#) from Terra Soft Solutions is a distribution for Apple PowerPC®-based processors, and embedded processors based on PowerPC and Cell processors.
- [Linspire](#) is a desktop version of Linux that can be found on some preloaded systems.
- The [Slackware Linux Project](#) by Patrick Volkerding has been around since 1993 and aims to be the most "UNIX®-like" Linux distribution out there.
- [Debian GNU/Linux](#) was started in 1993 as a distribution that was created openly, in the spirit of Linux and GNU.
- [Ubuntu Linux](#) is a relatively new distribution of Linux based on Debian. It focuses on ease-of-use and has related projects, Kubuntu (a version using the KDE desktop), Edubuntu (designed for school environments), and Xubuntu (a lightweight version using the Xfce desktop environment).
- [Gentoo Linux](#) is a distribution that can be automatically optimized and customized for just about any application or need. Packages are distributed as source and built to suit the target environment.
- [Mandriva](#) is a distribution featuring ease-of-use. The company was formed from the merger of several open source pioneers such as Mandrakesoft in France, Conectiva in Brazil, Edge IT in France, and Lycoris in the US.

You can find summary information on and links to a large number of Linux distributions at [DistroWatch.com](#). Tabular information on each distribution tells you what levels of which major packages are included in each version, when the version was released, and much other useful information.

Hardware and software vendors

Many hardware and software vendors have added Linux support to their products in recent years. At their sites, you can find information about which hardware supports Linux, software development tools, released sources, downloads of Linux drivers for specific hardware, and other special Linux projects. For example:

- [IBM and Linux](#)
- [Compaq and Linux](#)
- [SGI and Linux](#)
- [HP and Linux](#)
- [Sun and Linux](#)
- [Sun's StarOffice office productivity suite](#)
- [Oracle and Linux](#)
- [BEA and Linux](#)

Open source projects

Many open source projects have home pages where you will find information on the project. Some projects are sponsored by a foundation such as the Apache Software Foundation. Some examples are:

- [Apache Software Foundation](#) is the home of the Apache Web server, and many, many tools.
- [Eclipse Foundation](#) is focused on providing a vendor-neutral open development platform and application frameworks for building software.
- [OpenOffice.org](#) is multiplatform and multilingual office suite.
- [The GNOME Foundation](#) is the home of the GNOME desktop.
- [The KDE project](#) is the home of KDE, the K Desktop Environment.

A large number of open source projects are hosted on [SourceForge.net](#). These are grouped into categories such as clustering, database, desktop, financial, multimedia, security, and so on. Project pages include links for downloading, bug reporting, user forums, and a link to a project's home page (if available) where you will usually find more information about the project.

Other resources

Another great place for Linux information is the [IBM developerWorks Linux zone](#), the home of this tutorial as well as many other fine articles and tutorials for Linux developers.

Many print magazines also have online sites, and some news sites exist only on the Web. Some examples are:

- [LinuxWorld.com](#)

- [Slashdot](#)
- [freshmeat](#)
- [Linux Magazine](#) (German)
- [Linux+](#) (six languages)

Newsgroups

Internet *newsgroups* are, more accurately, a form of discussion lists. They grew out of bulletin boards, which were an early means of sharing information, usually over a dial-up link. Newsgroups use a protocol called *Network News Transfer Protocol* (NNTP), which is defined in IETF RFC 997 (February 1986).

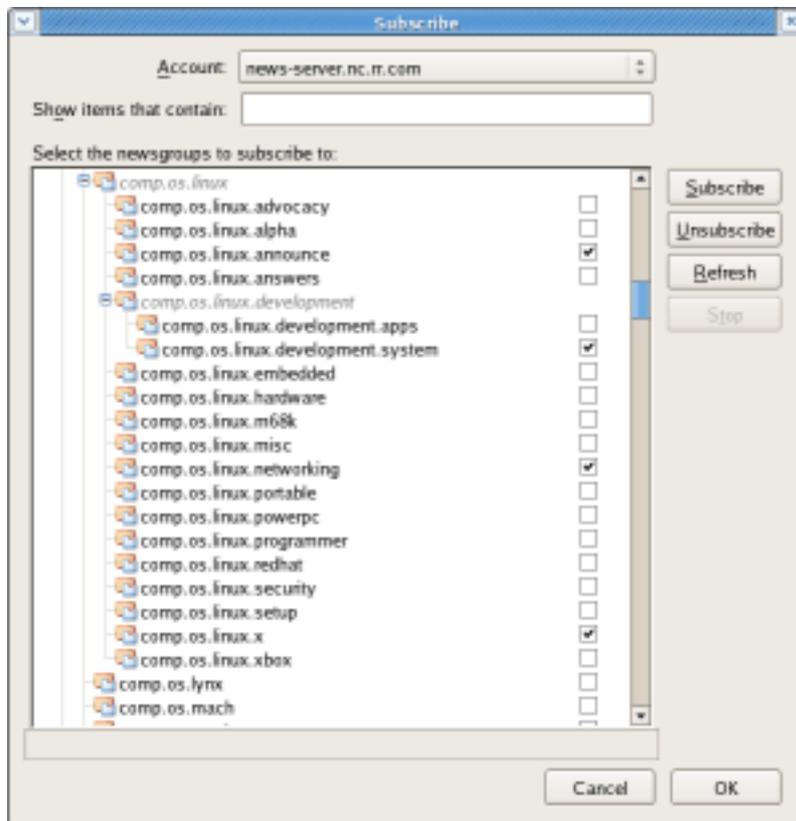
To participate, you use a news reader, which is also known as an NNTP client. There are many Linux clients including *evolution*, *gnus*, *pan*, *slrn*, *thunderbird*, and *tin*. Some of these use a text-mode interface, and some are graphical. The main advantage of a newsgroup is that you take part in the discussion only when you want to, instead of having it continually arrive in your in-box.

Usenet is the largest source of newsgroups. There are several major categories, such as *comp* for computing, *sci* for scientific subjects, and *rec* for recreational topics such as hobbies and games. Computing is further categorized into subjects, and these are still further categorized, so the newsgroups of primary interest to Linux users start with *comp.os.linux*. You can browse a [list on the LDP site](#).

Your Internet Service Provider probably mirrors a range of newsgroups, although news articles may not be retained for a very long period, particularly for active newsgroups. Several newsgroup providers offer a paid service that may provide longer retention, faster access, or a wider selection of newsgroups.

Figure 6 shows the *comp.os.linux* tree as carried on one ISP, using Mozilla's Thunderbird as a newsreader. You *subscribe* to newsgroups, and your newsreader displays only the subscribed groups. Subscribed groups are shown here with a checkmark.

Figure 6. Subscribing to *comp.os.linux.** newsgroups



Newsgroup discussions are often archived. A popular newsgroup for many years was Deja News. When it finally ceased, the newsgroup archives were acquired by Google and reintroduced as [Google Groups](#).

More recently, various Web-based *forums* have arisen. These typically function in a way quite similar to newsgroups, but require only a browser and no configuration. An example is the [Linux tech support forum](#) on IBM's developerWorks Web site where you can ask questions about this series of tutorials along with other topics.

Mailing lists

Mailing lists provide probably the most important point of collaboration for Linux developers. Often projects are developed by contributors who live far apart, possibly even on opposite sides of the globe. Mailing lists overcome time zone differences and thus provide a method for each developer on a project to contact all the others, and to hold group discussions via e-mail. One of the most famous development mailing lists is the [Linux Kernel Mailing List](#).

Mailing lists allow members to send a message to the list, and the list server then broadcasts the message to all members of the group. Individual members do not need to know the e-mail addresses of every member of the group, and they do not need to maintain lists of current members. To avoid a flood of messages from busy

lists, most lists allow a user to request a daily *digest* or single message containing all the list postings for the day.

In addition to development, mailing lists can provide a method for asking questions and receiving answers from knowledgeable developers, or even other users. For example, individual distributions often provide mailing lists for newcomers. You can check your distribution's Web site for information on the mailing lists it provides.

If you took the time to read the LKML FAQ at the link above, you might have noticed that mailing list subscribers often don't take kindly to questions being asked repeatedly. It's always wise to search the archives for a given mailing list before writing your question. Chances are, it will save you time, too. And, speaking of archives, these are often mirrored in multiple sites, so use the closest mirror, typically one in your country or continent.

Section 4. Notifying users

This section covers material for topic 1.108.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Notify the users about current issues related to the system through logon messages

Logon messages

The final short section of this tutorial introduces you to three different logon messages. These have their roots in non-graphical ASCII terminal access to multiuser UNIX® systems and are of diminishing importance today when many workstations are single-user systems, and much access uses a graphical workstation running a desktop such as GNOME or KDE, where these facilities are all but inoperative.

`/etc/issue` and `/etc/issue.net`

The first two of these, `/etc/issue` and `/etc/issue.net`, are displayed on an ASCII terminal that is connected locally (`/etc/issue`) or remotely (`/etc/issue.net`). Listing 16 illustrates these two files as found on a stock Fedora Core 5 system.

Listing 16. `/etc/issue` and `/etc/issue.net`

```
[ian@attic4 ~]$ cat /etc/issue
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m

[ian@attic4 ~]$ cat /etc/issue.net
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m
[ian@attic4 ~]$
```

Notice the control sequences `\r` and `\m`. These allow information such as date or system name to be inserted in the message. The control sequences are shown in Table 4 and are the same as allowed for the `mingetty` command.

Table 4. Control sequences for <code>/etc/issue</code> and <code>/etc/issue.net</code>	
Sequence	Purpose
<code>\d</code>	Inserts the current day according to <code>localtime</code>
<code>\l</code>	Inserts the line on which <code>mingetty</code> is running
<code>\m</code>	Inserts the machine architecture (equivalent to <code>uname -m</code>)
<code>\n</code>	Inserts the machine's network node hostname (equivalent to <code>uname -n</code>)
<code>\o</code>	Inserts the domain name
<code>\r</code>	Inserts the operating system release (equivalent to <code>uname -r</code>)
<code>\t</code>	Inserts the current time according to <code>localtime</code>
<code>\s</code>	Inserts the operating system name
<code>\u</code> or <code>\U</code>	Inserts the current number of users logged in. <code>\U</code> inserts "n users", while <code>\u</code> inserts only "n".
<code>\v</code>	Inserts the operating system version (equivalent to <code>uname -v</code>)

So you can see that the examples in Listing 16 insert the operating system release level and the machine architecture. Connecting via telnet to this system will cause the `/etc/issue.net` message to be displayed before the login prompt as shown in Listing 17.

Listing 17. Telnet connections display `/etc/issue.net`

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
login: ian
Password:
```

If you update `/etc/issue.net` to include a few more control sequences as shown in Listing 18, your logon prompt might look like that in Listing 19.

Listing 18. Updated `/etc/issue.net`

```
[ian@attic4 ~]$ cat /etc/issue.net
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m

\n
Date \d
Time \t
```

Listing 19. Revised telnet logon prompt

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
localhost.localdomain
Date 22:55 on Friday, 15 September 2006
Time 22:55 on Friday, 15 September 2006

login: ian
Password:
```

Notice that on this system `\d` and `\t` produce the same result. As it happens, neither `\u` nor `\U` insert the number of users logged in. This perhaps reflects the fact that these messages have very little use these days. The use of telnet with its passwords flowing in the clear is strongly discouraged. Since a connection using ssh passes the login id and therefore bypasses a login prompt, and since real ASCII terminals remotely connected are rare, the contents of `/etc/issue.net` are rarely seen, and probably not tested too well either.

You will see the contents of `/etc/issue` if you do not use a graphical login. Even if you do, you can usually get a non-graphical login at the system console using `Ctrl-Alt-F1` through `Ctrl-Alt-F6`, with `Ctrl-Alt-F7` returning you to the graphical terminal.

Message of the day

Both `/etc/issue` and `/etc/issue.net` provide user feedback in the form of a logon prompt and could also be used to advise users of issues such as impending outages. However, this is usually done with a *message of the day* or *motd*, which is stored in `/etc/motd`. The contents of `/etc/motd` are displayed after a successful login but just before the login shell is started. Listing 20 shows an example of a motd file, and Listing 21 shows how it and `/etc/issue.net` appear to a user logging in through a telnet session.

Listing 20. Sample message of the day (motd)

```
[ian@attic4 ~]$ cat /etc/motd
PLEASE NOTE!
All systems will shut down this weekend for emergency power testing.
Save your work or lose it.
```

Listing 20. Sample message of the day (motd)

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
localhost.localdomain
Date 22:55 on Friday, 15 September 2006
Time 22:55 on Friday, 15 September 2006

login: ian
Password:
Last login: Fri Sep 15 22:54:18 from 192.168.0.101

PLEASE NOTE!
All systems will shut down this weekend for emergency power testing.
Save your work or lose it.
[ian@attic4 ~]$
```

Again, the motd is really only useful on ASCII terminal sessions. Neither KDE nor GNOME desktops have an easy and satisfactory way of displaying it.

One final notification method that you should know about is the `wall` command, which sends a warning to all logged-in users using text from either a file or stdin. Again, these are not seen by users using the standard GNOME or KDE desktops.

Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, Feb 2006), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- [LPI Linux Certification in a Nutshell, Second Edition](#) (O'Reilly, 2006) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S.

and Ph.D. in computer science from North Carolina State University.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 108: Linux documentation

Junior Level Administration (LPIC-1) topic 108

Skill Level: Intermediate

[Ian Shields](#)

Senior Programmer
IBM developerWorks

20 Sep 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this fourth in a [series of nine tutorials](#), Ian introduces you to Linux® documentation. By the end of this tutorial, you will know how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding

developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	(This tutorial). Learn how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events. See detailed objectives below.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Linux documentation," the fourth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.108.1 Use and manage local system documentation	Weight 4	Find relevant man pages and search man page sections. Find commands and the man pages related to them. Configure access to man sources and the man system. Prepare man pages for printouts. Use the system documentation stored in /usr/share/doc/ and determine what documentation to keep in /usr/share/doc/.
1.108.2 Find Linux documentation on the Internet	Weight 3	Use Linux documentation at sources such as the Linux Documentation Project (LDP), vendor and third-party Web sites, newsgroups, newsgroup archives, and mailing lists.
1.108.5 Notify users of system-related issues	Weight 1	Notify the users about current issues related to the system through logon messages.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial. You will also need a connection to the Internet.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Local documentation

This section covers material for topic 1.108.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Find relevant man pages
- Search man page sections
- Find commands and man pages related to them
- Configure access to man sources and the man system
- Prepare man pages for printouts
- Use the system documentation stored in `/usr/share/doc/` and determine what documentation to keep in `/usr/share/doc/`

Find man pages

The primary (and traditional) source of documentation is the *manual pages*, which you can access using the `man` command. Ideally, you can look up the man page for any command, configuration file, or library routine. In practice, Linux is free software, and some pages haven't been written or are showing their age. Nonetheless, man pages are the first place to look when you need help. Figure 1 illustrates the manual page for the `man` command itself. Use the command `man man` to display this information.

Figure 1. Man page for the man command

```

ian@echidna:~
File Edit View Terminal Go Help
1 man(1) man(1)
2 NAME
  man - format and display the on-line manual pages
  manpath - determine user's search path for man pages
3 SYNOPSIS
  man [-acdfhkkTww] [--path] [-m system] [-p string] [-C config_file]
  [-M pathlist] [-P pager] [-S section_list] [section] name ...
4 DESCRIPTION
  man formats and displays the on-line manual pages.  If you specify sec-
  tion, man only looks in that section of the manual.  name is normally
  the name of the manual page, which is typically the name of a command,
  function, or file.  However, if name contains a slash (/) then man
  interprets it as a file specification, so that you can do man ./foo.5
  or even man /cd/foo/bar.1.gz.

  See below for a description of where man looks for the manual page
  files.
5 OPTIONS
  -C config_file
    Specify the configuration file to use; the default is
    /etc/man.config. (See man.conf(5).)

  -M path
    Specify the list of directories to search for man pages.  Sepa-
    rate the directories with colons.  An empty list is the same as
    not specifying -M at all.  See SEARCH PATH FOR MANUAL PAGES.

  -P pager
    Specify which pager to use.  This option overrides the MANPAGER
    environment variable, which in turn overrides the PAGER vari-
    able.  By default, man uses /usr/bin/less -isr.
:

```

Figure 1 shows some typical items in man pages:

1. A heading with the name of the command followed by its section number in parentheses
2. The name of the command and any related commands that are described on the same man page
3. A synopsis of the options and parameters applicable to the command
4. A short description of the command
5. Detailed information on each of the options

You may find other sections on usage, how to report bugs, author information, and a list of any related commands. For example, the man page for `man` tells us that related commands (and their manual sections) are: `apropos(1)`, `whatis(1)`, `less(1)`, `groff(1)`, and `man.conf(5)`.

Man pages are displayed using a *pager*, which is usually the `less` command on Linux systems. You can set this using the `$PAGER` environment variable, or by using the `-P` or `--pager` option, along with another pager name, on the `man` command. The pager will receive its input on `stdin`, so something like an editor that expects a file to manipulate does not work as a pager.

There are eight common manual page sections. Manual pages are usually installed when you install a package, so if you do not have a package installed, you probably won't have a manual page for it. Similarly, some of your manual sections may be empty or nearly empty. The common manual sections, with some example contents are:

1. User commands (`env`, `ls`, `echo`, `mkdir`, `tty`)
2. System calls or kernel functions (`link`, `sethostname`, `mkdir`)
3. Library routines (`acosh`, `asctime`, `btree`, `locale`, `XML::Parser`)
4. Device-related information (`isdn_audio`, `mouse`, `tty`, `zero`)
5. File format descriptions (`keymaps`, `motd`, `wvdial.conf`)
6. Games (note that many games are now graphical and have graphical help outside the man page system)
7. Miscellaneous (`arp`, `boot`, `regex`, `unix utf8`)
8. System administration (`debugfs`, `fdisk`, `fsck`, `mount`, `renice`, `rpm`)

Other man page sections that you might find include `9` for Linux kernel documentation, `n` for new documentation, `o` for old documentation, and `l` for local documentation.

Some entries appear in multiple sections. Our examples show `mkdir` in sections 1 and 2, and `tty` in sections 1 and 4.

The `info` command

In addition to the standard manual pages, the Free Software Foundation has created a number of *info* files that are processed with the `info` program. These provide extensive navigation facilities including the ability to jump to other sections. Try `man info` or `info info` for more information. Not all commands are documented with `info`, so you will find yourself using both `man` and `info` if you become an `info` user. You can also start at the top of the `info` tree by using `info` without parameters as shown in Listing 1.

Listing 1. The info command

```
File: dir,      Node: Top      This is the top of the INFO tree

This (the Directory node) gives a menu of major topics.
Typing "q" exits, "?" lists all Info commands, "d" returns here,
"h" gives a primer for first-timers,
"mEmacs<Return>" visits the Emacs manual, etc.

In Emacs, you can click mouse button 2 on a menu item or cross reference
to select it.

* Menu:

Utilities
* Bash: (bash).           The GNU Bourne-Again SHell.
* Enscript: (enscript).   GNU Enscript
* Gzip: (gzip).           The gzip command for compressing files.
* ZSH: (zsh).             The Z Shell Manual.

Libraries
* AA-lib: (aalib).        An ASCII-art graphics library
* History: (history).     The GNU history library API
* Libxmi: (libxmi).       The GNU libxmi 2-D rasterization library.
* Readline: (readline).   The GNU readline library API

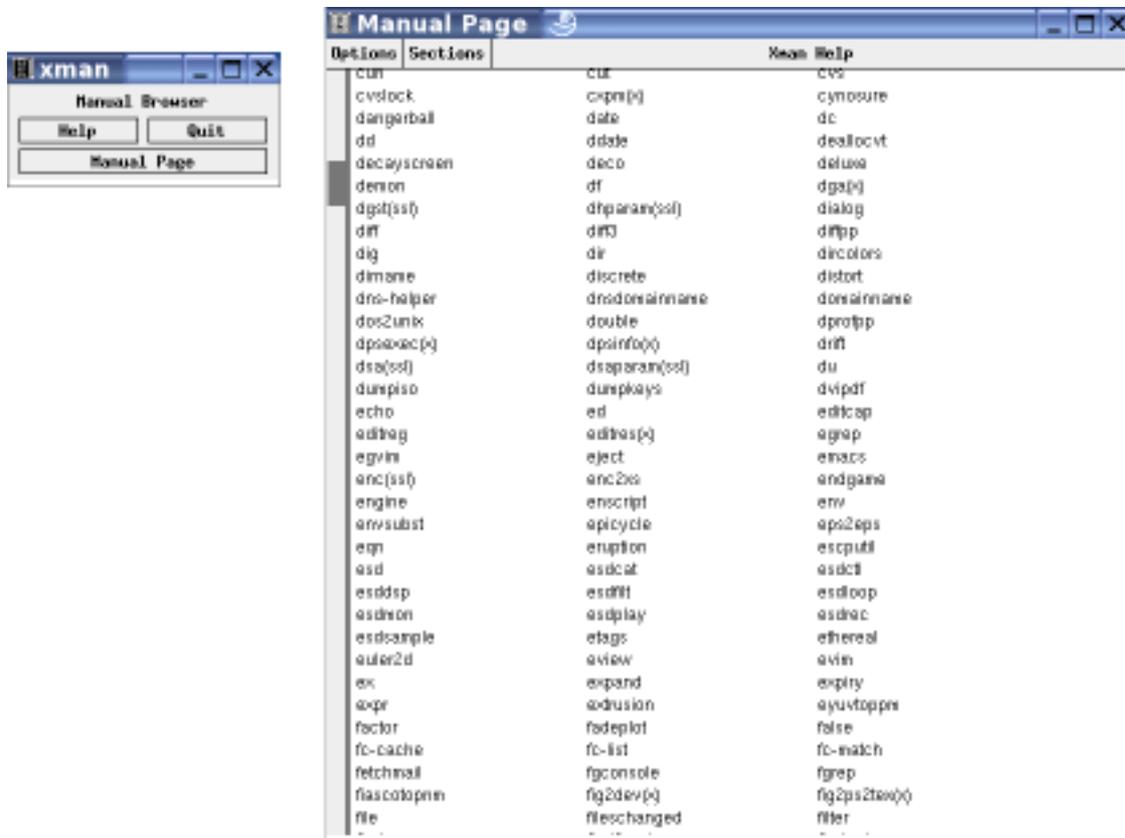
Texinfo documentem
* Info: (info).           Documentation browsing system.
-----Info: (dir)Top, 2104 lines --Top-----
Welcome to Info version 4.6. Type ? for help, m for menu item.
```

Graphical man page interfaces

In addition to the standard `man` command, which uses a terminal window and a pager, your system may also have one or more graphical interfaces to manual pages, such as `xman` (from the XFree86 Project) and `yelp` (the Gnome help browser).

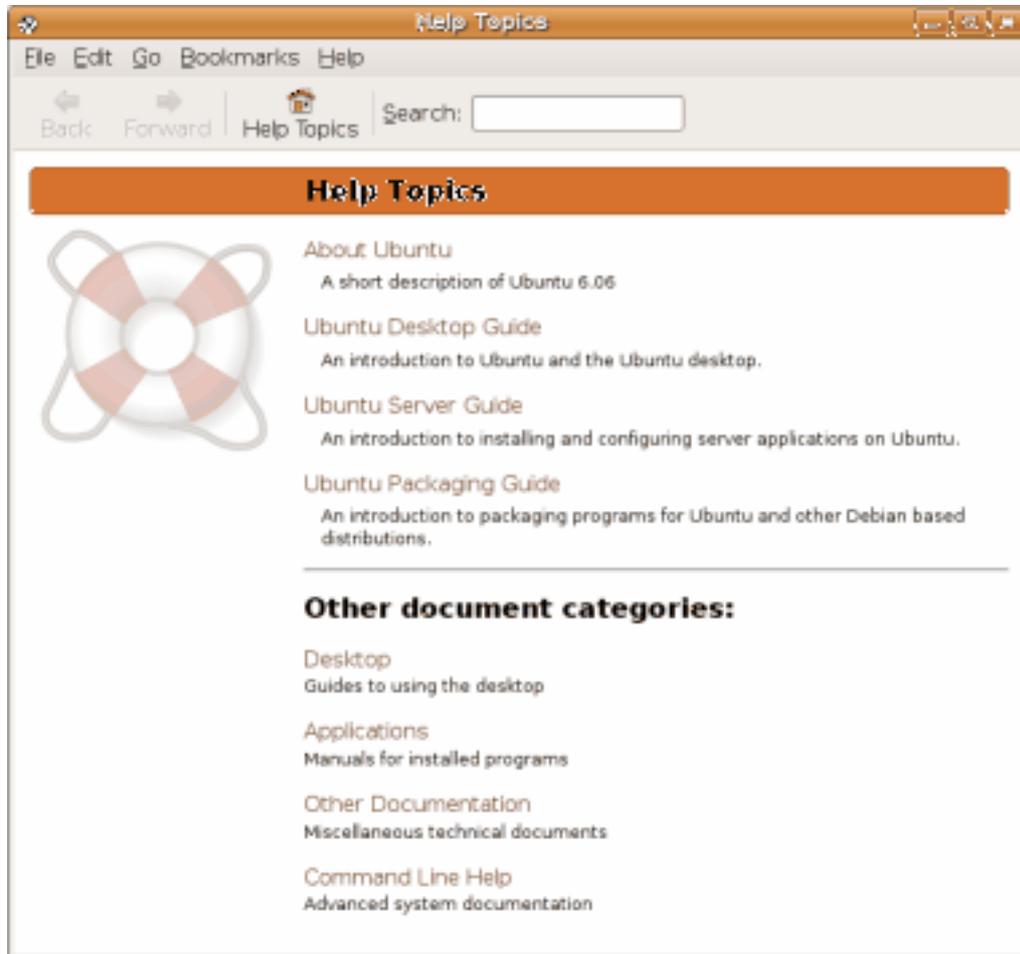
When you start `xman`, you will see a small window with three buttons. Click the **Manual Page** button to open a larger window where you can navigate through manual pages or search for information. Figure 2 shows an example of both windows.

Figure 2. Using xman



The `yelp` browser usually looks somewhat different from system to system. Figure 3 shows an example on Ubuntu 6.06. You can access either the man pages or the info pages using the **Command Line Help** item at the bottom of the display.

Figure 3. Using yelp on Ubuntu



Search man pages

If you know that a topic occurs in a particular section, you can specify the section. For example, `man 4 tty` or `man 2 mkdir`. An alternative is to use the `-a` option to display all applicable manual sections. If you specify `-a`, you will be prompted after quitting the page for each section. You may skip the next page, view it, or quit altogether.

As you saw earlier, some topics exist in more than one section. If you don't want to search through each section, you can use the `-aw` options of `man` to get a list of all available man pages for a topic. Listing 2 shows an example for `printf`. If you were writing a portable shell script, you might be interested in `man 1p printf` to learn about the POSIX version of the `printf` command. On the other hand, if you were writing a C or C++ program, you would be more interested in `man 3 printf`, which would show you the documentation for the `printf`, `fprintf`, `sprintf`, `snprintf`, `vprintf`, `vfprintf`, `vsprintf`, and `vsnprintf` library functions.

Listing 2. Available man pages for printf

```
ian@lyrebird:~> man -aw printf
/usr/share/man/man1/printf.1.gz
/usr/share/man/man1p/printf.1p.gz
/usr/share/man/man3/printf.3.gz
```

The `man` command pages output onto your display using a paging program. On most Linux systems, this is likely to be the `less` program. Another choice might be the older `more` program.

The `less` pager has several commands that help you search for strings within the displayed output. These are similar to `vi` editing commands. Use `man less` to find out more about `/` (search forwards), `?` (search backwards), and `n` (repeat last search), among many other commands.

The `info` command comes from the makers of `emacs`, so the searching commands are more like `emacs` commands. For example, `ctrl-s` searches forwards and `ctrl-r` searches backwards using an incremental search. You can also move around with the arrow keys, follow links (indicated with a star) using the Enter key, and quit using `q`. Use the `--vi-keys` option with `info` if you'd prefer similar key bindings to those used for `man`.

Find commands

Two important commands related to `man` are `whatis` and `apropos`. The `whatis` command searches man pages for the name you give and displays the name information from the appropriate manual pages. The `apropos` command does a keyword search of manual pages and lists ones containing your keyword. Listing 3 illustrates these commands.

Listing 3. Whatis and apropos examples

```
[ian@lyrebird ian]$ whatis man
man                (1) - format and display the on-line manual pages
man                (7) - macros to format man pages
man [manpath]     (1) - format and display the on-line manual pages
man.conf [man]    (5) - configuration data for man
[ian@lyrebird ian]$ whatis mkdir
mkdir              (1) - make directories
mkdir              (2) - create a directory
[ian@lyrebird ian]$ apropos mkdir
mkdir              (1) - make directories
mkdir              (2) - create a directory
mkdirhier          (1x) - makes a directory hierarchy
```

By the way, if you cannot find the manual page for `man.conf`, try running `man man.config` instead, which works on some systems.

The `apropos` command can produce a lot of output, so you may need to use more complex regular expressions rather than simple keywords. Alternatively, you may wish to filter the output through `grep` or another filter to reduce the output to something more of interest. As a practical example, you can use the `e2label` to display or change the label on an `ext2` or `ext3` filesystem, but you have to use another command to change the label on a ReiserFS filesystem. Suppose you run `mount` to display the mounted ReiserFS filesystems as shown in Listing 4.

Listing 4. Mounted ReiserFS filesystems

```
ian@lyrebird:~> mount -t reiserfs
LABEL=SLES9 on / type reiserfs (rw,acl,user_xattr)
```

Now you'd like to know what partition corresponds to the label `SLES9`, but you can't remember the command. Using `apropos label` might get you a couple of dozen responses, which isn't too bad to sift through. But wait. This command must have something to do with a filesystem of a volume. So you try the regular expressions shown in Listing 5.

Listing 5. Using `apropos` with regular expressions

```
ian@lyrebird:~> apropos "label.*file"
e2label (8)          - Change the label on an ext2/ext3 filesystem
ntfslabel (8)       - display/change the label on an ntfs file system
ian@lyrebird:~> apropos "label.*volume"
label.*volume: nothing appropriate.
```

Not exactly what you were looking for. You could try reversing the order of the terms in the regular expressions, or you could try filtering through `grep` or `egrep` as shown in Listing 6.

Listing 6. Filtering the output of `apropos`

```
ian@lyrebird:~> apropos label | grep -E "file|volume"
e2label (8)          - Change the label on an ext2/ext3 filesystem
mlabel (1)          - make an MSDOS volume label
ntfslabel (8)       - display/change the label on an ntfs file system
findfs (8)          - Find a filesystem by label or UUID
```

And there's the command that we need, `findfs`. Using it as shown in Listing 7 shows that the filesystem is on `/dev/hda10` on this particular system.

Listing 7. Finding the device for a mounted filesystem label

```
ian@lyrebird:~> /sbin/findfs LABEL=SLES9
/dev/hda10
```

Note that non-root users will usually have to give the full path to the `findfs` command.

As you can find out in the man page for the `man` command, you can also use `man -k` instead of `apropos` and `man -f` instead of `whatis`. Since these call the `apropos` or `whatis` command under the covers, there is probably little point in so doing.

Configuration

Manual pages may be in many locations on your system. You can determine the current search path using the `manpath` command. If the `MANPATH` environment variable is set, this will be used for searching for manual pages; otherwise, a path will be built automatically using information from a configuration file that we'll discuss in a moment. If the `MANPATH` environment variable is set, the `manpath` command will issue a warning message to this effect before displaying the path.

Listing 8. Displaying your MANPATH

```
[ian@echidna ian]$ manpath
/usr/local/share/man:/usr/share/man:/usr/X11R6/man:/usr/local/man

ian@lyrebird:~> manpath
manpath: warning: $MANPATH set, ignoring /etc/manpath.config
/usr/local/man:/usr/share/man:/usr/X11R6/man:/opt/gnome/share/man
```

Depending on your system, configuration information for the man system is stored in `/etc/man.config` or `/etc/manpath.config`. Older systems use `/etc/man.conf`. A current `man.config` file contains a list of directories (MANPATHs) that will be searched for manual pages, such as those shown in Listing 9.

Listing 9. MANPATH entries from /etc/man.config

```
MANPATH /usr/share/man
MANPATH /usr/man
MANPATH /usr/local/share/man
MANPATH /usr/local/man
MANPATH /usr/X11R6/man
```

In a `manpath.config` file, these entries will be `MANDATORY_MANPATH` entries, rather than `MANPATH` entries.

Besides these entries, you will also find entries giving a mapping between paths where executables may be found, and paths where the corresponding man pages might be, as shown in Listing 10.

Listing 10. MANPATH_MAP entries from /etc/man.config

```
MANPATH_MAP    /bin                /usr/share/man
MANPATH_MAP    /sbin              /usr/share/man
MANPATH_MAP    /usr/bin           /usr/share/man
MANPATH_MAP    /usr/sbin         /usr/share/man
MANPATH_MAP    /usr/local/bin    /usr/local/share/man
```

The `man` command uses a complicated method for searching for man pages, and setting these values will result in less wasted effort when searching for pages.

Another entry in the configuration file defines the search order for manual pages. Recall that the default is to display the first page found, so this ordering is important. Look near the bottom of `man.config` for a `MANSECT` line, or near the bottom of `manpath.config` for a `SECTION` line. Examine the configuration file on your system to see what other things can be configured.

You may have noticed that the `apropos` and `whatis` commands ran quickly. This is because they do not actually search the individual manual pages. Rather, they use a database created by the `makewhatis` command. This is usually run by the system either daily or weekly as a cron job.

Listing 11. Running `makewhatis`

```
[root@echidna root]# makewhatis
```

The command completes normally without any output message, but the `whatis` database is refreshed. This is usually stored in a location such as `/var/cache/man/whatis`. Note that some SUSE systems do not use the `whatis` database and therefore do not have a `makewhatis` command.

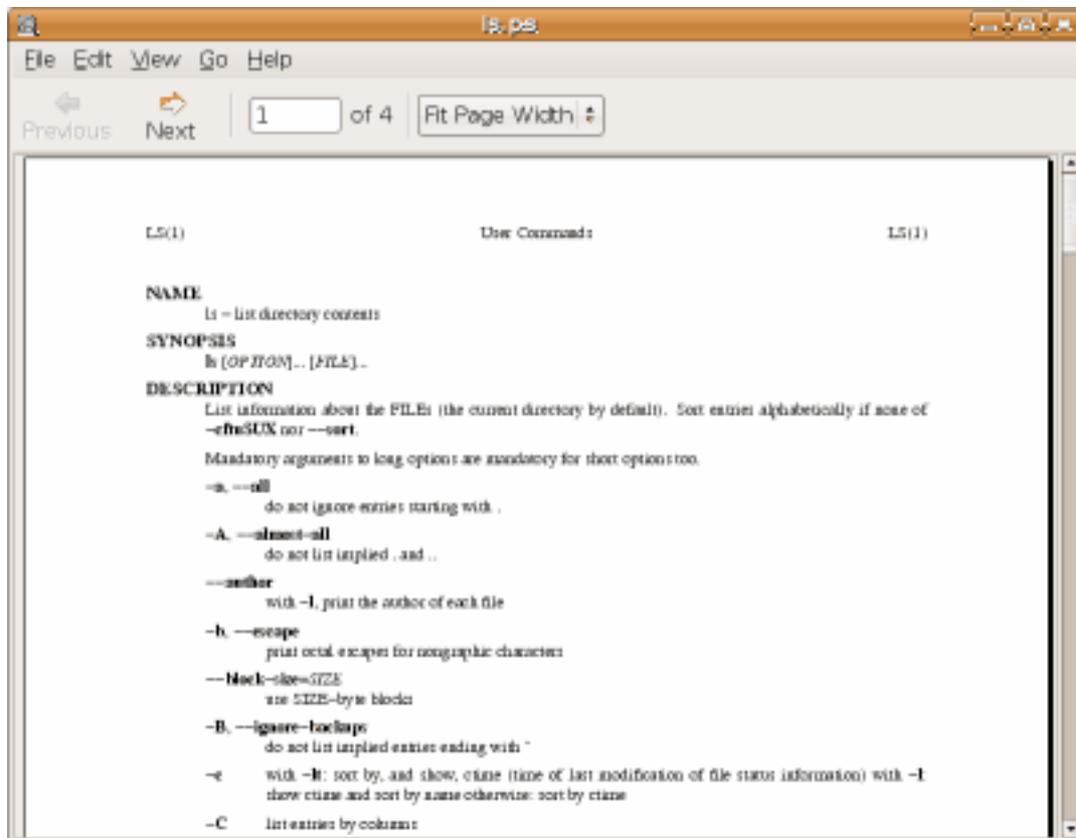
Printing man pages

If you wish to print the page, specify the `-t` option to format the page for printing using the `groff` or `troff` program. This will format the page for the default printer and send the output to `stdout`. Listing 12 shows how to format the man page for the `ls` command and save the output in a file, `ls.ps`. Figure 4 shows the formatted output.

Listing 12. Formatting the `ls` manpage for printing

```
ian@pinguino:~$ man -t ls > ls.ps
```

Figure 4. Formatted `ls` man page



If you need to format the page for a different device type, use the `-T` options with a device type, such as `dvi` or `ps`. See the man page for `man` for additional information.

/usr/share/doc/

In addition to the manual pages and info pages that you have already seen, your Linux system probably includes a lot more documentation. The customary place to store this is in `/usr/share/doc`, or `/usr/doc` on older systems. This additional documentation may be in any of several formats, such as text, PDF, PostScript, or HTML.

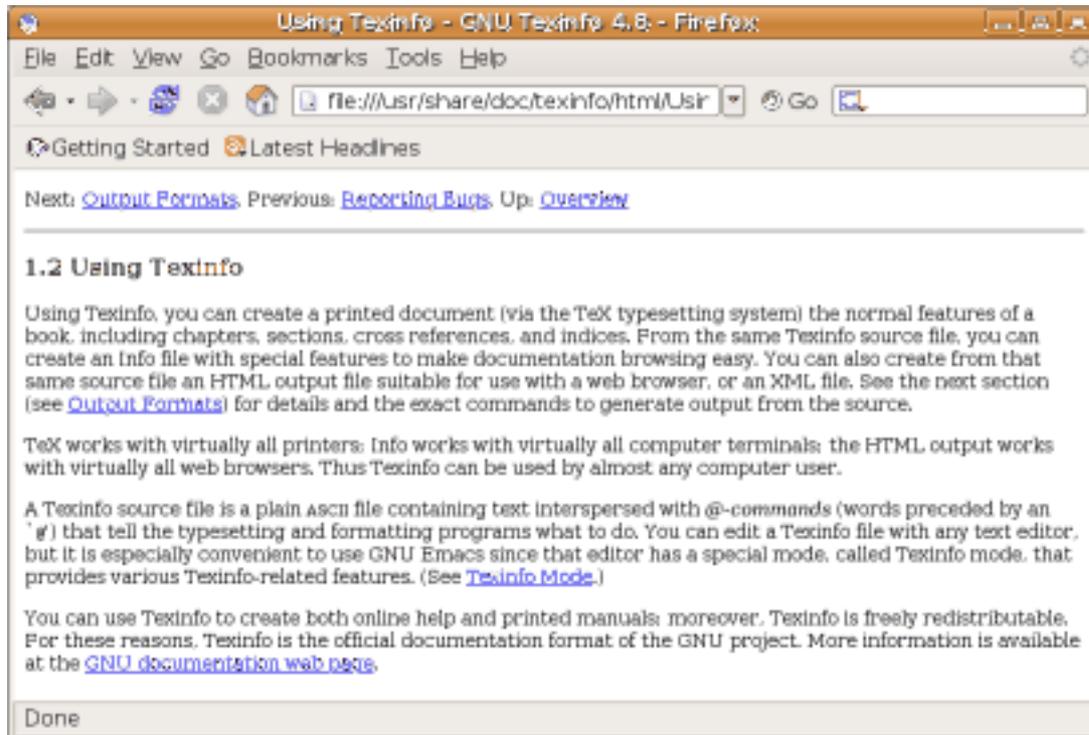
Searching through this documentation can often reveal gems that aren't available as man pages or info pages, such as tutorials or additional technical documentation. As Listing 13 shows, there can be a large number of files in `/usr/share/doc`, so you have plenty of reading resources.

Listing 13. Files in /usr/share/doc

```
ian@pinguino:~$ find /usr/share/doc -type f | wc -l
10144
```

Figure 5 shows an example of the HTML help for the Texinfo system that is used for the `info` command that you saw earlier.

Figure 5. Texinfo HTML help from `/usr/share/doc`



Sometimes, a man page will direct you to another source for documentation. For example, the man page for the `pngtopnm` command is shown in Listing 14. It directs you to a local copy in HTML format at `/usr/share/doc/packages/netpbm/doc/pngtopnm.html`, or to an online version if you do not have the local copy.

Listing 14. Pointer man page for `pngtopnm`

```
pngtopnm(1)           Netpbm pointer man pages           pngtopnm(1)

pngtopnm is part of the Netpbm package.  Netpbm documentation is
kept in HTML format.

Please refer to
<http://netpbm.sourceforge.net/doc/pngtopnm.html>.

If that doesn't work, also try <http://netpbm.sourceforge.net>
and emailing Bryan Henderson, bryanh@giraffe-data.com.

Local copy of the page is here:
/usr/share/doc/packages/netpbm/doc/pngtopnm.html
```

Other command help

Finally, if you can't find help for a command, try running the command with the `--help`, `--h`, or `--?` option. This may provide the command's help, or it may tell you how to get the help you need. Listing 15 shows an example for the `kdesu` command, which is usually present on systems with a KDE desktop.

Listing 15. Getting help for `kdesu` command

```
ian@lyrebird:~> man kdesu
No manual entry for kdesu
ian@lyrebird:~> kdesu --help
Usage: kdesu [Qt-options] [KDE-options] command

Runs a program with elevated privileges.

Generic options:
  --help                Show help about options
  --help-qt             Show Qt specific options
  --help-kde           Show KDE specific options
  --help-all           Show all options
  --author              Show author information
  -v, --version        Show version information
  --license             Show license information
  --                   End of options

Arguments:
  command               Specifies the command to run.

Options:
  -c <command>        Specifies the command to run. []
```

The next section covers online resources for help with Linux.

Section 3. Internet documentation

This section covers material for topic 1.108.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to find:

- Online documentation
- Newsgroups
- Mailing lists

Online documentation

In addition to the documentaiton on your system, there are many online sources of

documentation and help.

The Linux Documentation Project

The [Linux Documentation Project](#) is volunteer effort that is putting together the complete set of free Linux documentation. This project exists to consolidate various pieces of Linux documentation into a location that is easy to search and use.

The LDP is made up of the following areas:

HOWTOs

are subject-specific help, such as the [Linux IPv6 HOWTO](#).

Guides

are longer, in-depth books, such as [Introduction to Linux - A Hands on Guide](#).

FAQs

are Frequently Asked Questions, such as the [Linux Documentation Project \(LDP\) FAQ](#).

man pages

are help on individual commands, as you used in the previous section of this tutorial.

Linux Gazette

is an online magazine, currently available in English, French, German, Indonesian, Italian, Portuguese, Russian, and Spanish.

The examples here take you to the multiple-page HTML versions of the documentation. You will find most articles come in several formats, including single-page HTML, PDF, or plain text, among others.

The LDP also has links to [information in languages other than English](#).

The LDP site is well laid out with excellent navigation. If you aren't sure which section to peruse, you can take advantage of the search box, which helps you find things by topic.

If you'd like to help the LDP with Linux documentation, be sure to consult the [LDP Author Guide](#).

Distributor Web sites

Web sites for the various Linux distributions often provide updated documentation, installation instructions, hardware compatibility/incompatibility statements, and other support such as a knowledge base search tool. Some of these are:

- [Redhat Linux](#) is a large distributor of enterprise Linux products based in the United States.
- [SUSE Linux](#) was founded in Germany and is now owned by Novell.
- [Asianux](#) is an Asian Linux distributor, founded by Haansoft, Inc., Red Flag Software Co., Ltd., and Miracle Linux Corporation.
- [Turbolinux](#) is headquartered in Japan but distributes outside Asia as well.
- [Yellow Dog Linux](#) from Terra Soft Solutions is a distribution for Apple PowerPC®-based processors, and embedded processors based on PowerPC and Cell processors.
- [Linspire](#) is a desktop version of Linux that can be found on some preloaded systems.
- The [Slackware Linux Project](#) by Patrick Volkerding has been around since 1993 and aims to be the most "UNIX®-like" Linux distribution out there.
- [Debian GNU/Linux](#) was started in 1993 as a distribution that was created openly, in the spirit of Linux and GNU.
- [Ubuntu Linux](#) is a relatively new distribution of Linux based on Debian. It focuses on ease-of-use and has related projects, Kubuntu (a version using the KDE desktop), Edubuntu (designed for school environments), and Xubuntu (a lightweight version using the Xfce desktop environment).
- [Gentoo Linux](#) is a distribution that can be automatically optimized and customized for just about any application or need. Packages are distributed as source and built to suit the target environment.
- [Mandriva](#) is a distribution featuring ease-of-use. The company was formed from the merger of several open source pioneers such as Mandrakesoft in France, Conectiva in Brazil, Edge IT in France, and Lycoris in the US.

You can find summary information on and links to a large number of Linux distributions at [DistroWatch.com](#). Tabular information on each distribution tells you what levels of which major packages are included in each version, when the version was released, and much other useful information.

Hardware and software vendors

Many hardware and software vendors have added Linux support to their products in recent years. At their sites, you can find information about which hardware supports Linux, software development tools, released sources, downloads of Linux drivers for specific hardware, and other special Linux projects. For example:

- [IBM and Linux](#)
- [Compaq and Linux](#)
- [SGI and Linux](#)
- [HP and Linux](#)
- [Sun and Linux](#)
- [Sun's StarOffice office productivity suite](#)
- [Oracle and Linux](#)
- [BEA and Linux](#)

Open source projects

Many open source projects have home pages where you will find information on the project. Some projects are sponsored by a foundation such as the Apache Software Foundation. Some examples are:

- [Apache Software Foundation](#) is the home of the Apache Web server, and many, many tools.
- [Eclipse Foundation](#) is focused on providing a vendor-neutral open development platform and application frameworks for building software.
- [OpenOffice.org](#) is multiplatform and multilingual office suite.
- [The GNOME Foundation](#) is the home of the GNOME desktop.
- [The KDE project](#) is the home of KDE, the K Desktop Environment.

A large number of open source projects are hosted on [SourceForge.net](#). These are grouped into categories such as clustering, database, desktop, financial, multimedia, security, and so on. Project pages include links for downloading, bug reporting, user forums, and a link to a project's home page (if available) where you will usually find more information about the project.

Other resources

Another great place for Linux information is the [IBM developerWorks Linux zone](#), the home of this tutorial as well as many other fine articles and tutorials for Linux developers.

Many print magazines also have online sites, and some news sites exist only on the Web. Some examples are:

- [LinuxWorld.com](#)

- [Slashdot](#)
- [freshmeat](#)
- [Linux Magazine](#) (German)
- [Linux+](#) (six languages)

Newsgroups

Internet *newsgroups* are, more accurately, a form of discussion lists. They grew out of bulletin boards, which were an early means of sharing information, usually over a dial-up link. Newsgroups use a protocol called *Network News Transfer Protocol* (NNTP), which is defined in IETF RFC 997 (February 1986).

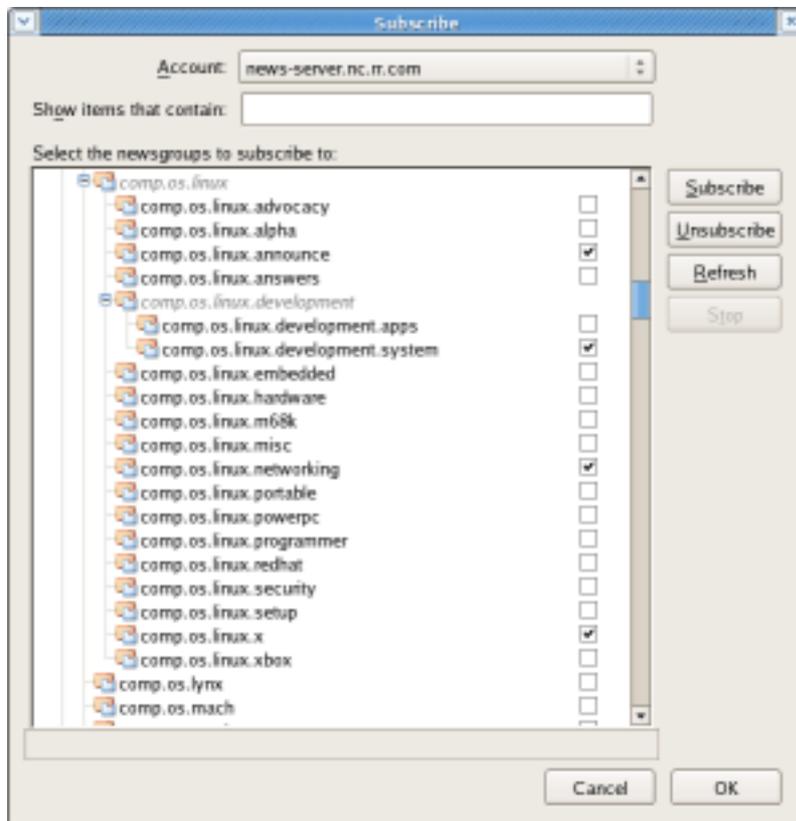
To participate, you use a news reader, which is also known as an NNTP client. There are many Linux clients including `evolution`, `gnus`, `pan`, `slrn`, `thunderbird`, and `tin`. Some of these use a text-mode interface, and some are graphical. The main advantage of a newsgroup is that you take part in the discussion only when you want to, instead of having it continually arrive in your in-box.

Usenet is the largest source of newsgroups. There are several major categories, such as *comp* for computing, *sci* for scientific subjects, and *rec* for recreational topics such as hobbies and games. Computing is further categorized into subjects, and these are still further categorized, so the newsgroups of primary interest to Linux users start with *comp.os.linux*. You can browse a [list on the LDP site](#).

Your Internet Service Provider probably mirrors a range of newsgroups, although news articles may not be retained for a very long period, particularly for active newsgroups. Several newsgroup providers offer a paid service that may provide longer retention, faster access, or a wider selection of newsgroups.

Figure 6 shows the *comp.os.linux* tree as carried on one ISP, using Mozilla's Thunderbird as a newsreader. You *subscribe* to newsgroups, and your newsreader displays only the subscribed groups. Subscribed groups are shown here with a checkmark.

Figure 6. Subscribing to *comp.os.linux.** newsgroups



Newsgroup discussions are often archived. A popular newsgroup for many years was Deja News. When it finally ceased, the newsgroup archives were acquired by Google and reintroduced as [Google Groups](#).

More recently, various Web-based *forums* have arisen. These typically function in a way quite similar to newsgroups, but require only a browser and no configuration. An example is the [Linux tech support forum](#) on IBM's developerWorks Web site where you can ask questions about this series of tutorials along with other topics.

Mailing lists

Mailing lists provide probably the most important point of collaboration for Linux developers. Often projects are developed by contributors who live far apart, possibly even on opposite sides of the globe. Mailing lists overcome time zone differences and thus provide a method for each developer on a project to contact all the others, and to hold group discussions via e-mail. One of the most famous development mailing lists is the [Linux Kernel Mailing List](#).

Mailing lists allow members to send a message to the list, and the list server then broadcasts the message to all members of the group. Individual members do not need to know the e-mail addresses of every member of the group, and they do not need to maintain lists of current members. To avoid a flood of messages from busy

lists, most lists allow a user to request a daily *digest* or single message containing all the list postings for the day.

In addition to development, mailing lists can provide a method for asking questions and receiving answers from knowledgeable developers, or even other users. For example, individual distributions often provide mailing lists for newcomers. You can check your distribution's Web site for information on the mailing lists it provides.

If you took the time to read the LKML FAQ at the link above, you might have noticed that mailing list subscribers often don't take kindly to questions being asked repeatedly. It's always wise to search the archives for a given mailing list before writing your question. Chances are, it will save you time, too. And, speaking of archives, these are often mirrored in multiple sites, so use the closest mirror, typically one in your country or continent.

Section 4. Notifying users

This section covers material for topic 1.108.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Notify the users about current issues related to the system through logon messages

Logon messages

The final short section of this tutorial introduces you to three different logon messages. These have their roots in non-graphical ASCII terminal access to multiuser UNIX® systems and are of diminishing importance today when many workstations are single-user systems, and much access uses a graphical workstation running a desktop such as GNOME or KDE, where these facilities are all but inoperative.

`/etc/issue` and `/etc/issue.net`

The first two of these, `/etc/issue` and `/etc/issue.net`, are displayed on an ASCII terminal that is connected locally (`/etc/issue`) or remotely (`/etc/issue.net`). Listing 16 illustrates these two files as found on a stock Fedora Core 5 system.

Listing 16. `/etc/issue` and `/etc/issue.net`

```
[ian@attic4 ~]$ cat /etc/issue
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m

[ian@attic4 ~]$ cat /etc/issue.net
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m
[ian@attic4 ~]$
```

Notice the control sequences `\r` and `\m`. These allow information such as date or system name to be inserted in the message. The control sequences are shown in Table 4 and are the same as allowed for the `mingetty` command.

Table 4. Control sequences for <code>/etc/issue</code> and <code>/etc/issue.net</code>	
Sequence	Purpose
<code>\d</code>	Inserts the current day according to <code>localtime</code>
<code>\l</code>	Inserts the line on which <code>mingetty</code> is running
<code>\m</code>	Inserts the machine architecture (equivalent to <code>uname -m</code>)
<code>\n</code>	Inserts the machine's network node hostname (equivalent to <code>uname -n</code>)
<code>\o</code>	Inserts the domain name
<code>\r</code>	Inserts the operating system release (equivalent to <code>uname -r</code>)
<code>\t</code>	Inserts the current time according to <code>localtime</code>
<code>\s</code>	Inserts the operating system name
<code>\u</code> or <code>\U</code>	Inserts the current number of users logged in. <code>\U</code> inserts "n users", while <code>\u</code> inserts only "n".
<code>\v</code>	Inserts the operating system version (equivalent to <code>uname -v</code>)

So you can see that the examples in Listing 16 insert the operating system release level and the machine architecture. Connecting via telnet to this system will cause the `/etc/issue.net` message to be displayed before the login prompt as shown in Listing 17.

Listing 17. Telnet connections display `/etc/issue.net`

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
login: ian
Password:
```

If you update `/etc/issue.net` to include a few more control sequences as shown in Listing 18, your logon prompt might look like that in Listing 19.

Listing 18. Updated `/etc/issue.net`

```
[ian@attic4 ~]$ cat /etc/issue.net
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m

\n
Date \d
Time \t
```

Listing 19. Revised telnet logon prompt

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
localhost.localdomain
Date 22:55 on Friday, 15 September 2006
Time 22:55 on Friday, 15 September 2006

login: ian
Password:
```

Notice that on this system `\d` and `\t` produce the same result. As it happens, neither `\u` nor `\U` insert the number of users logged in. This perhaps reflects the fact that these messages have very little use these days. The use of telnet with its passwords flowing in the clear is strongly discouraged. Since a connection using ssh passes the login id and therefore bypasses a login prompt, and since real ASCII terminals remotely connected are rare, the contents of `/etc/issue.net` are rarely seen, and probably not tested too well either.

You will see the contents of `/etc/issue` if you do not use a graphical login. Even if you do, you can usually get a non-graphical login at the system console using Ctrl-Alt-F1 through Ctrl-Alt-F6, with Ctrl-Alt-F7 returning you to the graphical terminal.

Message of the day

Both `/etc/issue` and `/etc/issue.net` provide user feedback in the form of a logon prompt and could also be used to advise users of issues such as impending outages. However, this is usually done with a *message of the day* or *motd*, which is stored in `/etc/motd`. The contents of `/etc/motd` are displayed after a successful login but just before the login shell is started. Listing 20 shows an example of a motd file, and Listing 21 shows how it and `/etc/issue.net` appear to a user logging in through a telnet session.

Listing 20. Sample message of the day (motd)

```
[ian@attic4 ~]$ cat /etc/motd
PLEASE NOTE!
All systems will shut down this weekend for emergency power testing.
Save your work or lose it.
```

Listing 20. Sample message of the day (motd)

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
localhost.localdomain
Date 22:55 on Friday, 15 September 2006
Time 22:55 on Friday, 15 September 2006

login: ian
Password:
Last login: Fri Sep 15 22:54:18 from 192.168.0.101

PLEASE NOTE!
All systems will shut down this weekend for emergency power testing.
Save your work or lose it.
[ian@attic4 ~]$
```

Again, the motd is really only useful on ASCII terminal sessions. Neither KDE nor GNOME desktops have an easy and satisfactory way of displaying it.

One final notification method that you should know about is the `wall` command, which sends a warning to all logged-in users using text from either a file or stdin. Again, these are not seen by users using the standard GNOME or KDE desktops.

Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, Feb 2006), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- [LPI Linux Certification in a Nutshell, Second Edition](#) (O'Reilly, 2006) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S.

and Ph.D. in computer science from North Carolina State University.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 108: Linux documentation

Junior Level Administration (LPIC-1) topic 108

Skill Level: Intermediate

[Ian Shields](#)

Senior Programmer
IBM developerWorks

20 Sep 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this fourth in a [series of nine tutorials](#), Ian introduces you to Linux® documentation. By the end of this tutorial, you will know how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding

developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	(This tutorial). Learn how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events. See detailed objectives below.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Linux documentation," the fourth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to use and manage local documentation, find documentation on the Internet, and use automated logon messages to notify users of system events.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.108.1 Use and manage local system documentation	Weight 4	Find relevant man pages and search man page sections. Find commands and the man pages related to them. Configure access to man sources and the man system. Prepare man pages for printouts. Use the system documentation stored in /usr/share/doc/ and determine what documentation to keep in /usr/share/doc/.
1.108.2 Find Linux documentation on the Internet	Weight 3	Use Linux documentation at sources such as the Linux Documentation Project (LDP), vendor and third-party Web sites, newsgroups, newsgroup archives, and mailing lists.
1.108.5 Notify users of system-related issues	Weight 1	Notify the users about current issues related to the system through logon messages.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial. You will also need a connection to the Internet.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Local documentation

This section covers material for topic 1.108.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Find relevant man pages
- Search man page sections
- Find commands and man pages related to them
- Configure access to man sources and the man system
- Prepare man pages for printouts
- Use the system documentation stored in `/usr/share/doc/` and determine what documentation to keep in `/usr/share/doc/`

Find man pages

The primary (and traditional) source of documentation is the *manual pages*, which you can access using the `man` command. Ideally, you can look up the man page for any command, configuration file, or library routine. In practice, Linux is free software, and some pages haven't been written or are showing their age. Nonetheless, man pages are the first place to look when you need help. Figure 1 illustrates the manual page for the `man` command itself. Use the command `man man` to display this information.

Figure 1. Man page for the man command

```

ian@echidna:~
File Edit View Terminal Go Help
1 man(1) man(1)
2 NAME
  man - format and display the on-line manual pages
  manpath - determine user's search path for man pages
3 SYNOPSIS
  man [-acdfhkkTww] [--path] [-m system] [-p string] [-C config_file]
  [-M pathlist] [-P pager] [-S section_list] [section] name ...
4 DESCRIPTION
  man formats and displays the on-line manual pages.  If you specify sec-
  tion, man only looks in that section of the manual.  name is normally
  the name of the manual page, which is typically the name of a command,
  function, or file.  However, if name contains a slash (/) then man
  interprets it as a file specification, so that you can do man ./foo.5
  or even man /cd/foo/bar.1.gz.

  See below for a description of where man looks for the manual page
  files.
5 OPTIONS
  -C config_file
    Specify the configuration file to use; the default is
    /etc/man.config. (See man.conf(5).)

  -M path
    Specify the list of directories to search for man pages.  Sepa-
    rate the directories with colons.  An empty list is the same as
    not specifying -M at all.  See SEARCH PATH FOR MANUAL PAGES.

  -P pager
    Specify which pager to use.  This option overrides the MANPAGER
    environment variable, which in turn overrides the PAGER vari-
    able.  By default, man uses /usr/bin/less -isr.
:

```

Figure 1 shows some typical items in man pages:

1. A heading with the name of the command followed by its section number in parentheses
2. The name of the command and any related commands that are described on the same man page
3. A synopsis of the options and parameters applicable to the command
4. A short description of the command
5. Detailed information on each of the options

You may find other sections on usage, how to report bugs, author information, and a list of any related commands. For example, the man page for `man` tells us that related commands (and their manual sections) are: `apropos(1)`, `whatis(1)`, `less(1)`, `groff(1)`, and `man.conf(5)`.

Man pages are displayed using a *pager*, which is usually the `less` command on Linux systems. You can set this using the `$PAGER` environment variable, or by using the `-P` or `--pager` option, along with another pager name, on the `man` command. The pager will receive its input on `stdin`, so something like an editor that expects a file to manipulate does not work as a pager.

There are eight common manual page sections. Manual pages are usually installed when you install a package, so if you do not have a package installed, you probably won't have a manual page for it. Similarly, some of your manual sections may be empty or nearly empty. The common manual sections, with some example contents are:

1. User commands (`env`, `ls`, `echo`, `mkdir`, `tty`)
2. System calls or kernel functions (`link`, `sethostname`, `mkdir`)
3. Library routines (`acosh`, `asctime`, `btree`, `locale`, `XML::Parser`)
4. Device-related information (`isdn_audio`, `mouse`, `tty`, `zero`)
5. File format descriptions (`keymaps`, `motd`, `wvdial.conf`)
6. Games (note that many games are now graphical and have graphical help outside the man page system)
7. Miscellaneous (`arp`, `boot`, `regex`, `unix utf8`)
8. System administration (`debugfs`, `fdisk`, `fsck`, `mount`, `renice`, `rpm`)

Other man page sections that you might find include `9` for Linux kernel documentation, `n` for new documentation, `o` for old documentation, and `l` for local documentation.

Some entries appear in multiple sections. Our examples show `mkdir` in sections 1 and 2, and `tty` in sections 1 and 4.

The `info` command

In addition to the standard manual pages, the Free Software Foundation has created a number of *info* files that are processed with the `info` program. These provide extensive navigation facilities including the ability to jump to other sections. Try `man info` or `info info` for more information. Not all commands are documented with `info`, so you will find yourself using both `man` and `info` if you become an `info` user. You can also start at the top of the `info` tree by using `info` without parameters as shown in Listing 1.

Listing 1. The info command

```
File: dir,      Node: Top      This is the top of the INFO tree

This (the Directory node) gives a menu of major topics.
Typing "q" exits, "?" lists all Info commands, "d" returns here,
"h" gives a primer for first-timers,
"mEmacs<Return>" visits the Emacs manual, etc.

In Emacs, you can click mouse button 2 on a menu item or cross reference
to select it.

* Menu:

Utilities
* Bash: (bash).           The GNU Bourne-Again SHell.
* Enscript: (enscript).   GNU Enscript
* Gzip: (gzip).           The gzip command for compressing files.
* ZSH: (zsh).             The Z Shell Manual.

Libraries
* AA-lib: (aalib).        An ASCII-art graphics library
* History: (history).    The GNU history library API
* Libxmi: (libxmi).      The GNU libxmi 2-D rasterization library.
* Readline: (readline).  The GNU readline library API

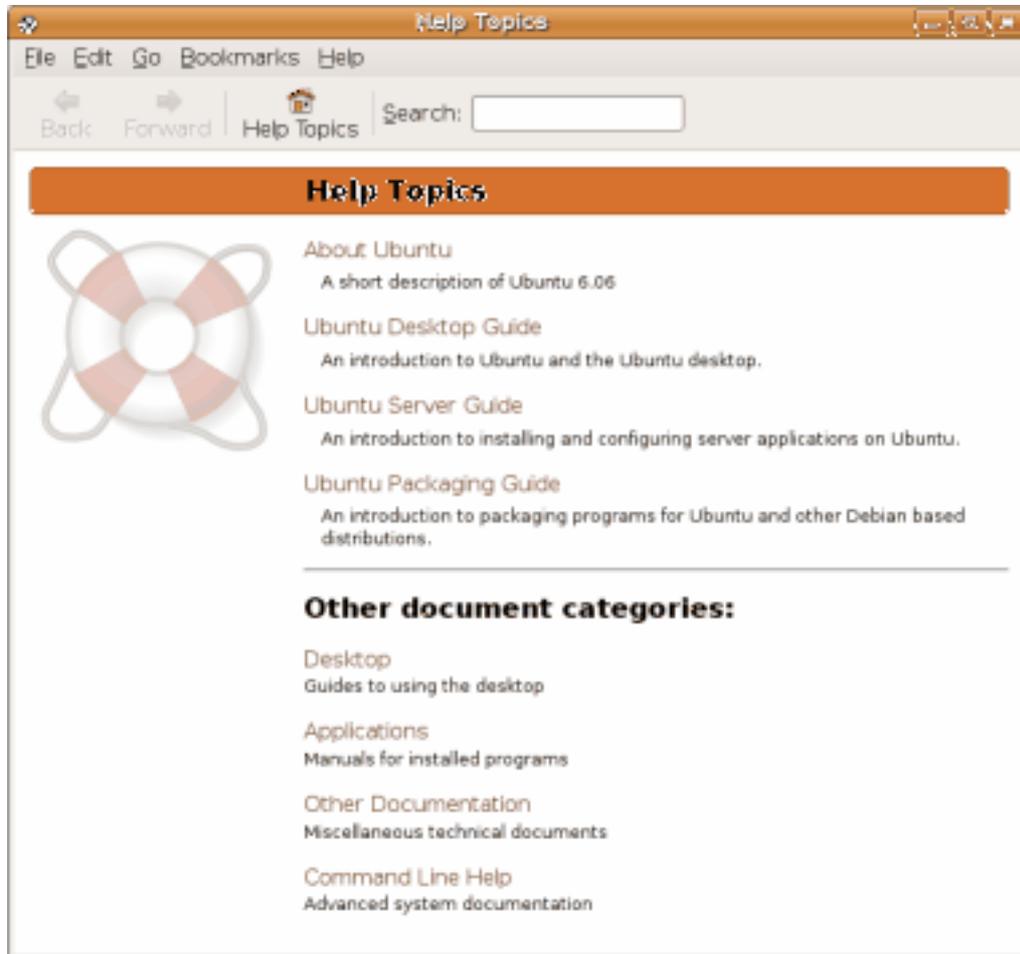
Texinfo document system
* Info: (info).          Documentation browsing system.
-----Info: (dir)Top, 2104 lines --Top-----
Welcome to Info version 4.6. Type ? for help, m for menu item.
```

Graphical man page interfaces

In addition to the standard `man` command, which uses a terminal window and a pager, your system may also have one or more graphical interfaces to manual pages, such as `xman` (from the XFree86 Project) and `yelp` (the Gnome help browser).

When you start `xman`, you will see a small window with three buttons. Click the **Manual Page** button to open a larger window where you can navigate through manual pages or search for information. Figure 2 shows an example of both windows.

Figure 2. Using `xman`



Search man pages

If you know that a topic occurs in a particular section, you can specify the section. For example, `man 4 tty` or `man 2 mkdir`. An alternative is to use the `-a` option to display all applicable manual sections. If you specify `-a`, you will be prompted after quitting the page for each section. You may skip the next page, view it, or quit altogether.

As you saw earlier, some topics exist in more than one section. If you don't want to search through each section, you can use the `-aw` options of `man` to get a list of all available man pages for a topic. Listing 2 shows an example for `printf`. If you were writing a portable shell script, you might be interested in `man 1p printf` to learn about the POSIX version of the `printf` command. On the other hand, if you were writing a C or C++ program, you would be more interested in `man 3 printf`, which would show you the documentation for the `printf`, `fprintf`, `sprintf`, `snprintf`, `vprintf`, `vfprintf`, `vsprintf`, and `vsnprintf` library functions.

Listing 2. Available man pages for printf

```
ian@lyrebird:~> man -aw printf
/usr/share/man/man1/printf.1.gz
/usr/share/man/man1p/printf.1p.gz
/usr/share/man/man3/printf.3.gz
```

The `man` command pages output onto your display using a paging program. On most Linux systems, this is likely to be the `less` program. Another choice might be the older `more` program.

The `less` pager has several commands that help you search for strings within the displayed output. These are similar to `vi` editing commands. Use `man less` to find out more about `/` (search forwards), `?` (search backwards), and `n` (repeat last search), among many other commands.

The `info` command comes from the makers of `emacs`, so the searching commands are more like `emacs` commands. For example, `ctrl-s` searches forwards and `ctrl-r` searches backwards using an incremental search. You can also move around with the arrow keys, follow links (indicated with a star) using the Enter key, and quit using `q`. Use the `--vi-keys` option with `info` if you'd prefer similar key bindings to those used for `man`.

Find commands

Two important commands related to `man` are `whatis` and `apropos`. The `whatis` command searches man pages for the name you give and displays the name information from the appropriate manual pages. The `apropos` command does a keyword search of manual pages and lists ones containing your keyword. Listing 3 illustrates these commands.

Listing 3. Whatis and apropos examples

```
[ian@lyrebird ian]$ whatis man
man          (1) - format and display the on-line manual pages
man          (7) - macros to format man pages
man [manpath] (1) - format and display the on-line manual pages
man.conf [man] (5) - configuration data for man
[ian@lyrebird ian]$ whatis mkdir
mkdir       (1) - make directories
mkdir       (2) - create a directory
[ian@lyrebird ian]$ apropos mkdir
mkdir       (1) - make directories
mkdir       (2) - create a directory
mkdirhier   (1x) - makes a directory hierarchy
```

By the way, if you cannot find the manual page for `man.conf`, try running `man man.config` instead, which works on some systems.

The `apropos` command can produce a lot of output, so you may need to use more complex regular expressions rather than simple keywords. Alternatively, you may wish to filter the output through `grep` or another filter to reduce the output to something more of interest. As a practical example, you can use the `e2label` to display or change the label on an `ext2` or `ext3` filesystem, but you have to use another command to change the label on a ReiserFS filesystem. Suppose you run `mount` to display the mounted ReiserFS filesystems as shown in Listing 4.

Listing 4. Mounted ReiserFS filesystems

```
ian@lyrebird:~> mount -t reiserfs
LABEL=SLES9 on / type reiserfs (rw,acl,user_xattr)
```

Now you'd like to know what partition corresponds to the label `SLES9`, but you can't remember the command. Using `apropos label` might get you a couple of dozen responses, which isn't too bad to sift through. But wait. This command must have something to do with a filesystem of a volume. So you try the regular expressions shown in Listing 5.

Listing 5. Using `apropos` with regular expressions

```
ian@lyrebird:~> apropos "label.*file"
e2label (8)          - Change the label on an ext2/ext3 filesystem
ntfslabel (8)       - display/change the label on an ntfs file system
ian@lyrebird:~> apropos "label.*volume"
label.*volume: nothing appropriate.
```

Not exactly what you were looking for. You could try reversing the order of the terms in the regular expressions, or you could try filtering through `grep` or `egrep` as shown in Listing 6.

Listing 6. Filtering the output of `apropos`

```
ian@lyrebird:~> apropos label | grep -E "file|volume"
e2label (8)          - Change the label on an ext2/ext3 filesystem
mlabel (1)          - make an MSDOS volume label
ntfslabel (8)       - display/change the label on an ntfs file system
findfs (8)          - Find a filesystem by label or UUID
```

And there's the command that we need, `findfs`. Using it as shown in Listing 7 shows that the filesystem is on `/dev/hda10` on this particular system.

Listing 7. Finding the device for a mounted filesystem label

```
ian@lyrebird:~> /sbin/findfs LABEL=SLES9
/dev/hda10
```

Note that non-root users will usually have to give the full path to the `findfs` command.

As you can find out in the man page for the `man` command, you can also use `man -k` instead of `apropos` and `man -f` instead of `whatis`. Since these call the `apropos` or `whatis` command under the covers, there is probably little point in so doing.

Configuration

Manual pages may be in many locations on your system. You can determine the current search path using the `manpath` command. If the `MANPATH` environment variable is set, this will be used for searching for manual pages; otherwise, a path will be built automatically using information from a configuration file that we'll discuss in a moment. If the `MANPATH` environment variable is set, the `manpath` command will issue a warning message to this effect before displaying the path.

Listing 8. Displaying your MANPATH

```
[ian@echidna ian]$ manpath
/usr/local/share/man:/usr/share/man:/usr/X11R6/man:/usr/local/man

ian@lyrebird:~> manpath
manpath: warning: $MANPATH set, ignoring /etc/manpath.config
/usr/local/man:/usr/share/man:/usr/X11R6/man:/opt/gnome/share/man
```

Depending on your system, configuration information for the man system is stored in `/etc/man.config` or `/etc/manpath.config`. Older systems use `/etc/man.conf`. A current `man.config` file contains a list of directories (MANPATHs) that will be searched for manual pages, such as those shown in Listing 9.

Listing 9. MANPATH entries from /etc/man.config

```
MANPATH /usr/share/man
MANPATH /usr/man
MANPATH /usr/local/share/man
MANPATH /usr/local/man
MANPATH /usr/X11R6/man
```

In a `manpath.config` file, these entries will be `MANDATORY_MANPATH` entries, rather than `MANPATH` entries.

Besides these entries, you will also find entries giving a mapping between paths where executables may be found, and paths where the corresponding man pages might be, as shown in Listing 10.

Listing 10. MANPATH_MAP entries from /etc/man.config

```
MANPATH_MAP    /bin                /usr/share/man
MANPATH_MAP    /sbin              /usr/share/man
MANPATH_MAP    /usr/bin           /usr/share/man
MANPATH_MAP    /usr/sbin         /usr/share/man
MANPATH_MAP    /usr/local/bin    /usr/local/share/man
```

The `man` command uses a complicated method for searching for man pages, and setting these values will result in less wasted effort when searching for pages.

Another entry in the configuration file defines the search order for manual pages. Recall that the default is to display the first page found, so this ordering is important. Look near the bottom of `man.config` for a `MANSECT` line, or near the bottom of `manpath.config` for a `SECTION` line. Examine the configuration file on your system to see what other things can be configured.

You may have noticed that the `apropos` and `whatis` commands ran quickly. This is because they do not actually search the individual manual pages. Rather, they use a database created by the `makewhatis` command. This is usually run by the system either daily or weekly as a cron job.

Listing 11. Running `makewhatis`

```
[root@echidna root]# makewhatis
```

The command completes normally without any output message, but the `whatis` database is refreshed. This is usually stored in a location such as `/var/cache/man/whatis`. Note that some SUSE systems do not use the `whatis` database and therefore do not have a `makewhatis` command.

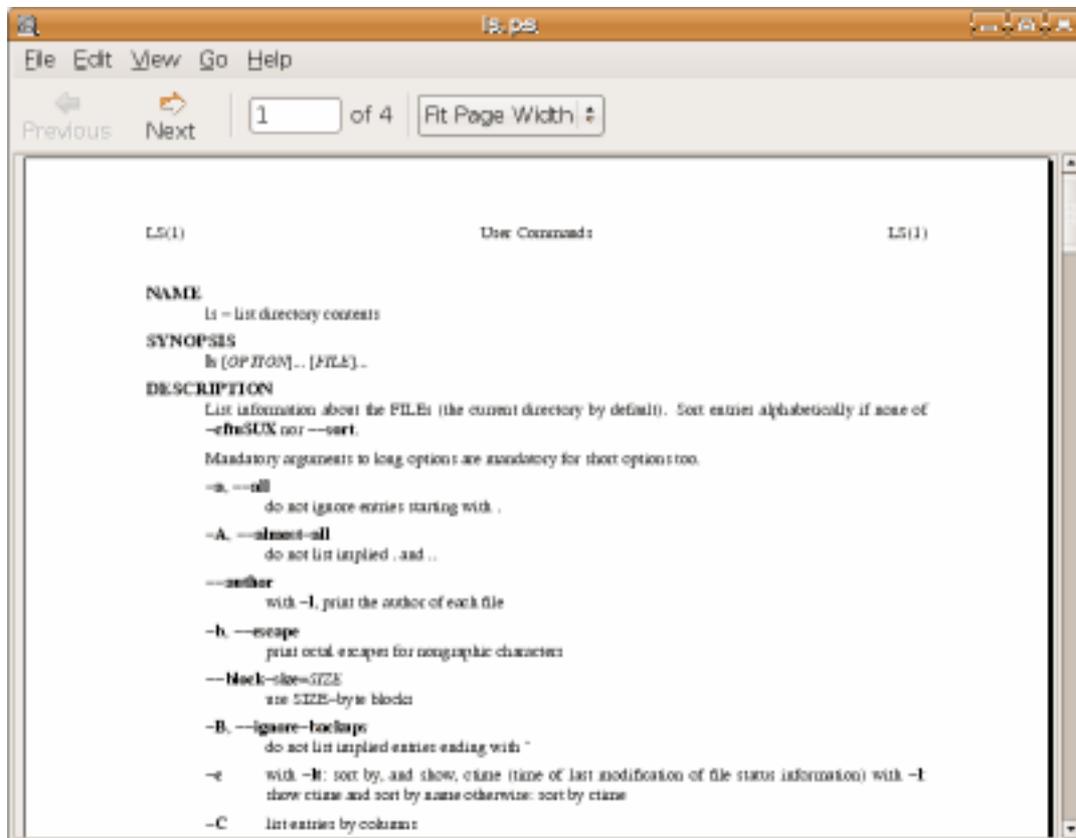
Printing man pages

If you wish to print the page, specify the `-t` option to format the page for printing using the `groff` or `troff` program. This will format the page for the default printer and send the output to `stdout`. Listing 12 shows how to format the man page for the `ls` command and save the output in a file, `ls.ps`. Figure 4 shows the formatted output.

Listing 12. Formatting the `ls` manpage for printing

```
ian@pinguino:~$ man -t ls > ls.ps
```

Figure 4. Formatted `ls` man page



If you need to format the page for a different device type, use the `-T` options with a device type, such as `dvi` or `ps`. See the man page for `man` for additional information.

/usr/share/doc/

In addition to the manual pages and info pages that you have already seen, your Linux system probably includes a lot more documentation. The customary place to store this is in `/usr/share/doc`, or `/usr/doc` on older systems. This additional documentation may be in any of several formats, such as text, PDF, PostScript, or HTML.

Searching through this documentation can often reveal gems that aren't available as man pages or info pages, such as tutorials or additional technical documentation. As Listing 13 shows, there can be a large number of files in `/usr/share/doc`, so you have plenty of reading resources.

Listing 13. Files in /usr/share/doc

```
ian@pinguino:~$ find /usr/share/doc -type f | wc -l
10144
```

Figure 5 shows an example of the HTML help for the Texinfo system that is used for the `info` command that you saw earlier.

Figure 5. Texinfo HTML help from `/usr/share/doc`



Sometimes, a man page will direct you to another source for documentation. For example, the man page for the `pngtopnm` command is shown in Listing 14. It directs you to a local copy in HTML format at `/usr/share/doc/packages/netpbm/doc/pngtopnm.html`, or to an online version if you do not have the local copy.

Listing 14. Pointer man page for `pngtopnm`

```
pngtopnm(1)           Netpbm pointer man pages           pngtopnm(1)

pngtopnm is part of the Netpbm package.  Netpbm documentation is
kept in HTML format.

Please refer to
<http://netpbm.sourceforge.net/doc/pngtopnm.html>.

If that doesn't work, also try <http://netpbm.sourceforge.net>
and emailing Bryan Henderson, bryanh@giraffe-data.com.

Local copy of the page is here:
/usr/share/doc/packages/netpbm/doc/pngtopnm.html
```

Other command help

Finally, if you can't find help for a command, try running the command with the `--help`, `--h`, or `--?` option. This may provide the command's help, or it may tell you how to get the help you need. Listing 15 shows an example for the `kdesu` command, which is usually present on systems with a KDE desktop.

Listing 15. Getting help for `kdesu` command

```
ian@lyrebird:~> man kdesu
No manual entry for kdesu
ian@lyrebird:~> kdesu --help
Usage: kdesu [Qt-options] [KDE-options] command

Runs a program with elevated privileges.

Generic options:
  --help                Show help about options
  --help-qt             Show Qt specific options
  --help-kde            Show KDE specific options
  --help-all           Show all options
  --author              Show author information
  -v, --version         Show version information
  --license             Show license information
  --                   End of options

Arguments:
  command               Specifies the command to run.

Options:
  -c <command>         Specifies the command to run. []
```

The next section covers online resources for help with Linux.

Section 3. Internet documentation

This section covers material for topic 1.108.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to find:

- Online documentation
- Newsgroups
- Mailing lists

Online documentation

In addition to the documentaiton on your system, there are many online sources of

documentation and help.

The Linux Documentation Project

The [Linux Documentation Project](#) is volunteer effort that is putting together the complete set of free Linux documentation. This project exists to consolidate various pieces of Linux documentation into a location that is easy to search and use.

The LDP is made up of the following areas:

HOWTOs

are subject-specific help, such as the [Linux IPv6 HOWTO](#).

Guides

are longer, in-depth books, such as [Introduction to Linux - A Hands on Guide](#).

FAQs

are Frequently Asked Questions, such as the [Linux Documentation Project \(LDP\) FAQ](#).

man pages

are help on individual commands, as you used in the previous section of this tutorial.

Linux Gazette

is an online magazine, currently available in English, French, German, Indonesian, Italian, Portuguese, Russian, and Spanish.

The examples here take you to the multiple-page HTML versions of the documentation. You will find most articles come in several formats, including single-page HTML, PDF, or plain text, among others.

The LDP also has links to [information in languages other than English](#).

The LDP site is well laid out with excellent navigation. If you aren't sure which section to peruse, you can take advantage of the search box, which helps you find things by topic.

If you'd like to help the LDP with Linux documentation, be sure to consult the [LDP Author Guide](#).

Distributor Web sites

Web sites for the various Linux distributions often provide updated documentation, installation instructions, hardware compatibility/incompatibility statements, and other support such as a knowledge base search tool. Some of these are:

- [Redhat Linux](#) is a large distributor of enterprise Linux products based in the United States.
- [SUSE Linux](#) was founded in Germany and is now owned by Novell.
- [Asianux](#) is an Asian Linux distributor, founded by Haansoft, Inc., Red Flag Software Co., Ltd., and Miracle Linux Corporation.
- [Turbolinux](#) is headquartered in Japan but distributes outside Asia as well.
- [Yellow Dog Linux](#) from Terra Soft Solutions is a distribution for Apple PowerPC®-based processors, and embedded processors based on PowerPC and Cell processors.
- [Linspire](#) is a desktop version of Linux that can be found on some preloaded systems.
- The [Slackware Linux Project](#) by Patrick Volkerding has been around since 1993 and aims to be the most "UNIX®-like" Linux distribution out there.
- [Debian GNU/Linux](#) was started in 1993 as a distribution that was created openly, in the spirit of Linux and GNU.
- [Ubuntu Linux](#) is a relatively new distribution of Linux based on Debian. It focuses on ease-of-use and has related projects, Kubuntu (a version using the KDE desktop), Edubuntu (designed for school environments), and Xubuntu (a lightweight version using the Xfce desktop environment).
- [Gentoo Linux](#) is a distribution that can be automatically optimized and customized for just about any application or need. Packages are distributed as source and built to suit the target environment.
- [Mandriva](#) is a distribution featuring ease-of-use. The company was formed from the merger of several open source pioneers such as Mandrakesoft in France, Conectiva in Brazil, Edge IT in France, and Lycoris in the US.

You can find summary information on and links to a large number of Linux distributions at [DistroWatch.com](#). Tabular information on each distribution tells you what levels of which major packages are included in each version, when the version was released, and much other useful information.

Hardware and software vendors

Many hardware and software vendors have added Linux support to their products in recent years. At their sites, you can find information about which hardware supports Linux, software development tools, released sources, downloads of Linux drivers for specific hardware, and other special Linux projects. For example:

- [IBM and Linux](#)
- [Compaq and Linux](#)
- [SGI and Linux](#)
- [HP and Linux](#)
- [Sun and Linux](#)
- [Sun's StarOffice office productivity suite](#)
- [Oracle and Linux](#)
- [BEA and Linux](#)

Open source projects

Many open source projects have home pages where you will find information on the project. Some projects are sponsored by a foundation such as the Apache Software Foundation. Some examples are:

- [Apache Software Foundation](#) is the home of the Apache Web server, and many, many tools.
- [Eclipse Foundation](#) is focused on providing a vendor-neutral open development platform and application frameworks for building software.
- [OpenOffice.org](#) is multiplatform and multilingual office suite.
- [The GNOME Foundation](#) is the home of the GNOME desktop.
- [The KDE project](#) is the home of KDE, the K Desktop Environment.

A large number of open source projects are hosted on [SourceForge.net](#). These are grouped into categories such as clustering, database, desktop, financial, multimedia, security, and so on. Project pages include links for downloading, bug reporting, user forums, and a link to a project's home page (if available) where you will usually find more information about the project.

Other resources

Another great place for Linux information is the [IBM developerWorks Linux zone](#), the home of this tutorial as well as many other fine articles and tutorials for Linux developers.

Many print magazines also have online sites, and some news sites exist only on the Web. Some examples are:

- [LinuxWorld.com](#)

- [Slashdot](#)
- [freshmeat](#)
- [Linux Magazine](#) (German)
- [Linux+](#) (six languages)

Newsgroups

Internet *newsgroups* are, more accurately, a form of discussion lists. They grew out of bulletin boards, which were an early means of sharing information, usually over a dial-up link. Newsgroups use a protocol called *Network News Transfer Protocol* (NNTP), which is defined in IETF RFC 997 (February 1986).

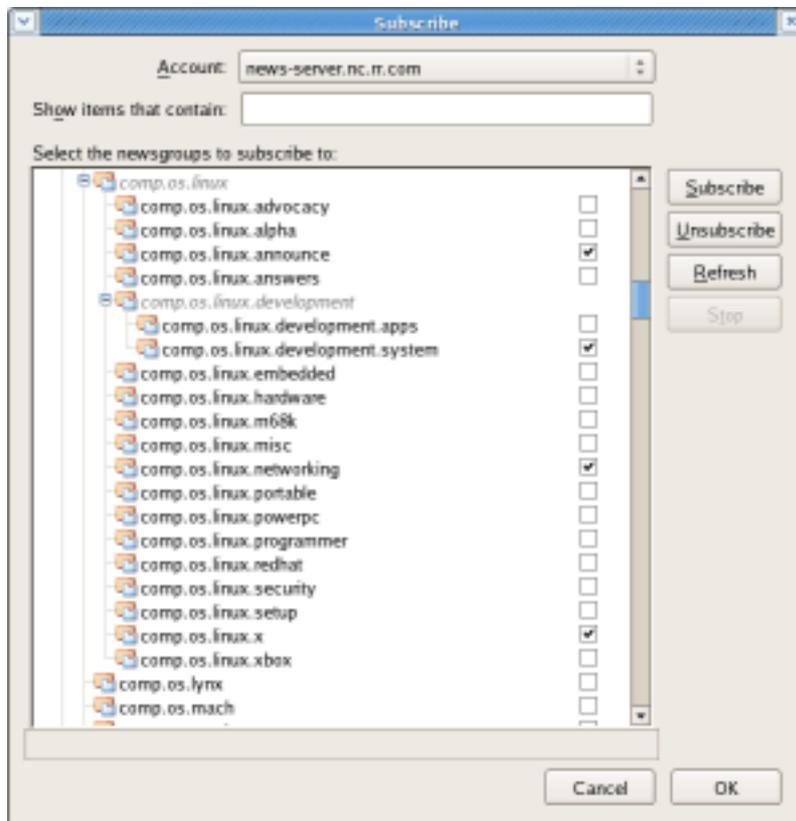
To participate, you use a news reader, which is also known as an NNTP client. There are many Linux clients including `evolution`, `gnus`, `pan`, `slrn`, `thunderbird`, and `tin`. Some of these use a text-mode interface, and some are graphical. The main advantage of a newsgroup is that you take part in the discussion only when you want to, instead of having it continually arrive in your in-box.

Usenet is the largest source of newsgroups. There are several major categories, such as *comp* for computing, *sci* for scientific subjects, and *rec* for recreational topics such as hobbies and games. Computing is further categorized into subjects, and these are still further categorized, so the newsgroups of primary interest to Linux users start with *comp.os.linux*. You can browse a [list on the LDP site](#).

Your Internet Service Provider probably mirrors a range of newsgroups, although news articles may not be retained for a very long period, particularly for active newsgroups. Several newsgroup providers offer a paid service that may provide longer retention, faster access, or a wider selection of newsgroups.

Figure 6 shows the *comp.os.linux* tree as carried on one ISP, using Mozilla's Thunderbird as a newsreader. You *subscribe* to newsgroups, and your newsreader displays only the subscribed groups. Subscribed groups are shown here with a checkmark.

Figure 6. Subscribing to *comp.os.linux.** newsgroups



Newsgroup discussions are often archived. A popular newsgroup for many years was Deja News. When it finally ceased, the newsgroup archives were acquired by Google and reintroduced as [Google Groups](#).

More recently, various Web-based *forums* have arisen. These typically function in a way quite similar to newsgroups, but require only a browser and no configuration. An example is the [Linux tech support forum](#) on IBM's developerWorks Web site where you can ask questions about this series of tutorials along with other topics.

Mailing lists

Mailing lists provide probably the most important point of collaboration for Linux developers. Often projects are developed by contributors who live far apart, possibly even on opposite sides of the globe. Mailing lists overcome time zone differences and thus provide a method for each developer on a project to contact all the others, and to hold group discussions via e-mail. One of the most famous development mailing lists is the [Linux Kernel Mailing List](#).

Mailing lists allow members to send a message to the list, and the list server then broadcasts the message to all members of the group. Individual members do not need to know the e-mail addresses of every member of the group, and they do not need to maintain lists of current members. To avoid a flood of messages from busy

lists, most lists allow a user to request a daily *digest* or single message containing all the list postings for the day.

In addition to development, mailing lists can provide a method for asking questions and receiving answers from knowledgeable developers, or even other users. For example, individual distributions often provide mailing lists for newcomers. You can check your distribution's Web site for information on the mailing lists it provides.

If you took the time to read the LKML FAQ at the link above, you might have noticed that mailing list subscribers often don't take kindly to questions being asked repeatedly. It's always wise to search the archives for a given mailing list before writing your question. Chances are, it will save you time, too. And, speaking of archives, these are often mirrored in multiple sites, so use the closest mirror, typically one in your country or continent.

Section 4. Notifying users

This section covers material for topic 1.108.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 1.

In this section, learn how to:

- Notify the users about current issues related to the system through logon messages

Logon messages

The final short section of this tutorial introduces you to three different logon messages. These have their roots in non-graphical ASCII terminal access to multiuser UNIX® systems and are of diminishing importance today when many workstations are single-user systems, and much access uses a graphical workstation running a desktop such as GNOME or KDE, where these facilities are all but inoperative.

`/etc/issue` and `/etc/issue.net`

The first two of these, `/etc/issue` and `/etc/issue.net`, are displayed on an ASCII terminal that is connected locally (`/etc/issue`) or remotely (`/etc/issue.net`). Listing 16 illustrates these two files as found on a stock Fedora Core 5 system.

Listing 16. `/etc/issue` and `/etc/issue.net`

```
[ian@attic4 ~]$ cat /etc/issue
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m

[ian@attic4 ~]$ cat /etc/issue.net
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m
[ian@attic4 ~]$
```

Notice the control sequences `\r` and `\m`. These allow information such as date or system name to be inserted in the message. The control sequences are shown in Table 4 and are the same as allowed for the `mingetty` command.

Table 4. Control sequences for <code>/etc/issue</code> and <code>/etc/issue.net</code>	
Sequence	Purpose
<code>\d</code>	Inserts the current day according to localtime
<code>\l</code>	Inserts the line on which <code>mingetty</code> is running
<code>\m</code>	Inserts the machine architecture (equivalent to <code>uname -m</code>)
<code>\n</code>	Inserts the machine's network node hostname (equivalent to <code>uname -n</code>)
<code>\o</code>	Inserts the domain name
<code>\r</code>	Inserts the operating system release (equivalent to <code>uname -r</code>)
<code>\t</code>	Inserts the current time according to localtime
<code>\s</code>	Inserts the operating system name
<code>\u</code> or <code>\U</code>	Inserts the current number of users logged in. <code>\U</code> inserts "n users", while <code>\u</code> inserts only "n".
<code>\v</code>	Inserts the operating system version (equivalent to <code>uname -v</code>)

So you can see that the examples in Listing 16 insert the operating system release level and the machine architecture. Connecting via telnet to this system will cause the `/etc/issue.net` message to be displayed before the login prompt as shown in Listing 17.

Listing 17. Telnet connections display `/etc/issue.net`

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
login: ian
Password:
```

If you update `/etc/issue.net` to include a few more control sequences as shown in Listing 18, your logon prompt might look like that in Listing 19.

Listing 18. Updated `/etc/issue.net`

```
[ian@attic4 ~]$ cat /etc/issue.net
Fedora Core release 5 (Bordeaux)
Kernel \r on an \m

\n
Date \d
Time \t
```

Listing 19. Revised telnet logon prompt

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
localhost.localdomain
Date 22:55 on Friday, 15 September 2006
Time 22:55 on Friday, 15 September 2006

login: ian
Password:
```

Notice that on this system `\d` and `\t` produce the same result. As it happens, neither `\u` nor `\U` insert the number of users logged in. This perhaps reflects the fact that these messages have very little use these days. The use of telnet with its passwords flowing in the clear is strongly discouraged. Since a connection using ssh passes the login id and therefore bypasses a login prompt, and since real ASCII terminals remotely connected are rare, the contents of `/etc/issue.net` are rarely seen, and probably not tested too well either.

You will see the contents of `/etc/issue` if you do not use a graphical login. Even if you do, you can usually get a non-graphical login at the system console using `Ctrl-Alt-F1` through `Ctrl-Alt-F6`, with `Ctrl-Alt-F7` returning you to the graphical terminal.

Message of the day

Both `/etc/issue` and `/etc/issue.net` provide user feedback in the form of a logon prompt and could also be used to advise users of issues such as impending outages. However, this is usually done with a *message of the day* or *motd*, which is stored in `/etc/motd`. The contents of `/etc/motd` are displayed after a successful login but just before the login shell is started. Listing 20 shows an example of a motd file, and Listing 21 shows how it and `/etc/issue.net` appear to a user logging in through a telnet session.

Listing 20. Sample message of the day (motd)

```
[ian@attic4 ~]$ cat /etc/motd
PLEASE NOTE!
All systems will shut down this weekend for emergency power testing.
Save your work or lose it.
```

Listing 20. Sample message of the day (motd)

```
Fedora Core release 5 (Bordeaux)
Kernel 2.6.17-1.2174_FC5 on an x86_64
localhost.localdomain
Date 22:55 on Friday, 15 September 2006
Time 22:55 on Friday, 15 September 2006

login: ian
Password:
Last login: Fri Sep 15 22:54:18 from 192.168.0.101

PLEASE NOTE!
All systems will shut down this weekend for emergency power testing.
Save your work or lose it.
[ian@attic4 ~]$
```

Again, the motd is really only useful on ASCII terminal sessions. Neither KDE nor GNOME desktops have an easy and satisfactory way of displaying it.

One final notification method that you should know about is the `wall` command, which sends a warning to all logged-in users using text from either a file or stdin. Again, these are not seen by users using the standard GNOME or KDE desktops.

Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, Feb 2006), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *[LPI Linux Certification in a Nutshell, Second Edition](#)* (O'Reilly, 2006) and *[LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S.

and Ph.D. in computer science from North Carolina State University.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 109: Shells, scripting, programming, and compiling

Junior Level Administration (LPIC-1) topic 109

Skill Level: Intermediate

[Ian Shields](#)
Senior Programmer
IBM

30 Jan 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this fifth in a [series of nine tutorials](#), Ian introduces you to the Bash shell, and scripts and programming in the Bash shell. By the end of this tutorial, you will know how to customize your shell environment, use shell programming structures to create functions and scripts, set and unset environment variables, and use the various login scripts.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial

on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	(This tutorial.) Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts. See the detailed objectives below.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot

- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Shells, scripting, programming, and compiling," the fifth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to use the Bash shell, how to use shell programming structures to create functions and scripts, how to customize your shell environment, how to set and unset environment variables, and how to use the various login scripts.

The title for this tutorial duplicates the corresponding topic in the LPI 102 exam, and therefore includes "programming and compiling," but the LPI objectives limit "programming" to that required for writing shell functions and scripts. And no objectives for compiling programs are included in the topic.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.109.1 Customize and use the shell environment	Weight 5	Customize shell environments to meet user needs. Set environment variables (at login or when spawning a new shell). Write Bash functions for frequently used sequences of commands.
1.109.2 Customize or write simple scripts	Weight 3	Write simple Bash scripts and customize existing ones.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be thoroughly familiar with the material from the "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)" tutorial, as many of the building blocks for this tutorial were covered in that tutorial, especially the section "Using the command line."

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Shell customization

This section covers material for topic 1.109.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 5.

In this section, learn how to:

- Set and unset environment variables
- Use profiles to set environment variables at login or when spawning a new shell
- Write shell functions for frequently used sequences of commands
- Use command lists

Shells and environments

Before the advent of graphical interfaces, programmers used a typewriter terminal or an ASCII display terminal to connect to a UNIX® system. A typewriter terminal allowed them to type commands, and the output was usually printed on continuous paper. Most ASCII display terminals had 80 characters per line and about 25 lines on the screen, although both larger and smaller terminals existed. Programmers typed a command and pressed **Enter**, and the system interpreted and then executed the command.

While this may seem somewhat primitive today in an era of drag-and-drop graphical interfaces, it was a huge step forward from writing a program, punching cards, compiling the card deck, and running the program. With the advent of editors, programmers could even create programs as card images and compile them in a terminal session.

The stream of characters typed at a terminal provided a *standard input* stream to the

shell, and the stream of characters that the shell returned on either paper or display represented the *standard output*.

The program that accepts the commands and executes them is called a *shell*. It provides a layer between you and the intricacies of an operating system. UNIX and Linux shells are extremely powerful in that you can build quite complex operations by combining basic functions. Using programming constructs you can then build functions for direct execution in the shell or save functions as *shell scripts* so that you can reuse them over and over.

Sometimes you need to execute commands before the system has booted far enough to allow terminal connections, and sometimes you need to execute commands periodically, whether or not you are logged on. A shell can do this for you, too. The standard input and output do not have to come from or be directed to a real user at a terminal.

In this section, you learn more about shells. In particular, you learn about the *bash* or *Bourne again* shell, which is an enhancement of the original Bourne shell, along with some features from other shells and some changes from the Bourne shell to make it more POSIX compliant.

POSIX is the *Portable Operating System Interface for uniX*, which is a series of IEEE standards collectively referred to as IEEE 1003. The first of these was IEEE Standard 1003.1-1988, released in 1988. Other well known shells include the *Korn* shell (ksh), the *C* shell (csh) and its derivative tcsh, the *Almquist* shell (ash) and its Debian derivative (dash). You need to know something about many of these shells, if only to recognize when a particular script requires features from one of them.

Many aspects of your interaction with a computer will be the same from one session to another. Recall from the tutorial "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)" that when you are running in a Bash shell, you have a shell *environment*, which defines such things as the form of your prompt, your home directory, your working directory, the name of your shell, files that you have opened, functions that you have defined, and so on. The environment is made available to every shell process. Shells, including bash, allow you to create and modify *shell variables*, which you may *export* to your environment for use by other processes running in the shell or by other shells that you may spawn from the current shell.

Both environment variables and shell variables have a *name*. You reference the value of a variable by prefixing its name with '\$'. Some of the common bash environment variables that are set for you are shown in Table 3.

Table 3. Common bash environment variables	
Name	Function
USER	The name of the logged-in user
UID	The numeric user id of the

	logged-in user
HOME	The user's home directory
PWD	The current working directory
SHELL	The name of the shell
\$	The process id (or <i>PID</i> of the running Bash shell (or other) process)
PPID	The process id of the process that started this process (that is, the id of the parent process)
?	The exit code of the last command

Setting variables

In the Bash shell, you create or *set* a shell variable by typing a name followed immediately by an equal sign (=). Variable names (or *identifiers*) are words consisting only of alphanumeric characters and underscores, that begin with an alphabetic character or an underscore. Variables are case sensitive, so `var1` and `VAR1` are different variables. By convention, variables, particularly exported variables, are upper case, but this is not a requirement. Technically, `$$` and `$?` are shell *parameters* rather than variables. They may only be referenced; you cannot assign a value to them.

When you create a shell variable, you will often want to *export* it to the environment so it will be available to other processes that you start from this shell. Variables that you export are **not** available to a parent shell. You use the `export` command to export a variable name. As a shortcut in bash, you can assign and export in one step.

To illustrate assignment and exporting, let's run the bash command while in the Bash shell and then run the Korn shell (ksh) from the new Bash shell. We will use the `ps` command to display information about the command that is running.

Listing 1. Setting and exporting shell variables

```
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
30576 30575 -bash
[ian@echidna ian]$ bash
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16353 30576 bash
[ian@echidna ian]$ VAR1=var1
[ian@echidna ian]$ VAR2=var2
[ian@echidna ian]$ export VAR2
```

```

[ian@echidna ian]$ export VAR3=var3
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3
var1 var2 var3
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $SHELL
var1 var2 var3 /bin/bash
[ian@echidna ian]$ ksh
$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16448 16353 ksh
$ export VAR4=var4
$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
var2 var3 var4 /bin/bash
$ exit
$ [ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
var1 var2 var3 /bin/bash
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16353 30576 bash
[ian@echidna ian]$ exit
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
30576 30575 -bash
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
/bin/bash

```

Notes:

1. At the start of this sequence, the Bash shell had PID 30576.
2. The second Bash shell has PID 16353, and its parent is PID 30576, the original Bash shell.
3. We created VAR1, VAR2, and VAR3 in the second Bash shell, but only exported VAR2 and VAR3.
4. In the Korn shell, we created VAR4. The `echo` command displayed values only for VAR2, VAR3, and VAR4, confirming that VAR1 was not exported. Were you surprised to see that the value of the SHELL variable had not changed, even though the prompt had changed? You cannot always rely on SHELL to tell you what shell you are running under, but the `ps` command does tell you the actual command. Note that `ps` puts a hyphen (-) in front of the first Bash shell to indicate that this is the *login shell*.
5. Back in the second Bash shell, we can see VAR1, VAR2, and VAR3.
6. And finally, when we return to the original shell, none of our new variables still exist.

Listing 2 shows what you might see in some of these common bash variables.

Listing 2. Environment and shell variables

```
[ian@echidna ian]$ echo $USER $UID
ian 500
[ian@echidna ian]$ echo $SHELL $HOME $PWD
/bin/bash /home/ian /home/ian
[ian@echidna ian]$ (exit 0);echo $?;(exit 4);echo $?
0
4
[ian@echidna ian]$ echo $$ $PPID
30576 30575
```

Environments and the C shell

In shells such as the C and tcsh shells, you use the `set` command to set variables in your shell, and the `setenv` command to set and export variables. The syntax differs slightly from that of the `export` command as illustrated in Listing 3. Note the equals (=) sign when using `set`.

Listing 3. Setting environment variables in the C shell

```
ian@attic4:~$ echo $VAR1 $VAR2

ian@attic4:~$ csh
% set VAR1=var1
% setenv VAR2 var2
% echo $VAR1 $VAR2
var1 var2
% bash
ian@attic4:~$ echo $VAR1 $VAR2
var2
```

Unsetting variables

You remove a variable from the Bash shell using the `unset` command. You can use the `-v` option to be sure that you are removing a variable definition. Functions can have the same name as variables, so use the `-f` if you want to remove a function definition. Without either `-f` or `-v`, the bash `unset` command removes a variable definition if it exists; otherwise, it removes a function definition if one exists. (Functions are covered in more detail later in the [Shell functions](#) section.)

Listing 4. The bash unset command

```
ian@attic4:~$ VAR1=var1
ian@attic4:~$ VAR2=var2
ian@attic4:~$ echo $VAR1 $VAR2
var1 var2
ian@attic4:~$ unset VAR1
ian@attic4:~$ echo $VAR1 $VAR2
var2
ian@attic4:~$ unset -v VAR2
```

```
ian@attic4:~$ echo $VAR1 $VAR2
```

The bash default is to treat unset variables as if they had an empty value, so you might wonder why you would unset a variable rather than just assign it an empty value. Bash and many other shells allow you to generate an error if an undefined variable is referenced. Use the command `set -u` to generate an error for undefined variables, and `set +u` to disable the warning. Listing 5 illustrates these points.

Listing 5. Generating errors with unset variables

```
ian@attic4:~$ set -u
ian@attic4:~$ VAR1=var1
ian@attic4:~$ echo $VAR1
var1
ian@attic4:~$ unset VAR1
ian@attic4:~$ echo $VAR1
-bash: VAR1: unbound variable
ian@attic4:~$ VAR1=
ian@attic4:~$ echo $VAR1

ian@attic4:~$ unset VAR1
ian@attic4:~$ echo $VAR1
-bash: VAR1: unbound variable
ian@attic4:~$ unset -v VAR1
ian@attic4:~$ set +u
ian@attic4:~$ echo $VAR1

ian@attic4:~$
```

Note that it is not an error to unset a variable that does not exist, even when `set -u` has been specified.

Profiles

When you log in to a Linux system, your id has a default shell, which is your *login* shell. If this shell is bash, then it executes several profile scripts before you get control. If `/etc/profile` exists, it is executed first. Depending on your distribution, other scripts in the `/etc` tree may also be executed, for example, `/etc/bash.bashrc` or `/etc/bashrc`. Once the system scripts have run, a script in your home directory is run if it exists. Bash looks for the files `~/.bash_profile`, `~/.bash_login`, and `~/.profile` in that order. The first one found is executed.

When you log off, bash executes the `~/.bash_logout` script from your home directory if it exists.

Once you have logged in and are already using bash, you may start another shell, called an *interactive* shell to run a command, for example to run a command in the background. In this case, bash executes only the `~/.bashrc` script, assuming one exists. It is common to check for this script in your `~/.bash_profile`, so that you can

execute it at login as well as when starting an interactive shell, using commands such as those shown in Listing 6.

Listing 6. Checking for ~/.bashrc

```
# include .bashrc if it exists
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

You may force bash to read profiles as if it were a login shell using the `--login` option. If you do not want to execute the profiles for a login shell, specify the `--noprofile` option. Similarly, if you want to disable execution of the `~/.bashrc` file for an interactive shell, start bash with the `--norc` option. You can also force bash to use a file other than `~/.bashrc` by specifying the `--rcfile` option with the name of the file you want to use. Listing 8 illustrates creation of a simple file called `testrc` and its use with the `--rcfile` option. Note that the `VAR1` variable is **not** set in the outer shell, but has been set for the inner shell by the `testrc` file.

Listing 7. Using the --rcfile option

```
ian@attic4:~$ echo VAR1=var1>testrc
ian@attic4:~$ echo $VAR1

ian@attic4:~$ bash --rcfile testrc
ian@attic4:~$ echo $VAR1
var1
```

Starting bash in other ways

In addition to the standard ways of running bash from a terminal as outlined above, bash may also be used in other ways.

Unless you *source* a script to run in the current shell, it will run in its own *non-interactive* shell, and the above profiles are not read. However, if the `BASH_ENV` variable is set, bash expands the value and assumes it is the name of a file. If the file exists, then bash executes the file before whatever script or command it is executing in the non-interactive shell. Listing 8 uses two simple files to illustrate this.

Listing 8. Using BASH_ENV

```
ian@attic4:~$ cat testenv.sh
#!/bin/bash
echo "Testing the environment"
ian@attic4:~$ cat somescript.sh
#!/bin/bash
```

```
echo "Doing nothing"
ian@attic4:~$ export BASH_ENV=~/.testenv.sh
ian@attic4:~$ ./somescript.sh
Testing the environment
Doing nothing
```

Non-interactive shells may also be started with the `--login` option to force execution of the profile files.

Bash may also be started in *POSIX* mode using the `--posix` option. This mode is similar to the non-interactive shell, except that the file to execute is determined from the ENV environment variable.

It is common in Linux systems to run bash as `/bin/sh` using a symbolic link. When bash detects that it is being run under the name `sh`, it attempts to follow the startup behavior of the older Bourne shell while still conforming to POSIX standards. When run as a login shell, bash attempts to read and execute `/etc/profile` and `~/.profile`. When run as an interactive shell using the `sh` command, bash attempts to execute the file specified by the ENV variable as it does when invoked in POSIX mode. When run interactively as `sh`, it **only** uses a file specified by the ENV variable; the `--rcfile` option will always be ignored.

If bash is invoked by the remote shell daemon, then it behaves as an interactive shell, using the `~/.bashrc` file if it exists.

Shell aliases

The Bash shell allows you to define *aliases* for commands. The most common reasons for aliases are to provide an alternate name for a command, or to provide some default parameters for the command. The `vi` editor has been a staple of UNIX and Linux systems for many years. The `vim` (Vi IMproved) editor is like `vi`, but with many improvements. So if you are used to typing "vi" when you want an editor, but you would really prefer to use `vim`, then an alias is for you. Listing 9 shows how to use the `alias` command to accomplish this.

Listing 9. Using vi as an alias for vim

```
[ian@pinguino ~]$ alias vi='vim'
[ian@pinguino ~]$ which vi
alias vi='vim'
/usr/bin/vim
[ian@pinguino ~]$ /usr/bin/which vi
/bin/vi
```

Notice in this example that if you use the `which` command to see where the `vi` program lives, you get two lines of output: the first shows the alias, and the second

the location of vim (/usr/bin/vim). However, if you use the `which` command with its full path (/usr/bin/which), you get the location of the `vi` command. If you guessed that this might mean that the `which` command itself is aliased on this system you would be right.

You can also use the `alias` command to display all the aliases if you use it with no options or with just the `-p` option, and you can display the aliases for one or more names by giving the names as arguments without assignments. Listing 10 shows the aliases for `which` and `vi`.

Listing 10. Aliases for which and vi

```
[ian@pinguino ~]$ alias which vi
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
alias vi='vim'
```

The alias for the `which` command is rather curious. Why pipe the output of the `alias` command (with no arguments) to `/usr/bin/which`? If you check the man pages for the `which` command, you will find that the `--read-alias` option instructs `which` to read a list of aliases from stdin and report matches on stdout. This allows the `which` command to report aliases as well as commands from your PATH, and is so common that your distribution may have set it up as a default for you. This is a good thing to do since the shell will execute an alias before a command of the same name. So now that you know this, you can check it using `alias which`. You can also learn whether this type of alias has been set for `which` by running `which which`.

Another common use for aliases is to add parameters automatically to commands, as you saw above for the `--read-alias` and several other parameters on the `which` command. This technique is often done for the root user with the `cp`, `mv`, and `rm` commands so that a prompt is issued before files are deleted or overwritten. This is illustrated in Listing 11.

Listing 11. Adding parameters for safety

```
[root@pinguino ~]# alias cp mv rm
alias cp='cp -i'
alias mv='mv -i'
alias rm='rm -i'
```

Command lists

In the earlier tutorial "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)," you learned about command *sequences* or *lists*. You have just seen the pipe (`|`) operator used with an alias, and you can use command lists as well. Suppose, for a

simple example, that you want a command to list the contents of the current directory and also the amount of space used by it and all its subdirectories. Let's call it the `lsdu` command. So you simply assign a sequence of the `ls` and `du` commands to the alias `lsdu`. Listing 12 shows the wrong way to do this and also the right way. Look carefully at it before you read, and think about why the first attempt did not work.

Listing 12. Aliases for command sequences

```
[ian@pinguino developerworks]$ alias lsdu=ls;du -sh # Wrong way
2.9M .
[ian@pinguino developerworks]$ lsdu
a tutorial new-article.sh new-tutorial.sh readme tools xsl
my-article new-article.vbs new-tutorial.vbs schema web
[ian@pinguino developerworks]$ alias 'lsdu=ls;du -sh' # Right way way
[ian@pinguino developerworks]$ lsdu
a tutorial new-article.sh new-tutorial.sh readme tools xsl
my-article new-article.vbs new-tutorial.vbs schema web
2.9M .
```

You need to be very careful to quote the full sequence that will make up the alias. You also need to be very careful about whether you use double or single quotes if you have shell variables as part of the alias. Do you want the shell to expand the variables when the alias is defined or when it is executed? Listing 13 shows the wrong way to create a custom command called `mywd` intended to print your working directory name

Listing 13. Custom pwd - attempt 1

```
[ian@pinguino developerworks]$ alias mywd="echo \"My working directory is $PWD\""
[ian@pinguino developerworks]$ mywd
My working directory is /home/ian/developerworks
[ian@pinguino developerworks]$ cd ..
[ian@pinguino ~]$ mywd
My working directory is /home/ian/developerworks
```

Remember that the double quotes cause bash to expand variables before executing a command. Listing 14 uses the `alias` command to show what the resulting alias actually is, from which our error is evident. Listing 14 also shows a correct way to define this alias.

Listing 14. Custom pwd - attempt 2

```
[ian@pinguino developerworks]$ alias mywd
alias mywd='echo \"My working directory is $PWD\"'
[ian@pinguino developerworks]$ mywd
"My working directory is /home/ian/developerworks"
[ian@pinguino developerworks]$ cd ..
[ian@pinguino ~]$ mywd
"My working directory is /home/ian"
```

Success at last.

Shell functions

Aliases allow you to use an abbreviation or alternate name for a command or command list. You may have noticed that you can add additional things, such as the program name you are seeking with the `which` command. When your input is executed, the alias is expanded, and anything else you type after that is added to the expansion before the final command or list is executed. This means that you can only add parameters to the end of the command or list, and you can use them only with the final command. Functions provide additional capability, including the ability to process parameters. Functions are part of the POSIX shell definition. They are available in shells such as `bash`, `dash`, and `ksh`, but are not available in `csh` or `tosh`.

In the next few paragraphs, you'll build a complex command piece-by-piece from smaller building blocks, refining it each step of the way and turning it into a function that you will further refine.

A hypothetical problem

You can use the `ls` command to list a variety of information about directories and files in your file system. Suppose you would like a command, let's call it `ldirs`, that will list directory names with output like that in Listing 15.

Listing 15. The `ldirs` command output

```
[ian@pinguino developerworks]$ ldirs *[st]* tools/*a*
my dw article
schema
tools
tools/java
xsl
```

To keep things relatively simple, the examples in this section use the directories and files from the developerWorks author package (see [Resources](#)), which you can use if you'd like to write articles or tutorials for developerWorks. In these examples, we used the `new-article.sh` script from the package to create a template for a new article that we've called "my dw article".

At the time of writing, the version of the developerWorks author package is 5.6, so you may see differences if you use a later version. Or just use your own files and directories. The `ldirs` command will handle those too. You'll find additional bash function examples in the tools that come with the developerWorks author package.

Finding directory entries

Ignoring the `*[st]* tools/*a*` for the moment, if you use the `ls` command with the color options as shown in the aliases above, you will see output similar to that shown in Figure 1.

Figure 1. Distinguishing files and directories with the `ls` command

```

ian@pinguino:~/developerworks
File Edit View Terminal Tabs Help
[ian@pinguino developerworks]$ ls
my dw article  new-article.vbs new-tutorial.vbs schema web
new-article.sh new-tutorial.sh readme      tools  xsl
[ian@pinguino developerworks]$ ls -l
total 80
drwxrwxr-x 2 ian ian 4096 Jan 24 17:06 my dw article
-rwxr--r-- 1 ian ian  215 Sep 27 16:34 new-article.sh
-rwxr--r-- 1 ian ian 1078 Sep 27 16:34 new-article.vbs
-rwxr--r-- 1 ian ian  216 Sep 27 16:34 new-tutorial.sh
-rwxr--r-- 1 ian ian 1079 Sep 27 16:34 new-tutorial.vbs
drwxrwxr-x 2 ian ian 4096 Jan 18 16:23 readme
drwxrwxr-x 3 ian ian 4096 Jan 19 07:41 schema
drwxrwxr-x 3 ian ian 4096 Jan 19 15:08 tools
drwxrwxr-x 3 ian ian 4096 Jan 17 16:03 web
drwxrwxr-x 3 ian ian 4096 Jan 19 10:59 xsl
[ian@pinguino developerworks]$

```

The directories are shown in dark blue in this example, but that's a bit hard to decode with the skills you have developed in this series of tutorials. Using the `-l` option, though, gives a clue on how to proceed: directory listings have a 'd' in the first position. So your first step might be to simply filter these from the long listing using `grep` as shown in Listing 16.

Listing 16. Using `grep` to find just directory entries

```

[ian@pinguino developerworks]$ ls -l | grep "^d"
drwxrwxr-x 2 ian ian 4096 Jan 24 17:06 my dw article
drwxrwxr-x 2 ian ian 4096 Jan 18 16:23 readme
drwxrwxr-x 3 ian ian 4096 Jan 19 07:41 schema
drwxrwxr-x 3 ian ian 4096 Jan 19 15:08 tools
drwxrwxr-x 3 ian ian 4096 Jan 17 16:03 web
drwxrwxr-x 3 ian ian 4096 Jan 19 10:59 xsl

```

Trimming the directory entries

You might consider using `awk` instead of `grep` so that in one pass you can filter the list and strip off the last part of each line, which is the directory name, as shown in Listing 17.

Listing 17. Using `awk` instead

```

[ian@pinguino developerworks]$ ls -l | awk '/^d/ { print $NF } '
article
readme
schema

```

```
tools
web
xsl
```

The problem with the approach in Listing 17 is that it doesn't handle the directory with spaces in the name, such as "my dw article". As with most things in Linux and life, there are often several ways to solve a problem, but the objective here is to learn about functions, so let's return to using `grep`. Another tool you learned about earlier in this series is `cut`, which cuts fields out of a file, including stdin. Looking back at Listing 16 again, you see eight blank-delimited fields before the filename. Adding `cut` to the previous command gives you output as shown in Listing 18. Note that the `-f9-` option tells `cut` to print fields 9 and above.

Listing 18. Using `cut` to trim names

```
[ian@pinguino developerworks]$ ls -l | grep "^d" | cut -d" " -f9-
my dw article
readme
schema
tools
web
xsl
```

A small problem with our approach is made obvious if we try our command on the `tools` directory instead of on the current directory as shown in Listing 19.

Listing 19. A problem with `cut`

```
[ian@pinguino developerworks]$ ls -l tools | grep "^d" | cut -d" " -f9-
11:25 java
[ian@pinguino developerworks]$ ls -ld tools/[fjt]*
-rw-rw-r-- 1 ian ian 4798 Jan 8 14:38 tools/figure1.gif
drwxrwxr-x 2 ian ian 4096 Oct 31 11:25 tools/java
-rw-rw-r-- 1 ian ian 39431 Jan 18 23:31 tools/template-dw-article-5.6.xml
-rw-rw-r-- 1 ian ian 39407 Jan 18 23:32 tools/template-dw-tutorial-5.6.xml
```

How did the timestamp get in there? The two template files have 5-digit sizes, while the `java` directory has only a 4-digit size, so `cut` interpreted the extra space as another field separator.

Use `seq` to find a cut point

The `cut` command can also cut using character positions instead of fields. Rather than counting characters, the Bash shell has lots of utilities that you can use, so you might try using the `seq` and `printf` commands to print a ruler above your long directory listing so you can easily figure where to cut the lines of output. The `seq` command takes up to three arguments, which allow you to print all the numbers up to a given value, all the numbers from one value to another, or all the numbers from one value, stepping by a given value, up to a third value. See the man pages for all

the other fancy things you can do with `seq`, including printing octal or hexadecimal numbers. For now let's use `seq` and `printf` to print a ruler with positions marked every 10 characters as shown in Listing 20.

Listing 20. Printing a ruler with `seq` and `printf`

```
[ian@pinguino developerworks]$ printf "....+...%2.d" `seq 10 10 60`;printf "\n";ls -l
....+...10....+...20....+...30....+...40....+...50....+...60
total 88
drwxrwxr-x 2 ian ian 4096 Jan 24 17:06 my dw article
-rwxr--r-- 1 ian ian 215 Sep 27 16:34 new-article.sh
-rwxr--r-- 1 ian ian 1078 Sep 27 16:34 new-article.vbs
-rwxr--r-- 1 ian ian 216 Sep 27 16:34 new-tutorial.sh
-rwxr--r-- 1 ian ian 1079 Sep 27 16:34 new-tutorial.vbs
drwxrwxr-x 2 ian ian 4096 Jan 18 16:23 readme
drwxrwxr-x 3 ian ian 4096 Jan 19 07:41 schema
drwxrwxr-x 3 ian ian 4096 Jan 19 15:08 tools
drwxrwxr-x 3 ian ian 4096 Jan 17 16:03 web
drwxrwxr-x 3 ian ian 4096 Jan 19 10:59 xsl
```

Aha! Now you can use the command `ls -l | grep "^d" | cut -c40-` to cut lines starting at position 40. A moment's reflection reveals that this doesn't really solve the problem either, because larger files will move the correct cut position to the right. Try it for yourself.

Sed to the rescue

Sometimes called the "Swiss army knife" of the UNIX and Linux toolbox, `sed` is an extremely powerful editing filter that uses regular expressions. You now understand that the challenge is to strip off the first 8 words and the blanks that follow them from every line of output that begins with 'd'. You can do it all with `sed`: select only those lines you are interested in using the pattern-matching expression `/^d/`, substituting a null string for the first eight words using the substitute command `s/^d\([^]* *\)\{8\}/`. Use the `-n` option to print only lines that you specify with the `p` command as shown in Listing 21.

Listing 21. Trimming directory names with `sed`

```
[ian@pinguino developerworks]$ ls -l | sed -ne 's/^d\([ ^ ]* *\)\{8\}/p'
my dw article
readme
schema
tools
web
xsl
[ian@pinguino developerworks]$ ls -l tools | sed -ne 's/^d\([ ^ ]* *\)\{8\}/p'
java
```

To learn more about `sed`, see the [Resources](#) section.

A function at last

Now that you have the complex command that you want for your `ldirs` function, it's time to learn about making it a function. A function consists of a name followed by `()` and then a compound command. For now, a compound command will be any command or command list, terminated by a semicolon and surrounded by braces (which must be separated from other tokens by white space). You will learn about other compound commands in the [Shell scripts](#) section.

Note: In the Bash shell, a function name may be preceded by the word 'function', but this is not part of the POSIX specification and is not supported by more minimalist shells such as dash. In the [Shell scripts](#) section, you will learn how to make sure that a script is interpreted by a particular shell, even if you normally use a different shell.

Inside the function, you can refer to the parameters using the bash special variables in Table 4. You prefix these with a `$` symbol to reference them as with other shell variables.

Table 4. Shell parameters for functions	
Parameter	Purpose
0, 1, 2, ...	The positional parameters starting from parameter 0. Parameter 0 refers to the name of the program that started bash, or the name of the shell script if the function is running within a shell script. See the bash man pages for information on other possibilities, such as when bash is started with the <code>-c</code> parameter. A string enclosed in single or double quotes will be passed as a single parameter, and the quotes will be stripped. In the case of double quotes, any shell variables such as <code>\$HOME</code> will be expanded before the function is called. You will need to use single or double quotes to pass parameters that contain embedded blanks or other characters that might have special meaning to the shell.
*	The positional parameters starting from parameter 1. If the expansion is done within double quotes, then the expansion is a single word with the first character of the interfield separator (IFS) special variable separating the parameters or no intervening space if IFS is null. The default IFS value is a blank, tab, and newline. If IFS is unset, then the separator used is a blank, just as for the default IFS.

@	The positional parameters starting from parameter 1. If the expansion is done within double quotes, then each parameter becomes a single word, so that "\$@" is equivalent to "\$1" "\$2" If your parameters are likely to contain embedded blanks, you will want to use this form.
#	The number of parameters, not including parameter 0.

Note: If you have more than 9 parameters, you cannot use \$10 to refer to the tenth one. You must first either process or save the first parameter (\$1), then use the `shift` command to drop parameter 1 and move all remaining parameters down 1, so that \$10 becomes \$9 and so on. The value of \$# will be updated to reflect the remaining number of parameters.

Now you can define a simple function to do nothing more than tell you how many parameters it has and display them as shown in Listing 22.

Listing 22. Function parameters

```
[ian@pinguino developerworks]$ testfunc () { echo "$# parameters"; echo "$@"; }
[ian@pinguino developerworks]$ testfunc
0 parameters

[ian@pinguino developerworks]$ testfunc a b c
3 parameters
a b c
[ian@pinguino developerworks]$ testfunc a "b c"
2 parameters
a b c
```

Whether you use \$*, "\$*", \$@, or "\$@", you won't see much difference in the output of the above function, but rest assured that when things become more complex, the distinctions will matter very much.

Now take the complex command that we tested up to this point and create a `ldirs` function with it, using "\$@" to represent the parameters. You can enter all of the function on a single line as you did in the previous example, or `bash` lets you enter commands on multiple lines, in which case a semicolon will be added automatically as shown in Listing 23. Listing 23 also shows the use of the `type` command to display the function definition. Note from the output of `type` that the `ls` command has been replaced by the expanded value of its alias. You could use `/bin/ls` instead of plain `ls` if you needed to avoid this.

Listing 23. Your first `ldirs` function

```
[ian@pinguino developerworks]$ # Enter the function on a single line
[ian@pinguino developerworks]$ ldirs () { ls -l "$@"|sed -ne 's/^d\([^ ]* *\)\{8\} //p'; }
[ian@pinguino developerworks]$ # Enter the function on multiple lines
[ian@pinguino developerworks]$ ldirs ()
> {
> ls -l "$@"|sed -ne 's/^d\([^ ]* *\)\{8\} //p'
> }
[ian@pinguino developerworks]$ type ldirs
ldirs is a function
ldirs ()
{
    ls --color=tty -l "$@" | sed -ne 's/^d\([^ ]* *\)\{8\} //p'
}
[ian@pinguino developerworks]$ ldirs
my dw article
readme
schema
tools
web
xsl
[ian@pinguino developerworks]$ ldirs tools
java
```

So now your function appears to be working. But what happens if you run `ldirs *` as shown in Listing 24?

Listing 24. Running `ldirs *`

```
[ian@pinguino developerworks]$ ldirs *
5.6
java
www.ibm.com
5.6
```

Are you surprised? You didn't find directories in the current directory, but rather second-level subdirectories. Review the man page for the `ls` command or our earlier tutorials in this series to understand why. Or run the `find` command as shown in Listing 25 to print the names of second-level subdirectories.

Listing 25. Finding second-level subdirectories

```
[ian@pinguino developerworks]$ find . -mindepth 2 -maxdepth 2 -type d
./tools/java
./web/www.ibm.com
./xsl/5.6
./schema/5.6
```

Adding some tests

Using wildcards has exposed a problem with the logic in this approach. We blithely ignored the fact that `ldirs` without any parameters displayed the subdirectories in the current directory, while `ldirs tools` displayed the `java` subdirectory of the `tools` directory rather than the `tools` directory itself as you would expect using `ls` with files rather than directories. Ideally, you should use `ls -l` if no parameters are

given and `ls -ld` if some parameters are given. You can use the `test` command to test the number of parameters and then use `&&` and `||` to build a command list that executes the appropriate command. Using the `[test expression]` form of `test`, your expression might look like `{ [$# -gt 0] &&/bin/ls -ld "$@" || /bin/ls -l } | sed -ne`

There is a small issue with this code, though, in that if the `ls -ld` command doesn't find any matching files or directories, it will issue an error message and return with a non-zero exit code, thus causing the `ls -l` command to be executed as well. Perhaps not what you wanted. One answer is to construct a compound command for the first `ls` command so that the number of parameters is tested again if the command fails. Expand the function to include this, and your function should now appear as in Listing 26. Try using it with some of the parameters in Listing 26, or experiment with your own parameters to see how it behaves.

Listing 26. Handling wildcards with `ldirs`

```
[ian@pinguino ~]$ type ldirs
ldirs is a function
ldirs ()
{
    {
        [ $# -gt 0 ] && {
            /bin/ls -ld "$@" || [ $# -gt 0 ]
        } || /bin/ls -l
    } | sed -ne 's/^d\([^ ]* *\)\{8\} //p'
}
[ian@pinguino developerworks]$ ldirs *
my dw article
readme
schema
tools
web
xsl
[ian@pinguino developerworks]$ ldirs tools/*
tools/java
[ian@pinguino developerworks]$ ldirs *xxx*
/bin/ls: *xxx*: No such file or directory
[ian@pinguino developerworks]$ ldirs *a* *s*
my dw article
readme
schema
schema
tools
xsl
```

Final touchup

At this point you might get a directory listed twice as in the last example of Listing 26. You could extend the pipeline by piping the `sed` output through `sort | uniq` if you wish.

Starting from some small building blocks, you have now built quite a complex shell function.

Customizing keystrokes

The keystrokes you type at a terminal session, and also those used in programs such as FTP, are processed by the readline library and can be configured. By default, the customization file is `.inputrc` in your home directory, which will be read during bash startup if it exists. You can configure a different file by setting the `INPUTRC` variable. If it is not set, `.inputrc` in your home directory will be used. Many systems have a default key mapping in `/etc/inputrc`, so you will normally want to include these using the `$include` directive.

Listing 27 illustrates how you might bind your `ldirs` function to the Ctrl-t key combination (press and hold Ctrl, then press t). If you want the command to be executed with no parameters, add `\n` to the end of the configuration line.

Listing 27. Sample `.inputrc` file

```
# My custom key mappings
$include /etc/inputrc
```

You can force the `INPUTRC` file to be read again by pressing Ctrl-x then Ctrl-r. Note that some distributions will set `INPUTRC=/etc/inputrc` if you do not have your own `.inputrc`, so if you create one on such a system, you will need to log out and log back in to pick up your new definitions. Just resetting `INPUTRC` to null or to point to your new file will reread the original file, not the new specification.

The `INPUTRC` file can include conditional specifications. For example, the behavior of your keyboard should be different according to whether you are using emacs editing mode (the bash default) or vi mode. See the man pages for bash for more details on how to customize your keyboard.

Saving aliases and functions

You will probably add your aliases and functions to your `~/.bashrc` file, although you may save them in any file you like. Whichever you do, remember to source the file or files using the `source` or `.` command so that the contents of your file will be read and executed in the current environment. If you create a script and just execute it, it will be executed in a subshell and all your valuable customization will be lost when the subshell exits and returns control to you.

In the next section, you learn how to go beyond simple functions. You learn how to add programming constructs such as conditional tests and looping constructs and combine these with multiple functions to create or modify Bash shell scripts.

Section 3. Shell scripts

This section covers material for topic 1.109.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Use standard shell syntax, such as loops and tests
- Use command substitution
- Test return values for success or failure or other information provided by a command
- Perform conditional mailing to the superuser
- Select the correct script interpreter through the shebang (#!) line
- Manage the location, ownership, execution, and suid-rights of scripts

This section builds on what you learned about simple functions in the last section and demonstrates some of the techniques and tools that add programming capability to the shell. You have already see simple logic using the `&&` and `||` operators, which allow you to execute one command based on whether the previous command exits normally or with an error. In the `ldirs` function, you used this to alter the call to `ls` according to whether or not parameters were passed to your `ldirs` function. Now you will learn how to extend these basic techniques to more complex shell programming.

Tests

The first thing you need to do in any kind of programming language after learning how to assign values to variables and pass parameters is to test those values and parameters. In shells the tests you do set the return status, which is the same thing that other commands do. In fact, `test` is a builtin *command*!

test and [

The `test` builtin command returns 0 (True) or 1 (False) depending on the evaluation of an expression `expr`. You can also use square brackets so that `test expr` and `[expr]` are equivalent. You can examine the return value by displaying `$?`; you can use the return value as you have before with `&&` and `||`; or you can test it using the various conditional constructs that are covered later in this section.

Listing 28. Some simple tests

```
[ian@pinguino ~]$ test 3 -gt 4 && echo True || echo false
false
[ian@pinguino ~]$ [ "abc" != "def" ];echo $?
0
[ian@pinguino ~]$ test -d "$HOME" ;echo $?
0
```

In the first of these examples, the `-gt` operator was used to perform an arithmetic comparison between two literal values. In the second, the alternate `[]` form was used to compare two strings for inequality. In the final example, the value of the `HOME` variable is tested to see if it is a directory using the `-d` unary operator.

Arithmetic values may be compared using one of `-eq`, `-ne`, `-lt`, `-le`, `-gt`, or `-ge`, meaning equal, not equal, less than, less than or equal, greater than, and greater than or equal, respectively.

Strings may be compared for equality, inequality, or whether the first string sorts before or after the second one using the operators `=`, `!=`, `<` and `>`, respectively. The unary operator `-z` tests for a null string, while `-n` or no operator at all returns True if a string is not empty.

Note: the `<` and `>` operators are also used by the shell for redirection, so you must escape them using `\<` or `\>`. Listing 29 shows some more examples of string tests. Check that they are as you expect.

Listing 29. Some string tests

```
[ian@pinguino ~]$ test "abc" = "def" ;echo $?
1
[ian@pinguino ~]$ [ "abc" != "def" ];echo $?
0
[ian@pinguino ~]$ [ "abc" \< "def" ];echo $?
0
[ian@pinguino ~]$ [ "abc" \> "def" ];echo $?
1
[ian@pinguino ~]$ [ "abc" \<"abc" ];echo $?
1
[ian@pinguino ~]$ [ "abc" \> "abc" ];echo $?
1
```

Some of the more common file tests are shown in Table 5. The result is True if the file tested is a file that exists and that has the specified characteristic.

Table 5. Some file tests	
Operator	Characteristic
<code>-d</code>	Directory
<code>-e</code>	Exists (also <code>-a</code>)

-f	Regular file
-h	Symbolic link (also -L)
-p	Named pipe
-r	Readable by you
-s	Not empty
-S	Socket
-w	Writable by you
-N	Has been modified since last being read

In addition to the unary tests above, two files can be compared with the binary operators shown in Table 6.

Table 6. Testing pairs of files	
Operator	True if
-nt	Test if file1 is newer than file 2. The modification date is used for this and the next comparison.
-ot	Test if file1 is older than file 2.
-ef	Test if file1 is a hard link to file2.

Several other tests allow you to check things such as the permissions of the file. See the man pages for `bash` for more details or use `help test` to see brief information on the test builtin. You can use the `help` command for other builtins too.

The `-o` operator allows you to test various shell options that may be set using `set -o option`, returning True (0) if the option is set and False (1) otherwise, as shown in Listing 30.

Listing 30. Testing shell options

```
[ian@pinguino ~]$ set +o nounset
[ian@pinguino ~]$ [ -o nounset ];echo $?
1
[ian@pinguino ~]$ set -u
[ian@pinguino ~]$ test -o nounset; echo $?
0
```

Finally, the `-a` and `-o` options allow you to combine expressions with logical AND and OR, respectively, while the unary `!` operator inverts the sense of the test. You

may use parentheses to group expressions and override the default precedence. Remember that the shell will normally run an expression between parentheses in a subshell, so you will need to escape the parentheses using `\(` and `\)` or enclosing these operators in single or double quotes. Listing 31 illustrates the application of de Morgan's laws to an expression.

Listing 31. Combining and grouping tests

```
[ian@pinguino ~]$ test "a" != "$HOME" -a 3 -ge 4 ; echo $?
1
[ian@pinguino ~]$ [ ! \( "a" = "$HOME" -o 3 -lt 4 \) ]; echo $?
1
[ian@pinguino ~]$ [ ! \( "a" = "$HOME" -o '(' 3 -lt 4 ')' " )" ]; echo $?
1
```

((and [[

The `test` command is very powerful, but somewhat unwieldy with its requirement for escaping and the difference between string and arithmetic comparisons. Fortunately `bash` has two other ways of testing that are somewhat more natural for people who are familiar with C, C++, or Java syntax. The `(())` *compound command* evaluates an arithmetic expression and sets the exit status to 1 if the expression evaluates to 0, or to 0 if the expression evaluates to a non-zero value. You do not need to escape operators between `((` and `))`. Arithmetic is done on integers. Division by 0 causes an error, but overflow does not. You may perform the usual C language arithmetic, logical, and bitwise operations. The `let` command can also execute one or more arithmetic expressions. It is usually used to assign values to arithmetic variables.

Listing 32. Assigning and testing arithmetic expressions

```
[ian@pinguino ~]$ let x=2 y=2**3 z=y*3;echo $? $x $y $z
0 2 8 24
[ian@pinguino ~]$ (( w=(y/x) + ( (~ ++x) & 0x0f ) )); echo $? $x $y $w
0 3 8 16
[ian@pinguino ~]$ (( w=(y/x) + ( (~ ++x) & 0x0f ) )); echo $? $x $y $w
0 4 8 13
```

As with `(())`, the `[[]]` compound command allows you to use more natural syntax for filename and string tests. You can combine tests that are allowed for the `test` command using parentheses and logical operators.

Listing 33. Using the [[compound

```
[ian@pinguino ~]$ [[ ( -d "$HOME" ) && ( -w "$HOME" ) ]] &&
> echo "home is a writable directory"
home is a writable directory
```

The `[]` compound can also do pattern matching on strings when the `=` or `!=` operators are used. The match behaves as for wildcard globbing as illustrated in Listing 34.

Listing 34. Wildcard tests with `[]`

```
[ian@pinguino ~]$ [[ "abc def .d,x--" == a[abc]*\ ?d* ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def c" == a[abc]*\ ?d* ]]; echo $?
1
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* ]]; echo $?
1
```

You can even do arithmetic tests within `[]` compounds, but be careful. Unless within a `(())` compound, the `<` and `>` operators will compare the operands as strings and test their order in the current collating sequence. Listing 35 illustrates this with some examples.

Listing 35. Including arithmetic tests with `[]`

```
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || (( 3 > 2 )) ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || 3 -gt 2 ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || 3 > 2 ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || a > 2 ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || a -gt 2 ]]; echo $?
-bash: a: unbound variable
```

Conditionals

While you could accomplish a huge amount of programming with the above tests and the `&&` and `||` control operators, bash includes the more familiar "if, then, else" and case constructs. After you learn about these, you will learn about looping constructs and your toolbox will really expand.

If, then, else statements

The bash `if` command is a compound command that tests the return value of a test or command (`$?` and branches based on whether it is True (0) or False (not 0). Although the tests above returned only 0 or 1 values, commands may return other values. You will learn more about testing those a little later in this tutorial. The `if` command in bash has a `then` clause containing a list of commands to be executed if the test or command returns 0. The command also has one or more optional `elif` clauses. Each of these optional `elif` clauses has an additional test and `then`

clause with an associated list of commands, an optional final `else` clause, and list of commands to be executed if neither the original test, nor any of the tests used in the `elif` clauses was true. A terminal `fi` marks the end of the construct.

Using what you have learned so far, you could now build a simple calculator to evaluate arithmetic expressions as shown in Listing 36.

Listing 36. Evaluating expressions with `if`, `then`, `else`

```
[ian@pinguino ~]$ function mycalc ()
> {
>   local x
>   if [ $# -lt 1 ]; then
>     echo "This function evaluates arithmetic for you if you give it some"
>   elif (( $* )); then
>     let x="$*"
>     echo "$* = $x"
>   else
>     echo "$* = 0 or is not an arithmetic expression"
>   fi
> }
[ian@pinguino ~]$ mycalc 3 + 4
3 + 4 = 7
[ian@pinguino ~]$ mycalc 3 + 4**3
3 + 4**3 = 67
[ian@pinguino ~]$ mycalc 3 + (4**3 /2)
-bash: syntax error near unexpected token `('
[ian@pinguino ~]$ mycalc 3 + "(4**3 /2)"
3 + (4**3 /2) = 35
[ian@pinguino ~]$ mycalc xyz
xyz = 0 or is not an arithmetic expression
[ian@pinguino ~]$ mycalc xyz + 3 + "(4**3 /2)" + abc
xyz + 3 + (4**3 /2) + abc = 35
```

The calculator makes use of the `local` statement to declare `x` as a local variable that is available only within the scope of the `mycalc` function. The `let` function has several possible options, as does the `declare` function to which it is closely related. Check the man pages for `bash`, or use `help let` for more information.

As you see in Listing 36, you need to be careful making sure that your expressions are properly escaped if they use shell metacharacters such as `(,)`, `*`, `>`, and `<`. Nevertheless, you have quite a handy little calculator for evaluating arithmetic as the shell does it.

You may have noticed the `else` clause and the last two examples. As you see, it is not an error to pass `xyz` to `mycalc`, but it evaluates to 0. This function is not smart enough to identify the character values in the final example of use and thus be able to warn the user. You could use a string pattern matching test such as

```
[[ ! ("$*" == *[a-zA-Z]* ) ]]
```

(or the appropriate form for your locale) to eliminate any expression containing alphabetic characters, but that would prevent using hexadecimal notation in your input, since you might use `0x0f` to represent 15 using hexadecimal notation. In fact,

the shell allows bases up to 64 (using *base#value* notation), so you could legitimately use any alphabetic character, plus `_` and `@` in your input. Octal and hexadecimal use the usual notation of a leading 0 for octal and leading 0x or 0X for hexadecimal. Listing 37 shows some examples.

Listing 37. Calculating with different bases

```
[ian@pinguino ~]$ mycalc 015
015 = 13
[ian@pinguino ~]$ mycalc 0xff
0xff = 255
[ian@pinguino ~]$ mycalc 29#37
29#37 = 94
[ian@pinguino ~]$ mycalc 64#1az
64#1az = 4771
[ian@pinguino ~]$ mycalc 64#1azA
64#1azA = 305380
[ian@pinguino ~]$ mycalc 64#1azA_@
64#1azA_@ = 1250840574
[ian@pinguino ~]$ mycalc 64#1az*64**3 + 64#A_@
64#1az*64**3 + 64#A_@ = 1250840574
```

Additional laundering of the input is beyond the scope of this tutorial, so use your calculator with appropriate care.

The `elif` statement is really a convenience. It will help you in writing scripts by allowing you to simplify the indenting. You may be surprised to see the output of the `type` command for the `mycalc` function as shown in Listing 38.

Listing 38. Type mycalc

```
[ian@pinguino ~]$ type mycalc
mycalc is a function
mycalc ()
{
    local x;
    if [ $# -lt 1 ]; then
        echo "This function evaluates arithmetic for you if you give it some";
    else
        if (( $* )); then
            let x="$*";
            echo "$* = $x";
        else
            echo "$* = 0 or is not an arithmetic expression";
        fi;
    fi;
}
```

Case statements

The `case` compound command simplifies testing when you have a list of possibilities and you want to take action based on whether a value matches a particular possibility. The `case` compound is introduced by `case WORD in` and terminated by `esac` ("case" spelled backwards). Each case consists of a single pattern, or multiple

patterns separated by |, followed by), a list of statements, and finally a pair of semicolons (;).

To illustrate, imagine a store that serves coffee, decaffeinated coffee (decaf), tea, or soda. The function in Listing 39 might be used to determine the response to an order.

Listing 39. Using case commands

```
[ian@pinguino ~]$ type myorder
myorder is a function
myorder ()
{
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here"
            ;;
    esac
}
[ian@pinguino ~]$ myorder decaf
Hot coffee coming right up
[ian@pinguino ~]$ myorder tea
Hot tea on its way
[ian@pinguino ~]$ myorder milk
Sorry, we don't serve that here
```

Note the use of '*' to match anything that had not already been matched.

Bash has another construct similar to `case` that can be used for printing output to a terminal and having a user select items. It is the `select` statement, which will not be covered here. See the `bash` man pages, or type `help select` to learn more about it.

Of course, there are many problems with such a simple approach to the problem; you can't order two drinks at once, and the function doesn't handle anything but lower-case input. Can you do a case-insensitive match? The answer is "yes", so let's see how.

Return values

The Bash shell has a `shopt` builtin that allows you to set or unset many shell options. One is `nocasematch`, which, if set, instructs the shell to ignore case in string matching. Your first thought might be to use the `-o` operand that you learned

about with the `test` command. Unfortunately, `nocasematch` is not one of the options you can test with `-o`, so you'll have to resort to something else.

The `shopt` command, like most UNIX and Linux commands sets a return value that you can examine using `$?` . The tests that you learned earlier are not the only things with return values. If you think about the tests that you do in an `if` statement, they really test the return value of the underlying test command for being True (0) or False (1 or anything other than 0). This works even if you don't use a test, but use some other command. Success is indicated by a return value of 0, and failure by a non-zero return value.

Armed with this knowledge, you can now test the `nocasematch` option, set it if it is not already set, and then return it to the user's preference when your function terminates. The `shopt` command has four convenient options, `-pqsu` to print the current value, don't print anything, set the option, or unset the option. The `-p` and `-q` options set a return value of 0 to indicate that the shell option is set, and 1 to indicate it is unset. The `-p` options prints out the command required to set the option to its current value, while the `-q` option simply sets a return value of 0 or 1.

Your modified function will use the return value from `shopt` to set a local variable representing the current state of the `nocasematch` option, set the option, run the case command, then reset the `nocasematch` option to its original value. One way to do this is shown in Listing 40.

Listing 40. Testing return values from commands

```
[ian@pinguino ~]$ type myorder
myorder is a function
myorder ()
{
    local restorecase;
    if shopt -q nocasematch; then
        restorecase="-s";
    else
        restorecase="-u";
        shopt -s nocasematch;
    fi;
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here"
            ;;
    esac;
    shopt $restorecase nocasematch
}
[ian@pinguino ~]$ shopt -p nocasematch
shopt -u nocasematch
```

```
[ian@pinguino ~]$ # nocasematch is currently unset
[ian@pinguino ~]$ myorder DECAF
Hot coffee coming right up
[ian@pinguino ~]$ myorder Soda
Your ice-cold soda will be ready in a moment
[ian@pinguino ~]$ shopt -p nocasematch
shopt -u nocasematch
[ian@pinguino ~]$ # nocasematch is unset again after running the myorder function
```

If you want your function (or script) to return a value that other functions or commands can test, use the `return` statement in your function. Listing 41 shows how to return 0 for a drink that you can serve and 1 if the customer requests something else.

Listing 41. Setting your own return values from functions

```
[ian@pinguino ~]$ type myorder
myorder is a function
myorder ()
{
    local restorecase=$(shopt -p nocasematch) rc=0;
    shopt -s nocasematch;
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here";
            rc=1
            ;;
    esac;
    $restorecase;
    return $rc
}
[ian@pinguino ~]$ myorder coffee;echo $?
Hot coffee coming right up
0
[ian@pinguino ~]$ myorder milk;echo $?
Sorry, we don't serve that here
1
```

If you don't specify your own return value, the return value will be that of the last command executed. Functions have a habit of being reused in situations that you never anticipated, so it is good practice to set your own value.

Commands may return values other than 0 and 1, and sometimes you will want to distinguish between them. For example, the `grep` command returns 0 if the pattern is matched and 1 if it is not, but it also returns 2 if the pattern is invalid or if the file specification doesn't match any files. If you need to distinguish more return values besides just success (0) or failure (non-zero), then you will probably use a `case` command or perhaps an `if` command with several `elif` parts.

Command substitution

You met command substitution in the "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)" tutorial, but let's do a quick review.

Command substitution allows you to use the output of a command as input to another command by simply surrounding the command with `$(` and `)` or with a pair of backticks ```. You will find the `$(` form advantageous if you want to nest output from one command as part of the command that will generate the final output, and it can be easier to figure out what's really going on as the parentheses have a left and right form as opposed to two identical backticks. However, the choice is yours, and backticks are still very common,

You will often use command substitution with loops (covered later under [Loops](#)). However, you can also use it to simplify the `myorder` function that you just created. Since `shopt -p nocasematch` actually prints the command that you need to set the `nocasematch` option to its current value, you only need to save that output and then execute it at the end of the `case` statement. This will restore the `nocasematch` option regardless of whether you actually changed it or not. Your revised function might now look like Listing 42. Try it for yourself.

Listing 42. Command substitution instead of return value tests

```
[ian@pinguino ~]$ type myorder
myorder is a function
myorder ()
{
    local restorecase=$(shopt -p nocasematch) rc=0;
    shopt -s nocasematch;
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here"
            rc=1
            ;;
    esac;
    $restorecase
    return $rc
}
[ian@pinguino ~]$ shopt -p nocasematch
shopt -u nocasematch
[ian@pinguino ~]$ myorder DECAF
Hot coffee coming right up
[ian@pinguino ~]$ myorder TeA
Hot tea on its way
[ian@pinguino ~]$ shopt -p nocasematch
shopt -u nocasematch
```

Debugging

If you have typed functions yourself and made typing errors that left you wondering what was wrong, you might also be wondering how to debug functions. Fortunately the shell lets you set the `-x` option to trace commands and their arguments as the shell executes them. Listing 43 shows how this works for the `myorder` function of Listing 42.

Listing 43. Tracing execution

```
[ian@pinguino ~]$ set -x
++ echo -ne '\033]0;ian@pinguino:~'

[ian@pinguino ~]$ myorder tea
+ myorder tea
++ shopt -p nocasematch
+ local 'restorecase=shopt -u nocasematch' rc=0
+ shopt -s nocasematch
+ case "$*" in
+ echo 'Hot tea on its way'
Hot tea on its way
+ shopt -u nocasematch
+ return 0
++ echo -ne '\033]0;ian@pinguino:~'

[ian@pinguino ~]$ set +x
+ set +x
```

You can use this technique for your aliases, functions, or scripts. If you need more information, add the `-v` option for verbose output.

Loops

Bash and other shells have three looping constructs that are somewhat similar to those in the C language. Each will execute a list of commands zero or more times. The list of commands is surrounded by the words `do` and `done`, each preceded by a semicolon.

for

loops come in two flavors. The most common form in shell scripting iterates over a set of values, executing the command list once for each value. The set may be empty, in which case the command list is not executed. The other form is more like a traditional C for loop, using three arithmetic expressions to control a starting condition, step function, and end condition.

while

loops evaluate a condition each time the loop starts and execute the command

list if the condition is true. If the condition is not initially true, the commands are never executed.

until

loops execute the command list and evaluate a condition each time the loop ends. If the condition is true the loop is executed again. Even if the condition is not initially true, the commands are executed at least once.

If the conditions that are tested are a list of commands, then the return value of the last one executed is the one used. Listing 44 illustrates the loop commands.

Listing 44. For, while, and until loops

```
[ian@pinguino ~]$ for x in abd 2 "my stuff"; do echo $x; done
abd
2
my stuff
[ian@pinguino ~]$ for (( x=2; x<5; x++ )); do echo $x; done
2
3
4
[ian@pinguino ~]$ let x=3; while [ $x -ge 0 ] ; do echo $x ;let x--;done
3
2
1
0
[ian@pinguino ~]$ let x=3; until echo -e "x=\c"; (( x-- == 0 )) ; do echo $x ; done
x=2
x=1
x=0
```

These examples are somewhat artificial, but they illustrate the concepts. You will most often want to iterate over the parameters to a function or shell script, or a list created by command substitution. Earlier you discovered that the shell may refer to the list of passed parameters as `$*` or `$@` and that whether you quoted these expressions or not affected how they were interpreted. Listing 45 shows a function that prints out the number of parameters and then prints the parameters according to the four alternatives. Listing 46 shows the function in action, with an additional character added to the front of the IFS variable for the function execution.

Listing 45. A function to print parameter information

```
[ian@pinguino ~]$ type testfunc
testfunc is a function
testfunc ()
{
    echo "$# parameters";
    echo Using '$*';
    for p in $*;
    do
        echo "[$p]";
    done;
    echo Using '$@';
    for p in "$@";

```

```

do
    echo "[$p]";
done;
echo Using '$@';
for p in $@;
do
    echo "[$p]";
done;
echo Using '"$@"';
for p in "$@";
do
    echo "[$p]";
done
}

```

Listing 46. Printing parameter information with testfunc

```

[ian@pinguino ~]$ IFS="|${IFS}" testfunc abc "a bc" "1 2
> 3"
3 parameters
Using $*
[abc]
[a]
[bc]
[1]
[2]
[3]
Using "$*"
[abc|a bc|1 2
3]
Using $@
[abc]
[a]
[bc]
[1]
[2]
[3]
Using "$@"
[abc]
[a bc]
[1 2
3]

```

Study the differences carefully, particularly for the quoted forms and the parameters that include white space such as blanks or newline characters.

Break and continue

The `break` command allows you to exit from a loop immediately. You can optionally specify a number of levels to break out of if you have nested loops. So if you had an `until` loop inside a `for` loop inside another `for` loop and all inside a `while` loop, then `break 3` would immediately terminate the `until` loop and the two `for` loops, and return control to the code in the `while` loop.

The `continue` statement allows you to bypass remaining statements in the command list and go immediately to the next iteration of the loop.

Listing 47. Using break and continue

```
[ian@pinguino ~]$ for word in red blue green yellow violet; do
> if [ "$word" = blue ]; then continue; fi
> if [ "$word" = yellow ]; then break; fi
> echo "$word"
> done
red
green
```

Revisiting ldirs

Remember how much work you did to get the `ldirs` function to extract the file name from a long listing and also figure out if it was a directory or not? The final function that you developed was not too bad, but suppose you had all the information you now have. Would you have created the same function? Perhaps not. You know how to test whether a name is a directory or not using `[-d $name]`, and you know about the `for` compound. Listing 48 shows another way you might have coded the `ldirs` function.

Listing 48. Another approach to ldirs

```
[ian@pinguino developerworks]$ type ldirs
ldirs is a function
ldirs ()
{
    if [ $# -gt 0 ]; then
        for file in "$@";
        do
            [ -d "$file" ] && echo "$file";
        done;
    else
        for file in *;
        do
            [ -d "$file" ] && echo "$file";
        done;
    fi;
    return 0
}
[ian@pinguino developerworks]$ ldirs
my dw article
my-tutorial
readme
schema
tools
web
xsl
[ian@pinguino developerworks]$ ldirs *s* tools/*
schema
tools
xsl
tools/java
[ian@pinguino developerworks]$ ldirs *www*
[ian@pinguino developerworks]$
```

You will note that the function quietly returns if there are no directories matching your criteria. This may or may not be what you want, but if it is, this form of the function is perhaps easier to understand than the version that used `sed` to parse

output from `ls`. At least you now have another tool in your toolbox.

Creating scripts

Recall that `myorder` could handle only one drink at a time? You could now combine that single drink function with a `for` compound to iterate through the parameters and handle multiple drinks. This is as simple as placing your function in a file and adding the `for` instruction. Listing 49 illustrates the new `myorder.sh` script.

Listing 49. Ordering multiple drinks

```
[ian@pinguino ~]$ cat myorder.sh
function myorder ()
{
    local restorecase=$(shopt -p nocasematch) rc=0;
    shopt -s nocasematch;
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here";
            rc=1
            ;;
    esac;
    $restorecase;
    return $rc
}

for file in "$@"; do myorder "$file"; done

[ian@pinguino ~]$ . myorder.sh coffee tea "milk shake"
Hot coffee coming right up
Hot tea on its way
Sorry, we don't serve that here
```

Note that the script was *sourced* to run in the current shell environment rather than its own shell using the `.` command. To be able to execute a script, either you have to source it, or the script file must be marked executable using the `chmod -x` command as illustrated in Listing 50.

Listing 50. Making the script executable

```
[ian@pinguino ~]$ chmod +x myorder.sh
[ian@pinguino ~]$ ./myorder.sh coffee tea "milk shake"
Hot coffee coming right up
Hot tea on its way
Sorry, we don't serve that here
```

Specify a shell

Now that you have a brand-new shell script to play with, you might ask whether it works in all shells. Listing 51 shows what happens if you run the exact same shell script on a Ubuntu system using first the Bash shell, then the dash shell.

Listing 51. Shell differences

```
ian@attic4:~$ ./myorder tea soda
-bash: ./myorder: No such file or directory
ian@attic4:~$ ./myorder.sh tea soda
Hot tea on its way
Your ice-cold soda will be ready in a moment
ian@attic4:~$ dash
$ ./myorder.sh tea soda
./myorder.sh: 1: Syntax error: "(" unexpected
```

That's not too good.

Remember earlier when we mentioned that the word 'function' was optional in a bash function definition, but that it wasn't part of the POSIX shell specification? Well, dash is a smaller and lighter shell than bash and it doesn't support that optional feature. Since you can't guarantee what shell your potential users might prefer, you should always ensure that your script is portable to all shell environments, which can be quite difficult, or use the so-called *shebang* (`#!`) to instruct the shell to run your script in a particular shell. The shebang line must be the first line of your script, and the rest of the line contains the path to the shell that your program must run under, so it would be `#!/bin/bash` the `myorder.sh` script.

Listing 52. Using shebang

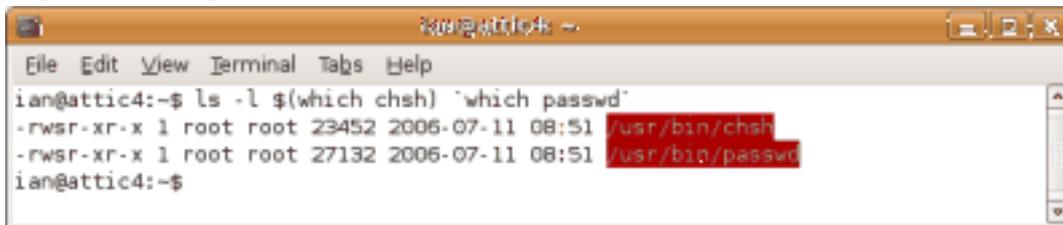
```
$ head -n3 myorder.sh
#!/bin/bash
function myorder ()
{
$ ./myorder.sh Tea Coffee
Hot tea on its way
Hot coffee coming right up
```

You can use the `cat` command to display `/etc/shells`, which is the list of shells on your system. Some systems do list shells that are not installed, and some listed shells (possibly `/dev/null`) may be there to ensure that FTP users cannot accidentally escape from their limited environment. If you need to change your default shell, you can do so with the `chsh` command, which updates the entry for your userid in `/etc/passwd`.

Suid rights and script locations

In the earlier tutorial [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#) you learned how to change a file's owner and group and how to set the suid and sgid permissions. An executable file with either of these permissions set will run in a shell with *effective* permissions of the file's owner (for suid) or group (for sgid). Thus, the program will be able to do anything that the owner or group could do, according to which permission bit is set. There are good reasons why some programs need to do this. For example, the `passwd` program needs to update `/etc/shadow`, and the `chsh` command, which you use to change your default shell, needs to update `/etc/passwd`. If you use an alias for `ls`, listing these programs is likely to result in a red, highlighted listing to warn you, as shown in Figure 2. Note that both of these programs have the suid big (s) set and thus operate as if root were running them.

Figure 2. Programs with suid permission



```

ian@attic4:~$ ls -l $(which chsh) $(which passwd)
-rwsr-xr-x 1 root root 23452 2006-07-11 08:51 /usr/bin/chsh
-rwsr-xr-x 1 root root 27132 2006-07-11 08:51 /usr/bin/passwd
ian@attic4:~$

```

Listing 53 shows that an ordinary user can run these and update files owned by root.

Listing 53. Using suid programs

```

ian@attic4:~$ passwd
Changing password for ian
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
ian@attic4:~$ chsh
Password:
Changing the login shell for ian
Enter the new value, or press ENTER for the default
  Login Shell [/bin/bash]: /bin/dash
ian@attic4:~$ find /etc -mmin -2 -ls
308865      4 drwxr-xr-x 108 root      root           4096 Jan 29 22:52 /etc
find: /etc/cups/ssl: Permission denied
find: /etc/lvm/archive: Permission denied
find: /etc/lvm/backup: Permission denied
find: /etc/ssl/private: Permission denied
311170      4 -rw-r--r--   1 root      root           1215 Jan 29 22:52 /etc/passwd
309744      4 -rw-r-----  1 root      shadow          782 Jan 29 22:52 /etc/shadow
ian@attic4:~$ grep ian /etc/passwd
ian:x:1000:1000:Ian Shields,,,:/home/ian:/bin/dash

```

You can set suid and sgid permissions for shell scripts, but most modern shells ignore these bits for scripts. As you have seen, the shell has a powerful scripting language, and there are even more features that are not covered in this tutorial, such as the ability to interpret and execute arbitrary expressions. These features make it a very unsafe environment to allow such wide permission. So, if you set suid

or `sgid` permission for a shell script, don't expect it to be honored when the script is executed.

Earlier, you changed the permissions of `myorder.sh` to mark it *executable* (`x`). Despite that, you still had to qualify the name by prefixing `./` to actually run it, unless you sourced it in the current shell. To execute a shell by name only, it needs to be on your path, as represented by the `PATH` variable. Normally, you do **not** want the current directory on your path, as it is a potential security exposure. Once you have tested your script and found it satisfactory, you should place it in `~/nom` if it is a personal script, or `/usr/local/bin` if it is to be available for others on the system. If you simply used `chmod -x` to mark it executable, it is executable by everyone (owner, group and world). This is generally what you want, but refer back to the earlier tutorial, [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#), if you need to restrict the script so that only members of a certain group can execute it.

You may have noticed that shells are usually located in `/bin` rather than in `/usr/bin`. According to the Filesystem Hierarchy Standard, `/usr/bin` may be on a filesystem shared among systems, and so it may not be available at initialization time. Therefore, certain functions, such as shells, should be in `/bin` so they are available even if `/usr/bin` is not yet mounted. User-created scripts do not usually need to be in `/bin` (or `/sbin`), as the programs in these directories should give you enough tools to get your system up and running to the point where you can mount the `/usr` filesystem.

Mail to root

If your script is running some administrative task on your system in the dead of night while you're sound asleep, what happens when something goes wrong? Fortunately, it's easy to mail error information or log files to yourself or to another administrator or to root. Simply pipe the message to the `mail` command, and use the `-s` option to add a subject line as shown in Listing 54.

Listing 54. Mailing an error message to a user

```
ian@attic4:~$ echo "Midnight error message" | mail -s "Admin error" ian
ian@attic4:~$ mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/ian": 1 message 1 new
>N 1 ian@localhost      Mon Jan 29 23:58   14/420   Admin error
&
Message 1:
From ian@localhost  Mon Jan 29 23:58:27 2007
X-Original-To: ian
To: ian@localhost
Subject: Admin error
Date: Mon, 29 Jan 2007 23:58:27 -0500 (EST)
From: ian@localhost (Ian Shields)
```

```
Midnight error message
```

```
& d  
& q
```

If you need to mail a log file, use the `<` redirection function to redirect it as input to the mail command. If you need to send several files, you can use `cat` to combine them and pipe the output to mail. In Listing 54, mail was sent to user `ian` who happened to also be the one running the command, but admin scripts are more likely to direct mail to root or another administrator. As usual, consult the man pages for mail to learn about other options that you can specify.

This brings us to the end of this tutorial. We have covered a lot of material on shells and scripting. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Shell Command Language](#)" defines the shell command language as specified by The Open Group and IEEE.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- Read these developerWorks articles for other ways to work with Bash:
 - [Bash by example, Part 1](#)
 - [Bash by example, Part 2](#)
 - [Bash by example, Part 3](#)
 - [System Administration Toolkit: Get the most out of bash](#)
 - [Working in the bash shell](#)
- Sed and awk each deserve a tutorial on their own, but read these developerWorks articles for a more complete background.
Sed:
 - [Common threads: Sed by example, Part 1](#)
 - [Common threads: Sed by example, Part 2](#)
 - [Common threads: Sed by example, Part 3](#)**Awk:**
 - [Common threads: Awk by example, Part 1](#)
 - [Common threads: Awk by example, Part 2](#)
 - [Common threads: Awk by example, Part 3](#)
 - [Get started with GAWK: AWK language fundamentals](#)
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.

- [LPI Linux Certification in a Nutshell, Second Edition](#) (O'Reilly, 2006) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- Download the developerWorks author package from "[Authoring with the developerWorks XML templates](#)" (developerWorks, January 2007).
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 109: Shells, scripting, programming, and compiling

Junior Level Administration (LPIC-1) topic 109

Skill Level: Intermediate

[Ian Shields](#)
Senior Programmer
IBM

30 Jan 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this fifth in a [series of nine tutorials](#), Ian introduces you to the Bash shell, and scripts and programming in the Bash shell. By the end of this tutorial, you will know how to customize your shell environment, use shell programming structures to create functions and scripts, set and unset environment variables, and use the various login scripts.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial

on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	(This tutorial.) Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts. See the detailed objectives below.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot

- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Shells, scripting, programming, and compiling," the fifth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to use the Bash shell, how to use shell programming structures to create functions and scripts, how to customize your shell environment, how to set and unset environment variables, and how to use the various login scripts.

The title for this tutorial duplicates the corresponding topic in the LPI 102 exam, and therefore includes "programming and compiling," but the LPI objectives limit "programming" to that required for writing shell functions and scripts. And no objectives for compiling programs are included in the topic.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.109.1 Customize and use the shell environment	Weight 5	Customize shell environments to meet user needs. Set environment variables (at login or when spawning a new shell). Write Bash functions for frequently used sequences of commands.
1.109.2 Customize or write simple scripts	Weight 3	Write simple Bash scripts and customize existing ones.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be thoroughly familiar with the material from the "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)" tutorial, as many of the building blocks for this tutorial were covered in that tutorial, especially the section "Using the command line."

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Shell customization

This section covers material for topic 1.109.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 5.

In this section, learn how to:

- Set and unset environment variables
- Use profiles to set environment variables at login or when spawning a new shell
- Write shell functions for frequently used sequences of commands
- Use command lists

Shells and environments

Before the advent of graphical interfaces, programmers used a typewriter terminal or an ASCII display terminal to connect to a UNIX® system. A typewriter terminal allowed them to type commands, and the output was usually printed on continuous paper. Most ASCII display terminals had 80 characters per line and about 25 lines on the screen, although both larger and smaller terminals existed. Programmers typed a command and pressed **Enter**, and the system interpreted and then executed the command.

While this may seem somewhat primitive today in an era of drag-and-drop graphical interfaces, it was a huge step forward from writing a program, punching cards, compiling the card deck, and running the program. With the advent of editors, programmers could even create programs as card images and compile them in a terminal session.

The stream of characters typed at a terminal provided a *standard input* stream to the

shell, and the stream of characters that the shell returned on either paper or display represented the *standard output*.

The program that accepts the commands and executes them is called a *shell*. It provides a layer between you and the intricacies of an operating system. UNIX and Linux shells are extremely powerful in that you can build quite complex operations by combining basic functions. Using programming constructs you can then build functions for direct execution in the shell or save functions as *shell scripts* so that you can reuse them over and over.

Sometimes you need to execute commands before the system has booted far enough to allow terminal connections, and sometimes you need to execute commands periodically, whether or not you are logged on. A shell can do this for you, too. The standard input and output do not have to come from or be directed to a real user at a terminal.

In this section, you learn more about shells. In particular, you learn about the *bash* or *Bourne again* shell, which is an enhancement of the original Bourne shell, along with some features from other shells and some changes from the Bourne shell to make it more POSIX compliant.

POSIX is the *Portable Operating System Interface for uniX*, which is a series of IEEE standards collectively referred to as IEEE 1003. The first of these was IEEE Standard 1003.1-1988, released in 1988. Other well known shells include the *Korn* shell (ksh), the *C* shell (csh) and its derivative tcsh, the *Almquist* shell (ash) and its Debian derivative (dash). You need to know something about many of these shells, if only to recognize when a particular script requires features from one of them.

Many aspects of your interaction with a computer will be the same from one session to another. Recall from the tutorial "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)" that when you are running in a Bash shell, you have a shell *environment*, which defines such things as the form of your prompt, your home directory, your working directory, the name of your shell, files that you have opened, functions that you have defined, and so on. The environment is made available to every shell process. Shells, including bash, allow you to create and modify *shell variables*, which you may *export* to your environment for use by other processes running in the shell or by other shells that you may spawn from the current shell.

Both environment variables and shell variables have a *name*. You reference the value of a variable by prefixing its name with '\$'. Some of the common bash environment variables that are set for you are shown in Table 3.

Table 3. Common bash environment variables	
Name	Function
USER	The name of the logged-in user
UID	The numeric user id of the

	logged-in user
HOME	The user's home directory
PWD	The current working directory
SHELL	The name of the shell
\$	The process id (or <i>PID</i> of the running Bash shell (or other) process)
PPID	The process id of the process that started this process (that is, the id of the parent process)
?	The exit code of the last command

Setting variables

In the Bash shell, you create or *set* a shell variable by typing a name followed immediately by an equal sign (=). Variable names (or *identifiers*) are words consisting only of alphanumeric characters and underscores, that begin with an alphabetic character or an underscore. Variables are case sensitive, so `var1` and `VAR1` are different variables. By convention, variables, particularly exported variables, are upper case, but this is not a requirement. Technically, `$$` and `$?` are shell *parameters* rather than variables. They may only be referenced; you cannot assign a value to them.

When you create a shell variable, you will often want to *export* it to the environment so it will be available to other processes that you start from this shell. Variables that you export are **not** available to a parent shell. You use the `export` command to export a variable name. As a shortcut in bash, you can assign and export in one step.

To illustrate assignment and exporting, let's run the bash command while in the Bash shell and then run the Korn shell (ksh) from the new Bash shell. We will use the `ps` command to display information about the command that is running.

Listing 1. Setting and exporting shell variables

```
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
30576 30575 -bash
[ian@echidna ian]$ bash
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16353 30576 bash
[ian@echidna ian]$ VAR1=var1
[ian@echidna ian]$ VAR2=var2
[ian@echidna ian]$ export VAR2
```

```

[ian@echidna ian]$ export VAR3=var3
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3
var1 var2 var3
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $SHELL
var1 var2 var3 /bin/bash
[ian@echidna ian]$ ksh
$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16448 16353 ksh
$ export VAR4=var4
$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
var2 var3 var4 /bin/bash
$ exit
$ [ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
var1 var2 var3 /bin/bash
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
16353 30576 bash
[ian@echidna ian]$ exit
[ian@echidna ian]$ ps -p $$ -o "pid ppid cmd"
  PID  PPID  CMD
30576 30575 -bash
[ian@echidna ian]$ echo $VAR1 $VAR2 $VAR3 $VAR4 $SHELL
/bin/bash

```

Notes:

1. At the start of this sequence, the Bash shell had PID 30576.
2. The second Bash shell has PID 16353, and its parent is PID 30576, the original Bash shell.
3. We created VAR1, VAR2, and VAR3 in the second Bash shell, but only exported VAR2 and VAR3.
4. In the Korn shell, we created VAR4. The `echo` command displayed values only for VAR2, VAR3, and VAR4, confirming that VAR1 was not exported. Were you surprised to see that the value of the SHELL variable had not changed, even though the prompt had changed? You cannot always rely on SHELL to tell you what shell you are running under, but the `ps` command does tell you the actual command. Note that `ps` puts a hyphen (-) in front of the first Bash shell to indicate that this is the *login shell*.
5. Back in the second Bash shell, we can see VAR1, VAR2, and VAR3.
6. And finally, when we return to the original shell, none of our new variables still exist.

Listing 2 shows what you might see in some of these common bash variables.

Listing 2. Environment and shell variables

```
[ian@echidna ian]$ echo $USER $UID
ian 500
[ian@echidna ian]$ echo $SHELL $HOME $PWD
/bin/bash /home/ian /home/ian
[ian@echidna ian]$ (exit 0);echo $?;(exit 4);echo $?
0
4
[ian@echidna ian]$ echo $$ $PPID
30576 30575
```

Environments and the C shell

In shells such as the C and tcsh shells, you use the `set` command to set variables in your shell, and the `setenv` command to set and export variables. The syntax differs slightly from that of the `export` command as illustrated in Listing 3. Note the equals (=) sign when using `set`.

Listing 3. Setting environment variables in the C shell

```
ian@attic4:~$ echo $VAR1 $VAR2

ian@attic4:~$ csh
% set VAR1=var1
% setenv VAR2 var2
% echo $VAR1 $VAR2
var1 var2
% bash
ian@attic4:~$ echo $VAR1 $VAR2
var2
```

Unsetting variables

You remove a variable from the Bash shell using the `unset` command. You can use the `-v` option to be sure that you are removing a variable definition. Functions can have the same name as variables, so use the `-f` if you want to remove a function definition. Without either `-f` or `-v`, the bash `unset` command removes a variable definition if it exists; otherwise, it removes a function definition if one exists. (Functions are covered in more detail later in the [Shell functions](#) section.)

Listing 4. The bash unset command

```
ian@attic4:~$ VAR1=var1
ian@attic4:~$ VAR2=var2
ian@attic4:~$ echo $VAR1 $VAR2
var1 var2
ian@attic4:~$ unset VAR1
ian@attic4:~$ echo $VAR1 $VAR2
var2
ian@attic4:~$ unset -v VAR2
```

```
ian@attic4:~$ echo $VAR1 $VAR2
```

The bash default is to treat unset variables as if they had an empty value, so you might wonder why you would unset a variable rather than just assign it an empty value. Bash and many other shells allow you to generate an error if an undefined variable is referenced. Use the command `set -u` to generate an error for undefined variables, and `set +u` to disable the warning. Listing 5 illustrates these points.

Listing 5. Generating errors with unset variables

```
ian@attic4:~$ set -u
ian@attic4:~$ VAR1=var1
ian@attic4:~$ echo $VAR1
var1
ian@attic4:~$ unset VAR1
ian@attic4:~$ echo $VAR1
-bash: VAR1: unbound variable
ian@attic4:~$ VAR1=
ian@attic4:~$ echo $VAR1

ian@attic4:~$ unset VAR1
ian@attic4:~$ echo $VAR1
-bash: VAR1: unbound variable
ian@attic4:~$ unset -v VAR1
ian@attic4:~$ set +u
ian@attic4:~$ echo $VAR1

ian@attic4:~$
```

Note that it is not an error to unset a variable that does not exist, even when `set -u` has been specified.

Profiles

When you log in to a Linux system, your id has a default shell, which is your *login* shell. If this shell is bash, then it executes several profile scripts before you get control. If `/etc/profile` exists, it is executed first. Depending on your distribution, other scripts in the `/etc` tree may also be executed, for example, `/etc/bash.bashrc` or `/etc/bashrc`. Once the system scripts have run, a script in your home directory is run if it exists. Bash looks for the files `~/.bash_profile`, `~/.bash_login`, and `~/.profile` in that order. The first one found is executed.

When you log off, bash executes the `~/.bash_logout` script from your home directory if it exists.

Once you have logged in and are already using bash, you may start another shell, called an *interactive* shell to run a command, for example to run a command in the background. In this case, bash executes only the `~/.bashrc` script, assuming one exists. It is common to check for this script in your `~/.bash_profile`, so that you can

execute it at login as well as when starting an interactive shell, using commands such as those shown in Listing 6.

Listing 6. Checking for ~/.bashrc

```
# include .bashrc if it exists
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

You may force bash to read profiles as if it were a login shell using the `--login` option. If you do not want to execute the profiles for a login shell, specify the `--noprofile` option. Similarly, if you want to disable execution of the `~/.bashrc` file for an interactive shell, start bash with the `--norc` option. You can also force bash to use a file other than `~/.bashrc` by specifying the `--rcfile` option with the name of the file you want to use. Listing 8 illustrates creation of a simple file called `testrc` and its use with the `--rcfile` option. Note that the `VAR1` variable is **not** set in the outer shell, but has been set for the inner shell by the `testrc` file.

Listing 7. Using the --rcfile option

```
ian@attic4:~$ echo VAR1=var1>testrc
ian@attic4:~$ echo $VAR1

ian@attic4:~$ bash --rcfile testrc
ian@attic4:~$ echo $VAR1
var1
```

Starting bash in other ways

In addition to the standard ways of running bash from a terminal as outlined above, bash may also be used in other ways.

Unless you *source* a script to run in the current shell, it will run in its own *non-interactive* shell, and the above profiles are not read. However, if the `BASH_ENV` variable is set, bash expands the value and assumes it is the name of a file. If the file exists, then bash executes the file before whatever script or command it is executing in the non-interactive shell. Listing 8 uses two simple files to illustrate this.

Listing 8. Using BASH_ENV

```
ian@attic4:~$ cat testenv.sh
#!/bin/bash
echo "Testing the environment"
ian@attic4:~$ cat somescript.sh
#!/bin/bash
```

```
echo "Doing nothing"
ian@attic4:~$ export BASH_ENV=~/.testenv.sh
ian@attic4:~$ ./somescript.sh
Testing the environment
Doing nothing
```

Non-interactive shells may also be started with the `--login` option to force execution of the profile files.

Bash may also be started in *POSIX* mode using the `--posix` option. This mode is similar to the non-interactive shell, except that the file to execute is determined from the ENV environment variable.

It is common in Linux systems to run bash as `/bin/sh` using a symbolic link. When bash detects that it is being run under the name `sh`, it attempts to follow the startup behavior of the older Bourne shell while still conforming to POSIX standards. When run as a login shell, bash attempts to read and execute `/etc/profile` and `~/.profile`. When run as an interactive shell using the `sh` command, bash attempts to execute the file specified by the ENV variable as it does when invoked in POSIX mode. When run interactively as `sh`, it **only** uses a file specified by the ENV variable; the `--rcfile` option will always be ignored.

If bash is invoked by the remote shell daemon, then it behaves as an interactive shell, using the `~/.bashrc` file if it exists.

Shell aliases

The Bash shell allows you to define *aliases* for commands. The most common reasons for aliases are to provide an alternate name for a command, or to provide some default parameters for the command. The `vi` editor has been a staple of UNIX and Linux systems for many years. The `vim` (Vi IMproved) editor is like `vi`, but with many improvements. So if you are used to typing "vi" when you want an editor, but you would really prefer to use `vim`, then an alias is for you. Listing 9 shows how to use the `alias` command to accomplish this.

Listing 9. Using vi as an alias for vim

```
[ian@pinguino ~]$ alias vi='vim'
[ian@pinguino ~]$ which vi
alias vi='vim'
/usr/bin/vim
[ian@pinguino ~]$ /usr/bin/which vi
/bin/vi
```

Notice in this example that if you use the `which` command to see where the `vi` program lives, you get two lines of output: the first shows the alias, and the second

the location of vim (/usr/bin/vim). However, if you use the `which` command with its full path (/usr/bin/which), you get the location of the `vi` command. If you guessed that this might mean that the `which` command itself is aliased on this system you would be right.

You can also use the `alias` command to display all the aliases if you use it with no options or with just the `-p` option, and you can display the aliases for one or more names by giving the names as arguments without assignments. Listing 10 shows the aliases for `which` and `vi`.

Listing 10. Aliases for which and vi

```
[ian@pinguino ~]$ alias which vi
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
alias vi='vim'
```

The alias for the `which` command is rather curious. Why pipe the output of the `alias` command (with no arguments) to `/usr/bin/which`? If you check the man pages for the `which` command, you will find that the `--read-alias` option instructs `which` to read a list of aliases from stdin and report matches on stdout. This allows the `which` command to report aliases as well as commands from your PATH, and is so common that your distribution may have set it up as a default for you. This is a good thing to do since the shell will execute an alias before a command of the same name. So now that you know this, you can check it using `alias which`. You can also learn whether this type of alias has been set for `which` by running `which which`.

Another common use for aliases is to add parameters automatically to commands, as you saw above for the `--read-alias` and several other parameters on the `which` command. This technique is often done for the root user with the `cp`, `mv`, and `rm` commands so that a prompt is issued before files are deleted or overwritten. This is illustrated in Listing 11.

Listing 11. Adding parameters for safety

```
[root@pinguino ~]# alias cp mv rm
alias cp='cp -i'
alias mv='mv -i'
alias rm='rm -i'
```

Command lists

In the earlier tutorial "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)," you learned about command *sequences* or *lists*. You have just seen the pipe (`|`) operator used with an alias, and you can use command lists as well. Suppose, for a

simple example, that you want a command to list the contents of the current directory and also the amount of space used by it and all its subdirectories. Let's call it the `lsdu` command. So you simply assign a sequence of the `ls` and `du` commands to the alias `lsdu`. Listing 12 shows the wrong way to do this and also the right way. Look carefully at it before you read, and think about why the first attempt did not work.

Listing 12. Aliases for command sequences

```
[ian@pinguino developerworks]$ alias lsdu=ls;du -sh # Wrong way
2.9M .
[ian@pinguino developerworks]$ lsdu
a tutorial new-article.sh new-tutorial.sh readme tools xsl
my-article new-article.vbs new-tutorial.vbs schema web
[ian@pinguino developerworks]$ alias 'lsdu=ls;du -sh' # Right way way
[ian@pinguino developerworks]$ lsdu
a tutorial new-article.sh new-tutorial.sh readme tools xsl
my-article new-article.vbs new-tutorial.vbs schema web
2.9M .
```

You need to be very careful to quote the full sequence that will make up the alias. You also need to be very careful about whether you use double or single quotes if you have shell variables as part of the alias. Do you want the shell to expand the variables when the alias is defined or when it is executed? Listing 13 shows the wrong way to create a custom command called `mywd` intended to print your working directory name

Listing 13. Custom pwd - attempt 1

```
[ian@pinguino developerworks]$ alias mywd="echo \"My working directory is $PWD\""
[ian@pinguino developerworks]$ mywd
My working directory is /home/ian/developerworks
[ian@pinguino developerworks]$ cd ..
[ian@pinguino ~]$ mywd
My working directory is /home/ian/developerworks
```

Remember that the double quotes cause bash to expand variables before executing a command. Listing 14 uses the `alias` command to show what the resulting alias actually is, from which our error is evident. Listing 14 also shows a correct way to define this alias.

Listing 14. Custom pwd - attempt 2

```
[ian@pinguino developerworks]$ alias mywd
alias mywd='echo \"My working directory is $PWD\"'
[ian@pinguino developerworks]$ mywd
"My working directory is /home/ian/developerworks"
[ian@pinguino developerworks]$ cd ..
[ian@pinguino ~]$ mywd
"My working directory is /home/ian"
```

Success at last.

Shell functions

Aliases allow you to use an abbreviation or alternate name for a command or command list. You may have noticed that you can add additional things, such as the program name you are seeking with the `which` command. When your input is executed, the alias is expanded, and anything else you type after that is added to the expansion before the final command or list is executed. This means that you can only add parameters to the end of the command or list, and you can use them only with the final command. Functions provide additional capability, including the ability to process parameters. Functions are part of the POSIX shell definition. They are available in shells such as `bash`, `dash`, and `ksh`, but are not available in `csh` or `tosh`.

In the next few paragraphs, you'll build a complex command piece-by-piece from smaller building blocks, refining it each step of the way and turning it into a function that you will further refine.

A hypothetical problem

You can use the `ls` command to list a variety of information about directories and files in your file system. Suppose you would like a command, let's call it `ldirs`, that will list directory names with output like that in Listing 15.

Listing 15. The `ldirs` command output

```
[ian@pinguino developerworks]$ ldirs *[st]* tools/*a*
my dw article
schema
tools
tools/java
xsl
```

To keep things relatively simple, the examples in this section use the directories and files from the developerWorks author package (see [Resources](#)), which you can use if you'd like to write articles or tutorials for developerWorks. In these examples, we used the `new-article.sh` script from the package to create a template for a new article that we've called "my dw article".

At the time of writing, the version of the developerWorks author package is 5.6, so you may see differences if you use a later version. Or just use your own files and directories. The `ldirs` command will handle those too. You'll find additional bash function examples in the tools that come with the developerWorks author package.

Finding directory entries

Ignoring the `*[st]* tools/*a*` for the moment, if you use the `ls` command with the color options as shown in the aliases above, you will see output similar to that shown in Figure 1.

Figure 1. Distinguishing files and directories with the `ls` command

```

ian@pinguino:~/developerworks
File Edit View Terminal Tabs Help
[ian@pinguino developerworks]$ ls
my dw article  new-article.vbs new-tutorial.vbs schema web
new-article.sh new-tutorial.sh readme          tools  xsl
[ian@pinguino developerworks]$ ls -l
total 80
drwxrwxr-x 2 ian ian 4096 Jan 24 17:06 my dw article
-rwxr--r-- 1 ian ian  215 Sep 27 16:34 new-article.sh
-rwxr--r-- 1 ian ian 1078 Sep 27 16:34 new-article.vbs
-rwxr--r-- 1 ian ian  216 Sep 27 16:34 new-tutorial.sh
-rwxr--r-- 1 ian ian 1079 Sep 27 16:34 new-tutorial.vbs
drwxrwxr-x 2 ian ian 4096 Jan 18 16:23 readme
drwxrwxr-x 3 ian ian 4096 Jan 19 07:41 schema
drwxrwxr-x 3 ian ian 4096 Jan 19 15:08 tools
drwxrwxr-x 3 ian ian 4096 Jan 17 16:03 web
drwxrwxr-x 3 ian ian 4096 Jan 19 10:59 xsl
[ian@pinguino developerworks]$

```

The directories are shown in dark blue in this example, but that's a bit hard to decode with the skills you have developed in this series of tutorials. Using the `-l` option, though, gives a clue on how to proceed: directory listings have a 'd' in the first position. So your first step might be to simply filter these from the long listing using `grep` as shown in Listing 16.

Listing 16. Using `grep` to find just directory entries

```

[ian@pinguino developerworks]$ ls -l | grep "^d"
drwxrwxr-x 2 ian ian 4096 Jan 24 17:06 my dw article
drwxrwxr-x 2 ian ian 4096 Jan 18 16:23 readme
drwxrwxr-x 3 ian ian 4096 Jan 19 07:41 schema
drwxrwxr-x 3 ian ian 4096 Jan 19 15:08 tools
drwxrwxr-x 3 ian ian 4096 Jan 17 16:03 web
drwxrwxr-x 3 ian ian 4096 Jan 19 10:59 xsl

```

Trimming the directory entries

You might consider using `awk` instead of `grep` so that in one pass you can filter the list and strip off the last part of each line, which is the directory name, as shown in Listing 17.

Listing 17. Using `awk` instead

```

[ian@pinguino developerworks]$ ls -l | awk '/^d/ { print $NF } '
article
readme
schema

```

```
tools
web
xsl
```

The problem with the approach in Listing 17 is that it doesn't handle the directory with spaces in the name, such as "my dw article". As with most things in Linux and life, there are often several ways to solve a problem, but the objective here is to learn about functions, so let's return to using `grep`. Another tool you learned about earlier in this series is `cut`, which cuts fields out of a file, including stdin. Looking back at Listing 16 again, you see eight blank-delimited fields before the filename. Adding `cut` to the previous command gives you output as shown in Listing 18. Note that the `-f9-` option tells `cut` to print fields 9 and above.

Listing 18. Using `cut` to trim names

```
[ian@pinguino developerworks]$ ls -l | grep "^d" | cut -d" " -f9-
my dw article
readme
schema
tools
web
xsl
```

A small problem with our approach is made obvious if we try our command on the `tools` directory instead of on the current directory as shown in Listing 19.

Listing 19. A problem with `cut`

```
[ian@pinguino developerworks]$ ls -l tools | grep "^d" | cut -d" " -f9-
11:25 java
[ian@pinguino developerworks]$ ls -ld tools/[fjt]*
-rw-rw-r-- 1 ian ian 4798 Jan 8 14:38 tools/figure1.gif
drwxrwxr-x 2 ian ian 4096 Oct 31 11:25 tools/java
-rw-rw-r-- 1 ian ian 39431 Jan 18 23:31 tools/template-dw-article-5.6.xml
-rw-rw-r-- 1 ian ian 39407 Jan 18 23:32 tools/template-dw-tutorial-5.6.xml
```

How did the timestamp get in there? The two template files have 5-digit sizes, while the `java` directory has only a 4-digit size, so `cut` interpreted the extra space as another field separator.

Use `seq` to find a cut point

The `cut` command can also cut using character positions instead of fields. Rather than counting characters, the Bash shell has lots of utilities that you can use, so you might try using the `seq` and `printf` commands to print a ruler above your long directory listing so you can easily figure where to cut the lines of output. The `seq` command takes up to three arguments, which allow you to print all the numbers up to a given value, all the numbers from one value to another, or all the numbers from one value, stepping by a given value, up to a third value. See the man pages for all

the other fancy things you can do with `seq`, including printing octal or hexadecimal numbers. For now let's use `seq` and `printf` to print a ruler with positions marked every 10 characters as shown in Listing 20.

Listing 20. Printing a ruler with `seq` and `printf`

```
[ian@pinguino developerworks]$ printf "....+...%2.d" `seq 10 10 60`;printf "\n";ls -l
....+...10....+...20....+...30....+...40....+...50....+...60
total 88
drwxrwxr-x 2 ian ian 4096 Jan 24 17:06 my dw article
-rwxr--r-- 1 ian ian 215 Sep 27 16:34 new-article.sh
-rwxr--r-- 1 ian ian 1078 Sep 27 16:34 new-article.vbs
-rwxr--r-- 1 ian ian 216 Sep 27 16:34 new-tutorial.sh
-rwxr--r-- 1 ian ian 1079 Sep 27 16:34 new-tutorial.vbs
drwxrwxr-x 2 ian ian 4096 Jan 18 16:23 readme
drwxrwxr-x 3 ian ian 4096 Jan 19 07:41 schema
drwxrwxr-x 3 ian ian 4096 Jan 19 15:08 tools
drwxrwxr-x 3 ian ian 4096 Jan 17 16:03 web
drwxrwxr-x 3 ian ian 4096 Jan 19 10:59 xsl
```

Aha! Now you can use the command `ls -l | grep "^d" | cut -c40-` to cut lines starting at position 40. A moment's reflection reveals that this doesn't really solve the problem either, because larger files will move the correct cut position to the right. Try it for yourself.

Sed to the rescue

Sometimes called the "Swiss army knife" of the UNIX and Linux toolbox, `sed` is an extremely powerful editing filter that uses regular expressions. You now understand that the challenge is to strip off the first 8 words and the blanks that follow them from every line of output that begins with 'd'. You can do it all with `sed`: select only those lines you are interested in using the pattern-matching expression `/^d/`, substituting a null string for the first eight words using the substitute command `s/^d\([^\]* *\)\{8\}/`. Use the `-n` option to print only lines that you specify with the `p` command as shown in Listing 21.

Listing 21. Trimming directory names with `sed`

```
[ian@pinguino developerworks]$ ls -l | sed -ne 's/^d\([^\ ]* *\)\{8\}/p'
my dw article
readme
schema
tools
web
xsl
[ian@pinguino developerworks]$ ls -l tools | sed -ne 's/^d\([^\ ]* *\)\{8\}/p'
java
```

To learn more about `sed`, see the [Resources](#) section.

A function at last

Now that you have the complex command that you want for your `ldirs` function, it's time to learn about making it a function. A function consists of a name followed by `()` and then a compound command. For now, a compound command will be any command or command list, terminated by a semicolon and surrounded by braces (which must be separated from other tokens by white space). You will learn about other compound commands in the [Shell scripts](#) section.

Note: In the Bash shell, a function name may be preceded by the word 'function', but this is not part of the POSIX specification and is not supported by more minimalist shells such as dash. In the [Shell scripts](#) section, you will learn how to make sure that a script is interpreted by a particular shell, even if you normally use a different shell.

Inside the function, you can refer to the parameters using the bash special variables in Table 4. You prefix these with a `$` symbol to reference them as with other shell variables.

Table 4. Shell parameters for functions	
Parameter	Purpose
0, 1, 2, ...	The positional parameters starting from parameter 0. Parameter 0 refers to the name of the program that started bash, or the name of the shell script if the function is running within a shell script. See the bash man pages for information on other possibilities, such as when bash is started with the <code>-c</code> parameter. A string enclosed in single or double quotes will be passed as a single parameter, and the quotes will be stripped. In the case of double quotes, any shell variables such as <code>\$HOME</code> will be expanded before the function is called. You will need to use single or double quotes to pass parameters that contain embedded blanks or other characters that might have special meaning to the shell.
*	The positional parameters starting from parameter 1. If the expansion is done within double quotes, then the expansion is a single word with the first character of the interfield separator (IFS) special variable separating the parameters or no intervening space if IFS is null. The default IFS value is a blank, tab, and newline. If IFS is unset, then the separator used is a blank, just as for the default IFS.

@	The positional parameters starting from parameter 1. If the expansion is done within double quotes, then each parameter becomes a single word, so that "\$@" is equivalent to "\$1" "\$2" If your parameters are likely to contain embedded blanks, you will want to use this form.
#	The number of parameters, not including parameter 0.

Note: If you have more than 9 parameters, you cannot use \$10 to refer to the tenth one. You must first either process or save the first parameter (\$1), then use the `shift` command to drop parameter 1 and move all remaining parameters down 1, so that \$10 becomes \$9 and so on. The value of \$# will be updated to reflect the remaining number of parameters.

Now you can define a simple function to do nothing more than tell you how many parameters it has and display them as shown in Listing 22.

Listing 22. Function parameters

```
[ian@pinguino developerworks]$ testfunc () { echo "$# parameters"; echo "$@"; }
[ian@pinguino developerworks]$ testfunc
0 parameters

[ian@pinguino developerworks]$ testfunc a b c
3 parameters
a b c
[ian@pinguino developerworks]$ testfunc a "b c"
2 parameters
a b c
```

Whether you use \$*, "\$*", \$@, or "\$@", you won't see much difference in the output of the above function, but rest assured that when things become more complex, the distinctions will matter very much.

Now take the complex command that we tested up to this point and create a `ldirs` function with it, using "\$@" to represent the parameters. You can enter all of the function on a single line as you did in the previous example, or `bash` lets you enter commands on multiple lines, in which case a semicolon will be added automatically as shown in Listing 23. Listing 23 also shows the use of the `type` command to display the function definition. Note from the output of `type` that the `ls` command has been replaced by the expanded value of its alias. You could use `/bin/ls` instead of plain `ls` if you needed to avoid this.

Listing 23. Your first `ldirs` function

```
[ian@pinguino developerworks]$ # Enter the function on a single line
[ian@pinguino developerworks]$ ldirs () { ls -l "$@"|sed -ne 's/^d\([^ ]* *\)\{8\} //p'; }
[ian@pinguino developerworks]$ # Enter the function on multiple lines
[ian@pinguino developerworks]$ ldirs ()
> {
> ls -l "$@"|sed -ne 's/^d\([^ ]* *\)\{8\} //p'
> }
[ian@pinguino developerworks]$ type ldirs
ldirs is a function
ldirs ()
{
    ls --color=tty -l "$@" | sed -ne 's/^d\([^ ]* *\)\{8\} //p'
}
[ian@pinguino developerworks]$ ldirs
my dw article
readme
schema
tools
web
xsl
[ian@pinguino developerworks]$ ldirs tools
java
```

So now your function appears to be working. But what happens if you run `ldirs *` as shown in Listing 24?

Listing 24. Running `ldirs *`

```
[ian@pinguino developerworks]$ ldirs *
5.6
java
www.ibm.com
5.6
```

Are you surprised? You didn't find directories in the current directory, but rather second-level subdirectories. Review the man page for the `ls` command or our earlier tutorials in this series to understand why. Or run the `find` command as shown in Listing 25 to print the names of second-level subdirectories.

Listing 25. Finding second-level subdirectories

```
[ian@pinguino developerworks]$ find . -mindepth 2 -maxdepth 2 -type d
./tools/java
./web/www.ibm.com
./xsl/5.6
./schema/5.6
```

Adding some tests

Using wildcards has exposed a problem with the logic in this approach. We blithely ignored the fact that `ldirs` without any parameters displayed the subdirectories in the current directory, while `ldirs tools` displayed the `java` subdirectory of the `tools` directory rather than the `tools` directory itself as you would expect using `ls` with files rather than directories. Ideally, you should use `ls -l` if no parameters are

given and `ls -ld` if some parameters are given. You can use the `test` command to test the number of parameters and then use `&&` and `||` to build a command list that executes the appropriate command. Using the `[test expression]` form of `test`, your expression might look like `{ [$# -gt 0] &&/bin/ls -ld "$@" || /bin/ls -l } | sed -ne`

There is a small issue with this code, though, in that if the `ls -ld` command doesn't find any matching files or directories, it will issue an error message and return with a non-zero exit code, thus causing the `ls -l` command to be executed as well. Perhaps not what you wanted. One answer is to construct a compound command for the first `ls` command so that the number of parameters is tested again if the command fails. Expand the function to include this, and your function should now appear as in Listing 26. Try using it with some of the parameters in Listing 26, or experiment with your own parameters to see how it behaves.

Listing 26. Handling wildcards with `ldirs`

```
[ian@pinguino ~]$ type ldirs
ldirs is a function
ldirs ()
{
    {
        [ $# -gt 0 ] && {
            /bin/ls -ld "$@" || [ $# -gt 0 ]
        } || /bin/ls -l
    } | sed -ne 's/^d\([^ ]* *\)\{8\} //p'
}
[ian@pinguino developerworks]$ ldirs *
my dw article
readme
schema
tools
web
xsl
[ian@pinguino developerworks]$ ldirs tools/*
tools/java
[ian@pinguino developerworks]$ ldirs *xxx*
/bin/ls: *xxx*: No such file or directory
[ian@pinguino developerworks]$ ldirs *a* *s*
my dw article
readme
schema
schema
tools
xsl
```

Final touchup

At this point you might get a directory listed twice as in the last example of Listing 26. You could extend the pipeline by piping the `sed` output through `sort | uniq` if you wish.

Starting from some small building blocks, you have now built quite a complex shell function.

Customizing keystrokes

The keystrokes you type at a terminal session, and also those used in programs such as FTP, are processed by the readline library and can be configured. By default, the customization file is `.inputrc` in your home directory, which will be read during bash startup if it exists. You can configure a different file by setting the `INPUTRC` variable. If it is not set, `.inputrc` in your home directory will be used. Many systems have a default key mapping in `/etc/inputrc`, so you will normally want to include these using the `$include` directive.

Listing 27 illustrates how you might bind your `ldirs` function to the Ctrl-t key combination (press and hold Ctrl, then press t). If you want the command to be executed with no parameters, add `\n` to the end of the configuration line.

Listing 27. Sample `.inputrc` file

```
# My custom key mappings
$include /etc/inputrc
```

You can force the `INPUTRC` file to be read again by pressing Ctrl-x then Ctrl-r. Note that some distributions will set `INPUTRC=/etc/inputrc` if you do not have your own `.inputrc`, so if you create one on such a system, you will need to log out and log back in to pick up your new definitions. Just resetting `INPUTRC` to null or to point to your new file will reread the original file, not the new specification.

The `INPUTRC` file can include conditional specifications. For example, the behavior of your keyboard should be different according to whether you are using emacs editing mode (the bash default) or vi mode. See the man pages for bash for more details on how to customize your keyboard.

Saving aliases and functions

You will probably add your aliases and functions to your `~/.bashrc` file, although you may save them in any file you like. Whichever you do, remember to source the file or files using the `source` or `.` command so that the contents of your file will be read and executed in the current environment. If you create a script and just execute it, it will be executed in a subshell and all your valuable customization will be lost when the subshell exits and returns control to you.

In the next section, you learn how to go beyond simple functions. You learn how to add programming constructs such as conditional tests and looping constructs and combine these with multiple functions to create or modify Bash shell scripts.

Section 3. Shell scripts

This section covers material for topic 1.109.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Use standard shell syntax, such as loops and tests
- Use command substitution
- Test return values for success or failure or other information provided by a command
- Perform conditional mailing to the superuser
- Select the correct script interpreter through the shebang (#!) line
- Manage the location, ownership, execution, and suid-rights of scripts

This section builds on what you learned about simple functions in the last section and demonstrates some of the techniques and tools that add programming capability to the shell. You have already see simple logic using the `&&` and `||` operators, which allow you to execute one command based on whether the previous command exits normally or with an error. In the `ldirs` function, you used this to alter the call to `ls` according to whether or not parameters were passed to your `ldirs` function. Now you will learn how to extend these basic techniques to more complex shell programming.

Tests

The first thing you need to do in any kind of programming language after learning how to assign values to variables and pass parameters is to test those values and parameters. In shells the tests you do set the return status, which is the same thing that other commands do. In fact, `test` is a builtin *command*!

test and [

The `test` builtin command returns 0 (True) or 1 (False) depending on the evaluation of an expression *expr*. You can also use square brackets so that `test expr` and `[expr]` are equivalent. You can examine the return value by displaying `$?`; you can use the return value as you have before with `&&` and `||`; or you can test it using the various conditional constructs that are covered later in this section.

Listing 28. Some simple tests

```
[ian@pinguino ~]$ test 3 -gt 4 && echo True || echo false
false
[ian@pinguino ~]$ [ "abc" != "def" ];echo $?
0
[ian@pinguino ~]$ test -d "$HOME" ;echo $?
0
```

In the first of these examples, the `-gt` operator was used to perform an arithmetic comparison between two literal values. In the second, the alternate `[]` form was used to compare two strings for inequality. In the final example, the value of the `HOME` variable is tested to see if it is a directory using the `-d` unary operator.

Arithmetic values may be compared using one of `-eq`, `-ne`, `-lt`, `-le`, `-gt`, or `-ge`, meaning equal, not equal, less than, less than or equal, greater than, and greater than or equal, respectively.

Strings may be compared for equality, inequality, or whether the first string sorts before or after the second one using the operators `=`, `!=`, `<` and `>`, respectively. The unary operator `-z` tests for a null string, while `-n` or no operator at all returns True if a string is not empty.

Note: the `<` and `>` operators are also used by the shell for redirection, so you must escape them using `\<` or `\>`. Listing 29 shows some more examples of string tests. Check that they are as you expect.

Listing 29. Some string tests

```
[ian@pinguino ~]$ test "abc" = "def" ;echo $?
1
[ian@pinguino ~]$ [ "abc" != "def" ];echo $?
0
[ian@pinguino ~]$ [ "abc" \< "def" ];echo $?
0
[ian@pinguino ~]$ [ "abc" \> "def" ];echo $?
1
[ian@pinguino ~]$ [ "abc" \<"abc" ];echo $?
1
[ian@pinguino ~]$ [ "abc" \> "abc" ];echo $?
1
```

Some of the more common file tests are shown in Table 5. The result is True if the file tested is a file that exists and that has the specified characteristic.

Table 5. Some file tests	
Operator	Characteristic
<code>-d</code>	Directory
<code>-e</code>	Exists (also <code>-a</code>)

-f	Regular file
-h	Symbolic link (also -L)
-p	Named pipe
-r	Readable by you
-s	Not empty
-S	Socket
-w	Writable by you
-N	Has been modified since last being read

In addition to the unary tests above, two files can be compared with the binary operators shown in Table 6.

Table 6. Testing pairs of files	
Operator	True if
-nt	Test if file1 is newer than file 2. The modification date is used for this and the next comparison.
-ot	Test if file1 is older than file 2.
-ef	Test if file1 is a hard link to file2.

Several other tests allow you to check things such as the permissions of the file. See the man pages for `bash` for more details or use `help test` to see brief information on the test builtin. You can use the `help` command for other builtins too.

The `-o` operator allows you to test various shell options that may be set using `set -o option`, returning True (0) if the option is set and False (1) otherwise, as shown in Listing 30.

Listing 30. Testing shell options

```
[ian@pinguino ~]$ set +o nounset
[ian@pinguino ~]$ [ -o nounset ];echo $?
1
[ian@pinguino ~]$ set -u
[ian@pinguino ~]$ test -o nounset; echo $?
0
```

Finally, the `-a` and `-o` options allow you to combine expressions with logical AND and OR, respectively, while the unary `!` operator inverts the sense of the test. You

may use parentheses to group expressions and override the default precedence. Remember that the shell will normally run an expression between parentheses in a subshell, so you will need to escape the parentheses using `\(` and `\)` or enclosing these operators in single or double quotes. Listing 31 illustrates the application of de Morgan's laws to an expression.

Listing 31. Combining and grouping tests

```
[ian@pinguino ~]$ test "a" != "$HOME" -a 3 -ge 4 ; echo $?
1
[ian@pinguino ~]$ [ ! \( "a" = "$HOME" -o 3 -lt 4 \) ]; echo $?
1
[ian@pinguino ~]$ [ ! \( "a" = "$HOME" -o '(' 3 -lt 4 ')' ')' ]; echo $?
1
```

((and [[

The `test` command is very powerful, but somewhat unwieldy with its requirement for escaping and the difference between string and arithmetic comparisons. Fortunately `bash` has two other ways of testing that are somewhat more natural for people who are familiar with C, C++, or Java syntax. The `(())` *compound command* evaluates an arithmetic expression and sets the exit status to 1 if the expression evaluates to 0, or to 0 if the expression evaluates to a non-zero value. You do not need to escape operators between `((` and `))`. Arithmetic is done on integers. Division by 0 causes an error, but overflow does not. You may perform the usual C language arithmetic, logical, and bitwise operations. The `let` command can also execute one or more arithmetic expressions. It is usually used to assign values to arithmetic variables.

Listing 32. Assigning and testing arithmetic expressions

```
[ian@pinguino ~]$ let x=2 y=2**3 z=y*3;echo $? $x $y $z
0 2 8 24
[ian@pinguino ~]$ (( w=(y/x) + ( (~ ++x) & 0x0f ) )); echo $? $x $y $w
0 3 8 16
[ian@pinguino ~]$ (( w=(y/x) + ( (~ ++x) & 0x0f ) )); echo $? $x $y $w
0 4 8 13
```

As with `(())`, the `[[]]` compound command allows you to use more natural syntax for filename and string tests. You can combine tests that are allowed for the `test` command using parentheses and logical operators.

Listing 33. Using the [[compound

```
[ian@pinguino ~]$ [[ ( -d "$HOME" ) && ( -w "$HOME" ) ]] &&
> echo "home is a writable directory"
home is a writable directory
```

The `[]` compound can also do pattern matching on strings when the `=` or `!=` operators are used. The match behaves as for wildcard globbing as illustrated in Listing 34.

Listing 34. Wildcard tests with `[]`

```
[ian@pinguino ~]$ [[ "abc def .d,x--" == a[abc]*\ ?d* ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def c" == a[abc]*\ ?d* ]]; echo $?
1
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* ]]; echo $?
1
```

You can even do arithmetic tests within `[]` compounds, but be careful. Unless within a `(())` compound, the `<` and `>` operators will compare the operands as strings and test their order in the current collating sequence. Listing 35 illustrates this with some examples.

Listing 35. Including arithmetic tests with `[]`

```
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || (( 3 > 2 )) ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || 3 -gt 2 ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || 3 > 2 ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || a > 2 ]]; echo $?
0
[ian@pinguino ~]$ [[ "abc def d,x" == a[abc]*\ ?d* || a -gt 2 ]]; echo $?
-bash: a: unbound variable
```

Conditionals

While you could accomplish a huge amount of programming with the above tests and the `&&` and `||` control operators, bash includes the more familiar "if, then, else" and case constructs. After you learn about these, you will learn about looping constructs and your toolbox will really expand.

If, then, else statements

The bash `if` command is a compound command that tests the return value of a test or command (`$?` and branches based on whether it is True (0) or False (not 0). Although the tests above returned only 0 or 1 values, commands may return other values. You will learn more about testing those a little later in this tutorial. The `if` command in bash has a `then` clause containing a list of commands to be executed if the test or command returns 0. The command also has one or more optional `elif` clauses. Each of these optional `elif` clauses has an additional test and `then`

clause with an associated list of commands, an optional final `else` clause, and list of commands to be executed if neither the original test, nor any of the tests used in the `elif` clauses was true. A terminal `fi` marks the end of the construct.

Using what you have learned so far, you could now build a simple calculator to evaluate arithmetic expressions as shown in Listing 36.

Listing 36. Evaluating expressions with `if`, `then`, `else`

```
[ian@pinguino ~]$ function mycalc ()
> {
>   local x
>   if [ $# -lt 1 ]; then
>     echo "This function evaluates arithmetic for you if you give it some"
>   elif (( $* )); then
>     let x="$*"
>     echo "$* = $x"
>   else
>     echo "$* = 0 or is not an arithmetic expression"
>   fi
> }
[ian@pinguino ~]$ mycalc 3 + 4
3 + 4 = 7
[ian@pinguino ~]$ mycalc 3 + 4**3
3 + 4**3 = 67
[ian@pinguino ~]$ mycalc 3 + (4**3 /2)
-bash: syntax error near unexpected token `('
[ian@pinguino ~]$ mycalc 3 + "(4**3 /2)"
3 + (4**3 /2) = 35
[ian@pinguino ~]$ mycalc xyz
xyz = 0 or is not an arithmetic expression
[ian@pinguino ~]$ mycalc xyz + 3 + "(4**3 /2)" + abc
xyz + 3 + (4**3 /2) + abc = 35
```

The calculator makes use of the `local` statement to declare `x` as a local variable that is available only within the scope of the `mycalc` function. The `let` function has several possible options, as does the `declare` function to which it is closely related. Check the man pages for `bash`, or use `help let` for more information.

As you see in Listing 36, you need to be careful making sure that your expressions are properly escaped if they use shell metacharacters such as `(,)`, `*`, `>`, and `<`. Nevertheless, you have quite a handy little calculator for evaluating arithmetic as the shell does it.

You may have noticed the `else` clause and the last two examples. As you see, it is not an error to pass `xyz` to `mycalc`, but it evaluates to 0. This function is not smart enough to identify the character values in the final example of use and thus be able to warn the user. You could use a string pattern matching test such as

```
[[ ! ("$*" == *[a-zA-Z]* ) ]]
```

(or the appropriate form for your locale) to eliminate any expression containing alphabetic characters, but that would prevent using hexadecimal notation in your input, since you might use `0x0f` to represent 15 using hexadecimal notation. In fact,

the shell allows bases up to 64 (using *base#value* notation), so you could legitimately use any alphabetic character, plus `_` and `@` in your input. Octal and hexadecimal use the usual notation of a leading 0 for octal and leading 0x or 0X for hexadecimal. Listing 37 shows some examples.

Listing 37. Calculating with different bases

```
[ian@pinguino ~]$ mycalc 015
015 = 13
[ian@pinguino ~]$ mycalc 0xff
0xff = 255
[ian@pinguino ~]$ mycalc 29#37
29#37 = 94
[ian@pinguino ~]$ mycalc 64#1az
64#1az = 4771
[ian@pinguino ~]$ mycalc 64#1azA
64#1azA = 305380
[ian@pinguino ~]$ mycalc 64#1azA_@
64#1azA_@ = 1250840574
[ian@pinguino ~]$ mycalc 64#1az*64**3 + 64#A_@
64#1az*64**3 + 64#A_@ = 1250840574
```

Additional laundering of the input is beyond the scope of this tutorial, so use your calculator with appropriate care.

The `elif` statement is really a convenience. It will help you in writing scripts by allowing you to simplify the indenting. You may be surprised to see the output of the `type` command for the `mycalc` function as shown in Listing 38.

Listing 38. Type mycalc

```
[ian@pinguino ~]$ type mycalc
mycalc is a function
mycalc ()
{
    local x;
    if [ $# -lt 1 ]; then
        echo "This function evaluates arithmetic for you if you give it some";
    else
        if (( $* )); then
            let x="$*";
            echo "$* = $x";
        else
            echo "$* = 0 or is not an arithmetic expression";
        fi;
    fi;
}
```

Case statements

The `case` compound command simplifies testing when you have a list of possibilities and you want to take action based on whether a value matches a particular possibility. The `case` compound is introduced by `case WORD in` and terminated by `esac` ("case" spelled backwards). Each case consists of a single pattern, or multiple

patterns separated by |, followed by), a list of statements, and finally a pair of semicolons (;).

To illustrate, imagine a store that serves coffee, decaffeinated coffee (decaf), tea, or soda. The function in Listing 39 might be used to determine the response to an order.

Listing 39. Using case commands

```
[ian@pinguino ~]$ type myorder
myorder is a function
myorder ()
{
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here"
            ;;
    esac
}
[ian@pinguino ~]$ myorder decaf
Hot coffee coming right up
[ian@pinguino ~]$ myorder tea
Hot tea on its way
[ian@pinguino ~]$ myorder milk
Sorry, we don't serve that here
```

Note the use of '*' to match anything that had not already been matched.

Bash has another construct similar to `case` that can be used for printing output to a terminal and having a user select items. It is the `select` statement, which will not be covered here. See the bash man pages, or type `help select` to learn more about it.

Of course, there are many problems with such a simple approach to the problem; you can't order two drinks at once, and the function doesn't handle anything but lower-case input. Can you do a case-insensitive match? The answer is "yes", so let's see how.

Return values

The Bash shell has a `shopt` builtin that allows you to set or unset many shell options. One is `nocasematch`, which, if set, instructs the shell to ignore case in string matching. Your first thought might be to use the `-o` operand that you learned

about with the `test` command. Unfortunately, `nocasematch` is not one of the options you can test with `-o`, so you'll have to resort to something else.

The `shopt` command, like most UNIX and Linux commands sets a return value that you can examine using `$?` . The tests that you learned earlier are not the only things with return values. If you think about the tests that you do in an `if` statement, they really test the return value of the underlying test command for being True (0) or False (1 or anything other than 0). This works even if you don't use a test, but use some other command. Success is indicated by a return value of 0, and failure by a non-zero return value.

Armed with this knowledge, you can now test the `nocasematch` option, set it if it is not already set, and then return it to the user's preference when your function terminates. The `shopt` command has four convenient options, `-pqsu` to print the current value, don't print anything, set the option, or unset the option. The `-p` and `-q` options set a return value of 0 to indicate that the shell option is set, and 1 to indicate it is unset. The `-p` options prints out the command required to set the option to its current value, while the `-q` option simply sets a return value of 0 or 1.

Your modified function will use the return value from `shopt` to set a local variable representing the current state of the `nocasematch` option, set the option, run the case command, then reset the `nocasematch` option to its original value. One way to do this is shown in Listing 40.

Listing 40. Testing return values from commands

```
[ian@pinguino ~]$ type myorder
myorder is a function
myorder ()
{
    local restorecase;
    if shopt -q nocasematch; then
        restorecase="-s";
    else
        restorecase="-u";
        shopt -s nocasematch;
    fi;
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here"
            ;;
    esac;
    shopt $restorecase nocasematch
}
[ian@pinguino ~]$ shopt -p nocasematch
shopt -u nocasematch
```

```
[ian@pinguino ~]$ # nocasematch is currently unset
[ian@pinguino ~]$ myorder DECAF
Hot coffee coming right up
[ian@pinguino ~]$ myorder Soda
Your ice-cold soda will be ready in a moment
[ian@pinguino ~]$ shopt -p nocasematch
shopt -u nocasematch
[ian@pinguino ~]$ # nocasematch is unset again after running the myorder function
```

If you want your function (or script) to return a value that other functions or commands can test, use the `return` statement in your function. Listing 41 shows how to return 0 for a drink that you can serve and 1 if the customer requests something else.

Listing 41. Setting your own return values from functions

```
[ian@pinguino ~]$ type myorder
myorder is a function
myorder ()
{
    local restorecase=$(shopt -p nocasematch) rc=0;
    shopt -s nocasematch;
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here";
            rc=1
            ;;
    esac;
    $restorecase;
    return $rc
}
[ian@pinguino ~]$ myorder coffee;echo $?
Hot coffee coming right up
0
[ian@pinguino ~]$ myorder milk;echo $?
Sorry, we don't serve that here
1
```

If you don't specify your own return value, the return value will be that of the last command executed. Functions have a habit of being reused in situations that you never anticipated, so it is good practice to set your own value.

Commands may return values other than 0 and 1, and sometimes you will want to distinguish between them. For example, the `grep` command returns 0 if the pattern is matched and 1 if it is not, but it also returns 2 if the pattern is invalid or if the file specification doesn't match any files. If you need to distinguish more return values besides just success (0) or failure (non-zero), then you will probably use a `case` command or perhaps an `if` command with several `elif` parts.

Command substitution

You met command substitution in the "[LPI exam 101 prep \(topic 103\): GNU and UNIX commands](#)" tutorial, but let's do a quick review.

Command substitution allows you to use the output of a command as input to another command by simply surrounding the command with `$(` and `)` or with a pair of backticks ```. You will find the `$(` form advantageous if you want to nest output from one command as part of the command that will generate the final output, and it can be easier to figure out what's really going on as the parentheses have a left and right form as opposed to two identical backticks. However, the choice is yours, and backticks are still very common,

You will often use command substitution with loops (covered later under [Loops](#)). However, you can also use it to simplify the `myorder` function that you just created. Since `shopt -p nocasematch` actually prints the command that you need to set the `nocasematch` option to its current value, you only need to save that output and then execute it at the end of the `case` statement. This will restore the `nocasematch` option regardless of whether you actually changed it or not. Your revised function might now look like Listing 42. Try it for yourself.

Listing 42. Command substitution instead of return value tests

```
[ian@pinguino ~]$ type myorder
myorder is a function
myorder ()
{
    local restorecase=$(shopt -p nocasematch) rc=0;
    shopt -s nocasematch;
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here"
            rc=1
            ;;
    esac;
    $restorecase
    return $rc
}
[ian@pinguino ~]$ shopt -p nocasematch
shopt -u nocasematch
[ian@pinguino ~]$ myorder DECAF
Hot coffee coming right up
[ian@pinguino ~]$ myorder TeA
Hot tea on its way
[ian@pinguino ~]$ shopt -p nocasematch
shopt -u nocasematch
```

Debugging

If you have typed functions yourself and made typing errors that left you wondering what was wrong, you might also be wondering how to debug functions. Fortunately the shell lets you set the `-x` option to trace commands and their arguments as the shell executes them. Listing 43 shows how this works for the `myorder` function of Listing 42.

Listing 43. Tracing execution

```
[ian@pinguino ~]$ set -x
++ echo -ne '\033]0;ian@pinguino:~'

[ian@pinguino ~]$ myorder tea
+ myorder tea
++ shopt -p nocasematch
+ local 'restorecase=shopt -u nocasematch' rc=0
+ shopt -s nocasematch
+ case "$*" in
+ echo 'Hot tea on its way'
Hot tea on its way
+ shopt -u nocasematch
+ return 0
++ echo -ne '\033]0;ian@pinguino:~'

[ian@pinguino ~]$ set +x
+ set +x
```

You can use this technique for your aliases, functions, or scripts. If you need more information, add the `-v` option for verbose output.

Loops

Bash and other shells have three looping constructs that are somewhat similar to those in the C language. Each will execute a list of commands zero or more times. The list of commands is surrounded by the words `do` and `done`, each preceded by a semicolon.

for

loops come in two flavors. The most common form in shell scripting iterates over a set of values, executing the command list once for each value. The set may be empty, in which case the command list is not executed. The other form is more like a traditional C for loop, using three arithmetic expressions to control a starting condition, step function, and end condition.

while

loops evaluate a condition each time the loop starts and execute the command

list if the condition is true. If the condition is not initially true, the commands are never executed.

until

loops execute the command list and evaluate a condition each time the loop ends. If the condition is true the loop is executed again. Even if the condition is not initially true, the commands are executed at least once.

If the conditions that are tested are a list of commands, then the return value of the last one executed is the one used. Listing 44 illustrates the loop commands.

Listing 44. For, while, and until loops

```
[ian@pinguino ~]$ for x in abd 2 "my stuff"; do echo $x; done
abd
2
my stuff
[ian@pinguino ~]$ for (( x=2; x<5; x++ )); do echo $x; done
2
3
4
[ian@pinguino ~]$ let x=3; while [ $x -ge 0 ] ; do echo $x ;let x--;done
3
2
1
0
[ian@pinguino ~]$ let x=3; until echo -e "x=\c"; (( x-- == 0 )) ; do echo $x ; done
x=2
x=1
x=0
```

These examples are somewhat artificial, but they illustrate the concepts. You will most often want to iterate over the parameters to a function or shell script, or a list created by command substitution. Earlier you discovered that the shell may refer to the list of passed parameters as `$*` or `$@` and that whether you quoted these expressions or not affected how they were interpreted. Listing 45 shows a function that prints out the number of parameters and then prints the parameters according to the four alternatives. Listing 46 shows the function in action, with an additional character added to the front of the IFS variable for the function execution.

Listing 45. A function to print parameter information

```
[ian@pinguino ~]$ type testfunc
testfunc is a function
testfunc ()
{
    echo "$# parameters";
    echo Using '$*';
    for p in $*;
    do
        echo "[$p]";
    done;
    echo Using '$@';
    for p in "$@";

```

```

do
    echo "[$p]";
done;
echo Using '$@';
for p in $@;
do
    echo "[$p]";
done;
echo Using '"$@"';
for p in "$@";
do
    echo "[$p]";
done
}

```

Listing 46. Printing parameter information with testfunc

```

[ian@pinguino ~]$ IFS="|${IFS}" testfunc abc "a bc" "1 2
> 3"
3 parameters
Using $*
[abc]
[a]
[bc]
[1]
[2]
[3]
Using "$*"
[abc|a bc|1 2
3]
Using $@
[abc]
[a]
[bc]
[1]
[2]
[3]
Using "$@"
[abc]
[a bc]
[1 2
3]

```

Study the differences carefully, particularly for the quoted forms and the parameters that include white space such as blanks or newline characters.

Break and continue

The `break` command allows you to exit from a loop immediately. You can optionally specify a number of levels to break out of if you have nested loops. So if you had an `until` loop inside a `for` loop inside another `for` loop and all inside a `while` loop, then `break 3` would immediately terminate the `until` loop and the two `for` loops, and return control to the code in the `while` loop.

The `continue` statement allows you to bypass remaining statements in the command list and go immediately to the next iteration of the loop.

Listing 47. Using break and continue

```
[ian@pinguino ~]$ for word in red blue green yellow violet; do
> if [ "$word" = blue ]; then continue; fi
> if [ "$word" = yellow ]; then break; fi
> echo "$word"
> done
red
green
```

Revisiting ldirs

Remember how much work you did to get the `ldirs` function to extract the file name from a long listing and also figure out if it was a directory or not? The final function that you developed was not too bad, but suppose you had all the information you now have. Would you have created the same function? Perhaps not. You know how to test whether a name is a directory or not using `[-d $name]`, and you know about the `for` compound. Listing 48 shows another way you might have coded the `ldirs` function.

Listing 48. Another approach to ldirs

```
[ian@pinguino developerworks]$ type ldirs
ldirs is a function
ldirs ()
{
    if [ $# -gt 0 ]; then
        for file in "$@";
        do
            [ -d "$file" ] && echo "$file";
        done;
    else
        for file in *;
        do
            [ -d "$file" ] && echo "$file";
        done;
    fi;
    return 0
}
[ian@pinguino developerworks]$ ldirs
my dw article
my-tutorial
readme
schema
tools
web
xsl
[ian@pinguino developerworks]$ ldirs *s* tools/*
schema
tools
xsl
tools/java
[ian@pinguino developerworks]$ ldirs *www*
[ian@pinguino developerworks]$
```

You will note that the function quietly returns if there are no directories matching your criteria. This may or may not be what you want, but if it is, this form of the function is perhaps easier to understand than the version that used `sed` to parse

output from `ls`. At least you now have another tool in your toolbox.

Creating scripts

Recall that `myorder` could handle only one drink at a time? You could now combine that single drink function with a `for` compound to iterate through the parameters and handle multiple drinks. This is as simple as placing your function in a file and adding the `for` instruction. Listing 49 illustrates the new `myorder.sh` script.

Listing 49. Ordering multiple drinks

```
[ian@pinguino ~]$ cat myorder.sh
function myorder ()
{
    local restorecase=$(shopt -p nocasematch) rc=0;
    shopt -s nocasematch;
    case "$*" in
        "coffee" | "decaf")
            echo "Hot coffee coming right up"
            ;;
        "tea")
            echo "Hot tea on its way"
            ;;
        "soda")
            echo "Your ice-cold soda will be ready in a moment"
            ;;
        *)
            echo "Sorry, we don't serve that here";
            rc=1
            ;;
    esac;
    $restorecase;
    return $rc
}

for file in "$@"; do myorder "$file"; done

[ian@pinguino ~]$ . myorder.sh coffee tea "milk shake"
Hot coffee coming right up
Hot tea on its way
Sorry, we don't serve that here
```

Note that the script was *sourced* to run in the current shell environment rather than its own shell using the `.` command. To be able to execute a script, either you have to source it, or the script file must be marked executable using the `chmod -x` command as illustrated in Listing 50.

Listing 50. Making the script executable

```
[ian@pinguino ~]$ chmod +x myorder.sh
[ian@pinguino ~]$ ./myorder.sh coffee tea "milk shake"
Hot coffee coming right up
Hot tea on its way
Sorry, we don't serve that here
```

Specify a shell

Now that you have a brand-new shell script to play with, you might ask whether it works in all shells. Listing 51 shows what happens if you run the exact same shell script on a Ubuntu system using first the Bash shell, then the dash shell.

Listing 51. Shell differences

```
ian@attic4:~$ ./myorder tea soda
-bash: ./myorder: No such file or directory
ian@attic4:~$ ./myorder.sh tea soda
Hot tea on its way
Your ice-cold soda will be ready in a moment
ian@attic4:~$ dash
$ ./myorder.sh tea soda
./myorder.sh: 1: Syntax error: "(" unexpected
```

That's not too good.

Remember earlier when we mentioned that the word 'function' was optional in a bash function definition, but that it wasn't part of the POSIX shell specification? Well, dash is a smaller and lighter shell than bash and it doesn't support that optional feature. Since you can't guarantee what shell your potential users might prefer, you should always ensure that your script is portable to all shell environments, which can be quite difficult, or use the so-called *shebang* (`#!`) to instruct the shell to run your script in a particular shell. The shebang line must be the first line of your script, and the rest of the line contains the path to the shell that your program must run under, so it would be `#!/bin/bash` the `myorder.sh` script.

Listing 52. Using shebang

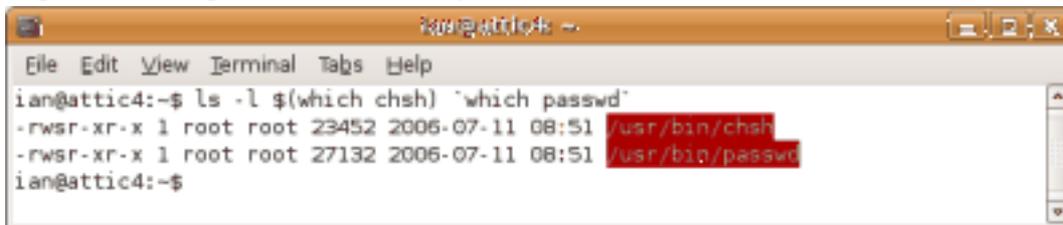
```
$ head -n3 myorder.sh
#!/bin/bash
function myorder ()
{
$ ./myorder.sh Tea Coffee
Hot tea on its way
Hot coffee coming right up
```

You can use the `cat` command to display `/etc/shells`, which is the list of shells on your system. Some systems do list shells that are not installed, and some listed shells (possibly `/dev/null`) may be there to ensure that FTP users cannot accidentally escape from their limited environment. If you need to change your default shell, you can do so with the `chsh` command, which updates the entry for your userid in `/etc/passwd`.

Suid rights and script locations

In the earlier tutorial [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#) you learned how to change a file's owner and group and how to set the suid and sgid permissions. An executable file with either of these permissions set will run in a shell with *effective* permissions of the file's owner (for suid) or group (for sgid). Thus, the program will be able to do anything that the owner or group could do, according to which permission bit is set. There are good reasons why some programs need to do this. For example, the `passwd` program needs to update `/etc/shadow`, and the `chsh` command, which you use to change your default shell, needs to update `/etc/passwd`. If you use an alias for `ls`, listing these programs is likely to result in a red, highlighted listing to warn you, as shown in Figure 2. Note that both of these programs have the suid big (s) set and thus operate as if root were running them.

Figure 2. Programs with suid permission



```

ian@attic4:~$ ls -l $(which chsh) $(which passwd)
-rwsr-xr-x 1 root root 23452 2006-07-11 08:51 /usr/bin/chsh
-rwsr-xr-x 1 root root 27132 2006-07-11 08:51 /usr/bin/passwd
ian@attic4:~$

```

Listing 53 shows that an ordinary user can run these and update files owned by root.

Listing 53. Using suid programs

```

ian@attic4:~$ passwd
Changing password for ian
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
ian@attic4:~$ chsh
Password:
Changing the login shell for ian
Enter the new value, or press ENTER for the default
  Login Shell [/bin/bash]: /bin/dash
ian@attic4:~$ find /etc -mmin -2 -ls
308865      4 drwxr-xr-x 108 root      root           4096 Jan 29 22:52 /etc
find: /etc/cups/ssl: Permission denied
find: /etc/lvm/archive: Permission denied
find: /etc/lvm/backup: Permission denied
find: /etc/ssl/private: Permission denied
311170      4 -rw-r--r--   1 root      root           1215 Jan 29 22:52 /etc/passwd
309744      4 -rw-r-----  1 root      shadow          782 Jan 29 22:52 /etc/shadow
ian@attic4:~$ grep ian /etc/passwd
ian:x:1000:1000:Ian Shields,,,:/home/ian:/bin/dash

```

You can set suid and sgid permissions for shell scripts, but most modern shells ignore these bits for scripts. As you have seen, the shell has a powerful scripting language, and there are even more features that are not covered in this tutorial, such as the ability to interpret and execute arbitrary expressions. These features make it a very unsafe environment to allow such wide permission. So, if you set suid

or `sgid` permission for a shell script, don't expect it to be honored when the script is executed.

Earlier, you changed the permissions of `myorder.sh` to mark it *executable* (`x`). Despite that, you still had to qualify the name by prefixing `./` to actually run it, unless you sourced it in the current shell. To execute a shell by name only, it needs to be on your path, as represented by the `PATH` variable. Normally, you do **not** want the current directory on your path, as it is a potential security exposure. Once you have tested your script and found it satisfactory, you should place it in `~/nom` if it is a personal script, or `/usr/local/bin` if it is to be available for others on the system. If you simply used `chmod -x` to mark it executable, it is executable by everyone (owner, group and world). This is generally what you want, but refer back to the earlier tutorial, [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#), if you need to restrict the script so that only members of a certain group can execute it.

You may have noticed that shells are usually located in `/bin` rather than in `/usr/bin`. According to the Filesystem Hierarchy Standard, `/usr/bin` may be on a filesystem shared among systems, and so it may not be available at initialization time. Therefore, certain functions, such as shells, should be in `/bin` so they are available even if `/usr/bin` is not yet mounted. User-created scripts do not usually need to be in `/bin` (or `/sbin`), as the programs in these directories should give you enough tools to get your system up and running to the point where you can mount the `/usr` filesystem.

Mail to root

If your script is running some administrative task on your system in the dead of night while you're sound asleep, what happens when something goes wrong? Fortunately, it's easy to mail error information or log files to yourself or to another administrator or to root. Simply pipe the message to the `mail` command, and use the `-s` option to add a subject line as shown in Listing 54.

Listing 54. Mailing an error message to a user

```
ian@attic4:~$ echo "Midnight error message" | mail -s "Admin error" ian
ian@attic4:~$ mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/ian": 1 message 1 new
>N 1 ian@localhost      Mon Jan 29 23:58   14/420   Admin error
&
Message 1:
From ian@localhost  Mon Jan 29 23:58:27 2007
X-Original-To: ian
To: ian@localhost
Subject: Admin error
Date: Mon, 29 Jan 2007 23:58:27 -0500 (EST)
From: ian@localhost (Ian Shields)
```

```
Midnight error message
```

```
& d  
& q
```

If you need to mail a log file, use the `<` redirection function to redirect it as input to the mail command. If you need to send several files, you can use `cat` to combine them and pipe the output to mail. In Listing 54, mail was sent to user `ian` who happened to also be the one running the command, but admin scripts are more likely to direct mail to root or another administrator. As usual, consult the man pages for mail to learn about other options that you can specify.

This brings us to the end of this tutorial. We have covered a lot of material on shells and scripting. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Shell Command Language](#)" defines the shell command language as specified by The Open Group and IEEE.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- Read these developerWorks articles for other ways to work with Bash:
 - [Bash by example, Part 1](#)
 - [Bash by example, Part 2](#)
 - [Bash by example, Part 3](#)
 - [System Administration Toolkit: Get the most out of bash](#)
 - [Working in the bash shell](#)
- Sed and awk each deserve a tutorial on their own, but read these developerWorks articles for a more complete background.
Sed:
 - [Common threads: Sed by example, Part 1](#)
 - [Common threads: Sed by example, Part 2](#)
 - [Common threads: Sed by example, Part 3](#)**Awk:**
 - [Common threads: Awk by example, Part 1](#)
 - [Common threads: Awk by example, Part 2](#)
 - [Common threads: Awk by example, Part 3](#)
 - [Get started with GAWK: AWK language fundamentals](#)
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.

- [LPI Linux Certification in a Nutshell, Second Edition](#) (O'Reilly, 2006) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- Download the developerWorks author package from "[Authoring with the developerWorks XML templates](#)" (developerWorks, January 2007).
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 101 prep: The X Window System

Junior Level Administration (LPIC-1) topic 110

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

02 Jul 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 101. In this fifth in a [series of five tutorials](#), Ian introduces you to the X Window System on Linux®. By the end of this tutorial, you will know how to install and maintain the X Window System. This tutorial covers both major packages for X on Linux: XFree86 and X.Org.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 101, the five topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 101: Tutorials and topics

LPI exam 101 topic	developerWorks tutorial	Tutorial summary
Topic 101	LPI exam 101 prep: Hardware and architecture	Learn to configure your system hardware with Linux. By the end of this tutorial, you will know how Linux configures the hardware found on a modern PC and where to look if you have problems.
Topic 102	LPI exam 101 prep: Linux installation and package management	Get an introduction to Linux installation and package management. By the end of this tutorial, you will know how Linux uses disk partitions, how Linux boots, and how to install and manage software packages.
Topic 103	LPI exam 101 prep: GNU and UNIX commands	Get an introduction to common GNU and UNIX commands. By the end of this tutorial, you will know how to use commands in the bash shell, including how to use text processing commands and filters, how to search files and directories, and how to manage processes.
Topic 104	LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard.	Learn how to create filesystems on disk partitions, as well as how to make them accessible to users, manage file ownership and user quotas, and repair filesystems as needed. Also learn about hard and symbolic links, and how to locate files in your filesystem and where files should be placed.
Topic 110	LPI exam 101 prep: The X Window system	(This tutorial). Learn how to install and maintain the X Window System. See detailed objectives below.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger

system, back up and restore, and shut down and reboot

- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "The X Window System," the fifth of five tutorials designed to prepare you for LPI exam 101. In this tutorial, you learn about setting up the X Window System on Linux. This tutorial covers both major packages for X on Linux: XFree86 and X.Org.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. The X Window System: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
1.110.1 Install and configure X	Weight 5	Configure and install X and an X font server. Check that the video card and monitor are supported by your X server, and customize and tune X for the card and monitor. Install an X font server, install fonts, and configure X to use the font server.
1.110.2 Set up a display manager	Weight 3	Set up and customize a display manager. Turn the display manager on or off and change its greeting and default bitplanes. Configure display managers for use by X stations, such as the X, GNOME, and KDE display managers.
1.110.4 Install and customize a window manager environment	Weight 5	Customize a system-wide desktop environment and window manager, including window manager menus and desktop panel menus. Select and configure an X terminal, and verify and resolve library

dependency issues for X applications. Export an X display to a client workstation.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial. You should also be familiar with using GUI applications, preferably under the X Window System.

This tutorial builds on content covered in the previous four tutorials in this series, so you may want to first review the [tutorials for topics 101, 102, 103, and 104](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Install and configure X

This section covers material for topic 1.110.1 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn how to:

- Check that your video card and monitor are supported by your X server
- Configure and install X
- Customize and tune X for the card and monitor
- Configure and install an X font server
- Install fonts

History of the X Window System

The X Window System, also known simply as X or X11, is a window system for graphical (bitmap) displays. X originated at MIT in 1984 and was developed as part of Project Athena, which provided a computing environment using disparate hardware. X separates the display functions into a *display server* and *clients*, which provide the application logic. It is network transparent, so the display server and the

client need not be on the same machine. Note that the sense of "client" and "server" is somewhat opposite to what you might normally think. In addition to handling displayed output, the server end also handles input from devices such as keyboards, mice, graphic tablets, and touchscreens.

X provides a toolkit for GUI applications, but it does not specify a user interface. On a typical Linux system, you will choose between KDE or GNOME desktops, and you may have several other window managers available too. Because X does not specify a user interface, these desktops and window managers have different appearances.

Because X was developed to serve a large community with disparate hardware types, you will find that different versions of X client and server will generally interoperate quite well.

XFree86 and X.Org

By 1987, MIT wanted to hand off control of X, and the MIT X Consortium was founded as a non-profit group to oversee development of X. After a few more stewardship changes, the Open Group formed X.Org in 1999. Since 1992, much of the active development of X was done by XFree86, which had originally created a port of X to the Intel® 386 hardware for use in Linux, hence the name XFree86. XFree86 joined X.Org as a non-paying member.

Although originally created for the 386, later versions of XFree86 supported several different platforms, and it became the most widely used X version on Linux. After some disputes over new licensing terms and the development model of XFree86, the X.Org Foundation was formed. Working from the last XFree86 version under the earlier license, it created X11R6.7 and X11R6.8. Many distributions still use XFree86, while many have picked up X.Org instead.

Video hardware support

Both the XFree86 and X.Org packages support a wide range of modern video cards. Consult the online documentation for your release (see [Resources](#)). Some manufacturers do not release open source drivers for all functions, so you may need to integrate a driver from the manufacturer into your XFree86 system. Check the manufacturer's Web site for improved or updated Linux drivers. This is often the case for accelerated 3D drivers. Even if the hardware capabilities of your card cannot be used by XFree86, it is possible that you may be able to run in VESA (Video Electronics Standards Association) *framebuffer* mode.

Modern monitors implement the VESA *Display Data Channel* (or *DDC*) specification, which allows monitor information and capabilities to be determined programatically. The XFree86 configuration tools (other than `xf86config`) use this information to

configure your X system.

One way to see how X works with your hardware is to boot a live CD distribution, such as Knoppix or Ubuntu. These generally have excellent ability to detect and use your hardware. Many distributions offer a graphical installation choice, which also requires correct detection and use of your hardware.

XFree86

Most distributions include a version of XFree86 or X.Org already packaged for the system. If not, you may be able to find an RPM or .deb package and install it according to the techniques you learned in the tutorial for topic 102, "[LPI exam 101 prep: Linux installation and package management](#)."

XFree86 installation

If you do not have an XFree86 package available, then you will need to download the files from XFree86 project Web site (see [Resources](#)). Prebuilt packages are available for Linux on several popular hardware platforms, or you may install from the source distribution. This tutorial assumes you will be installing a binary package of the current release (version 4.5.0).

You will need to download several binary packages. You should use the available md5 checksums and the GPG keys to validate your downloads. Table 3 lists the required files for XFree86.

File	Description
Xinstall.sh	Installation script
extract	Tarball extraction utility
Xbin.tgz	X clients, utilities, and run-time libraries
Xlib.tgz	Data files required at run-time
Xman.tgz	Manual pages
Xdoc.tgz	XFree86 documentation
Xfnts.tgz	Base set of fonts
Xfenc.tgz	Font encoding data
Xetc.tgz	Run-time configuration files - part 1
Xrc.tgz	Run-time configuration files - part 2

Xvar.tgz	Run-time data
Xxserv.tgz	XFree86 X server
Xmod.tgz	X server modules

If you are not sure which version to download, then download the Xinstall.sh file for the one you think is closest, and use the `-check` option to check your system as shown in Listing 1.

Listing 1. Checking for the correct XFree86 binary package

```
root@pinguino:~/xfree86# sh Xinstall.sh -check
Checking which OS you're running...
uname reports 'Linux' version '2.6.12-10-386', architecture 'i686'.
libc version is '6.3.5' (6.3).

Binary distribution name is 'Linux-ix86-glibc23'

If you don't find a binary distribution with this name, then
binaries for your platform are not available from XFree86.org.
```

For this example, you should look for the "Linux-ix86-glibc23" package.

Table 4 lists the optional files for XFree86. For this tutorial, you will need the font server and any other items that you wish to install.

Table 4. XFree86 optional files	
File	Description
Xdrm.tgz	Direct rendering manager (DRM) kernel modules source
Xfsrv.tgz	Font server
Xnest.tgz	Nested X server
Xprog.tgz	X header files, configuration files, and libraries for developing X applications
Xprt.tgz	X Print server
Xvfb.tgz	Virtual framebuffer X server
Xtinx.tgz	TinyX servers
Xf100.tgz	100dpi fonts
Xfcyr.tgz	Cyrillic fonts
Xfsc1.tgz	Scalable fonts (Speedo, Type1, and TrueType)

Xhtml.tgz	HTML version of the documentation
Xps.tgz	PostScript version of the documentation
Xpdf.tgz	PDF version of the documentation

Before you install XFree86, you should make backups of your `/usr/X11R6`, `/etc/X11`, and `/etc/fonts` directories as their contents may be changed by the XFree86 installation. You may use the `tar`, `cp`, or `zip` commands to do this. When you are ready to install XFree86, change to the directory where you downloaded the XFree86 files, and run the `Xinstall.sh` script as shown in Listing 2.

Listing 2. Installing XFree86

```
root@pinguino:~/xfree86# sh Xinstall.sh
```

You will be prompted for answers to several questions, which may vary according to whether or not you have a prior installation of X. After the mandatory components are installed, you will be prompted to install the optional components individually.

Following the file installation, the script will run the `ldconfig` command and offer to set up several symbolic links for you.

The easiest way to install XFree86 is to install all the components you want using the `Xinstall.sh` script. If you do not, you will either need to reinstall the whole package, potentially overwriting any customization you have done, or manually install other components.

XFree86 configuration

Historically, configuring XFree86 involved creating an `XF86Config` file, which contained information about the video card, mouse, keyboard, and display hardware as well as customization items, such as preferred display resolutions. The original configuration tool, `xf86config`, required a user to have and enter detailed information about video card and monitor timings. Recent versions of XFree86 are capable of dynamically determining the available hardware and can run with little or no configuration information.

The available configuration tools are:

XFree86 -autoconfig

Running `XFree86` with the `-autoconfig` option will attempt to automatically configure the X server. If your setup is correctly determined, you should be able to move the X cursor around the screen with your mouse. Hold down the **Ctrl**

and **Alt** keys and press the **Backspace** key to exit display. This confirms that automatic configuration will work. A configuration file is not written.

XFree86 -configure

Running `XFree86` with the `-configure` option may work if the `-autoconfig` option does not. This option may also give problems on some systems.

xf86cfg

The `xf86cfg` command attempts to start the display and input drivers. If it is successful, you will see a window with a diagram of your system. Right-click an item to view or update its configuration. On some systems you may need to use the numeric keypad instead of mouse buttons, because the mouse was not properly detected. You may wish to try creating a symbolic link to `/dev/mouse` from your actual mouse device before running `xf86cfg`. For example:

```
ln -s /dev/input/mice /dev/mouse
```

When you click **Quit**, you will be prompted to save your `/etc/X11R6/lib/X11/XF86Config` and `/etc/X11R6/lib/X11/xkb/X0-config.keyboard` configuration files.

xf86config

The `xf86config` command uses a text-mode interface to interactively prompt for information about your mouse, keyboard, video card, and display. You will need horizontal and vertical frequency information for your display. You can select most video cards from a database of known video cards. If not, you may need specific chipset and timing information for your card.

Notes:

1. If your system includes XFree86, your distributor may have included a tool, such as the `sax2` command used on SUSE systems or the `redhat-config-xfree86` command used on some Red Hat® systems. Always check your system documentation for such tools.
2. Another configuration tool, `XF86Setup`, is no longer distributed with XFree86.

X.Org

Most distributions include a version of XFree86 or X.Org already packaged for the system. If not, you may be able to find an RPM or .deb package and install it according to the techniques you learned in the tutorial for topic 102, "[LPI exam 101 prep: Linux installation and package management](#)."

X.Org installation

If you do not have an X.Org package available, then you will need to download and build the source from the X.Org Web site or a mirror (see [Resources](#)). At the time of this writing, these sites do not contain prebuilt binary packages for X11R6.9.0 or X11R7.0. The source is available from the CVS repository, or as tarballs that are compressed with either gzip or bzip2. You need to get either the gz or bz2 files, **not both**. You will find the *X.Org Modular Tree Developer's Guide* (see [Resources](#)) an invaluable aid when downloading and building X.Org yourself. Take note of the additional packages, such as freetype, fontconfig, and Mesa, that are recommended for a fully functional build.

X.Org configuration

The X.Org package is based on a recent version of XFree86 and has similar configuration capabilities, including dynamically determining the available hardware. The configuration file is named `xorg.conf` rather than `XF86Config`. You may find it in one of several places: `/etc/xorg.conf`, `/etc/X11/xorg.conf`, `/usr/X11R6/etc/xorg.conf`, `/usr/X11R6/lib/X11/xorg.conf.hostname`, or `/usr/X11R6/lib/X11/xorg.conf`.

The available configuration tools are:

X -configure

Running `X` with the `-configure` option causes the X server to load each driver module, probe for the driver, and create a configuration file that is saved in the home directory of the user who started the server (usually `/root`). The file is called `xorg.conf.new`.

xorgcfg

This tool is similar to `xf86cfg`.

xorg86config

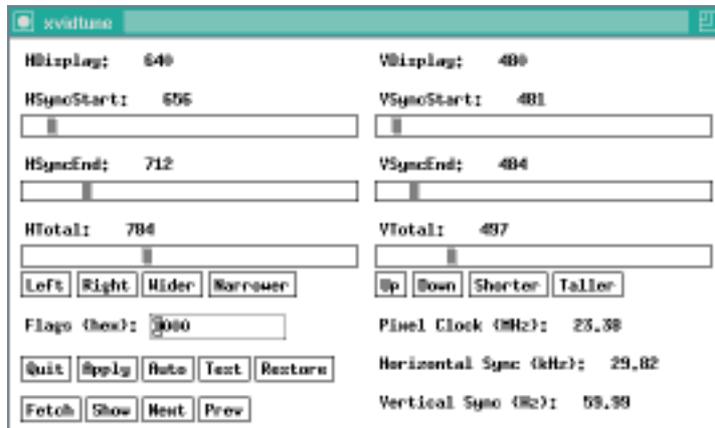
The `xorgconfig` command uses a text-mode interface to interactively prompt for information about your mouse, keyboard, video card, and display. As with `xf86config`, you will need horizontal and vertical frequency information for your display. You can select most video cards from a database of known video cards. If not, you may need specific chipset and timing information for your card.

Tuning X

Modern multisync CRT monitors usually have controls to set the size and position of the displayed image on the screen. If your monitor does not have this capability, you can use the `xvidtune` command to tune the size and position of your X display. When you run `xvidtune` from an X terminal session, you will see a window similar to Figure 1. Adjust the settings and click **Test** to see how they work, or click **Apply** to change the settings. If you click **Show**, the current settings will be printed to your

terminal window in a format that you can use as a Modeline setting in your CF86Config or xorg.conf file.

Figure 1. Running xvidtune



Consult the man page for additional information.

Overview of fonts in X

For many years, font handling on X systems was done by the *core X11 fonts system*. Recent versions of XFree86 (and X.Org) X servers include the *Xft fonts system*. The core fonts system was originally designed to support monochrome bitmap fonts, but it has been enhanced over time. The Xft system was designed to handle modern requirements, including anti-aliasing and sub-pixel rasterization and it allows applications to have extensive control over the rendering of glyphs. A major difference between the two systems is that the core fonts are handled on the server, while the Xft fonts are handled by the clients, which send the necessary glyphs to the server.

X originally used Type 1 (or Adobe Type 1) fonts, a font specification developed by Adobe. The Xft system can handle these as well as OpenType, TrueType, Speedo, and CID font types.

The xfs font server

With the core X11 fonts system, the X Server obtains fonts and font information from a *font server*. The X font server, `xfs`, usually runs as a daemon and is started at system startup, although it is possible to run it as an ordinary task. You will usually install a font server as part of your X installation. However, because X is a network protocol, it is possible to obtain fonts and font information over a network rather than from your local machine.

The X font server uses a configuration file, normally `/usr/X11R6/lib/X11/fs/config`. A

sample font configuration file is shown in Listing 3. The configuration file may also be located in or linked to `/etc/X11/fs`.

Listing 3. Sample `/usr/X11R6/lib/X11/fs/config`

```
# allow a max of 10 clients to connect to this font server
client-limit = 10

# when a font server reaches its limit, start up a new one
clone-self = on

# alternate font servers for clients to use
#alternate-servers = foo:7101,bar:7102

# where to look for fonts
#
catalogue = /usr/X11R6/lib/X11/fonts/misc:unscaled,
            /usr/X11R6/lib/X11/fonts/75dpi:unscaled,
            /usr/X11R6/lib/X11/fonts/100dpi:unscaled,
            /usr/X11R6/lib/X11/fonts/misc,
            /usr/X11R6/lib/X11/fonts/Type1,
            /usr/X11R6/lib/X11/fonts/Speedo,
            /usr/X11R6/lib/X11/fonts/cyrillic,
            /usr/X11R6/lib/X11/fonts/TrueType,
            /usr/share/fonts/default/Type1

# in 12 points, decipoints
default-point-size = 120

# 100 x 100 and 75 x 75
default-resolutions = 75,75,100,100

# how to log errors
use-syslog = on

# don't listen to TCP ports by default for security reasons
no-listen = tcp
```

This example is typical of a Linux workstation installation where the font server does not provide fonts over TCP network connections (`no-listen = tcp`).

The Xft library

The Xft library provides functions that allow client applications to select fonts based on pattern criteria and to generate glyphs to send to the server. Patterns take into account items such as font family (Helvetica, Times, and so on), point size, weight (regular, bold, italic), and many other possible characteristics. While the core font system allowed a client to find a match for the first available font on a server, the Xft system finds the best match for all the criteria and then sends the glyph information to the server. Xft interacts with FreeType for font rendering and with the X Render extensions, which help speed up font rendering operations. Xft is included with current versions of both XFree86 and X.Org.

Note: If your X server is running across a network and using a video card that does not support the X Render extensions, you may want to disable anti-aliasing as the network performance in this case may be a problem. You can use the `xdpinfo`

command to check your X server. Listing 4 shows part of the output from `xdpinfo`. Since the output from `xdpinfo` is large, you may want to use `grep` to filter for 'RENDER'.

Listing 4. Checking for RENDER extensions with `xdpinfo`

```
[ian@lyrebird ian]$ xdpinfo
name of display:      :0.0
version number:      11.0
vendor string:       The XFree86 Project, Inc
vendor release number: 40300000
XFree86 version:    4.3.0
maximum request size: 4194300 bytes
motion buffer size: 256
bitmap unit, bit order, padding: 32, LSBFirst, 32
image byte order:    LSBFirst
number of supported pixmap formats: 7
supported pixmap formats:
  depth 1, bits_per_pixel 1, scanline_pad 32
  depth 4, bits_per_pixel 8, scanline_pad 32
  depth 8, bits_per_pixel 8, scanline_pad 32
  depth 15, bits_per_pixel 16, scanline_pad 32
  depth 16, bits_per_pixel 16, scanline_pad 32
  depth 24, bits_per_pixel 32, scanline_pad 32
  depth 32, bits_per_pixel 32, scanline_pad 32
keycode range:      minimum 8, maximum 255
focus: window 0x2000011, revert to Parent
number of extensions: 30
  BIG-REQUESTS
  DOUBLE-BUFFER
  DPMS
  Extended-Visual-Information
  FontCache
  GLX
  LBX
  MIT-SCREEN-SAVER
  MIT-SHM
  MIT-SUNDRY-NONSTANDARD
  RANDR
  RECORD
  RENDER
  SECURITY
  SGI-GLX
  SHAPE
  SYNC
  TOG-CUP
  X-Resource
  XC-APPGROUP
  XC-MISC
  XFree86-Bigfont
  XFree86-DGA
  XFree86-DRI
  XFree86-Misc
  XFree86-VidModeExtension
  XInputExtension
  XKEYBOARD
  XTEST
  XVideo
default screen number: 0
number of screens: 1
```

The use of Xft instead of the core X system requires application changes, so you may find that some of your applications do not appear to take advantage of the

better font rendering of Xft. At the time of this writing, the Qt toolkit (used for KDE) and the GTK+ toolkit (used for GNOME), as well as Mozilla 1.2 and above, are examples of applications that use Xft.

Installing fonts

There are two methods of installing fonts, one for Xft and a more complex one for the core X11 fonts.

Fonts for Xft

Xft uses fonts located in a set of well-known font directories as well as fonts installed in the `.fonts` subdirectory of the user's home directory. The well-known font directories include subdirectories of `/usr/X11R6/lib/X11/lib/fonts` as listed in the catalog entry in `/usr/X11R6/lib/X11/fs/config`. Other font directories may be specified in the `FontPath` section `XF86Config` or `xorg.conf`, according to which X Server package you are using.

Simply copy your font files to the user's `.fonts` directory or to a directory such as `/usr/local/share/fonts` for system-wide use. The font server should pick up the new fonts and make them available at its next opportunity. You can trigger this update with the `fc-cache` command.

The current font technology in X uses loadable modules to provide font support for different font types as shown in Table 5.

Module	Description
bitmap	bitmap fonts (.bdf, .pcf, and .snf)
freetype	TrueType fonts (.ttf and .ttc), OpenType fonts (.otf and .otc) and Type 1 fonts (.pfa and .pfb)
type1	Alternate for Type 1 (.pfa and .pfb) and CID Fonts
xft	Alternate TrueType module (.ttf and .ttc)
speedo	Bitstream Speedo fonts (.spd)

If you are having trouble installing and using a font, check the server log (for

example, `/var/log/XFree86.0.log`) to make sure that the appropriate module was loaded. Module names are case sensitive. You can use the `xset` command to display (and set) X server settings, including the font path and the location of the configuration and log files, as illustrated in Listing 5.

Listing 5. Displaying X server settings with `xset`

```
[ian@lyrebird ian]$ xset -display 0:0 -q
Keyboard Control:
  auto repeat:  on      key click percent:  0      LED mask:  00000000
  auto repeat delay:  500    repeat rate:  30
  auto repeating keys:  00ffffffdffffbbf
                        fadffffffffffdfe5ff
                        ffffffffffffffff
                        ffffffffffffffff
  bell percent:  50      bell pitch:  400    bell duration:  100
Pointer Control:
  acceleration:  2/1      threshold:  4
Screen Saver:
  prefer blanking:  yes    allow exposures:  yes
  timeout:  0      cycle:  0
Colors:
  default colormap:  0x20    BlackPixel:  0      WhitePixel:  16777215
Font Path:
  /home/ian/.gnome2/share/cursor-fonts,unix/:7100,/home/ian/.gnome2/share/fonts
Bug Mode: compatibility mode is disabled
DPMS (Energy Star):
  Standby: 7200    Suspend: 7200    Off: 14340
  DPMS is Enabled
  Monitor is Off
Font cache:
  hi-mark (KB): 5120  low-mark (KB): 3840  balance (%): 70
File paths:
  Config file:  /etc/X11/XF86Config
  Modules path: /usr/X11R6/lib/modules
  Log file:    /var/log/XFree86.0.log
```

If you need to further control the behavior of Xft, you can use either the system-wide configuration file (`/etc/fonts/fonts.conf`) or a user-specific file (`.fonts.conf` in the home directory of the user). You can enable or disable anti-aliasing and control sub-pixel rendering (used on LCD displays), among other things. These are XML files, so you need to ensure that you maintain well-formed XML if you edit them. Consult the `man` (or `info`) pages for `fonts-conf` for additional information on the contents and format of these files.

Core X11 fonts

If you have files in Bitmap Distribution Format (`.bdf`), it is desirable to convert them to Portable Compiled Format (`.pcf`) using the `bdftoppcf` command and then compress them using `gzip` before installing them. Once this is done, you may copy your new fonts to a directory, such as `/usr/local/share/fonts/bitmap/`, and then run the `mkfontdir` command to create a font directory for use by the font server. These steps are illustrated in Listing 6.

Listing 6. Installing a bitmapped font

```
[root@lyrebird root]# bdf2pcf courier12.bdf -o courier12.pcf
[root@lyrebird root]# gzip courier12.pcf
[root@lyrebird root]# mkdir -p /usr/local/share/fonts/bitmap
[root@lyrebird root]# cp *.pcf.gz /usr/local/share/fonts/bitmap/
[root@lyrebird root]# mkfontdir /usr/local/share/fonts/bitmap/
[root@lyrebird root]# ls /usr/local/share/fonts/bitmap/
courier12.pcf.gz  fonts.dir
```

Note that the `mkfontdir` command created the `fonts.dir` file.

When installing scalable fonts, such as TrueType or Type1 fonts, an extra step is required to create scaling information. After you copy the font files to the target directory, run the `mkfontscale` command and then the `mkfontdir` command. The `mkfontscale` command will create an index of scalable font files in a file called `fonts.scale`.

Once you have set up the font directory and scaling information for your new fonts, you need to tell the server where to find them, by including the new directory in the font path. You can do this temporarily by using the `xset` command or permanently by adding a `FontPath` entry to `XF86Config` or `xorg.conf`. To add the new bitmap font directory to the front of the font path, use the `+fp` option of `xset` as shown in Listing 7.

Listing 7. Updating the fontpath with xset

```
[ian@lyrebird ian]$ xset +fp /usr/local/share/fonts/bitmap/ -display 0:0
```

Although not shown here, it is a good idea to add scalable fonts before bitmapped fonts in the path as this results in better font matching. To add directories to the back of the path, use the `fp+` option. Similarly, the use of `-fp` and `fp-` options will remove font directories from the front and back of the font path, respectively.

You can make permanent modifications to the font path by editing the `XF86Config` or `xorg.conf` file. You can add as many `FontPath` lines as you need in the Files section, as shown in Listing 8.

Listing 8. Updating XF86Config or xorg.conf

```
Section "Files"
# RgbPath is the location of the RGB database. Note, this is the name of the
# file minus the extension (like ".txt" or ".db"). There is normally
# no need to change the default.

# Multiple FontPath entries are allowed (they are concatenated together)
# By default, Red Hat 6.0 and later now use a font server independent of
# the X server to render fonts.

    RgbPath        "/usr/X11R6/lib/X11/rgb"
    FontPath       "unix/:7100"
```

```
FontPath      "/usr/local/share/fonts/bitmap/"
EndSection
[
```

For information on other modifications you can make to the X configuration files, see the man pages for either XF86Config or xorg.conf, as appropriate.

Section 3. Set up a display manager

This section covers material for topic 1.110.2 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn how to:

- Set up and customize a display manager
- Change the display manager greeting
- Change default bitplanes for the display manager
- Configure display managers for use by X stations

The display managers covered are XDM (X Display Manager), GDM (GNOME Display Manager), and KDM (KDE Display Manager).

Display managers

In the previous section, if you installed and configured X on a system that did not have X already installed, you probably noticed that, to get any kind of graphical display, you must initially log in to a terminal window and run the `startx` command. While this works for the local display, it is cumbersome. Furthermore, it does not work for a remote X terminal.

The solution is to use a *display manager* to present a graphical login screen and handle authentication. Once a user authenticates, the display manager starts a session for the user on the system where the display manager is running. The graphical output is displayed on the screen where the user entered his or her login credentials. This may be a local display or an X display connected across a network.

Both XFree86 and X.Org come with the XDM display manager. Two other display managers are also popular, KDE and GNOME. In this section, you learn how to set up and customize these three display managers.

To set up a graphical login, however, you need to understand Linux system initialization. You can learn more about this in the forthcoming tutorial, [LPI exam 102 prep \(topic 106\): Boot, initialization, shutdown, and runlevels](#), and in [LPI exam 201 prep \(topic 202\): System startup](#). In the remainder of this section, you will learn enough to start your system with a graphical login, but the main focus in this section is on setting up and customizing the display manager.

On systems such as Red Hat® and SUSE systems, X is usually started in runlevel 5. Debian systems treat runlevels 2 through 5 as equivalent and default to starting in runlevel 2. The determination of default runlevel is made in `/etc/inittab` as shown in Listing 9.

Listing 9. Setting the default runlevel in `/etc/inittab`.

```
# The default runlevel is defined here
id:5:initdefault:
```

Another line, such as that shown in Listing 10 (for a SUSE system) or Listing 11 (for a Ubuntu system), determines the script or program to be run first.

Listing 10. Initial script for SUSE (or Red Hat) system

```
# First script to be executed, if not booting in emergency (-b) mode
si::bootwait:/etc/init.d/boot
```

Listing 11. Initial script for Ubuntu (or Debian) system

```
# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS
```

The initialization scripts (`/etc/init.d/boot` or `/etc/init.d/rcS`) will then run other scripts. Eventually, a series of scripts for the chosen runlevel will be run. For the above examples, these might include `/etc/rc2.d/S13gdm` (Ubuntu) or `/etc/init.d/rc5.d/S16xdm` (SUSE), which are both scripts to run a display manager. You will find that the `rcn.d` directories in `/etc/init.d` usually contain symbolic links to scripts in `/etc/init.d` without the leading S (or K) and number. The **S** indicates that the script should be run when the runlevel is entered, and the **K** indicates that the script should be run when the runlevel is terminated. The digits specify an order from 1 to 99 in which the scripts should be run.

Hint: Look for scripts that end in **dm** if you are trying to determine how the display manager is started.

You may find that the script for running a display manager, say `/etc/init.d/rc5.d/S16xdm`, may be a small script that contains additional logic to determine which display manager will really be run. So, while many systems allow these to be controlled through configuration, you can also find out what display manager will run by examining your initialization files.

It should come as no surprise that you can control whether your display manager is started at system startup simply by creating symbolic links for starting and stopping it in the appropriate `rcn.d` directory. Furthermore, if you need to stop or start the display manager, you can use the script from `/etc/init.d` directly as shown in Listing 12.

Listing 12. Stopping and starting a display manager

```
root@pinguino:~# /etc/init.d/gdm stop
* Stopping GNOME Display Manager... [ ok ]
root@pinguino:~# /etc/init.d/gdm start
* Starting GNOME Display Manager... [ ok ]
```

Now that you know how to control starting and stopping a display manager, let's look at configuring each of our three display managers.

XDM

The X Display Manager (XDM) is included in the XFree86 and X.Org packages. Under the Filesystem Hierarchy Standard, the configuration files should be located in `/etc/X11/xdm`. The main configuration file is `/etc/X11/xdm/xdm-config`. This file contains the location of other files used by XDM, information on authorization requirements, the names of scripts run to perform the various tasks for a user, and some other configuration information.

The `Xservers` file determines which local display or displays should be managed by XDM. It usually contains a single line as shown in Listing 13.

Listing 13. Sample `Xservers` file

```
:0 local /usr/X11R6/bin/X :0 vt07
```

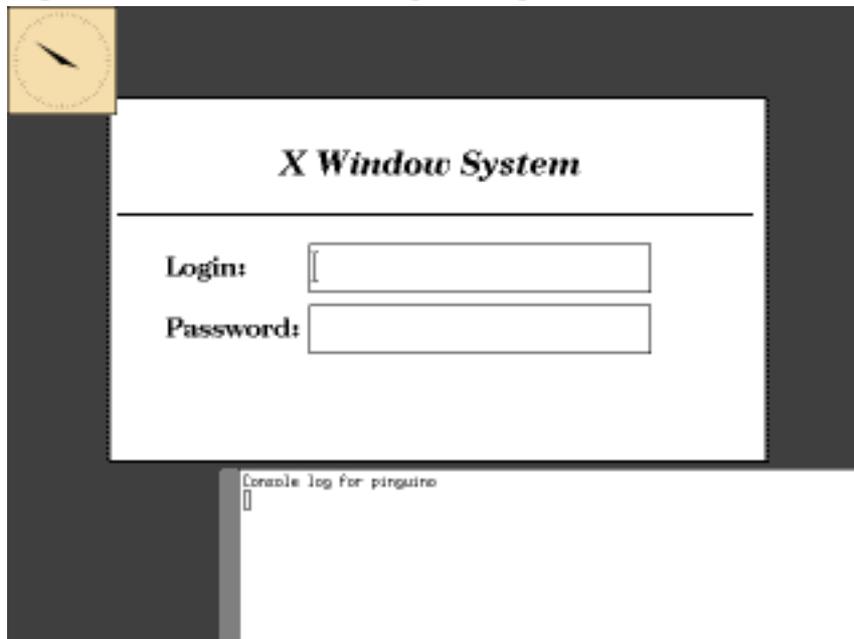
Listing 13 indicates that X should be run on virtual terminal 7. Most systems support using Ctrl-Alt-F1 through Ctrl-Alt-F7 to switch between virtual terminals, where vt01 through vt06 are text-mode terminals, and vt07 is the X terminal.

If you wish to support remote X terminals, then you need an `Xaccess` file. This file controls how XDM communicates with terminals that support the *X Display Manager Control Protocol (XDMCP)*. Terminals that do not support this protocol are defined in

the Xservers file. XDCMP uses the well-known UDP port 177. For security reasons, you should restrict XDCMP use to a trusted internal network with suitable firewall protection.

You can customize the way XDM works by updating the scripts in /etc/X11/xdm. In particular, the Xsetup (or Xsetup_0) script lets you customize the greeting. Figure 2 shows a simple XDM greeting with a digital clock added.

Figure 2. A modified XDM greeting



The source for the modified Xsetup_0 file is shown in Listing 14.

Listing 14. Sample Xsetup_0 file

```
#!/bin/sh
xclock -geometry 80x80 -bg wheat&
xconsole -geometry 480x130-0-0 -daemon -notify -verbose -fn fixed -exitOnFail
```

The greeting shown in Figure 2 came from a system using a 640x480 pixel screen resolution at 256 colors. XDM uses the default resolution from your XF86Config or xorg.conf file. To change your default system-wide screen resolution, you may edit this file or use the utilities that may have come with your system. Listing 15 shows the Screen section of an XF86Config file. Note that the DefaultDepth is 16, so the X server will try to run the screen at the first possible resolution specified for this depth, 1024x768 in this case.

Listing 15. Configuring screen resolution

```

Section "Screen"
    DefaultDepth 16
    SubSection "Display"
        Depth 15
        Modes "1280x1024" "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth 16
        Modes "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth 24
        Modes "1280x1024" "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth 32
        Modes "1280x1024" "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth 8
        Modes "1280x1024" "1024x768" "800x600" "640x480"
    EndSubSection
    Device "Device[0]"
    Identifier "Screen[0]"
    Monitor "Monitor[0]"
EndSection

```

Note that `Depth` refers to the number of bits that make up each pixel. You may also see this called *bits per pixel* or *bitplanes*. Thus, using 8 bitplanes or 8 bits for each color allows up to 256 colors, while a depth of 16 allows up to 65536 colors. With today's graphic cards, the higher depths of 24 and 32 are now common.

You can check the screen resolution using the `xwininfo` command with the `-root` option to see the characteristics of your running X server, as shown in Listing 16.

Listing 16. Checking the screen resolution

```

ian@lyrebird:~> xwininfo -display 0:0 -root
xwininfo: Window id: 0x36 (the root window) (has no name)

Absolute upper-left X: 0
Absolute upper-left Y: 0
Relative upper-left X: 0
Relative upper-left Y: 0
Width: 1024
Height: 768
Depth: 16
Visual Class: TrueColor
Border width: 0
Class: InputOutput
Colormap: 0x20 (installed)
Bit Gravity State: NorthWestGravity
Window Gravity State: NorthWestGravity
Backing Store State: NotUseful
Save Under State: no
Map State: IsViewable
Override Redirect State: no
Corners: +0+0 -0+0 -0-0 +0-0
-geometry 1024x768+0+0

```

KDM

KDM is the K Desktop Manager for the K Desktop Environment (KDE). KDE version 3 uses a `kdmrc` configuration file, a change from earlier versions, which used configuration information that was based on the `xdm` configuration files. This is located in the `$KDEDIR/share/config/kdm/` directory, where `$KDEDIR` might be `/etc/kde3/kdm/` or perhaps somewhere else. For example, on a SUSE SLES8 system, this is located in `/etc/opt/kde3/share/config/kdm`.

Listing 17. KDM configuration file - `kdmrc`

```
[Desktop0]
BackgroundMode=VerticalGradient
Color1=205,205,205
Color2=129,129,129
MultiWallpaperMode=NoMulti
Wallpaper=UnitedLinux-background.jpeg
WallpaperMode=Scaled

[X-*-Greeter]
GreetString=UnitedLinux 1.0 (%h)
EchoMode=OneStar
HiddenUsers=nobody,
BackgroundCfg=/etc/opt/kde3/share/config/kdm/kdmrc
MinShowUID=500
SessionTypes=kde,gnome,twm,failsafe

[General]
PidFile=/var/run/kdm.pid
Xservers=/etc/opt/kde3/share/config/kdm/Xservers

[Shutdown]
HaltCmd=/sbin/halt
LiloCmd=/sbin/lilo
LiloMap=/boot/map
RebootCmd=/sbin/reboot
UseLilo=false

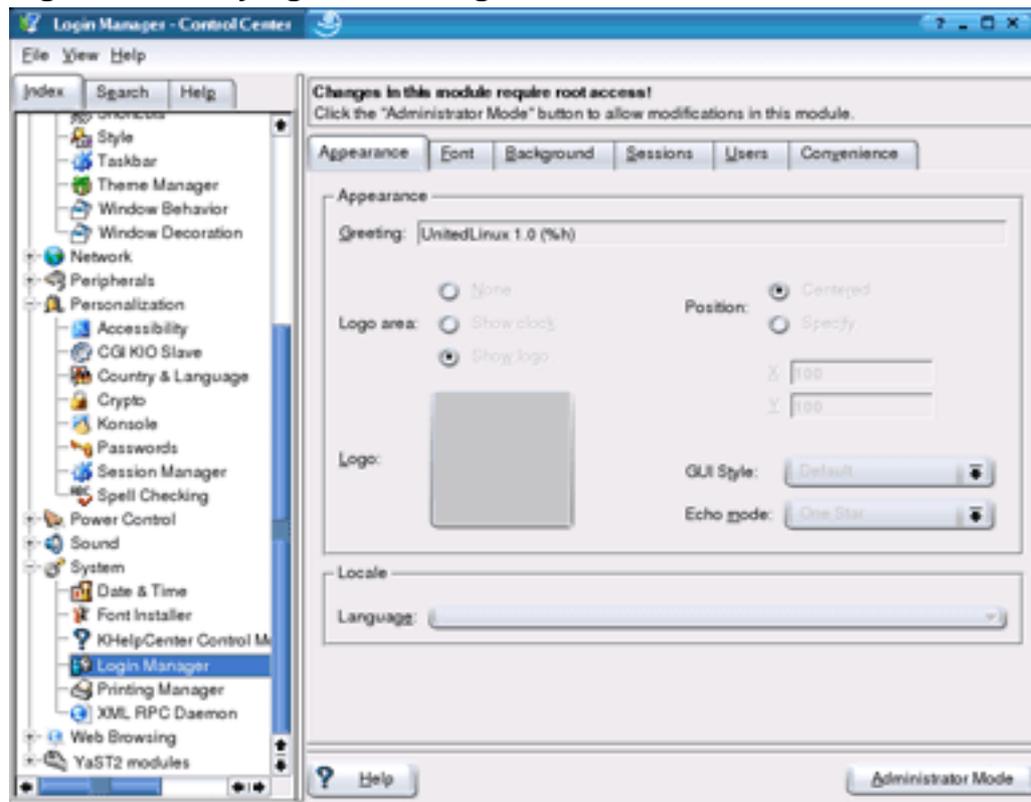
[X-*-Core]
Reset=/etc/X11/xdm/Xreset
Session=/etc/X11/xdm/Xsession
Setup=/opt/kde3/share/config/kdm/Xsetup
Startup=/etc/X11/xdm/Xstartup
AllowShutdown=Root

[Xdmcp]
Willing=/etc/X11/xdm/Xwilling
Xaccess=/etc/X11/xdm/Xaccess
```

Many sections contain the same type of configuration information as for XDM, but there are some differences. For example, the `SessionTypes` field allows KDM to start any one of several different session types; other commands allow KDM to shut down or reboot the system.

You can configure KDM by editing the `kdmrc` file. You can also change many of the Login Manager settings by using the KDE control center (`kcontrol`) as shown in

Figure 3.

Figure 3. Modifying KDM configuration with kcontrol

The KDM handbook (see [Resources](#)) contains extensive information on KDM configuration.

GDM

GDM is the GNOME Desktop Manager for the GNOME Desktop Environment. This desktop manager was not based on XDM, but was written from scratch. GDM uses a `gdm.conf` configuration file, normally located in the `/etc/X11/gdm` directory. Listing 18 shows part of a `gdm.conf` file.

Listing 18. Partial GDM configuration file - `gdm.conf`

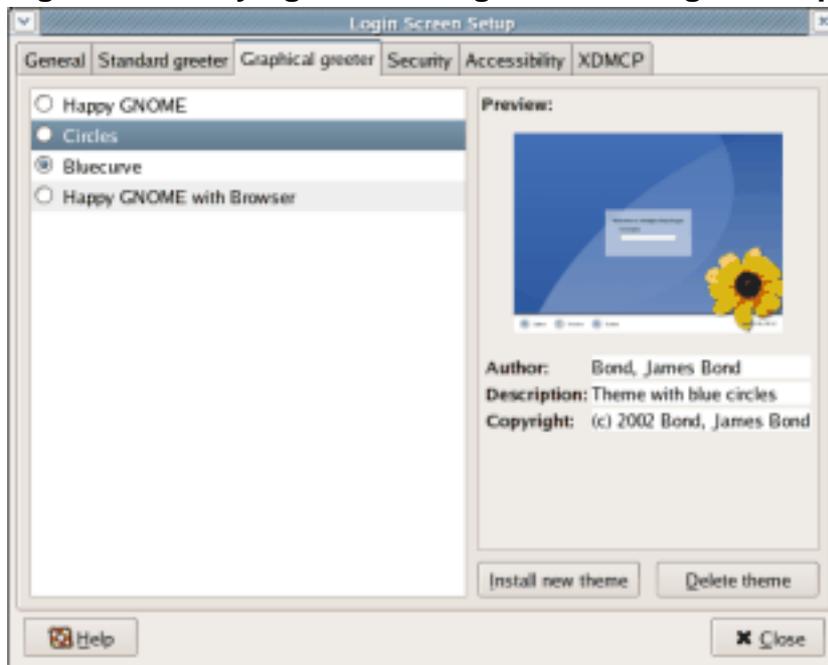
```
# You should probably never change this value unless you have a weird setup
PidFile=/var/run/gdm.pid
# Note that a post login script is run before a PreSession script.
# It is run after the login is successful and before any setup is
# run on behalf of the user
PostLoginScriptDir=/etc/X11/gdm/PostLogin/
PreSessionScriptDir=/etc/X11/gdm/PreSession/
PostSessionScriptDir=/etc/X11/gdm/PostSession/
DisplayInitDir=/etc/X11/gdm/Init
...
```

```
# Probably should not touch the below this is the standard setup
ServAuthDir=/var/gdm
# This is our standard startup script. A bit different from a normal
# X session, but it shares a lot of stuff with that. See the provided
# default for more information.
BaseXsession=/etc/X11/xdm/Xsession
# This is a directory where .desktop files describing the sessions live
# It is really a PATH style variable since 2.4.4.2 to allow actual
# interoperability with KDM. Note that <sysconfdir>/dm/Sessions is there
# for backwards compatibility reasons with 2.4.4.x
#SessionDesktopDir=/etc/X11/sessions/:/etc/X11/dm/Sessions/:/usr/share/gdm/Buil\
tInSessions/:/usr/share/xsessions/
# This is the default .desktop session. One of the ones in SessionDesktopDir
DefaultSession=default.desktop
```

Again, you will see some similarities in the type of configuration information used for GDM, KDM, and XDM, although `gdm.conf` is a much larger file with many more options.

You can configure GDM by editing the `gdm.conf` file. You can also change many of the settings by using the `gdmsetup` command. Figure 4 shows an alternate graphical greeter available on a Fedora system.

Figure 4. Modifying GDM configuration with `gdmsetup`



The GNOME Display Manager Reference Manual (available from the `gdmsetup` help, or see [Resources](#)) contains extensive information on GDM configuration.

Section 4. Customize a window manager

This section covers material for topic 1.110.4 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn how to:

- Customize a system-wide desktop environment or window manager
- Customize window manager menus and desktop panel menus
- Configure an X terminal
- Verify and resolve library dependency issues for X applications
- Export an X display

Window managers

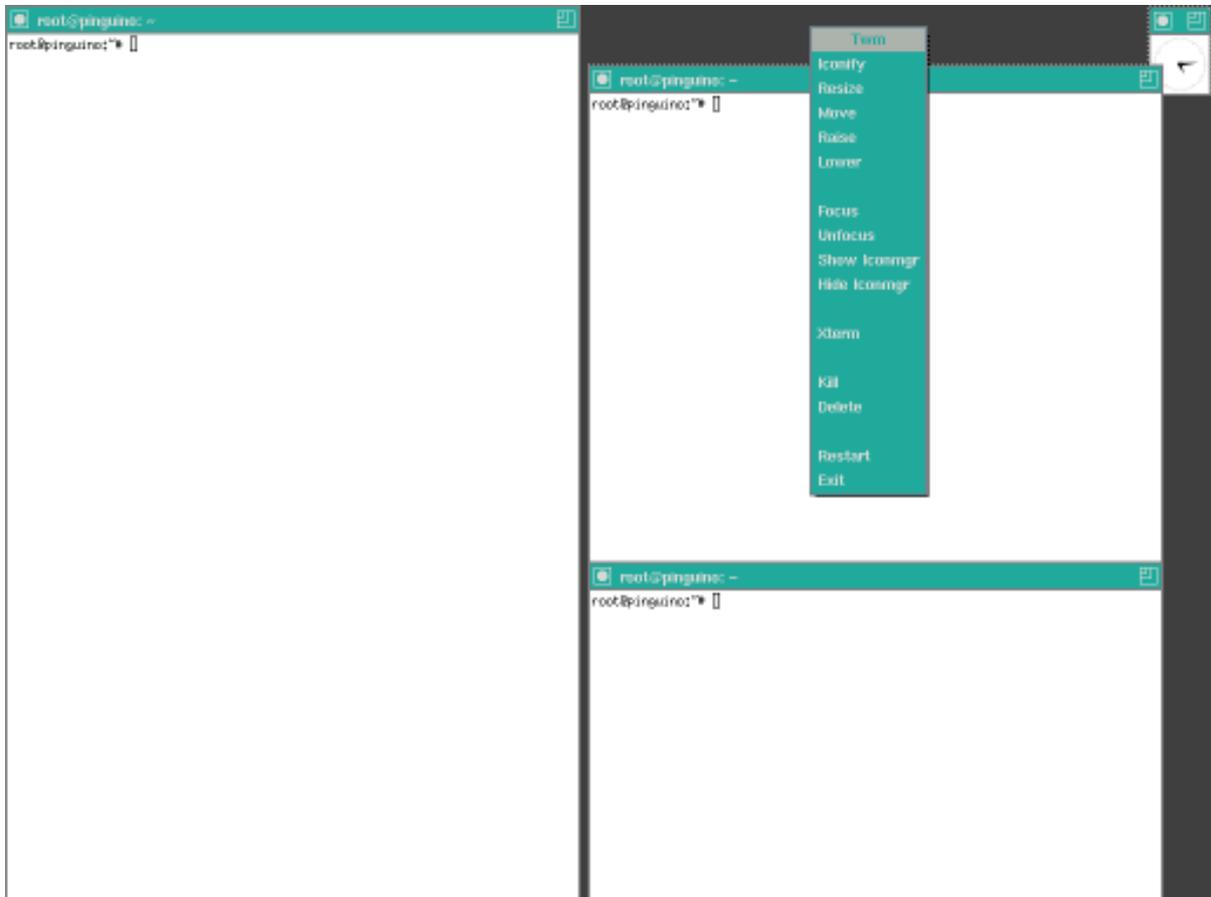
In the previous section, you learned about display managers and how to set them up. You also learned in this tutorial that while X is a toolkit enabling applications to create graphical windows, it does not specify a user interface. In this section, you learn more about user interfaces and how to configure what happens after you have an X session up and running.

You can imagine that, without any user interface specification, developer creativity might result in many different styles of windows, all vying for space on your screen, and all with different keystrokes, mouse operations, and styles for things like buttons, dialog boxes, and so on. To bring some order to chaos, higher level toolkits were developed. These in turn gave rise to *window managers*, such as twm, fwm, and fwm2 and finally to desktops such as KDE and GNOME.

Desktops provide a consistent user experience but also consume considerable CPU and memory resources. Before computers had the power to support the likes of a KDE or GNOME desktop, window managers were popular, and many users still like them for their light weight and fast response.

If you have just installed X and you type in the command `startx`, then you will see a display something like Figure 5.

Figure 5. Running twm with startx



This is the twm window manager, shown with the menu that is obtained by pressing mouse button 1 (normally the left button for right-handed users) over the background. You will notice three terminal windows and an analog clock, but no task bars, launchers, or other desktop paraphernalia.

The `startx` command is actually a front end command to `xinit`, which starts the X server process and some client applications. It is usually located in `/usr/X11R6/bin`, as is `xinit` and many other X utilities. X applications can take settings from an X resources database as well as a command line. Table 6 summarizes the names and purpose of each of the configuration files used by either `startx` or `xinit`. Note that some or all of these files may not be present on a particular system or user home directory.

Table 6. Configuration files for startx and xinit	
File	Description
<code>\$HOME/.xinitrc</code>	User-defined executable script that merges in resource files and starts client applications
<code>\$HOME/.xserverrc</code>	User-defined executable script that provides overrides for default X

	server configuration
/usr/X11R6/lib/X11/xinit/xinitrc	System default executable script that merges in resource files and starts client applications
/usr/X11R6/lib/X11/xinit/xserverrc	System default executable script that provides overrides for default X server configuration
\$HOME/.Xresources	User-defined file describing resources for X applications
\$HOME/.Xmodmap	User-defined file defining keyboard and mouse settings
/usr/X11R6/lib/X11/xinit/.Xresources	System default file describing resources for X applications
/usr/X11R6/lib/X11/xinit/.Xmodmap	System default file defining keyboard and mouse settings

Note carefully that the system `xinitrc` and `xserverrc` file omit the leading dot, while all the others have it.

Every window on a screen, and indeed every widget on the screen, has attributes such as height, width, and placement (geometry), foreground and background colors or images, title text and color, and so on. For a new client application, most of these values can be supplied on the command line. Since there are many attributes, it is easier to have defaults. Such defaults are stored in a *resource database*, which is built from resource files using the `xrdb` command.

Listing 19 shows the default `xinit` file shipped with XFree86 4.5.0.

Listing 19. Sample `xinit` file - `/usr/X11R6/lib/X11/xinit/xinitrc`

```
#!/bin/sh
# $Xorg: xinitrc.cpp,v 1.3 2000/08/17 19:54:30 cpqbld Exp $

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi
```

```
if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# start some nice programs

twm &
xclock -geometry 50x50-1+1 &
xterm -geometry 80x50+494+51 &
xterm -geometry 80x20+494-0 &
exec xterm -geometry 80x66+0+0 -name login
```

Note that the `xrdb` command is used to merge in the resources, and the `xmodmap` is used to update the keyboard and mouse definitions. Finally, several programs are started in the background with a final program started in the foreground using the `exec` command, which terminates the current script (`xinitrc`) and transfers control to the `xterm` window with geometry `80x66+0+0`. This is the login window, and terminating it will terminate the X server. There must be one such application, although some people prefer the window manager (`twm` in this example) to have this role. All other applications should be started in the background so the script can complete.

The first two values in the geometry specification define the size of the window. For a clock, this is in pixels, while for the `xterm` windows, it is in lines and columns. If present, the next two values define the placement of the window. If the first value is positive, the window is placed relative to the left edge of the screen, while a negative value causes placement relative to the right edge. Similarly, positive and negative second values refer to the top and bottom of the screen respectively.

Suppose you want your clock to be larger with a different colored face and to be located in the bottom right of the screen instead of the top right. If you just want this for one user, copy the above file to the user's home directory as `.xinitrc` (remember the dot) and edit the clock specification as shown in Listing 20. You can find all the color names in the `rgb.txt` file in your X installation directory tree (for example, `/usr/X11R6/lib/X11/rgb.txt`).

Listing 20. Modifying `xclock` startup in `xinitrc`

```
xclock -background mistyrose -geometry 100x100-1-1 &
```

If you want to update the defaults for the whole installation, you should update the `/usr/X11R6/lib/X11/xinit/.Xresources` and `/usr/X11R6/lib/X11/xinit/.Xmodmap` files instead of the dot versions for individual users.

Several tools can help you customize windows and keystrokes.

xrdb

Merges resources from a resource file into the X resource database for the

running X server. By default it passes the source resource file through a C++ compiler. Specify the `-nocpp` option if you do not have a compiler installed.

xmodmap

Sets keyboard and mouse bindings. For example, you can switch settings for left-handed mouse usage, or set up the backspace and delete keys to work the way you are accustomed to.

xwininfo

Tells you information about a window, including geometry information.

editres

Lets you customize the resources for windows on your screen, see the changes, and save them in a file that you can later use with `xrdb`.

xev

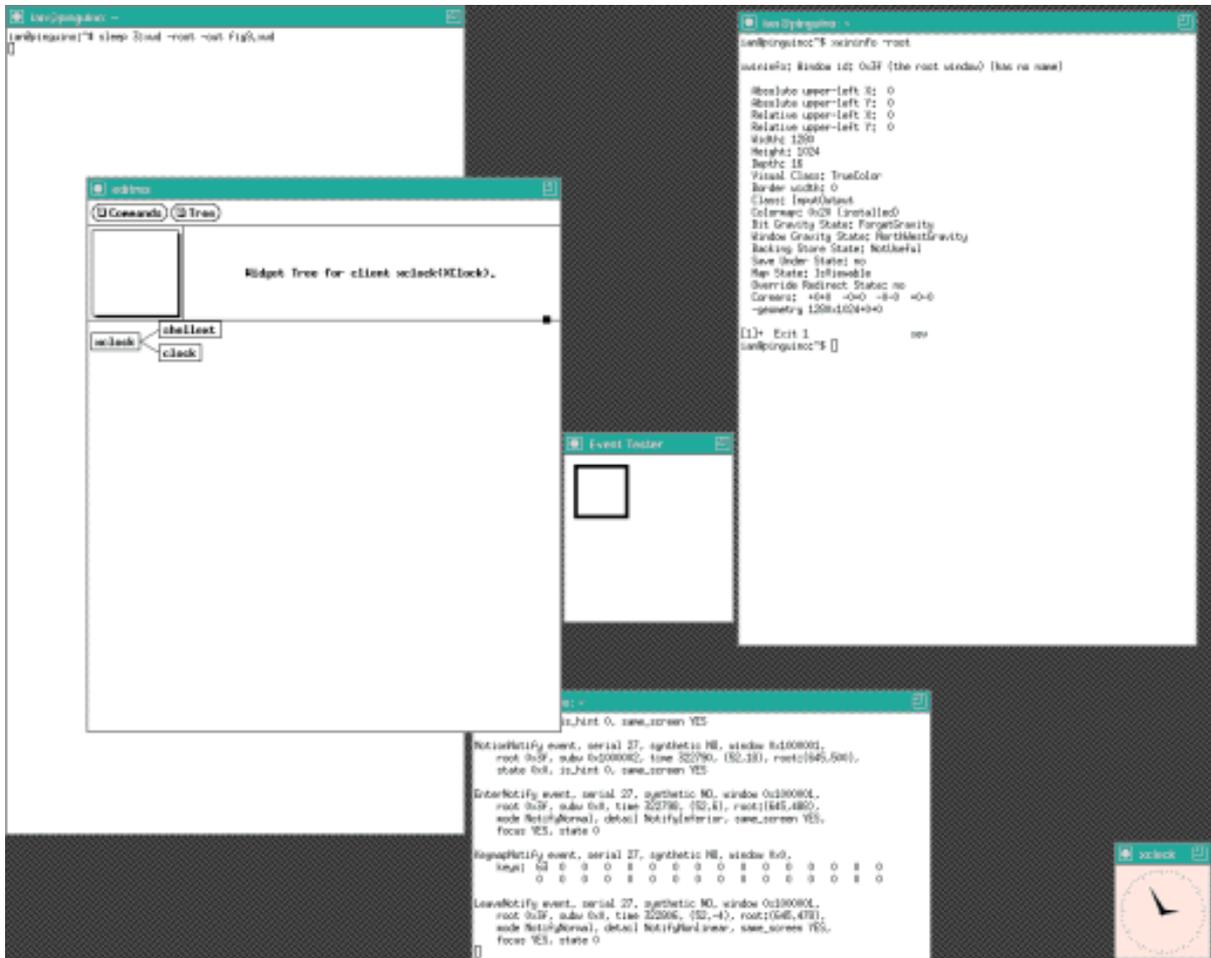
Starts a window and captures X events that are displayed in the originating xterm window. Use this to help determine the appropriate codes to use when customizing keyboards or checking mouse events.

Use the man pages to find out more about each of these commands.

Figure 6 shows a busy screen with several of these commands running.

- The upper left terminal window (the login window) is using `xwd` to capture the whole screen to a file.
- The window overlapping it is running `editres` to modify the resources for the clock window.
- The small central window is running `xev`, and the output is going to the terminal window below. The right-hand window shows the output of `xwininfo` for the root window (the whole screen).
- The customized clock with its face colored `mistyrose` is running in the bottom right corner of the screen.

Figure 6. Window information and configuration



Besides the windows themselves, the window manager also allows customization. For example, the menu that was shown in Figure 5 is configured in a twm customization file. The system default is in the X installation directory tree (/usr/X11R6/lib/X11/twm/system.twmrc), and individual users may have a .twmrc file. If a user has multiple displays, there can be files (such as .twmrc.0 or .twmrc.1) for each display number. Listing 21 shows the part of the system.twmrc file that defines the menu shown in Figure 5.

Listing 21. Menu configuration in twm

```

menu "defops"
{
  "Twm"      f.title
  "Iconify"  f.iconify
  "Resize"   f.resize
  "Move"     f.move
  "Raise"    f.raise
  "Lower"    f.lower
  " "        f.nop
  "Focus"    f.focus
  "Unfocus" f.unfocus
  "Show Iconmgr" f.showiconmgr
}
    
```

```

"Hide Iconmgr"  f.hideiconmgr
" "            f.nop
"Xterm"        f.exec "exec xterm &"
" "            f.nop
"Kill"         f.destroy
"Delete"       f.delete
" "            f.nop
"Restart"      f.restart
"Exit"         f.quit
}

```

See the man page for twm or your preferred window manager for more information.

Desktops

If you are using a display manager or desktop, you will find that these also can be configured. Indeed, you already saw the Xsetup_0 file for XDM in the previous section. As with the window manager configuration that you have just seen, desktop configuration can be done system-wide or per-user.

GNOME customization

GNOME is configured mostly using XML files. System defaults are found in directories in the /etc filesystem, such as /etc/gconf, /etc/gnome, and /etc/gnome-vfs2..0, along with other directories for specific GNOME applications. User configuration is usually found in subdirectories of the home directory that start with .g. Listing 22 shows several of the possible locations for GNOME configuration information.

Listing 22. GNOME configuration locations

```

[ian@lyrebird ian]$ ls -d /etc/g[nc]*
/etc/gconf /etc/gnome /etc/gnome-vfs-2.0 /etc/gnome-vfs-mime-magic
[ian@lyrebird ian]$ find . -maxdepth 1 -type d -name ".g[nc]*"
./.gnome2
./.gconfd
./.gconf
./.gnome
./.gnome2_private
./.gnome-desktop
./.gnome_private

```

Rather than extensive man pages, GNOME has an online manual; you can access the manual from the `gnome-help` command, or from a menu item such as Desktop > Help. As of this writing, the manual has three major sections: Desktop, Applications, and Other Documentation. The table of contents for a recent version of the Desktop Help is shown in Figure 7.

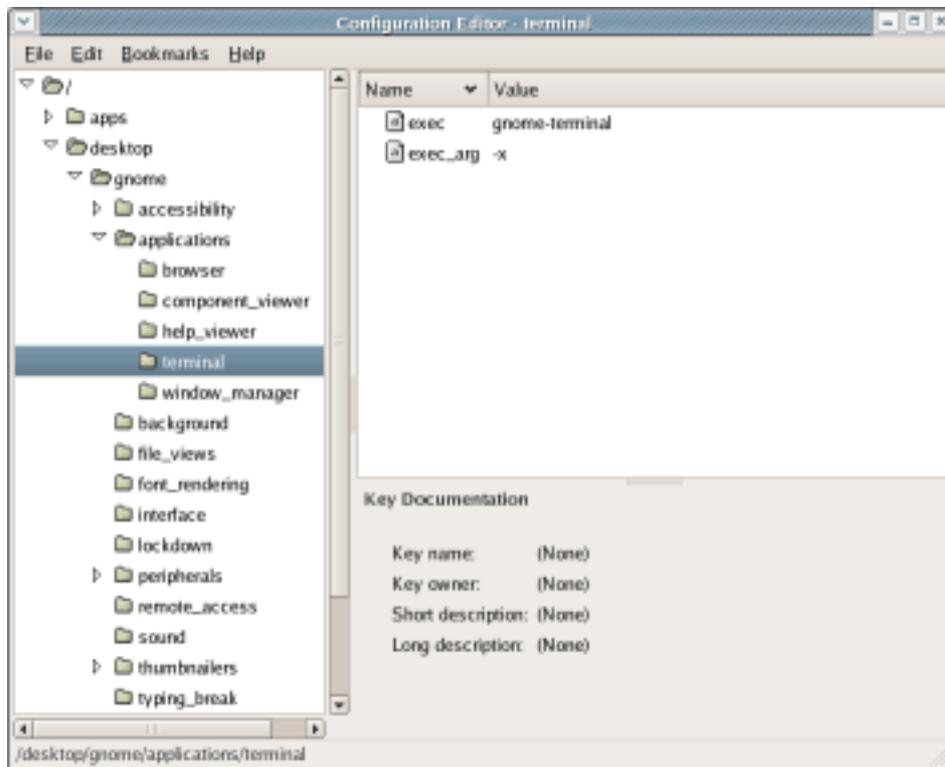
Figure 7. Window information and configuration



You will find information about configuration tools under the System Administration Guide in the Desktop section and also under the Configuration Editor Manual in the applications topic in the Desktop section.

You may launch the graphical configuration editor using the `gconf-editor` command or by selecting Configuration Editor from the Applications > System Tools menu. The configuration for the `gnome-terminal` application is shown in Figure 8.

Figure 8. Window information and configuration



In addition to graphical tools, there is also a `gconftool-2` command line program for interrogating and updating GNOME configuration information. See the above mentioned System Administration Guide for more details.

KDE customization

KDE is configured using plain text files with UTF-8 encoding for non-ASCII characters. As with GNOME, there may be many different configuration files. If multiple identically named files exist in different parts of the configuration tree, then the values from these are merged. System values are located in the `$KDEDIR/share/config` tree, where `$KDEDIR` might be `/etc/kde3/kdm/` or perhaps somewhere else. For example, on a SUSE SLES8 system, this is located in `/etc/opt/kde3/share/config`. User-specific customizations are located in the `.kde/share` tree in the user's home directory.

Configuration files have one or more group names enclosed in square brackets, followed by key and value pairs. The key may contain spaces and is separated from the value by an equal sign. The configuration file for the konqueror browser is shown in Listing 23.

Listing 23. KDE configuration file for konqueror browser

```
[HTML Settings]

[Java/JavaScript Settings]
```

```
ECMADomainSettings=localhost::Accept
JavaPath=/usr/lib/java2/jre/bin/java
EnableJava=true
EnableJavaScript=true

[EmbedSettings]
embed-text=true
embed-audio=false
embed-video=false

[Reusing]
MaxPreloadCount=1
PreloadOnStartup=true
```

The configuration files may be edited manually. Most systems will include a graphical editing tool, such as KConfigEditor or a tool customized for the distribution, such as the SUSE Control Center.

Different xterms

The usual `xterm` program that is installed with graphical desktops is very functional, but also uses quite a lot of system resource. If you are running a system with many X terminal clients running on a single processor, you may want to consider using a lighter-weight terminal. Two such are `rxvt` and `aterm` (which was built on top of `rxvt`). These are VT102 emulators, which are not usually installed by default, so you will need to install them.

Required libraries

By now you may be realizing that there are many different libraries and toolkits that are used for X applications. So how do you ensure that you have the right libraries? The `ldd` command lists library dependencies for any application. In its simplest form, it takes the name of an executable and prints out the libraries required to run the program. Note that `ldd` does not automatically search your `PATH`, so you usually need to give the path (relative or absolute) as well as the program name, unless the program is in the current directory. Listing 24 shows the library dependencies for the three terminal emulators that you saw above. The number of library dependencies for each gives you a clue about the relative system requirements of each.

Listing 24. Library dependencies for `xterm`, `aterm`, and `rxvt`

```
root@pinguino:~# ldd `which xterm`
linux-gate.so.1 => (0xffffe000)
libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7fab000)
libfontconfig.so.1 => /usr/X11R6/lib/libfontconfig.so.1 (0xb7f88000)
libfreetype.so.6 => /usr/X11R6/lib/libfreetype.so.6 (0xb7f22000)
libexpat.so.0 => /usr/X11R6/lib/libexpat.so.0 (0xb7f06000)
libXrender.so.1 => /usr/X11R6/lib/libXrender.so.1 (0xb7eff000)
```

```

libXaw.so.7 => /usr/X11R6/lib/libXaw.so.7 (0xb7ead000)
libXmu.so.6 => /usr/X11R6/lib/libXmu.so.6 (0xb7e99000)
libXt.so.6 => /usr/X11R6/lib/libXt.so.6 (0xb7e4f000)
libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0xb7e46000)
libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0xb7e30000)
libXpm.so.4 => /usr/X11R6/lib/libXpm.so.4 (0xb7e22000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7e15000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7d56000)
libncurses.so.5 => /lib/libncurses.so.5 (0xb7d15000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7be6000)
libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7be3000)
/lib/ld-linux.so.2 (0xb7fc3000)
root@pinguino:~# ldd `which aterm`
linux-gate.so.1 => (0xffffe000)
libXpm.so.4 => /usr/X11R6/lib/libXpm.so.4 (0xb7f81000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ec1000)
libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0xb7eb9000)
libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0xb7ea3000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7d75000)
libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7d72000)
/lib/ld-linux.so.2 (0x80000000)
root@pinguino:~# ldd `which rxvt`
linux-gate.so.1 => (0xffffe000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7eb0000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7d81000)
libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7d7e000)
/lib/ld-linux.so.2 (0x80000000)

```

Exporting a display

An X display is known by a name of the form *hostname:displaynumber.screennumber*. For Linux running on a workstation such as a PC, there is typically only one display with a single screen. In this case, the *displayname* may be, and usually is, omitted so the display is known as :0.0. The DISPLAY environment variable is usually set to the display name., so you can display it using the command `echo $DISPLAY`. Depending on your system, this variable may or may not be set if you use `su -` to switch to another user. In such a case, you may need to set and export the DISPLAY as shown in Listing 25. In this listing you see an attempt to start the `xclock` application after switching to root, but the attempt fails because the DISPLAY environment variable is not set. Even if the DISPLAY variable is set, you still may not be able to use the display, as you will also need authorization to do so.

Listing 25. Attempting to start xclock

```

ian@lyrebird:~> whoami
ian
ian@lyrebird:~> echo $DISPLAY
:0.0
ian@lyrebird:~> su -
Password:
lyrebird:~ # echo $DISPLAY

lyrebird:~ # xclock
Error: Can't open display:
lyrebird:~ # export DISPLAY=:0.0

```

```
lyrebird:~ # echo $DISPLAY
:0.0
lyrebird:~ # xclock
Xlib: connection to ":0.0" refused by server
Xlib: No protocol specified

Error: Can't open display: :0.0
lyrebird:~ # export XAUTHORITY=~ian/.Xauthority
lyrebird:~ # xclock
lyrebird:~ # ls -l ~ian/.Xauthority
-rw----- 1 ian users 206 Feb 18 16:20 /home/ian/.Xauthority
```

Let's take a look at what is going on here. In this case, the user `ian` logged in to the system and his `DISPLAY` environment was set to `:0.0` as we expect. When user `ian` switched to user `root`, the `DISPLAY` environment variable was not set, and an attempt to start `xclock` failed because the application did not know what display to use.

So the substituted user, `root`, set the `DISPLAY` environment variable, and exported it so that it would be available to other shells that might be started from this terminal window. Note that setting and exporting an environment variable does not use the leading `$` sign, while displaying or otherwise using the value does. Note too, that if the `su` command had omitted the `-` (minus) sign, the `DISPLAY` environment variable would have been set as it had been for user `ian`. Nevertheless, even with the environment variable set, `xclock` still failed.

The reason for the second failure lies in the client/server nature of X. Although `root` is running in a window on the one and only display on this system, the display is actually owned by the user who logged in originally, `ian` in this case. Let's take a look at X authorization.

Authorization methods

For local displays on Linux systems, authorization usually depends on a so-called **MIT-MAGIC-COOKIE-1**, which is usually regenerated every time the X server restarts. A user can *extract* the magic cookie from the `.Xauthority` file in his or her home directory (using the `xauth extract` command and give it to another user to *merge* into that user's `.Xauthority` file using the `xauth merge` command. Alternatively, a user can grant authority for other users to access the local system using the `xhost +local:` command.

XAUTHORITY

Another alternative is to set the `XAUTHORITY` environment variable to the location of a file containing the appropriate MIT-MAGIC-COOKIE-1. When switching to `root`, it is easy to do this, since `root` can read files owned by other users. This is, in fact, what we did in Listing 25, so after setting and exporting `XAUTHORITY` to `~ian/.Xauthority`, `root` is now able to open graphical windows on the desktop. We said we'd mention a difference with Red Hat systems. Using `su` to switch to `root` on

a Red Hat system is slightly different on a SUSE system, where the display setup is done automatically for you.

So what if you switch to another non-root user? You will notice from Listing 25 that the `.Xauthority` file for user `ian` allows only user read and write access. Even members of the same group cannot read it, which is what you want unless you'd like someone to open up an application that takes over your screen and prevents you from doing anything! So, if you do extract an MIT-MAGIC-COOKIE-1 from your `.Xauthority` file, then you have to find some secure way to give it to your friendly non-root user. Another approach is to use the `xhost` command to grant authority to any user on a particular host.

The `xhost` command

Because of the difficulty of securely passing an MIT-MAGIC-COOKIE-1 cookie to another user, you may find that, for a Linux system with a single user, `xhost` is easier to use, even though the `xauth` approach is generally preferred over the `xhost` command. However, keep in mind the network heritage of the X Window System so that you do not accidentally grant more access than you intended and thereby open up your system and allow arbitrary network users to open windows on your desktop.

To give all local users authority to open applications on the display (`:0.0`), user `ian` can use the `xhost` command. Open a terminal window on your desktop and enter this command:

```
xhost +local:
```

Note the trailing colon (`:`). This will allow other users on the same system to connect to the X server and open windows. Since you are a single-user system, this means that you can `su` to an arbitrary non-root user and can now launch `xclock` or other X applications.

You may also use `xhost` to authorize remote hosts. This is generally not a good idea except on restricted networks. You will also need to enable the appropriate ports on your firewall if you are using one.

Another alternative for using X applications from another system is to connect to the system using a *secure shell* (`ssh`) connection. If your default `ssh` client configuration does not enable X forwarding, you may need to add the `-X` parameter to your `ssh` command. The `ssh` server will also need to have X forwarding enabled. In general, this is a much more secure method of using X remotely than opening up your system to arbitrary connections using `xhost`.

For more details on using the `xauth` and `xhost` commands, you can use the commands `info xauth`, `man xauth`, `info xhost`, or `man xhost` as appropriate to view the online manual pages. If you are interested in security for X connections,

start with the manual pages for Xsecure.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, February 2006), learn how to open a terminal window or shell prompt and much more.
- Get [documentation for XFree86 version 4.5.0](#) and other versions on the XFree86 site.
- Get [documentation for the X Window System Version 11 Release 6.9 and 7.0](#) on the X.Org site.
- The [X.Org Modular Tree Developer's Guide](#) will help you build the X.Org releases from source.
- An [Xft tutorial](#) describes the Xft font mechanism introduced in XFree86 4.0.2.
- [kde.org](#) is the home of the K Desktop Environment and [KDE documentation](#), including *The kdm Handbook* and *KDE for System Administrators*.
- [The GNOME Foundation](#) is the home of the GNOME Desktop Environment and the [Gnome Display Manager Reference Manual](#).
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *The Concise Guide to Xfree86 for Linux* (Que, 1999) provides more detail on installing, configuring and using X.
- *LPI Linux Certification in a Nutshell* (O'Reilly, 2001) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are references for those who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Download XFree86](#) from The XFree86 Project, Inc.
- [Download X.Org](#) from The X.Org Foundation.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software

for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

- [Download IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content.](#)
- Read [developerWorks blogs](#), and get involved in the developerWorks community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

LPI exam 102 prep, Topic 111: Administrative tasks

Junior Level Administration (LPIC-1) topic 111

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

10 Jul 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this sixth in a [series of nine tutorials](#), Ian introduces you to administrative tasks. By the end of this tutorial, you will know how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *intermediate level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active intermediate level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the four junior and intermediate certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts.
Topic 111	LPI exam 102 prep: Administrative tasks	(This tutorial.) Learn how to manage user and group accounts and tune user and system environments, configure and use system log files, automate system administration tasks by scheduling jobs to run at another time, back up your system, and maintain system time. See the detailed objectives below.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Administrative tasks," the sixth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.111.1 User and group accounts	Weight 4	Add, remove, suspend, and change user accounts. Manage user and group information in password and group databases, including shadow databases. Create and manage special purpose and limited accounts.
1.111.2 Tune user and system environments	Weight 3	Modify global and user profiles. Set environment variables and maintain skeleton directories for new user accounts. Set command search paths.
1.111.3 Configure and use system log files to meet administrative and security needs	Weight 3	Configure and manage system logs, including the type and level of logged information. Scan and monitor log files for

		notable activity and track down noted problems. Rotate and archive log files.
1.111.4 Automate system administration tasks by scheduling jobs to run in the future	Weight 4	Use the <code>cron</code> or <code>anacron</code> commands to run jobs at regular intervals, and use the <code>at</code> command to run jobs at a specific time.
1.111.5 Maintain an effective data backup strategy	Weight 3	Plan a backup strategy and back up filesystems automatically to various media.
1.111.6 Maintain system time	Weight 4	Maintain the system time and time zone, and synchronize the clock via NTP. Set the BIOS clock to the correct time in UTC, and configure NTP, including correcting for clock drift.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be thoroughly familiar with the material from the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, which covers basic concepts of users, groups, and file permissions.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. User and group accounts

This section covers material for topic 1.111.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Add, modify, and remove users and groups
- Suspend and change user accounts
- Manage user and group information in the password databases and group databases
- Use the correct tools to manage shadow password databases and group databases
- Create and manage limited and special-purpose accounts

As you learned in the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, Linux is a multi-user system where each user belongs to one *primary* group and possibly to additional groups. Ownership of files in Linux is closely related to user ids and groups. Recall that you can log in as one user and become another user using the `su` or `sudo -s` commands, and that you can use the `whoami` command to check your current effective id and the `groups` command to find out what groups you belong to. In this section, you learn how to create, delete, and manage users and groups. You also learn about the files in `/etc`, where user and group information is stored.

Add and remove users and groups

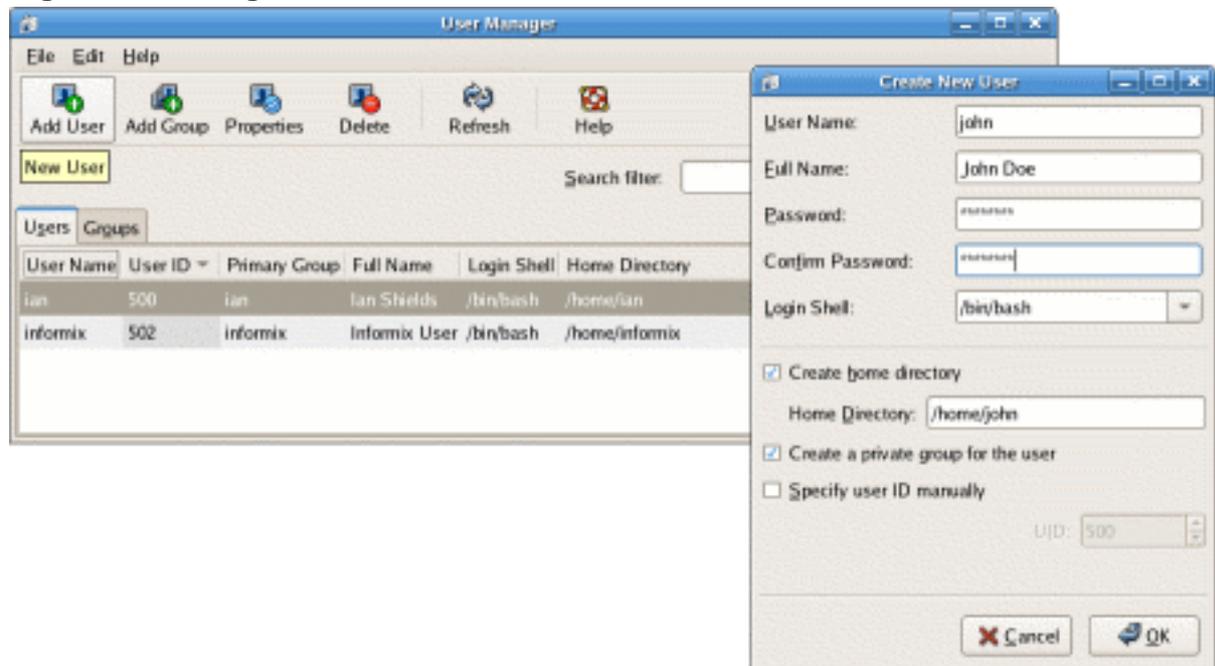
You add a user to a Linux system using the `useradd` command, and you delete a user using the `userdel` command. Similarly, you add or delete groups using the `groupadd` and `groupdel` commands.

Adding a user or group

Modern Linux desktops usually have graphical interfaces for user and group administration. The graphical interface is usually accessed through menu options for system administration. These interfaces do vary considerably, so the one on your system may not look much like the example here, but the underlying concepts and commands remain similar.

Let's start by adding a user to a Fedora Core 5 system graphically, and then examine the underlying commands. In the case of Fedora Core 5 with GNOME desktop, use **System > Administration > Users and Groups**, then click the **Add User** button.

Figure 1 depicts the User Manager panel with the Create New User panel showing basic information for a new user named 'john'. The full name of the user, John Doe, and a password have been entered. The panel provides a default login shell of `/bin/bash`. On Fedora systems, the default is to create a new group with the same name as the user, 'john' in this case, and a home directory of `/home/john`.

Figure 1. Adding a user

Listing 1 shows the use of the `id` command to display basic information about the new user. As you can see, john has user number 503 and a matching group, john, with group number 503. This is the only group of which john is a member.

Listing 1. Displaying user id information

```
[root@pinguino ~]# id john
uid=503(john) gid=503(john) groups=503(john)
```

To accomplish the same task from the command line, you use the `groupadd` and `useradd` commands to create the group and user, then use the `passwd` command to set the password for the newly created user. All of these commands require root authority. The basic use of these commands to add another user, jane, is illustrated in Listing 2.

Listing 2. Adding user jane

```
[root@pinguino ~]# groupadd jane
[root@pinguino ~]# useradd -c "Jane Doe" -g jane -m jane
[root@pinguino ~]# passwd jane
Changing password for user jane.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# id jane
uid=504(jane) gid=504(jane) groups=504(jane)
[root@pinguino ~]# ls -ld /home/jane
drwx----- 3 jane jane 4096 Jun 25 18:22 /home/jane
```

In these two examples, both the user id and the group id have values greater than 500. Be aware that some newer systems start user ids at 1000 rather than 500. These values normally signify ordinary users, while values below 500 (or 1000 if the system starts ordinary users at 1000) are reserved for *system users*. [System users](#) are covered later in this section. The actual cutoff points are set in `/etc/login.defs` as `UID_MIN` and `GID_MIN`.

In Listing 2 above, the `groupadd` command has a single parameter, `jane`, the name of the group to be added. Group names must begin with a lower case letter or an underscore, and usually contain only these along with hyphens or dashes. Options you may specify are shown in Table 3.

Option	Purpose
-f	Exit with success status if the group already exists. This is handy for scripting when you do not need to check if a group exists before attempting to create it.
-g	Specifies the group id manually. The default is to use the smallest value that is at least <code>GID_MIN</code> and also greater than the id of any existing group. Group ids are normally unique and must be non-negative
-o	Permits a group to have a non-unique id.
-K	Can be used to override defaults from <code>/etc/login.defs</code> .

In Listing 2 above, the `useradd` command has a single parameter, `jane`, the name of the user to be added, along with the `-c`, `-g`, and `-m` options. Common options for the `useradd` command are shown in Table 4.

Option	Purpose
-b --base-dir	The default base directory in which user home directories are created. This is usually <code>/home</code> , and the user's home directory is <code>/home/\$USER</code> .
-c --comment	A text string describing the id, such as the user's full name.
-d --home	Provides a specific directory name for the home directory.
-e	The date on which the account will

--expiredate	expire or be disabled in the form YYYY-MM_DD.
-g --gid	The name or number of the initial login group for the user. The group must exist, which is why group jane was created before user jane in Listing 2.
-G --groups	A comma-separated list of additional groups to which the user belongs.
-K	Can be used to override defaults from /etc/login.defs.
-m --create-home	Create the user's home directory if it does not exist. Copy the skeleton files and any directories from /etc/skel to the home directory.
-o --non-unique	Permits a user to have a non-unique id.
-p --password	The encrypted password. If a password is not specified, the default is to disable the account. You will usually use the <code>passwd</code> command in a subsequent step rather than generating an encrypted password and specifying it on the <code>useradd</code> command.
-s --shell	The name of the user's login shell if different from the default login shell.
-u --uid	The non-negative numerical userid, which must be unique if <code>-o</code> is not specified. The default is to use the smallest value that is at least <code>UID_MIN</code> and also greater than the id of any existing user.

Notes:

1. Some systems, including Fedora and Red Hat distributions, have extensions to the user-creation commands. For example, the default Fedora and Red Hat behavior is to create a new group for a user, and the `-n` option can be used on the `useradd` command to disable this function. Be aware of such possible system differences and refer to the man pages on your system when in doubt.
2. On SUSE systems, use YaST or YaST2 to access graphical user and group administration interfaces.
3. Graphical interfaces may perform additional tasks such as creating the user's mail file in `/var/spool/mail`.

Deleting a user or group

Deleting a user or group is much simpler than adding one, because there are fewer options. In fact, the `groupdel` command to delete a group requires only the group name; it has no options. You cannot delete any group that is the primary group of a user. If you use a graphical interface for deleting users and groups, the functions are very similar to the commands shown here.

Use the `userdel` command to delete a user. The `-r`, or `--remove` option requests removal of the user's home directory and anything it contains, along with the user's mail spool. When you delete a user, a group with the same name as the user will also be deleted if `USERGROUPS_ENAB` is set to `yes` in `/etc/login.defs`, but this will be done only if the group is not the primary group of another user.

In Listing 3 you see an example of deleting groups when multiple users share the same primary group. Here, another user, `jane2`, has previously been added to the system with the same group as `jane`.

Listing 3. Deleting users and groups

```
root@pinguino:~# groupdel jane
groupdel: cannot remove user's primary group.
root@pinguino:~# userdel -r jane
userdel: Cannot remove group jane which is a primary group for another
user.
root@pinguino:~# userdel -r jane2
root@pinguino:~# groupdel jane
```

Notes:

1. There is a `userdel` option, `-f` or `--force`, which can be used to delete users and their group. This option is dangerous, so you should use it only as a last resort. Read the man page carefully before you do.
2. Be aware that if you delete a user or group, and there are files that belong to that user or group on your filesystem, then the files are not automatically deleted or assigned to another user or group.

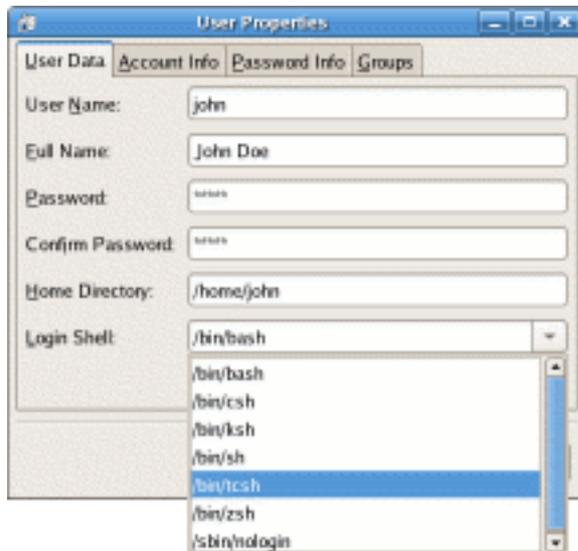
Suspend and change accounts

Now that you can create or delete a user id or a group, you may also find a need to modify one.

Modifying user accounts

Suppose user john wishes to have the tcsh shell as his default. From a graphical interface you will usually find a way to either edit a user (or group), or to examine the properties of the object. Figure 2 shows the properties dialog for the user john that we created earlier on a Fedora Core 5 system.

Figure 2. Modifying a user account



From the command line, you can use the `usermod` command to modify a user account. You can use most of the options that you use with `useradd`, except that you cannot create or populate a new home directory for the user. If you need to change the name of the user, specify the `-l` or `--login` option with the new name. You will probably want to rename the home directory to match the user id. You may also need to rename other items such as mail spool files. Finally, if the login shell is changed, some of the associated profile files may need to be altered. Listing 4 shows an example of the things you might need to do to change user john to john2 with `/bin/tcsh` as the default shell and renamed home directory `/home/john2`.

Listing 4. Modifying a user

```
[root@pinguino ~]# usermod -l john2 -s /bin/tcsh -d /home/john2 john
[root@pinguino ~]# ls -d ~john2
ls: /home/john2: No such file or directory
[root@pinguino ~]# mv /home/john /home/john2
[root@pinguino ~]# ls -d ~john2
/home/john2
```

Notes:

1. If you need to modify a user's additional groups, you must specify the complete list of additional groups. There is no command to simply add or delete a single group for a user.

2. There are restrictions on changing the name or id of a user who is logged in or who has running processes. Check the man pages for details.
3. If you change a user number, you may want to change files and directories owned by that user to match the new number.

Modifying groups

Not surprisingly, the `groupmod` command is used to modify group information. You can change the group number with the `-g` option, and the name with the `-n` option.

Listing 5. Renaming a group

```
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# groupmod -n john2 john
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

Notice in Listing 5 that the group name for the home directory of john2 magically changed when we used `groupmod` to change the group name. Are you surprised? Because groups are represented in the filesystem inodes by their number rather than by their name, this is not surprising. However, if you change a group's number, you should update any users for which that group is the primary group, and you may also want to update the files and directories belonging to that group to match the new number (in the same way as noted above for changing a user number). Listing 6 shows how to change the group number for john2 to 505, update the user account, and make appropriate changes to all the affected files in the `/home` filesystem. You probably want renumbering users and groups if at all possible.

Listing 6. Renumbering a group

```
[root@pinguino ~]# groupmod -g 505 john2
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# id john2
uid=503(john2) gid=503 groups=503
[root@pinguino ~]# usermod -g john2 john2
[root@pinguino ~]# id john2
uid=503(john2) gid=505(john2) groups=505(john2)
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# find /home -gid 503 -exec chgrp john2 {} \;
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

User and group passwords

You have already seen the `passwd` command, which is used to change a user password. The password is (or should be) unique to the user and may be changed by user. The root user may change any user's password as we have already seen.

Groups may also have passwords, and the `gpasswd` command is used to set them. Having a group password allows users to join a group temporarily with the `newgrp` command, if they know the group password. Of course, having multiple people knowing a password is somewhat problematic, so you will have to weigh the advantages of adding a user to a group using `usermod`, versus the security issue of having too many people knowing the group password.

Suspending or locking accounts

If you need to prevent a user from logging in, you can *suspend* or *lock* the account using the `-L` option of the `usermod` command. To *unlock* the account, use the `-U` option. Listing 7 shows how to lock account `john2` and what happens if `john2` attempts to log in to the system. Note that when the `john2` account is unlocked, the same password is restored.

Listing 7. Locking an account

```
[root@pinguino ~]# usermod -L john2
[root@pinguino ~]# ssh john2@pinguino
john2@pinguino's password:
Permission denied, please try again.
```

You may have noticed back in [Figure 2](#) that there were several tabs on the dialog box with additional user properties. We briefly mentioned the use of the `passwd` command for setting user passwords, but both it and the `usermod` command can perform many tasks related to user accounts, as can another command, the `chage` command. Some of these options are shown in Table 5. Refer to the appropriate man pages for more details on these and other options.

Table 5. Commands and options for changing user accounts			
	Option for command		Purpose
Usermod	Passwd	Chage	
-L	-l	N/A	Lock or suspend the account.
-U	-u	N/A	Unlock the account.
N/A	-d	N/A	Disable the account by setting it passwordless.

-e	-f	-E	Set the expiration date for an account.
N/A	-n	-m	The minimum password lifetime in days.
N/A	-x	-M	The maximum password lifetime in days.
N/A	-w	-W	The number of days of warning before a password must be changed.
-f	-i	-l	The number of days after a password expires until the account is disabled.
N/A	-S	-l	Output a short message about the current account status.

Manage user and group databases

The primary repositories for user and group information are four files in `/etc`.

`/etc/passwd`

is the *password* file containing basic information about users

`/etc/shadow`

is the *shadow password* file containing encrypted passwords

`/etc/group`

is the *group* file containing basic information about groups and which users belong to them

/etc/gshadow

is the *shadow group* file containing encrypted group passwords

These files are updated by the commands you have already seen in this tutorial and you will meet some more commands for working with them after we discuss the files themselves. All of these files are plain text files. In general, you should not edit them directly. Use the tools provided for updating them so they are properly locked and kept synchronized.

You will note that the *passwd* and *group* files are both *shadowed*. This is for security reasons. The *passwd* and *group* files themselves must be world readable, but the encrypted passwords should not be world readable. Therefore, the shadow files contain the encrypted passwords, and these files are only readable by root. The necessary authentication access is provided by an *suid* program that has root authority, but can be run by anyone. Make sure that your system has the permissions set appropriately. Listing 8 shows an example.

Listing 8. User and group database permissions

```
[ian@pinguino ~]$ ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow
-rw-r--r-- 1 root root 701 Jun 26 19:04 /etc/group
-r----- 1 root root 580 Jun 26 19:04 /etc/gshadow
-rw-r--r-- 1 root root 1939 Jun 26 19:43 /etc/passwd
-r----- 1 root root 1324 Jun 26 19:50 /etc/shadow
```

Note: Although it is still technically possible to run without shadowed password and group files, this is almost never done and is not recommended.

The /etc/passwd file

The */etc/passwd* file contains one line for each user in the system. Some example lines are shown in Listing 9.

Listing 9. /etc/password entries

```
root:x:0:0:root:/root:/bin/bash
jane:x:504:504:Jane Doe:/home/jane:/bin/bash
john2:x:503:505:John Doe:/home/john2:/bin/tcsh
```

Each line contains seven fields separated by colons (:), as shown in Table 6.

Table 6. Fields in /etc/passwd	
Field	Purpose
Username	The name used to log in to the system.

	For example, john2.
Password	The encrypted password. When using shadow passwords, it contains a single x character.
User id (UID)	The number used to represent this user name in the system. For example, 503 for user john2.
Group id (GID)	The number used to represent this user's primary group in the system. For example, 505 for user john2.
Comment (GECOS)	An optional field used to describe the user. For example, "John Doe". The field may contain multiple comma-separated entries. It is also used by programs such as <code>finger</code> . The GECOS name is historic. See details in <code>man 5 passwd</code> .
Home	The absolute path the user's home directory. For example, <code>/home/john2</code>
Shell	The program automatically launched when a user logs in to the system. This is usually an interactive shell such as <code>/bin/bash</code> or <code>/bin/tcsh</code> , but may be any program, not necessarily an interactive shell.

The `/etc/group` file

The `/etc/group` file contains one line for each group in the system. Some example lines are shown in Listing 10.

Listing 10. `/etc/group` entries

```
root:x:0:root
jane:x:504:john2
john2:x:505:
```

Each line contains four fields separated by colons (:), as shown in Table 7.

Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The encrypted password. When using shadow group passwords, it contains a single x character.

Group id (GID)	The number used to represent this group in the system. For example, 505 for group john2.
Members	A comma-separated list of group members, excepting those members for whom this is the primary group.

Shadow files

The file `/etc/shadow` should only be readable by root. It contains encrypted passwords, along with password and account expiration information. See the man page (`man 5 shadow`) for information on the field layout. Passwords may be encrypted using DES, but are more usually encrypted using MD5. The DES algorithm uses the low order 7 bits of the first 8 characters of the user password as a 56-bit key, while the MD5 algorithm uses the whole password. In either case, passwords are *salted* so that two otherwise identical passwords do not generate the same encrypted value. Listing 11 shows how to set identical passwords for users jane and john2, and then shows the resulting encoded MD5 passwords in `/etc/shadow`.

Listing 11. Passwords in `/etc/shadow`

```
[root@pinguino ~]# echo lpic1111 |passwd jane --stdin
Changing password for user jane.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# echo lpic1111 |passwd john2 --stdin
Changing password for user john2.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# grep "^j" /etc/shadow
jane:$1$eG0/KGQY$ZJ1.ltYtVw0sv.C5OrqUu/:13691:0:99999:7:::
john2:$1$grkxo6ie$J2muvoTpwo3dZAYYTDYNu.:13691:0:180:7:29::
```

The leading `1` indicates an MD5 password, and the salt is a variable length field of up to 8 characters ending with the next `$` sign. The encrypted password is the remaining string of 22 characters.

Tools for users and groups

You have already seen several commands that manipulate the account and group files and their shadows. Here you learn about:

- Group administrators
- Editing commands for password and group files
- Conversion programs

Group administrators

In some circumstances you may want users other than root to be able to administer one or more groups by adding or removing group members. Listing 12 shows how root can add user jane as an administrator of group john2, and then jane, in turn, can add user ian as a member.

Listing 12. Adding group administrators and members

```
[root@pinguino ~]# gpasswd -A jane john2
[root@pinguino ~]# su - jane
[jane@pinguino ~]$ gpasswd -a ian john2
Adding user ian to group john2
[jane@pinguino ~]$ id ian;id jane
uid=500(ian) gid=500(ian) groups=500(ian),505(john2)
uid=504(jane) gid=504(jane) groups=504(jane)
```

You may be surprised to note that, although jane is an administrator of group john2, she is not a member of it. An examination of the structure of `/etc/gshadow` shows why. The `/etc/gshadow` file contains four fields for each entry as shown in Table 8. Note that the third field is a comma-separated list of administrators for the group.

Table 8. Fields in <code>/etc/gshadow</code>	
Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The field used to contain the encrypted password if the group has a password. If there is no password, you may see 'x', '!' or '!!' here.
Admins	A comma-separated list of group administrators.
Members	A comma-separated list of group members.

As you can see, the administrator list and the member list are two distinct fields. The `-A` option of `gpasswd` allows the root user to add administrators to a group, while the `-M` option allows root to add members. The `-a` (note lower case) option allows an administrator to add a member, while the `-d` option allows an administrator to remove a member. Additional options allow a group password to be removed. See the man pages for details.

Editing commands for password and group files

Although not listed in the LPI objectives, you should also be aware of the `vipw` command for safely editing `/etc/passwd` and `visgr` for safely editing `/etc/group`. The

commands will lock the necessary files while you make changes using the `vi` editor. If you make changes to `/etc/passwd`, then `vipw` will prompt you to see if you also need to update `/etc/shadow`. Similarly, if you update `/etc/group` using `vigr`, you will be prompted to update `/etc/gshadow`. If you need to remove group administrators, you may need to use `vigr`, as `gpasswd` only allows addition of administrators.

Conversion programs

Four other related commands are also not listed in the LPI objectives. They are `pwconv`, `pwunconv`, `grpconv`, and `grpunconv`. They are used for converting between shadowed and non-shadowed password and group files. You may never need these, but be aware of their existence. See the man pages for details.

Limited and special-purpose accounts

By convention, system users usually have an id of less than 100, with root having id 0. Normal users start automatic numbering from the `UID_MIN` value set in `/etc/login.defs`, with this value commonly being set at 500 or 1000.

Besides regular user accounts and the root account on your system, you will usually have several special-purpose accounts, for daemons such as FTP, SSH, mail, news, and so on. Listing 13 shows some entries from `/etc/passwd` for these.

Listing 13. Limited and special-purpose accounts

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
```

Such accounts frequently control files but should not be accessed by normal login. Therefore, they usually have a login shell specified as `/sbin/nologin`, or `/bin/false` so that login attempts will fail.

Section 3. Environment tuning

This section covers material for topic 1.111.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to tune the user environment, including these tasks:

- Set and unset environment variables
- Maintain skeleton directories for new user accounts
- Set command search paths

Set and unset environment variables

When you create a new user, you usually initialize many variable according to your local needs. These are usually set in the profiles that you provide for new users, such as `.bash_profile` and `.bashrc`, or in the system-wide profiles `/etc/profile` and `/etc/bashrc`. Listing 14 shows an example of how the `PS1` system prompt is set in `/etc/profile` on an Ubuntu 7.04 system. The first `if` statement checks whether the `PS1` variable is set, indicating an interactive shell, since a non-interactive shell doesn't need a prompt. The second `if` statement checks whether the `BASH` environment variable is set. If so, it sets a complex prompt and sources (note the dot) `/etc/bash.bashrc`. If the `BASH` variable is not set, then a check is made for root (`id=0`), and the prompt is set to `#` or `$` accordingly.

Listing 14. Setting environment variables

```
if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
```

The tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) has detailed information about the commands used for setting and unsetting environment variables, as well as information about how and when the various profiles are used.

When customizing environments for users, be aware of two major points:

1. `/etc/profile` is read only at login time, and so it is not executed when each new shell is created.

2. Functions and aliases are not inherited by new shells. Therefore, you will usually set these and your environment variables in `/etc/bashrc`, or in the user's own profiles.

In addition to the system profiles, `/etc/profile` and `/etc/bashrc`, the Linux Standard Base (LSB) specifies that additional scripts may be placed in the directory `/etc/profile.d`. These scripts are sourced when an interactive login shell is created. They provide a convenient way of separating customization for different programs. Listing 15 shows an example.

Listing 15. `/etc/profile.d/vim.sh` on Fedora 7

```
[if [ -n "$BASH_VERSION" -o -n "$KSH_VERSION" -o -n "$ZSH_VERSION" ];
then
  [ -x //usr/bin/id ] || return
  [ `//usr/bin/id -u` -le 100 ] && return
  # for bash and zsh, only if no alias is already set
  alias vi >/dev/null 2>&1 || alias vi=vim
fi
```

Remember that you should usually `export` any variables that you set in a profile; otherwise, they will not be available to commands that run in a new shell.

Maintain skeleton directories for new users

You learned in the section [Add and remove users and groups](#) that you can create or populate a new home directory for the user. The source for this new directory is the subtree rooted at `/etc/skel`. Listing 16 shows the files in this subtree for a Fedora 7 system. Note that most files start with a period (dot), so you need the `-a` option to list them. The `-R` options lists subdirectories recursively, and the `-L` option follows any symbolic links.

Listing 16. `/etc/skel` on Fedora 7

```
[ian@lyrebird ~]$ ls -aRL /etc/skel
/etc/skel:
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .xemacs

/etc/skel/.xemacs:
.  ..  init.el
```

In addition to `.bash_logout`, `.bash_profile`, and `.bashrc`, which you might expect for the Bash shell, note that this example includes profile information for the emacs and xemacs editors. See the tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) if you need to review the functions of the various profile files.

Listing 17 shows `/etc/skel/.bashrc` from the above system. This file might be different on a different release or different distribution, but it gives you an idea of how the default user setup can be done.

Listing 17. `/etc/skel/.bashrc` on Fedora 7

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
```

As you can see, the global `/etc/bashrc` is sourced, then any user specific instructions can be added. Listing 18 shows the part of `/etc/bashrc` in which the `.sh` scripts from `/etc/profile.d` are sourced.

Listing 18. Sourcing `.sh` scripts from `/etc/profile.d`

```
for i in /etc/profile.d/*.sh; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i
```

Note that the variable, `i`, is unset after the loop.

Set command search paths

Your default profiles often include `PATH` variables for local functions or for products that you may have installed. You can set these in the skeleton files in `/etc/skel`, modify `/etc/profile`, `/etc/bashrc`, or create a file in `/etc/profile.d` if your system uses that. If you do modify the system files, be sure to check that your changes are intact after any system updates. Listing 19 shows how to add a new directory, `/opt/productxyz/bin`, to either the front or rear of your existing `PATH`.

Listing 19. Adding a path directory

```
PATH="$PATH${PATH:+:}/opt/productxyz/bin"
PATH="/opt/productxyz/bin${PATH:+:}$PATH"
```

Although not strictly required, the expression `${PATH:+:}` inserts a path separator (colon) only if the `PATH` variable is unset or null.

Section 4. System log files

This section covers material for topic 1.111.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to configure and manage system logs, including these tasks:

- Manage the type and level of information logged
- Rotate and archive log files automatically
- Scan log files for notable activity
- Monitor log files
- Track down problems reported in log files

Manage the type and level of information logged

The system logging facility on a Linux system provides system logging and kernel message trapping. Logging can be done on a local system or sent to a remote system, and the level of logging can be finely controlled through the `/etc/syslog.conf` configuration file. Logging is performed by the `syslogd` daemon, which normally receives input through the `/dev/log` socket, as shown in Listing 20.

Listing 20. `/dev/log` is a socket

```
ian@pinguino:~$ ls -l /dev/log
srw-rw-rw- 1 root root 0 2007-07-05 15:42 /dev/log
```

For local logging, the main file is usually `/var/log/messages`, but many other files are used in most installations, and you can customize these extensively. For example, you may want a separate log for messages from the mail system.

The `syslog.conf` configuration file

The `syslog.conf` file is the main configuration file for the `syslogd` daemon. Logging is based on a combination of facility and priority. The defined facilities are `auth` (or `security`), `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `syslog`, `user`, `uucp`, and `local0` through `local7`. The keyword `auth` should be used instead of `security`, and the keyword `mark` is for internal use.

The priorities (in ascending order) are:

1. debug
2. info
3. notice
4. warning (or warn)
5. err (or error)
6. crit
7. alert
8. emerg (or panic)

The parenthesized keywords (warn, error, and panic) are now deprecated.

Entries in `syslog.conf` specify logging rules. Each rule has a selector field and an action field, which are separated by one or more spaces or tabs. The selector field identifies the facility and the priorities that the rule applies to, and the action field identifies the logging action for the facility and priorities. The default behavior is to take the action for the specified level and for all higher levels, although it is possible to limit logging to specific levels. Each selector consists of a facility and a priority separated by a period (dot). Multiple facilities for a given action can be specified by separating them with a comma. Multiple facility/priority pairs for a given action can be specified by separating them with a semi-colon. Listing 21 shows an example of a simple `syslog.conf`.

Listing 21. Example `syslog.conf`

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.* /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages

# The authpriv file has restricted access.
authpriv.* /var/log/secure

# Log all the mail messages in one place.
mail.*
-/var/log/maillog

# Log cron stuff
cron.* /var/log/cron
```

```
# Everybody gets emergency messages
*.emerg                                     *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                             /var/log/spooler

# Save boot messages also to boot.log
local7.*
/var/log/boot.log
```

Notes:

- As with many configuration files, lines starting with # and blank lines are ignored.
- An * may be used to refer to all facilities or all priorities.
- The special priority keyword `none` indicates that no logging for this facility should be done with this action.
- The hyphen before a file name (such as `-/var/log/maillog`, in this example) indicates that the log file should not be synchronized after every write. You might lose information after a system crash, but you might gain performance by doing this.

The actions are generically referred to as "logfiles," although they do not have to be real files. Table 9 describe the possible logfiles.

Table 9. Actions in syslog.conf	
Action	Purpose
Regular File	Specify the full pathname, beginning with a slash (/). Prefix it with a hyphen (-) to omit syncing the file after each log entry. This may cause information loss if a crash occurs, but may improve performance.
Named Pipes	A fifo or named pipe can be used as a destination for log messages by putting a pipe symbol () before the filename. You must create the fifo using the <code>mkfifo</code> command before starting (or restarting) <code>syslogd</code> . Fifos are sometimes used for debugging.
Terminal and Console	A terminal such as <code>/dev/console</code> .
Remote Machine	To forward messages to another host, put an at (@) sign before the hostname. Note that messages are not forwarded from the receiving host.
List of Users	A comma-separated list of users to receive a message (if the user is

	logged in). The root user is frequently included here.
Everyone logged on	Specify an asterisk (*) to have everyone logged on notified using the wall command.

You may prefix ! to a priority to indicate that the action should not apply to this level and higher. Similarly you may prefix it with = to indicate that the rule applies only to this level or with != to indicate that the rule applies to all except this level. Listing 22 shows some examples, and the man page for syslog.conf has many more examples.

Listing 22. More syslog.conf examples

```
# Store all kernel messages in /var/log/kernel.
# Send critical and higher ones to remote host pinguino and to the
console
# Finally, Send info, notice and warning messages to
/var/log/kernel-info
#
kern.*                /var/log/kernel
kern.crit             @pinguino
kern.crit             /dev/console
kern.info;kern.!err  /var/log/kernel-info

# Store all mail messages except info priority in /var/log/mail.
mail.*;mail.!=info   /var/log/mail
```

Rotate and archive log files automatically

With the amount of logging that is possible, you need to be able to control the size of log files. This is done using the `logrotate` command, which is usually run as a cron job. Cron jobs are covered later in this tutorial in the section [Scheduling jobs](#). The general idea behind the `logrotate` command is that log files are periodically backed up and a new log is started. Several generations of log are kept, and when a log ages to the last generation, it may be archived. For example, it might be mailed to an archival user.

You use the `/etc/logrotate.conf` configuration file to specify how your log rotating and archiving should happen. You can specify different frequencies, such as daily, weekly, or monthly, for different log files, and you can control the number of generations to keep and when or whether to mail copies to an archival user. Listing 23 shows a sample `/etc/logrotate.conf` file.

Listing 23. Sample /etc/logrotate.conf

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

The `logrotate.conf` file has global options at the beginning. These are the defaults if nothing more specific is specified elsewhere. In this example, log files are rotated weekly, and four weeks worth of backups are kept. Once a log file is rotated, a new one is automatically created in place of the old one. Note that the `logrotate.conf` file may include specifications from other files. Here, all the files in `/etc/logrotate.d` are included.

This example also includes specific rules for `/var/log/wtmp` and `/var/log/btmp`, which are rotated monthly. No error message is issued if the files are missing. A new file is created, and only one backup is kept.

In this example, when a backup reaches the last generation, it is deleted because there is no specification of what else to do with it.

Note: The files `/var/log/wtmp` and `/var/log/btmp` record successful and unsuccessful login attempts, respectively. Unlike most log files, these are not clear text files. You may examine them using the `last` or `lastb` commands. See the man pages for these commands for details.

Log files may also be backed up when they reach a specific size, and commands may be scripted to run either prior to or after the backup operation. Listing 24 shows a more complex example.

Listing 24. Another logrotate configuration example

```
/var/log/messages {
```

```

rotate 5
mail logsave@pinguino
size 100k
postrotate
    /usr/bin/killall -HUP syslogd
endscript
}

```

In this example, `/var/log/messages` is rotated after it reaches 100KB in size. Five backups are kept, and when the oldest backup ages out, it is mailed to `logsave@pinguino`. The `postrotate` introduces a script that restarts the `syslogd` daemon after the rotation is complete, by sending it the HUP signal. The `endscript` statement is required to terminate the script and is also required if a `prerotate` script is present. See the `logrotate` man page for more complete information.

Scan log files for notable activity

Log files entries are usually time stamped and contain the hostname of the reporting process, along with the process name. Listing 25 shows a few lines from `/var/log/messages`, containing entries from `gconfd`, `ntpd`, `init`, and `yum`.

Listing 25. Sample log file entries

```

Jul  5 15:28:24 lyrebird gconfd (root-2832): Exiting
Jul  5 15:31:06 lyrebird ntpd[2063]: synchronized to 87.98.219.90,
stratum 2
Jul  5 15:31:06 lyrebird ntpd[2063]: kernel time sync status change 0001
Jul  5 15:31:24 lyrebird init: Trying to re-exec init
Jul  5 15:31:24 lyrebird yum: Updated: libselinux.i386 2.0.14-2.fc7
Jul  5 15:31:24 lyrebird yum: Updated: libsemanage.i386 2.0.3-4.fc7
Jul  5 15:31:25 lyrebird yum: Updated: cups-libs.i386 1.2.11-2.fc7
Jul  5 15:31:25 lyrebird yum: Updated: libXfont.i386 1.2.9-2.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager.i386 0.6.5-7.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager-glib.i386
0.6.5-7.fc7

```

You can scan log files using a pager, such as `less`, or search for specific entries (such as kernel messages from host `lyrebird`) using `grep` as shown in Listing 26.

Listing 26. Scanning log files

```

[root@lyrebird ~]# less /var/log/messages
[root@lyrebird ~]# grep "lyrebird kernel" /var/log/messages | tail -n 9
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HCI socket layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP ver 2.8
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM TTY layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM ver 1.8

```

```
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HIDP (Human Interface
Emulation) ver 1.2
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized drm 1.1.0 20060810
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized i915 1.6.0 20060119
on minor 0
```

Monitor log files

Occasionally you may need to monitor log files for events. For example, you might be trying to catch an infrequently occurring event at the time it happens. In such a case, you can use the `tail` command with the `-f` option to *follow* the log file. Listing 27 shows an example.

Listing 27. Following log file updates

```
[root@lyrebird ~]# tail -n 1 -f /var/log/messages
Jul  6 15:16:26 lyrebird syslogd 1.4.2: restart.
Jul  6 15:16:26 lyrebird kernel: klogd 1.4.2, log source = /proc/kmsg
started.
Jul  6 15:19:35 lyrebird yum: Updated: samba-common.i386 3.0.25b-2.fc7
Jul  6 15:19:35 lyrebird yum: Updated: procps.i386 3.2.7-14.fc7
Jul  6 15:19:36 lyrebird yum: Updated: samba-client.i386 3.0.25b-2.fc7
Jul  6 15:19:37 lyrebird yum: Updated: libsmbclient.i386 3.0.25b-2.fc7
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Received signal 15, shutting
down cleanly
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Exiting
Jul  6 15:19:57 lyrebird yum: Updated: bluez-gnome.i386 0.8-1.fc7
```

Track down problems reported in log files

When you find problems in log files, you will want to note the time, the hostname, and the process that generated the problem. If the message identifies the problem specifically enough for you to resolve it, you are done. If not, you might need to update `syslog.conf` to specify that more messages be logged for the appropriate facility. For example, you might need to show informational messages instead of warning messages or even debug level messages. Your application may have additional facilities that you can use.

Finally, if you need to put marks in the log file to help you know what messages were logged at what stage of your debugging activity, you can use the `logger` command from a terminal window or shell script to send a message of your choice to the syslog daemon for logging according to the rules in `syslog.conf`.

Section 5. Scheduling jobs

This section covers material for topic 1.111.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use the `cron` or `anacron` commands to run jobs at regular intervals
- Use the `at` command to run jobs at a specific time
- Manage cron and at jobs
- Configure user access to the cron and at services

In the previous section, you learned about the `logrotate` command and saw the need to run it periodically. You will see the same need to run commands regularly in the next two sections on backup and network time services. These are only some of the many administrative tasks that have to be done frequently and regularly. In this section, you learn about the tools that are used to automate periodic job scheduling and also the tools used to run a job at some specific time.

Run jobs at regular intervals

Running jobs at regular intervals is managed by the `cron` facility, which consists of the `crond` daemon and a set of tables describing what work is to be done and with what frequency. The daemon wakes up every minute and checks the crontabs to determine what needs to be done. Users manage crontabs using the `crontab` command. The `crond` daemon is usually started by the `init` process at system startup.

To keep things simple, let's suppose that you want to run the command shown in Listing 28 on a regular basis. This command doesn't actually do anything except report the day and the time, but it illustrates how to use `crontab` to set up cron jobs, and we'll know when it was run from the output. Setting up crontab entries requires a string with escaped shell metacharacters, so it is best done with simple commands and parameters, so in this example, the `echo` command will be run from within a script `/home/ian/mycrontab.sh`, which takes no parameters. This saves some careful work with escape characters.

Listing 28. A simple command example.

```
[ian@lyrebird ~]$ cat mycrontest.sh
#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
[ian@lyrebird ~]$ ./mycrontest.sh
It is now 18:37:42 on Friday
```

Creating a crontab

To create a crontab, you use the `crontab` command with the `-e` (for "edit") option. This will open the `vi` editor unless you have specified another editor in the `EDITOR` or `VISUAL` environment variable.

Each crontab entry contains six fields:

1. Minute
2. Hour
3. Day of the month
4. Month of the year
5. Day of the week
6. String to be executed by `sh`

Minutes and hours range from 0-59 and 0-12, respectively, while day or month and month of year range from 1-31 and 1-12, respectively. Day of week ranges from 0-6 with 0 being Sunday. Day of week may also be specified as `sun`, `mon`, `tue`, and so on. The sixth field is everything after the fifth field, is interpreted as a string to pass to `sh`. A percent sign (%) will be translated to a newline, so if you want a % or any other special character, precede it with a backslash (\). The line up to the first % is passed to the shell, while any line(s) after the % are passed as standard input.

The various time-related fields can specify an individual value, a range of values, such as 0-10 or `sun-wed`, or a comma-separated list of individual values and ranges. So a somewhat artificial crontab entry for our example command might be as shown in Listing 29.

Listing 29. A simple crontab example.

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
```

In this example, our command is executed at the 0th, 20th, and 40th minutes (every 20 minutes), for the hours between 10 P.M. and midnight on Fridays and Saturdays during July. See the man page for `crontab(5)` for details on additional ways to specify times.

What about the output?

You may be wondering what happens to any output from the command. Most

commands designed for use with the cron facility will log output using the syslog facility that you learned about in the previous section. However any output that is directed to stdout will be mailed to the user. Listing 30 shows the output you might receive from our example command.

Listing 30. Mailed cron output

```
From ian@lyrebird.raleigh.ibm.com Fri Jul 6 23:00:02 2007
Date: Fri, 6 Jul 2007 23:00:01 -0400
From: root@lyrebird.raleigh.ibm.com (Cron Daemon)
To: ian@lyrebird.raleigh.ibm.com
Subject: Cron <ian@lyrebird> /home/ian/mycronetest.sh
Content-Type: text/plain; charset=UTF-8
Auto-Submitted: auto-generated
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/home/ian>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=ian>
X-Cron-Env: <USER=ian>

It is now 23:00:01 on Friday
```

Where is my crontab?

The crontab that you created with the `crontab` command is stored in `/etc/spool/cron` under the name of the user who created it. So the above crontab is stored in `/etc/spool/cron/ian`. Given this, you will not be surprised to learn that the `crontab` command, like the `passwd` command you learned about earlier, is an `suid` program that runs with root authority.

`/etc/crontab`

In addition to the user crontab files in `/var/spool/cron`, `cron` also checks `/etc/crontab` and files in the `/etc/cron.d` directory. These system crontabs have one additional field between the fifth time entry (day) and the command. This additional field specifies the user for whom the command should be run, normally `root`. A `/etc/crontab` might look like the example in Listing 31.

Listing 31. `/etc/crontab`

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

In this example, the real work is done by the `run-parts` command, which runs

scripts from `/etc/cron.hourly`, `/etc/cron.daily`, and so on; `/etc/crontab` simply controls the timing of the recurring jobs. Note that the commands here all run as root. Note also that the crontab can include shell variables assignments that will be set before the commands are run.

Anacron

The cron facility works well for systems that run continuously. For systems that may be turned off much of the time, such as laptops, another facility, the *anacron* (for "anachronistic cron") can handle scheduling of the jobs usually done daily, weekly, or monthly by the cron facility. Anacron does not handle hourly jobs.

Anacron keeps timestamp files in `/var/spool/anacron` to record when jobs are run. When anacron runs, it checks to see if the required number of days has passed since the job was last run and runs it if necessary. The table of jobs for anacron is stored in `/etc/anacrontab`, which has a slightly different format than `/etc/crontab`. As with `/etc/crontab`, `/etc/anacrontab` may contain environment settings. Each job has four fields.

1. period
2. delay
3. job-identifier
4. command

The period is a number of days, but may be specified as `@monthly` to ensure that a job runs only once per month, regardless of the number of days in the month. The delay is the number of minutes to wait after the job is due to run before actually starting it. You can use this to prevent a flood of jobs when a system first starts. The job identifier can contain any non-blank character except slashes (`/`).

Both `/etc/crontab` and `/etc/anacrontab` are updated by direct editing. You do not use the `crontab` command to update these files or files in the `/etc/cron.d` directory.

Run jobs at specific times

Sometimes you may need to run a job just once, rather than regularly. For this you use the `at` command. The commands to be run are read from a file specified with the `-f` option, or from stdin if `-f` is not used. The `-m` option sends mail to the user even if there is no stdout from the command. The `-v` option will display the time at which the job will run before reading the job. The time is also displayed in the output. Listing 32 shows an example of running the `mycrontest.sh` script that you used earlier. Listing 33 shows the output that is mailed back to the user after the job runs.

Notice that it is somewhat more compact than the corresponding output from the cron job.

Listing 32. Using the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh -v 10:25
Sat Jul 7 10:25:00 2007

job 5 at Sat Jul 7 10:25:00 2007
```

Listing 33. Job output from at

```
From ian@lyrebird.raleigh.ibm.com Sat Jul 7 10:25:00 2007
Date: Sat, 7 Jul 2007 10:25:00 -0400
From: Ian Shields <ian@lyrebird.raleigh.ibm.com>
Subject: Output from your job 5
To: ian@lyrebird.raleigh.ibm.com

It is now 10:25:00 on Saturday
```

Time specifications can be quite complex. Listing 34 shows a few examples. See the man page for `at` or the file `/usr/share/doc/at/timespec` or a file such as `/usr/share/doc/at-3.1.10/timespec`, where 3.1.10 in this example is the version of the `at` package.

Listing 34. Time values with the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh 10pm tomorrow
job 14 at Sun Jul 8 22:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 tuesday
job 15 at Tue Jul 10 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 july 11
job 16 at Wed Jul 11 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 next week
job 17 at Sat Jul 14 02:00:00 2007
```

The `at` command also has a `-q` option. Increasing the queue increases the nice value for the job. There is also a `batch` command, which is similar to the `at` command except that jobs are run only when the system load is low enough. See the man pages for more details on these features.

Manage scheduled jobs

Listing scheduled jobs

You can manage your cron and `at` jobs. Use the `crontab` command with the `-l` option to list your crontab, and use the `atq` command to display the jobs you have queued using the `at` command, as shown in Listing 35.

Listing 35. Displaying scheduled jobs

```
[ian@lyrebird ~]$ crontab -l
0,20,40 22-23 * 7 fri-sat /home/ian/mycronstest.sh
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
```

If you want to review the actual command scheduled for execution by `at`, you can use the `at` command with the `-c` option and the job number. You will notice that most of the environment that was active at the time the `at` command was issued is saved with the scheduled job. Listing 36 shows part of the output for job 15.

Listing 36. Using `at -c` with a job number

```
#!/bin/sh
# atrun uid=500 gid=500
# mail ian 0
umask 2
HOSTNAME=lyrebird.raleigh.ibm.com; export HOSTNAME
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=9.67.219.151\ 3210\ 22; export SSH_CLIENT
SSH_TTY=/dev/pts/5; export SSH_TTY
USER=ian; export USER
...
HOME=/home/ian; export HOME
LOGNAME=ian; export LOGNAME
...
cd /home/ian || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
${SHELL:-/bin/sh} << `(dd if=/dev/urandom count=200 bs=1 \
  2>/dev/null|LC_ALL=C tr -d -c '[:alnum:]')`

#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
```

Note that the contents of our script file have been copied in as a here-document that will be executed by the shell specified by the `SHELL` variable or `/bin/sh` if the `SHELL` variable is not set. See the tutorial [LPI exam 101 prep, Topic 103: GNU and UNIX commands](#) if you need to review here-documents.

Deleting scheduled jobs

You can delete all scheduled cron jobs using the `crontab` command with the `-r` option as illustrated in Listing 37.

Listing 37. Displaying and deleting cron jobs

```
[ian@lyrebird ~]$ crontab -l
```

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycronetest.sh
[ian@lyrebird ~]$ crontab -r
[ian@lyrebird ~]$ crontab -l
no crontab for ian
```

To delete system cron or anacron jobs, edit `/etc/crontab`, `/etc/anacrontab`, or edit or delete files in the `/etc/cron.d` directory.

You can delete one or more jobs that were scheduled with the `at` command by using the `atrm` command with the job number. Multiple jobs should be separated by spaces. Listing 38 shows an example.

Listing 38. Displaying and removing jobs with `atq` and `atrm`

```
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
[ian@lyrebird ~]$ atrm 16 14 15
[ian@lyrebird ~]$ atq
17      Sat Jul 14 02:00:00 2007 a ian
```

Configure user access to job scheduling

If the file `/etc/cron.allow` exists, any non-root user must be listed in it in order to use `crontab` and the cron facility. If `/etc/cron.allow` does not exist, but `/etc/cron.deny` does exist, a non-root user who is listed in it cannot use `crontab` or the cron facility. If neither of these files exists, only the super user will be allowed to use this command. An empty `/etc/cron.deny` file allows all users to use the cron facility and is the default.

The corresponding `/etc/at.allow` and `/etc/at.deny` files have similar effects for the `at` facility.

Section 6. Data backup

This section covers material for topic 1.111.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Plan a backup strategy

- Dump a raw device to a file or restore a raw device from a file
- Perform partial and manual backups
- Verify the integrity of backup files
- Restore filesystems partially or fully from backups

Plan a backup strategy

Having a good backup is a necessary part of system administration, but deciding what to back up and when and how can be complex. Databases, such as customer orders or inventory, are usually critical to a business and many include specialized backup and recovery tools that are beyond the scope of this tutorial. At the other extreme, some files are temporary in nature and no backup is needed at all. In this section, we focus on system files and user data and discuss some of the considerations, methods, and tools for backup of such data.

There are three general approaches to backup:

1. A *full* backup is a complete backup, usually of a whole filesystem, directory, or group of related files. This takes the longest time to create, so it is usually used with one of the other two approaches.
2. A *differential* or *cumulative* backup is a backup of all things that have changed since the last full backup. Recovery requires the last full backup plus the latest differential backup.
3. An *incremental* backup is a backup of only those changes since the last incremental backup. Recovery requires the last full backup plus all of the incremental backups (in order) since the last full backup.

What to back up

When deciding what to back up, you should consider how volatile the data is. This will help you determine how often it should be backed up. Similarly, critical data should be backed up more often than non-critical data. Your operating system will probably be relatively easy to rebuild, particularly if you use a common image for several systems, although the files that customize each system would be more important to back up.

For programming staff, it may be sufficient to keep backups of repositories such as CVS repositories, while individual programmers' sandboxes may be less important. Depending on how important mail is to your operation, it may suffice to have infrequent mail backups, or it may be necessary to be able to recover mail to the

most recent date possible. You may want to keep backups of system cron files, but may not be so concerned about scheduled jobs for individual users.

The Filesystem Hierarchy Standard provides a classification of data that may help you with your backup choices. For details, see the tutorial [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#).

Once you have decided what to back up, you need to decide how often to do a full backup and whether to do differential or incremental backups in between those full backups. Having made those decisions, the following suggestions will help you choose appropriate tools.

Automating backups

In the previous section, you learned how to schedule jobs, and the cron facility is ideal for helping to automate the scheduling of your backups. However, backups are frequently made to removable media, particularly tape, so operator intervention is probably going to be needed. You should create and use backup scripts to ensure that the backup process is as automatic and repeatable as possible.

Dump and restore raw devices

One way to make a full backup of a filesystem is to make an image of the partition on which it resides. A *raw device*, such as `/dev/hda1` or `/dev/sda2`, can be opened and read as a sequential file. Similarly, it can be written from a backup as a sequential file. This requires no knowledge on the part of the backup tool as to the filesystem layout, but does require that the restore be done to space that is at least as large as the original. Some tools that handle raw devices are *filesystem aware*, meaning that they understand one or more of the Linux filesystems. These utilities can dump from a raw device but do not dump unused parts of the partition. They may or may not require restoration to the same or larger sized partition. The `dd` command is an example of the first type, while the `dump` command is an example of the second type that is specific to the `ext2` and `ext3` filesystems.

The `dd` command

In its simplest form, the `dd` command copies an input file to an output file, where either file may be a raw device. For backing up a raw device, such as `/dev/hda1` or `/dev/sda2`, the input file would be a raw device. Ideally, the filesystem on the device should be unmounted, or at least mounted read only, to ensure that data does not change during the backup. Listing 39 shows an example.

Listing 39. Backup a partition using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 of=backup-1
```

```
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 49.3103 s, 21.2 MB/s
```

The `if` and `of` parameters specify the input and output files respectively. In this example, the input file is a raw device, `dev/sda3`, and the output file is a file, `backup-1`, in the root user's home directory. To dump the file to tape or floppy disk, you would specify something like `of=/dev/fd0` or `of=/dev/st0`.

Note that 1,044,610,560 bytes of data was copied and the output file is indeed that large, even though only about 3% of this particular partition is actually used. Unless you are copying to a tape with hardware compression, you will probably want to compress the data. Listing 40 shows one way to accomplish this, along with the output of `ls` and `df` commands, which show you the file sizes and the usage percentage of the filesystem on `/dev/sda3`.

Listing 40. Backup with compression using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 | gzip > backup-2
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 117.723 s, 8.9 MB/s
[root@lyrebird ~]# ls -l backup-[12]
-rw-r--r-- 1 root root 1044610560 2007-07-08 15:17 backup-1
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
[root@lyrebird ~]# df -h /dev/sda3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        972M   28M  944M   3% /grubfile
```

The `gzip` compression reduced the file size to about 20% of the uncompressed size. However, unused blocks may contain arbitrary data, so even the compressed backup may be much larger than the total data on the partition.

If you divide the size by the number of records processed by `dd`, you will see that `dd` is writing 512-byte blocks of data. When copying to a raw output device such as tape, this can result in a very inefficient operation, so `dd` can read or write data in much larger blocks. Specify the `obs` option to change the output size or the `ibs` option to specify the input block size. You can also specify just `bs` to set both input and output block sizes to a common value.

If you need multiple tapes or other removable storage to store your backup, you will need to break it into smaller pieces using a utility such as `split`.

If you need to skip blocks such as disk or tape labels, you can do so with `dd`. See the man page for examples.

Besides just copying data, the `dd` command can do several conversions, such as between ASCII and EBCDIC, between big-endian and little-endian, or between variable-length data records and fixed-length data records. Obviously these

conversions are likely to be useful when copying real files rather than raw devices. Again, see the man page for details.

The dump command

The `dump` command can be used for full, differential, or incremental backups on `ext2` or `ext3` filesystems. Listing 41 shows an example.

Listing 41. Backup with compression using dump

```
[root@lyrebird ~]# dump -0 -f backup-4 -j -u /dev/sda3
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-4
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzlib)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 12285 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:47:48 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-4
DUMP: Volume 1 completed at: Sun Jul  8 16:47:57 2007
DUMP: Volume 1 took 0:00:09
DUMP: Volume 1 transfer rate: 819 kB/s
DUMP: Volume 1 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: 12260 blocks (11.97MB) on 1 volume(s)
DUMP: finished in 9 seconds, throughput 1362 kBytes/sec
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Date this dump completed: Sun Jul  8 16:47:57 2007
DUMP: Average transfer rate: 819 kB/s
DUMP: Wrote 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[2-4]
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
-rw-r--r-- 1 root root 266932272 2007-07-08 15:44 backup-3
-rw-r--r-- 1 root root  7554939 2007-07-08 16:47 backup-4
```

In this example, `-0` specifies the *dump level* which is an integer, historically from 0 to 9, where 0 specifies a full dump. The `-f` option specifies the output file, which may be a raw device. Specify `-` to direct the output to stdout. The `-j` option specifies compression, with a default level of 2, using `bzlib` compression. You can use the `-z` option to specify `zlib` compression if you prefer. The `-u` option causes the record of dump information, normally `/etc/dumpdates`, to be updated. Any parameters after the options represent a file or list of files, where the file may also be a raw device, as in this example. Notice how much smaller the backup is when the backup program is aware of the filesystem structure and can avoid the saving of unused blocks on the device.

If output is to a device such as tape, the `dump` command will prompt for another volume as each volume is filled. You can also provide multiple file names separated by commas. For example, if you wanted an unattended dump that required two tapes, you could load the tapes on `/dev/st0` and `/dev/st1`, schedule the `dump` command specifying both tapes as output, and go home to sleep.

When you specify a dump level greater than 0, an incremental dump is performed of all files that are new or have changed since the last dump at a lower level was taken. So a dump at level 1 will be a differential dump, even if a dump at level 2 or higher has been taken in the meantime. Listing 42 shows the result of updating the time stamp of an existing file on /dev/sda3 and creating a new file, then taking a dump at level 2. After that, another new file is created and a dump at level 1 is taken. The information from /etc/dumpdates is also shown. For brevity, part of the second dump output has been omitted.

Listing 42. Backup with compression using dump

```
[root@lyrebird ~]# dump -2 -f backup-5 -j -u /dev/sda3
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-5
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzlib)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 91 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:55:47 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-5
DUMP: Volume 1 completed at: Sun Jul  8 16:55:47 2007
DUMP: 90 blocks (0.09MB) on 1 volume(s)
DUMP: finished in less than a second
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date this dump completed: Sun Jul  8 16:55:47 2007
DUMP: Average transfer rate: 0 kB/s
DUMP: Wrote 90kB uncompressed, 15kB compressed, 6.000:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# echo "This data is even newer" >/grubfile/newerfile
[root@lyrebird ~]# dump -1 -f backup-6 -j -u -A backup-6-toc /dev/sda3
DUMP: Date of this level 1 dump: Sun Jul  8 17:08:18 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-6
...
DUMP: Wrote 100kB uncompressed, 16kB compressed, 6.250:1
DUMP: Archiving dump to backup-6-toc
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[4-6]
-rw-r--r-- 1 root root 7554939 2007-07-08 16:47 backup-4
-rw-r--r-- 1 root root  16198 2007-07-08 16:55 backup-5
-rw-r--r-- 1 root root  16560 2007-07-08 17:08 backup-6
[root@lyrebird ~]# cat /etc/dumpdates
/dev/sda3 0 Sun Jul  8 16:47:47 2007 -0400
/dev/sda3 2 Sun Jul  8 16:55:46 2007 -0400
/dev/sda3 1 Sun Jul  8 17:08:18 2007 -0400
```

Notice that backup-6 is, indeed, larger than backup 5. The level 1 dump illustrates the use of the `-A` option to create a table of contents that can be used to determine if a file is on an archive without actually mounting the archive. This is particularly useful with tape or other removable archive volumes. You will see these examples again when we discuss restoring data later in this section.

The `dump` command can dump files or subdirectories, but you cannot update

/etc/dumpdates and only level 0, of full dump, is supported.

Listing 43 illustrates the `dump` command dumping a directory, `/usr/include/bits`, and its contents to floppy disk. In this case, the dump will not fit on a single floppy, so a new volume is required. The prompt and response are shown in bold.

Listing 43. Backup a directory to multiple volumes using dump

```
[root@lyrebird ~]# dump -0 -f /dev/fd0 /usr/include/bits
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Dumping /dev/sdb9 (/ (dir usr/include/bits)) to /dev/fd0
DUMP: Label: /
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 2790 blocks.
DUMP: Volume 1 started with block 1 at: Mon Jul  9 16:03:30 2007
DUMP: dumping (Pass III) [directories]
DUMP: End of tape detected
DUMP: Closing /dev/fd0
DUMP: Volume 1 completed at: Mon Jul  9 16:04:49 2007
DUMP: Volume 1 1470 blocks (1.44MB)
DUMP: Volume 1 took 0:01:19
DUMP: Volume 1 transfer rate: 18 kB/s
DUMP: Change Volumes: Mount volume #2
DUMP: Is the new volume mounted and ready to go?: ("yes" or "no") y
DUMP: Volume 2 started with block 1441 at: Mon Jul  9 16:05:10 2007
DUMP: Volume 2 begins with blocks from inode 2
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing /dev/fd0
DUMP: Volume 2 completed at: Mon Jul  9 16:06:28 2007
DUMP: Volume 2 1410 blocks (1.38MB)
DUMP: Volume 2 took 0:01:18
DUMP: Volume 2 transfer rate: 18 kB/s
DUMP: 2850 blocks (2.78MB) on 2 volume(s)
DUMP: finished in 109 seconds, throughput 26 kBytes/sec
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Date this dump completed: Mon Jul  9 16:06:28 2007
DUMP: Average transfer rate: 18 kB/s
DUMP: DUMP IS DONE
```

If you back up to tape, remember that the tape will usually be rewound after each job. Devices with a name like `/dev/st0` or `/dev/st1` automatically rewind. The corresponding non-rewind equivalent devices are `/dev/nst0` and `/dev/nst1`. In any event, you can always use the `mt` command to perform magnetic tape operations such as forward spacing over files and records, back spacing, rewinding, and writing EOF marks. See the man pages for `mt` and `st` for additional information.

If you select the dump levels judiciously, you can minimize the number of archives you need to restore to any particular level. See the man pages for `dump` for a suggestion based on the Towers of Hanoi puzzle.

As with the `dd` command, there are many options that are not covered in this brief introduction. See the man pages for more details.

Partial and manual backups

So far, you have learned about tools that work well for backing up whole filesystems. Sometimes your backup needs to target selected files or subdirectories without backing up the whole filesystem. For example, you might need a weekly backup of most of your system, but daily backups of your mail files. Two other programs, `cpio` and `tar`, are more commonly used for this purpose. Both can write archives to files or to devices such as tape or floppy disk, and both can restore from such archives. Of the two, `tar` is more commonly used today, possibly because it handles complete directories better, and GNU `tar` supports both `gzip` and `bzip` compression.

Using `cpio`

The `cpio` command operates in *copy-out* mode to create an archive, *copy-in* mode to restore an archive, or *copy-pass* mode to copy a set of files from one location to another. You use the `-o` or `--create` option for copy-out mode, the `-i` or `--extract` option for copy-in mode, and the `-p` or `--pass-through` option for copy-pass mode. Input is a list of files provided on `stdin`. Output is either to `stdout` or to a device or file specified with the `-f` or `--file` option.

Listing 44 shows how to generate a list of files using the `find` command. Note the use of the `-print0` option on `find` to generate null-terminate strings for file names, and the corresponding `--null` option on `cpio` to read this format. This will correctly handle file names that have embedded blank or newline characters.

Listing 44. Back up a home directory using `cpio`

```
[root@lyrebird ~]# find ~ian -depth -print0 | cpio --null -o
>backup-cpio-1
18855 blocks
```

If you'd like to see the files listed as they are archived, add the `-v` option to `cpio`.

As with other commands that can archive to tape, the block size may be specified. For details on this and other options, see the man page.

Using `tar`

The `tar` (originally from *Tape ARchive*) creates an archive file, or *tarfile* or *tarball*, from a set of input files or directories; it also restores files from such an archive. If a directory is given as input to `tar`, all files and subdirectories are automatically included, which makes `tar` very convenient for archiving subtrees of your directory structure.

As with the other archiving commands we have discussed, output can be to a file, a

device such as tape or diskette, or stdout. The output location is specified with the `-f` option. Other common options are `-c` to create an archive, `-x` to extract an archive, `-v` for verbose output, which lists the files being processed, `-z` to use gzip compression, and `-j` to use bzip2 compression. Most `tar` options have a short form using a single hyphen and a long form using a pair of hyphens. The short forms are illustrated here. See the man pages for the long form and for additional options.

Listing 45 shows how to create a backup of the system cron jobs using `tar`.

Listing 45. Backup of system cron jobs using tar

```
[root@lyrebird ~]# tar -czvf backup-tar-1 /etc/*crontab /etc/cron.d
tar: Removing leading `/' from member names
/etc/anacrontab
/etc/crontab
/etc/cron.d/
/etc/cron.d/sa-update
/etc/cron.d/smolt
```

In the first line of output, you are told that `tar` will remove the leading slash (/) from member names. This allows files to be restored to some other location for verification before replacing system files. It is a good idea to avoid mixing absolute path names with relative path names when creating an archive, since all will be relative when restoring from the archive.

The `tar` command can append additional files to an archive using the `-r` or `--append` option. This may cause multiple copies of a file in the archive. In such a case, the *last* one will be restored during a restore operation. You can use the `--occurrence` option to select a specific file among multiples. If the archive is on a regular filesystem instead of tape, you may use the `-u` or `--update` option to update an archive. This works like appending to an archive, except that the time stamps of the files in the archive are compared with those on the filesystem, and only files that have been modified since the archived version are appended. As mentioned, this does not work for tape archives.

As with the other commands you have studied here, there are many options that are not covered in this brief introduction. See the man or info pages for more details.

Backup file integrity

Backup file integrity is extremely important. There is no point in having a backup if it is bad. A good backup strategy also involves checking your backups.

The first step to ensuring backup integrity is to ensure that you have properly captured the data you are backing up. If the filesystem is unmounted or mounted read only, this is usually straightforward as the data you are backing up cannot

change during your backup. If you must back up filesystems, directories, or files that are subject to modification while you are taking the backup, you should verify that no changes have been made during your backup. If changes were made, you will need to have a strategy for capturing them, either by repeating the backup, or perhaps by replacing or superseding the affected files in your backup. Needless to say, this will also affect your restore procedures.

Assuming you took good backups, you will periodically need to verify your backups. One way is to restore the backup to a spare volume and verify that it matches what you backed up. This is easiest to do right before you allow updates on the filesystem you are backing up. If you back up to media such as CD or DVD, you may be able to use the `diff` command as part of your backup procedure to ensure that your backup is good. Remember that even good backups can deteriorate in storage, so you should check periodically, even if you do verify at the time of backup. Keeping digests using programs such as `md5sum` or `sha1sum` is also a good check on the integrity of a backup file.

Restore filesystems from backups

A counterpart to backing up files is the ability to restore them when needed. Occasionally you will want to restore an entire filesystem, but it is far more common to need to restore only specific files or perhaps a set of directories. Almost always you will restore to some temporary space and verify that what you have restored is indeed what you want and is consistent with the current state of your system before actually making the restored files live.

A related issue is the need to verify that the items you want happen to be on a particular backup, as often happens when a user needs access to a version of a file that was modified or perhaps deleted "sometime in the last week or two." With these thoughts in mind, let's look at some of the restoration options.

Restoring a dd archive

Recall that the `dd` command was not filesystem aware, so you will need to restore a dump of a partition to find out what is on it. Listing 46 shows how to restore the partition that was dumped back in Listing 39 to a partition, `/dev/sdc7`, that was specially created on a removable USB drive just for this purpose.

Listing 46. Restoring a partition using dd

```
[root@lyrebird ~]# dd if=backup-1 of=/dev/sdc7
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 44.0084 s, 23.7 MB/s
```

Recall that we added some files to the filesystem on `/dev/sda3` after this backup was taken. If you mount the newly restore partition and compare it with the original, you will see that this is indeed the case, as shown in Listing 47. Note that the file whose timestamp was updated using `touch` is not shown here, as you would expect.

Listing 47. Comparing the restored partition with current state

```
[root@lyrebird ~]# mount /dev/sdc7 /mnt/temp-dd/
[root@lyrebird ~]# diff -rq /grubfile/ /mnt/temp-dd/
Only in /grubfile/: newerfile
Only in /grubfile/: newfile
```

Restoring a dump archive using restore

Recall that our final use of `dump` was a differential backup and that we created a table of contents. Listing 48 shows how to use `restore` to check the files in the archive created by `dump`, using the archive itself (`backup-5`) or the table of contents (`backup-6-toc`).

Listing 48. Checking the contents of archives

```
[root@lyrebird ~]# restore -t -f backup-5
Dump tape is compressed.
Dump   date: Sun Jul  8 16:55:46 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 2 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
[root@lyrebird ~]# restore -t -A backup-6-toc
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
Starting inode numbers by volume:
Volume 1: 2
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
      17     ./newerfile
```

The `restore` command can also compare the contents of an archive with the contents of the filesystem using the `-C` option. In Listing 49 we updated `newerfile` and then compared the backup with the filesystem.

Listing 49. Comparing an archive with a filesystem using restore

```
[root@lyrebird ~]# echo "something different" >/grubfile/newerfile
[root@lyrebird ~]# restore -C -f backup-6
Dump tape is compressed.
```

```
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
fileSYS = /grubfile
./newerfile: size has changed.
Some files were modified!  1 compare errors
```

The `restore` command can restore interactively or automatically. Listing 50 shows how to restore `newerfile` to root's home directory (so you could examine it before replacing the updated file if needed), then replace the updated file with the backup copy. This example illustrates interactive restoration.

Listing 50. Restoring a file using restore

```
[root@lyrebird ~]# restore -i -f backup-6
Dump tape is compressed.
restore > ?
Available commands are:
  ls [arg] - list directory
  cd arg - change directory
  pwd - print current directory
  add [arg] - add `arg' to list of files to be extracted
  delete [arg] - delete `arg' from list of files to be extracted
  extract - extract requested files
  setmodes - set modes of requested directories
  quit - immediately exit program
  what - list dump header information
  verbose - toggle verbose flag (useful with ``ls'')
  prompt - toggle the prompt display
  help or `?' - print this list
If no `arg' is supplied, the current directory is used
restore > ls new*
newerfile
newfile
restore > add newerfile
restore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume # (none if no more volumes): 1
set owner/mode for '.'? [yn] y
restore > q
[root@lyrebird ~]# mv -f newerfile /grubfile
```

Restoring a cpio archive

The `cpio` command in copy-in mode (option `-i` or `--extract`) can list the contents of an archive or restore selected files. When you list the files, specifying the `--absolute-filenames` option reduces the number of extraneous messages that `cpio` will otherwise issue as it strips any leading `/` characters from each path that has one. Partial output from listing our previous archive is shown in Listing 51.

Listing 51. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio -id --list --absolute-filenames <backup-cpio-1
```

```
/home/ian/.gstreamer-0.10/registry.i686.xml
/home/ian/.gstreamer-0.10
/home/ian/.Trash/gnome-terminal.desktop
/home/ian/.Trash
/home/ian/.bash_profile
```

Listing 52 shows how to restore all the files with "samp" in their path name or file name. The output has been piped through `uniq` to reduce the number of "Removing leading '/' ..." messages. You must specify the `-d` option to create directories; otherwise, all files are created in the current directory. Furthermore, `cpio` will not replace any newer files on the filesystem with archive copies unless you specify the `-u` or `--unconditional` option.

Listing 52. Restoring selected files using `cpio`

```
[root@lyrebird ~]# cpio -ivd "*samp*" < backup-cpio-1 2>&1 |uniq
cpio: Removing leading `/' from member names
home/ian/crontab.samp
cpio: Removing leading `/' from member names
home/ian/sample.file
cpio: Removing leading `/' from member names
18855 blocks
```

Restoring a tar archive

The `tar` command can also compare archives with the current filesystem as well as restore files from archives. Use the `-d`, `--compare`, or `--diff` option to perform comparisons. The output will show files whose contents differ as well as files whose time stamps differ. Listing 53 shows verbose output (using option `-v`), from a comparison of the file created earlier and the files in `/etc` after `/etc/crontab` has been touched to alter its time stamp. The option `directory /` instructs `tar` to perform the comparison starting from the root directory rather than the current directory.

Listing 53. Comparing archives and files using `tar`

```
[root@lyrebird ~]# touch /etc/crontab
[root@lyrebird ~]# tar --diff -vf backup-tar-1 --directory /
etc/anacrontab
etc/crontab
etc/crontab: Mod time differs
etc/cron.d/
etc/cron.d/sa-update
etc/cron.d/smolt
```

Listing 54 shows how to extract just `/etc/crontab` and `/etc/anacrontab` into the current directory.

Listing 54. Extracting archive files using `tar`

```
[root@lyrebird ~]# tar -xzvf backup-tar-1 "*tab"
```

```
etc/anacrontab
etc/crontab
```

Note that `tar`, in contrast to `cpio` creates the directory hierarchy for you automatically.

The next section of this tutorial shows you how to maintain system time.

Section 7. System time

This section covers material for topic 1.111.6 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

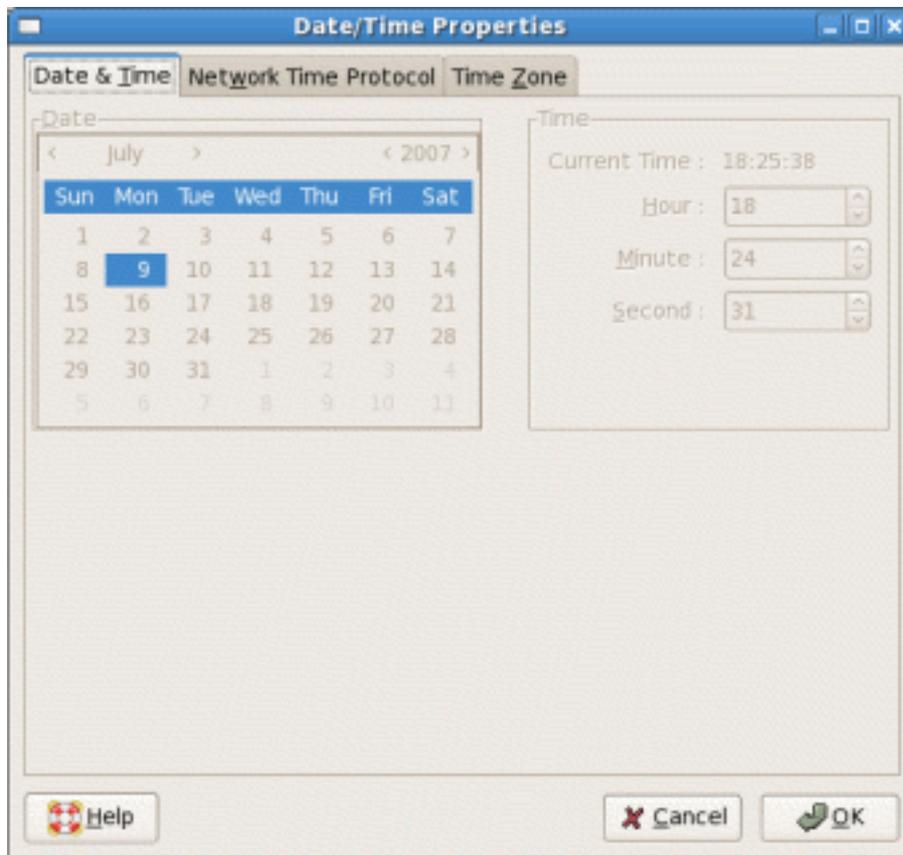
- Set the system date and time
- Set the BIOS clock to the correct UTC time
- Configure your time zone
- Configure the Network Time Protocol (NTP) service, including correcting for clock drift

Set the system date and time

System time on a Linux system is very important. You saw earlier how the cron and anacron facilities do things based on time, so they need an accurate time to base decisions on. Most of the backup and restore tools discussed in the previous section, along with development tools such as `make`, also depend on reliable time measurements. Most computers built since around 1980 include some kind of clock mechanism, and most built since 1984 or so have a persistent clock mechanism that keeps time even if the computer is turned off.

If you installed a Linux system graphically, you probably set the clock and chose a time zone suitable for your needs. You may have elected to use the Network Time Protocol (NTP) to set your clock, and you may or may not have elected to keep the system clock using Coordinated Universal Time or UTC. If you subsequently went to set the clock using graphical tools on a Fedora or Red Hat or similar system, you may have seen a dialog box like that in Figure 3.

Figure 3. Updating the date and time



Surprise! You can't actually set the clock yourself using this dialog. In this section you learn more about the difference between local clocks and NTP and how to set your system time.

No matter whether you live in New York, Budapest, Nakhodka, Ulan Bator, Bangkok, or Canberra, most of your Linux time computations are related to Coordinated Universal Time or UTC. If you run a dedicated Linux system, it is customary to keep the hardware clock set to UTC, but if you also boot another operating system such as Windows, you may need to set the hardware clock to local time. It really doesn't matter as far as Linux is concerned, except that there happen to be two different methods of keeping track of time zones internally in Linux, and if they don't agree, you can wind up with some odd time stamps on FAT filesystems, among other things. Listing 55 shows you how to use the `date` command to display the current date and time. The display is always in local time, even if your hardware clock keeps UTC time.

Listing 55. Displaying the current date and time

```
[root@lyrebird ~]# date;date -u
Mon Jul  9 22:40:01 EDT 2007
```

The `date` command supports a wide variety of possible output formats, some of which you already saw back in [Listing 28](#). See the man page for `date` if you'd like to learn more about the various date formats.

If you need to set the date, you can do this by providing a date and time as an argument. The required format is historical and is somewhat odd even to Americans and truly odd to the rest of the world. You must specify at least month, day, hour, and minute in MMDDhhmm format, and you may also append a two- or four-digit year (CCYY or YY) and optionally a period (.) followed by a two-digit number of seconds. Listing 56 shows an example that alters the system date by a little over a minute.

Listing 56. Setting the system date and time

```
[root@lyrebird ~]# date; date 0709221407;date
Mon Jul  9 23:12:37 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
```

Set the BIOS clock to UTC time

Your Linux system, along with most other current operating systems, actually has two clocks. The first is the hardware clock, sometimes called the Real Time Clock, RTC, or BIOS clock, which is usually tied to an oscillating quartz crystal that is accurate to within a few seconds per day. It is subject to variations such as ambient temperature. The second is the internal software clock, which is driven by counting system interrupts. It is subject to variations caused by high system load and interrupt latency. Nevertheless, your system typically reads the hardware clock at startup and from then on uses the software clock. The `date` command that you just learned about sets the software clock, not the hardware clock.

If you use the Network Time Protocol (NTP), you may possibly set the hardware clock when you first install the system and never worry about it again. If not, this part of the tutorial will show you how to display and set the hardware clock time.

You can use the `hwclock` command to display the current value of the hardware clock. Listing 57 shows the current value of both the system and hardware clocks.

Listing 57. System and hardware clock values

```
[root@lyrebird ~]# date;hwclock
Mon Jul  9 22:16:11 EDT 2007
Mon 09 Jul 2007 11:14:49 PM EDT -0.071616 seconds
```

Notice that the two values are different. You can synchronize the hardware clock

from the system clock using the `-w` or `--systohc` option of `hwclock`, and you can synchronize the system clock from the hardware clock using the `-s` or `--hctosys` option, as shown in Listing 58.

Listing 58. Setting the system clock from the hardware clock

```
[root@lyrebird ~]# date;hwclock;hwclock -s;date
Mon Jul  9 22:20:23 EDT 2007
Mon 09 Jul 2007 11:19:01 PM EDT  -0.414881 seconds
Mon Jul  9 23:19:02 EDT 2007
```

You may specify either the `--utc` or the `--localtime` option to have the system clock kept in UTC or local time. If no value is specified, the value is taken from the third line of `/etc/adjtime`.

The Linux kernel has a mode that copies the system time to the hardware clock every 11 minutes. This is off by default, but is turned on by NTP. Running anything that set the time the old fashioned way, such as `hwclock --hctosys`, turns it off, so it's a good idea to just let NTP do its work if you are using NTP. See the man page for `adjtimex` to find out how to check whether the clock is being updated every 11 minutes or not. You may need to install the `adjtimex` package as it is not always installed by default.

The `hwclock` command keeps track of changes made to the hardware clock in order to compensate for inaccuracies in the clock frequency. The necessary data points are kept in `/etc/adjtime`, which is an ASCII file. If you are not using the Network Time Protocol, you can use the `adjtimex` command to compensate for clock drift. Otherwise, the hardware clock will be adjusted approximately every 11 minutes by NTP. Besides showing whether your hardware clock is in local or UTC time, the first value in `/etc/adjtime` shows the amount of hardware clock drift per day (in seconds). Listing 59 shows two examples.

Listing 59. /etc/adjtime showing clock drift and local or UTC time.

```
[root@lyrebird ~]# cat /etc/adjtime
0.000990 1184019960 0.000000
1184019960
LOCAL
root@pinguino:~# cat /etc/adjtime
-0.003247 1182889954 0.000000
1182889954
LOCAL
```

Note that both these systems keep the hardware clock in local time, but the clock drifts are different — 0.000990 on lyrebird and -0.003247 on pinguino.

Configure your time zone

Your time zone is a measure of how far your local time differs from UTC. Information on available time zones that can be configured is kept in `/usr/share/zoneinfo`. Traditionally, `/etc/localtime` was a link to one of the time zone files in this directory tree, for example, `/usr/share/zoneinfo/Eire` or `/usr/share/zoneinfo/Australia/Hobart`. On modern systems it is much more likely to be a copy of the appropriate time zone data file since the `/usr/share` filesystem may not be mounted when the local time zone information is needed early in the boot process.

Similarly, another file, `/etc/timezone` was traditionally a link to `/etc/default/init` and was used to set the time zone environment variable `TZ`, and several locale-related environment variables. The file may or may not exist on your system. If it does, it may simply contain the name of the current time zone. You may also find time zone information in `/etc/sysconfig/clock`. Listing 60 shows these files from a Ubuntu 7.04 and a Fedora 7 system.

Listing 60. Time zone information in `/etc`

```
root@pinguino:~# cat /etc/timezone
America/New_York

[root@lyrebird ~]# cat /etc/sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
# The timezone of the system is defined by the contents of
/etc/localtime.
ZONE="America/New York"
UTC=false
ARC=false
```

Some systems such as Debian and Ubuntu have a `tzconfig` command to set the time zone. Others such as Fedora use `system-config-date` to set the time zone and to indicate whether the clock uses UTC or not. Listing 61 illustrates the use of the `tzconfig` command to display the current time zone.

Listing 61. Setting time zone with `tzconfig`

```
root@pinguino:~# tzconfig
Your current time zone is set to America/New_York
Do you want to change that? [n]:
Your time zone will not be changed
```

Configure the Network Time Protocol

The *Network Time Protocol (NTP)* is a protocol to synchronize computer clocks over a network. Synchronization is usually to UTC.

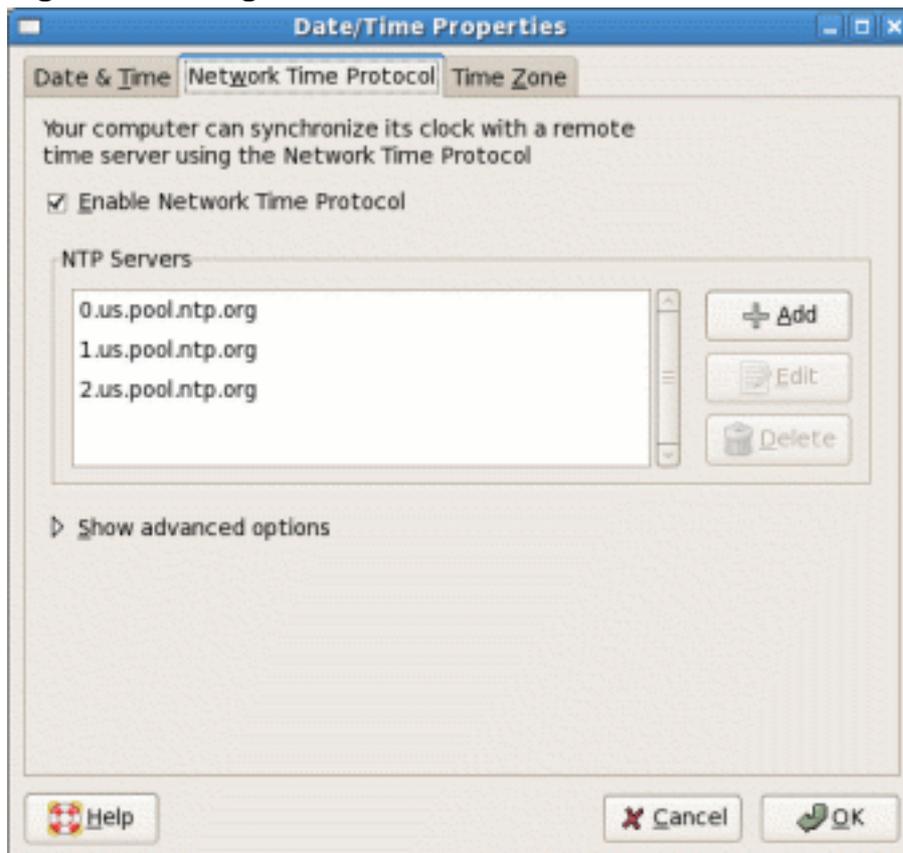
NTP version 3 is an Internet draft standard, formalized in RFC 1305. The current development version, NTP version 4 is a significant revision, which has not been formalized in an RFC. RFC 4330 describes Simple NTP (SNTP) version 4.

Time synchronization is accomplished by sending messages to *time servers*. The time returned is adjusted by an offset of half the round-trip delay. The accuracy of the time is therefore dependent on the network latency and the extent to which the latency is the same in both directions. The shorter the path to a time server, the more accurate the time is likely to be. See [Resources](#) for more detailed information than this simplistic description can provide.

There is a huge number of computers on the Internet, so time servers are organized into *strata*. A relatively small number of stratum 1 servers maintain very accurate time from a source such as an atomic clock. A larger number of stratum 2 servers get their time from stratum 1 servers and make it available to an even larger number of stratum 3 servers, and so on. To ease the load on time servers, a large number of volunteers donate time services through pool.ntp.org (see [Resources](#) for a link). Round robin DNS servers accomplish NTP load balancing by distributing NTP server requests among a pool of available servers.

If you use a graphical interface, you might be able to set your NTP time servers using a dialog similar to that in Figure 4. The fact that this system has enabled automatic time updates using NTP is why the dialog in Figure 3 did not allow the date and time to be changed.

Figure 4. Setting NTP servers



NTP configuration information is kept in `/etc/ntp.conf`, so you can also edit that file and then restart the `ntpd` daemon after you save it. Listing 62 shows an example `/etc/ntp.conf` file using the time servers from Figure 4.

Listing 62. Setting time zone with `tzconfig`

```
[root@lyrebird ~]# cat /etc/ntp.conf
# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#broadcast 192.168.1.255 key 42          # broadcast server
#broadcastclient          # broadcast client
#broadcast 224.0.1.1 key 42            # multicast server
#multicastclient 224.0.1.1            # multicast client
#manycastserver 239.255.254.254        # manycast server
#manycastclient 239.255.254.254 key 42 # manycast client

# Undisciplined Local Clock. This is a fake driver intended for backup
# and when no outside source of synchronized time is available.
#server 127.127.1.0 # local clock
#fudge 127.127.1.0 stratum 10

# Drift file. Put this in a directory which the daemon can write to.
# No symbolic links allowed, either, since the daemon updates the file
# by creating a temporary in the same directory and then rename()'ing
# it to the file.
driftfile /var/lib/ntp/drift

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8
server 0.us.pool.ntp.org
restrict 0.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 1.us.pool.ntp.org
restrict 1.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 2.us.pool.ntp.org
restrict 2.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
```

If you are using the `pool.ntp.org` time servers, these may be anywhere in the world. You will usually get better time by restricting your servers as in this example where `us.pool.ntp.org` is used, resulting in only U.S. servers being chosen. See [Resources](#)

for more information on the `ntp.pool.org` project.

NTP commands

You can use the `ntpdate` command to set your system time from an NTP time server as shown in Listing 63.

Listing 63. Setting system time from an NTP server using `ntpdate`

```
[root@lyrebird ~]# ntpdate 0.us.pool.ntp.org
10 Jul 10:27:39 ntpdate[15308]: adjust time server 66.199.242.154 offset
-0.007271 sec
```

Because the servers operate in round robin mode, the next time you run this command you will probably see a different server. Listing 64 shows the first few DNS responses for `0.us.ntp.pool.org` a few moments after the above `ntpdate` command was run.

Listing 64. Round robin NTP server pool

```
[root@lyrebird ~]# dig 0.pool.ntp.org +noall +answer | head -n 5
0.pool.ntp.org. 1062 IN A 217.116.227.3
0.pool.ntp.org. 1062 IN A 24.215.0.24
0.pool.ntp.org. 1062 IN A 62.66.254.154
0.pool.ntp.org. 1062 IN A 76.168.30.201
0.pool.ntp.org. 1062 IN A 81.169.139.140
```

The `ntpdate` command is now deprecated as the same function can be done using `ntpq` with the `-q` option, as shown in Listing 65.

Listing 65. Setting system time using `ntpd -q`

```
[root@lyrebird ~]# ntpd -q
ntpd: time slew -0.014406s
```

Note that the `ntpd` command uses the time server information from `/etc/ntp.conf`, or a configuration file provided on the command line. See the man page for more information and for information about other options for `ntpd`. Be aware also that if the `ntpd` daemon is running, `ntpd -q` will quietly exit, leaving a failure message in `/var/log/messages`.

Another related command is the `ntpq` command, which allows you to query the NTP daemon. See the man page for more details.

This brings us to the end of this tutorial. We have covered a lot of material on system administration. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- See the [Partimage homepage](#) for information on Partimage, a filesystem-aware partition dump and restore tool.
- "[/etc: Host-specific system configuration](#)" describes the Linux Standard Base (LSB) requirements for /etc.
- The [Network Time Protocol Project](#) produces a reference implementation of the NTP protocol, and implementation documentation.
- The [Network Time Synchronization Project](#) maintains an extensive array of documentation and background information, including briefing slides, on network time protocols.
- The [pool.ntp.org project](#) is a big virtual cluster of timeservers striving to provide reliable easy to use NTP service for millions of clients without putting a strain on the big popular timeservers.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content.](#)
- Get involved in the [developerWorks community](#) through our developer blogs, forums, podcasts, and community topics in our new [developerWorks spaces](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 111: Administrative tasks

Junior Level Administration (LPIC-1) topic 111

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

10 Jul 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this sixth in a [series of nine tutorials](#), Ian introduces you to administrative tasks. By the end of this tutorial, you will know how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *intermediate level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active intermediate level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the four junior and intermediate certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts.
Topic 111	LPI exam 102 prep: Administrative tasks	(This tutorial.) Learn how to manage user and group accounts and tune user and system environments, configure and use system log files, automate system administration tasks by scheduling jobs to run at another time, back up your system, and maintain system time. See the detailed objectives below.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Administrative tasks," the sixth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.111.1 User and group accounts	Weight 4	Add, remove, suspend, and change user accounts. Manage user and group information in password and group databases, including shadow databases. Create and manage special purpose and limited accounts.
1.111.2 Tune user and system environments	Weight 3	Modify global and user profiles. Set environment variables and maintain skeleton directories for new user accounts. Set command search paths.
1.111.3 Configure and use system log files to meet administrative and security needs	Weight 3	Configure and manage system logs, including the type and level of logged information. Scan and monitor log files for

		notable activity and track down noted problems. Rotate and archive log files.
1.111.4 Automate system administration tasks by scheduling jobs to run in the future	Weight 4	Use the <code>cron</code> or <code>anacron</code> commands to run jobs at regular intervals, and use the <code>at</code> command to run jobs at a specific time.
1.111.5 Maintain an effective data backup strategy	Weight 3	Plan a backup strategy and back up filesystems automatically to various media.
1.111.6 Maintain system time	Weight 4	Maintain the system time and time zone, and synchronize the clock via NTP. Set the BIOS clock to the correct time in UTC, and configure NTP, including correcting for clock drift.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be thoroughly familiar with the material from the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, which covers basic concepts of users, groups, and file permissions.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. User and group accounts

This section covers material for topic 1.111.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Add, modify, and remove users and groups
- Suspend and change user accounts
- Manage user and group information in the password databases and group databases
- Use the correct tools to manage shadow password databases and group databases
- Create and manage limited and special-purpose accounts

As you learned in the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, Linux is a multi-user system where each user belongs to one *primary* group and possibly to additional groups. Ownership of files in Linux is closely related to user ids and groups. Recall that you can log in as one user and become another user using the `su` or `sudo -s` commands, and that you can use the `whoami` command to check your current effective id and the `groups` command to find out what groups you belong to. In this section, you learn how to create, delete, and manage users and groups. You also learn about the files in `/etc`, where user and group information is stored.

Add and remove users and groups

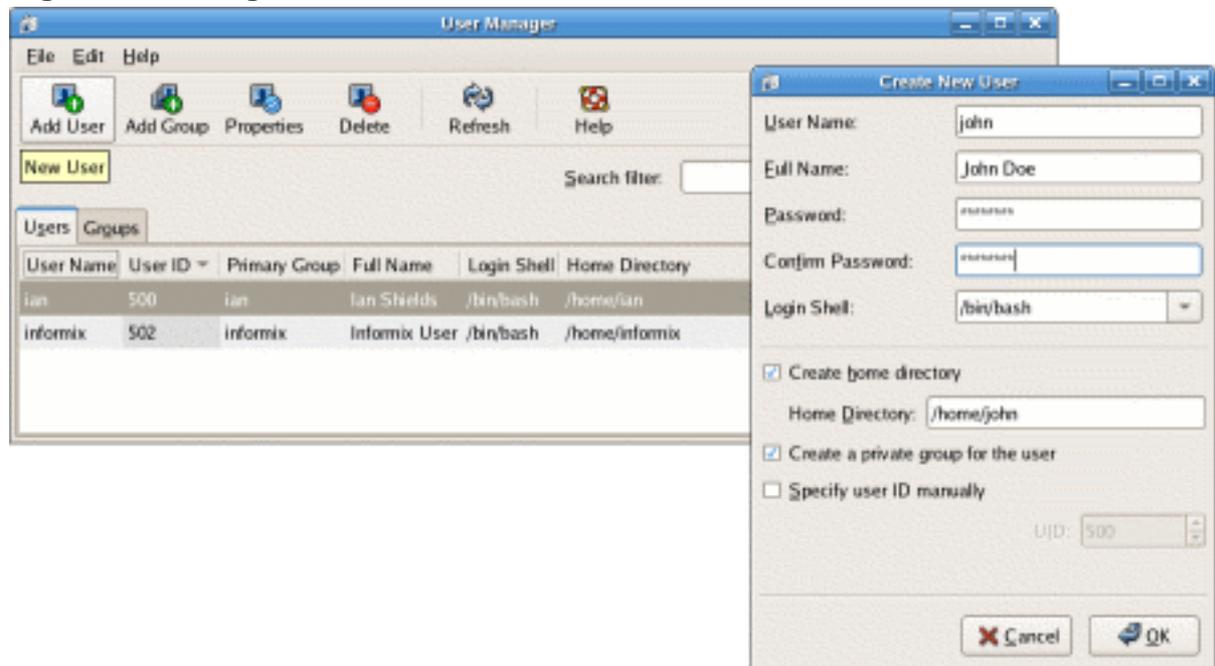
You add a user to a Linux system using the `useradd` command, and you delete a user using the `userdel` command. Similarly, you add or delete groups using the `groupadd` and `groupdel` commands.

Adding a user or group

Modern Linux desktops usually have graphical interfaces for user and group administration. The graphical interface is usually accessed through menu options for system administration. These interfaces do vary considerably, so the one on your system may not look much like the example here, but the underlying concepts and commands remain similar.

Let's start by adding a user to a Fedora Core 5 system graphically, and then examine the underlying commands. In the case of Fedora Core 5 with GNOME desktop, use **System > Administration > Users and Groups**, then click the **Add User** button.

Figure 1 depicts the User Manager panel with the Create New User panel showing basic information for a new user named 'john'. The full name of the user, John Doe, and a password have been entered. The panel provides a default login shell of `/bin/bash`. On Fedora systems, the default is to create a new group with the same name as the user, 'john' in this case, and a home directory of `/home/john`.

Figure 1. Adding a user

Listing 1 shows the use of the `id` command to display basic information about the new user. As you can see, john has user number 503 and a matching group, john, with group number 503. This is the only group of which john is a member.

Listing 1. Displaying user id information

```
[root@pinguino ~]# id john
uid=503(john) gid=503(john) groups=503(john)
```

To accomplish the same task from the command line, you use the `groupadd` and `useradd` commands to create the group and user, then use the `passwd` command to set the password for the newly created user. All of these commands require root authority. The basic use of these commands to add another user, jane, is illustrated in Listing 2.

Listing 2. Adding user jane

```
[root@pinguino ~]# groupadd jane
[root@pinguino ~]# useradd -c "Jane Doe" -g jane -m jane
[root@pinguino ~]# passwd jane
Changing password for user jane.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# id jane
uid=504(jane) gid=504(jane) groups=504(jane)
[root@pinguino ~]# ls -ld /home/jane
drwx----- 3 jane jane 4096 Jun 25 18:22 /home/jane
```

In these two examples, both the user id and the group id have values greater than 500. Be aware that some newer systems start user ids at 1000 rather than 500. These values normally signify ordinary users, while values below 500 (or 1000 if the system starts ordinary users at 1000) are reserved for *system users*. [System users](#) are covered later in this section. The actual cutoff points are set in `/etc/login.defs` as `UID_MIN` and `GID_MIN`.

In Listing 2 above, the `groupadd` command has a single parameter, `jane`, the name of the group to be added. Group names must begin with a lower case letter or an underscore, and usually contain only these along with hyphens or dashes. Options you may specify are shown in Table 3.

Option	Purpose
-f	Exit with success status if the group already exists. This is handy for scripting when you do not need to check if a group exists before attempting to create it.
-g	Specifies the group id manually. The default is to use the smallest value that is at least <code>GID_MIN</code> and also greater than the id of any existing group. Group ids are normally unique and must be non-negative
-o	Permits a group to have a non-unique id.
-K	Can be used to override defaults from <code>/etc/login.defs</code> .

In Listing 2 above, the `useradd` command has a single parameter, `jane`, the name of the user to be added, along with the `-c`, `-g`, and `-m` options. Common options for the `useradd` command are shown in Table 4.

Option	Purpose
-b --base-dir	The default base directory in which user home directories are created. This is usually <code>/home</code> , and the user's home directory is <code>/home/\$USER</code> .
-c --comment	A text string describing the id, such as the user's full name.
-d --home	Provides a specific directory name for the home directory.
-e	The date on which the account will

--expiredate	expire or be disabled in the form YYYY-MM_DD.
-g --gid	The name or number of the initial login group for the user. The group must exist, which is why group jane was created before user jane in Listing 2.
-G --groups	A comma-separated list of additional groups to which the user belongs.
-K	Can be used to override defaults from /etc/login.defs.
-m --create-home	Create the user's home directory if it does not exist. Copy the skeleton files and any directories from /etc/skel to the home directory.
-o --non-unique	Permits a user to have a non-unique id.
-p --password	The encrypted password. If a password is not specified, the default is to disable the account. You will usually use the <code>passwd</code> command in a subsequent step rather than generating an encrypted password and specifying it on the <code>useradd</code> command.
-s --shell	The name of the user's login shell if different from the default login shell.
-u --uid	The non-negative numerical userid, which must be unique if <code>-o</code> is not specified. The default is to use the smallest value that is at least <code>UID_MIN</code> and also greater than the id of any existing user.

Notes:

1. Some systems, including Fedora and Red Hat distributions, have extensions to the user-creation commands. For example, the default Fedora and Red Hat behavior is to create a new group for a user, and the `-n` option can be used on the `useradd` command to disable this function. Be aware of such possible system differences and refer to the man pages on your system when in doubt.
2. On SUSE systems, use YaST or YaST2 to access graphical user and group administration interfaces.
3. Graphical interfaces may perform additional tasks such as creating the user's mail file in `/var/spool/mail`.

Deleting a user or group

Deleting a user or group is much simpler than adding one, because there are fewer options. In fact, the `groupdel` command to delete a group requires only the group name; it has no options. You cannot delete any group that is the primary group of a user. If you use a graphical interface for deleting users and groups, the functions are very similar to the commands shown here.

Use the `userdel` command to delete a user. The `-r`, or `--remove` option requests removal of the user's home directory and anything it contains, along with the user's mail spool. When you delete a user, a group with the same name as the user will also be deleted if `USERGROUPS_ENAB` is set to `yes` in `/etc/login.defs`, but this will be done only if the group is not the primary group of another user.

In Listing 3 you see an example of deleting groups when multiple users share the same primary group. Here, another user, `jane2`, has previously been added to the system with the same group as `jane`.

Listing 3. Deleting users and groups

```
root@pinguino:~# groupdel jane
groupdel: cannot remove user's primary group.
root@pinguino:~# userdel -r jane
userdel: Cannot remove group jane which is a primary group for another
user.
root@pinguino:~# userdel -r jane2
root@pinguino:~# groupdel jane
```

Notes:

1. There is a `userdel` option, `-f` or `--force`, which can be used to delete users and their group. This option is dangerous, so you should use it only as a last resort. Read the man page carefully before you do.
2. Be aware that if you delete a user or group, and there are files that belong to that user or group on your filesystem, then the files are not automatically deleted or assigned to another user or group.

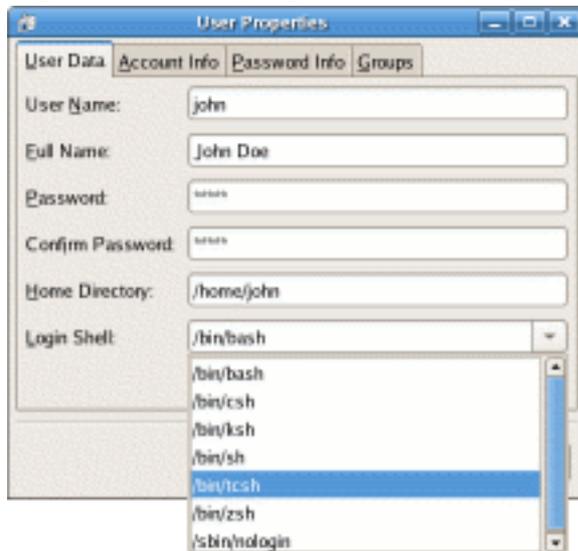
Suspend and change accounts

Now that you can create or delete a user id or a group, you may also find a need to modify one.

Modifying user accounts

Suppose user john wishes to have the tcsh shell as his default. From a graphical interface you will usually find a way to either edit a user (or group), or to examine the properties of the object. Figure 2 shows the properties dialog for the user john that we created earlier on a Fedora Core 5 system.

Figure 2. Modifying a user account



From the command line, you can use the `usermod` command to modify a user account. You can use most of the options that you use with `useradd`, except that you cannot create or populate a new home directory for the user. If you need to change the name of the user, specify the `-l` or `--login` option with the new name. You will probably want to rename the home directory to match the user id. You may also need to rename other items such as mail spool files. Finally, if the login shell is changed, some of the associated profile files may need to be altered. Listing 4 shows an example of the things you might need to do to change user john to john2 with `/bin/tcsh` as the default shell and renamed home directory `/home/john2`.

Listing 4. Modifying a user

```
[root@pinguino ~]# usermod -l john2 -s /bin/tcsh -d /home/john2 john
[root@pinguino ~]# ls -d ~john2
ls: /home/john2: No such file or directory
[root@pinguino ~]# mv /home/john /home/john2
[root@pinguino ~]# ls -d ~john2
/home/john2
```

Notes:

1. If you need to modify a user's additional groups, you must specify the complete list of additional groups. There is no command to simply add or delete a single group for a user.

2. There are restrictions on changing the name or id of a user who is logged in or who has running processes. Check the man pages for details.
3. If you change a user number, you may want to change files and directories owned by that user to match the new number.

Modifying groups

Not surprisingly, the `groupmod` command is used to modify group information. You can change the group number with the `-g` option, and the name with the `-n` option.

Listing 5. Renaming a group

```
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# groupmod -n john2 john
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

Notice in Listing 5 that the group name for the home directory of `john2` magically changed when we used `groupmod` to change the group name. Are you surprised? Because groups are represented in the filesystem inodes by their number rather than by their name, this is not surprising. However, if you change a group's number, you should update any users for which that group is the primary group, and you may also want to update the files and directories belonging to that group to match the new number (in the same way as noted above for changing a user number). Listing 6 shows how to change the group number for `john2` to 505, update the user account, and make appropriate changes to all the affected files in the `/home` filesystem. You probably want renumbering users and groups if at all possible.

Listing 6. Renumbering a group

```
[root@pinguino ~]# groupmod -g 505 john2
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# id john2
uid=503(john2) gid=503 groups=503
[root@pinguino ~]# usermod -g john2 john2
[root@pinguino ~]# id john2
uid=503(john2) gid=505(john2) groups=505(john2)
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# find /home -gid 503 -exec chgrp john2 {} \;
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

User and group passwords

You have already seen the `passwd` command, which is used to change a user password. The password is (or should be) unique to the user and may be changed by user. The root user may change any user's password as we have already seen.

Groups may also have passwords, and the `gpasswd` command is used to set them. Having a group password allows users to join a group temporarily with the `newgrp` command, if they know the group password. Of course, having multiple people knowing a password is somewhat problematic, so you will have to weigh the advantages of adding a user to a group using `usermod`, versus the security issue of having too many people knowing the group password.

Suspending or locking accounts

If you need to prevent a user from logging in, you can *suspend* or *lock* the account using the `-L` option of the `usermod` command. To *unlock* the account, use the `-U` option. Listing 7 shows how to lock account `john2` and what happens if `john2` attempts to log in to the system. Note that when the `john2` account is unlocked, the same password is restored.

Listing 7. Locking an account

```
[root@pinguino ~]# usermod -L john2
[root@pinguino ~]# ssh john2@pinguino
john2@pinguino's password:
Permission denied, please try again.
```

You may have noticed back in [Figure 2](#) that there were several tabs on the dialog box with additional user properties. We briefly mentioned the use of the `passwd` command for setting user passwords, but both it and the `usermod` command can perform many tasks related to user accounts, as can another command, the `chage` command. Some of these options are shown in Table 5. Refer to the appropriate man pages for more details on these and other options.

Table 5. Commands and options for changing user accounts			
	Option for command		Purpose
Usermod	Passwd	Chage	
-L	-l	N/A	Lock or suspend the account.
-U	-u	N/A	Unlock the account.
N/A	-d	N/A	Disable the account by setting it passwordless.

-e	-f	-E	Set the expiration date for an account.
N/A	-n	-m	The minimum password lifetime in days.
N/A	-x	-M	The maximum password lifetime in days.
N/A	-w	-W	The number of days of warning before a password must be changed.
-f	-i	-l	The number of days after a password expires until the account is disabled.
N/A	-S	-l	Output a short message about the current account status.

Manage user and group databases

The primary repositories for user and group information are four files in `/etc`.

`/etc/passwd`

is the *password* file containing basic information about users

`/etc/shadow`

is the *shadow password* file containing encrypted passwords

`/etc/group`

is the *group* file containing basic information about groups and which users belong to them

/etc/gshadow

is the *shadow group* file containing encrypted group passwords

These files are updated by the commands you have already seen in this tutorial and you will meet some more commands for working with them after we discuss the files themselves. All of these files are plain text files. In general, you should not edit them directly. Use the tools provided for updating them so they are properly locked and kept synchronized.

You will note that the passwd and group files are both *shadowed*. This is for security reasons. The passwd and group files themselves must be world readable, but the encrypted passwords should not be world readable. Therefore, the shadow files contain the encrypted passwords, and these files are only readable by root. The necessary authentication access is provided by an suid program that has root authority, but can be run by anyone. Make sure that your system has the permissions set appropriately. Listing 8 shows an example.

Listing 8. User and group database permissions

```
[ian@pinguino ~]$ ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow
-rw-r--r-- 1 root root 701 Jun 26 19:04 /etc/group
-r----- 1 root root 580 Jun 26 19:04 /etc/gshadow
-rw-r--r-- 1 root root 1939 Jun 26 19:43 /etc/passwd
-r----- 1 root root 1324 Jun 26 19:50 /etc/shadow
```

Note: Although it is still technically possible to run without shadowed password and group files, this is almost never done and is not recommended.

The /etc/passwd file

The /etc/passwd file contains one line for each user in the system. Some example lines are shown in Listing 9.

Listing 9. /etc/password entries

```
root:x:0:0:root:/root:/bin/bash
jane:x:504:504:Jane Doe:/home/jane:/bin/bash
john2:x:503:505:John Doe:/home/john2:/bin/tcsh
```

Each line contains seven fields separated by colons (:), as shown in Table 6.

Table 6. Fields in /etc/passwd	
Field	Purpose
Username	The name used to log in to the system.

	For example, john2.
Password	The encrypted password. When using shadow passwords, it contains a single x character.
User id (UID)	The number used to represent this user name in the system. For example, 503 for user john2.
Group id (GID)	The number used to represent this user's primary group in the system. For example, 505 for user john2.
Comment (GECOS)	An optional field used to describe the user. For example, "John Doe". The field may contain multiple comma-separated entries. It is also used by programs such as <code>finger</code> . The GECOS name is historic. See details in <code>man 5 passwd</code> .
Home	The absolute path the user's home directory. For example, <code>/home/john2</code>
Shell	The program automatically launched when a user logs in to the system. This is usually an interactive shell such as <code>/bin/bash</code> or <code>/bin/tcsh</code> , but may be any program, not necessarily an interactive shell.

The `/etc/group` file

The `/etc/group` file contains one line for each group in the system. Some example lines are shown in Listing 10.

Listing 10. `/etc/group` entries

```
root:x:0:root
jane:x:504:john2
john2:x:505:
```

Each line contains four fields separated by colons (:), as shown in Table 7.

Table 7. Fields in <code>/etc/group</code>	
Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The encrypted password. When using shadow group passwords, it contains a single x character.

Group id (GID)	The number used to represent this group in the system. For example, 505 for group john2.
Members	A comma-separated list of group members, excepting those members for whom this is the primary group.

Shadow files

The file `/etc/shadow` should only be readable by root. It contains encrypted passwords, along with password and account expiration information. See the man page (`man 5 shadow`) for information on the field layout. Passwords may be encrypted using DES, but are more usually encrypted using MD5. The DES algorithm uses the low order 7 bits of the first 8 characters of the user password as a 56-bit key, while the MD5 algorithm uses the whole password. In either case, passwords are *salted* so that two otherwise identical passwords do not generate the same encrypted value. Listing 11 shows how to set identical passwords for users jane and john2, and then shows the resulting encoded MD5 passwords in `/etc/shadow`.

Listing 11. Passwords in `/etc/shadow`

```
[root@pinguino ~]# echo lpic1111 |passwd jane --stdin
Changing password for user jane.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# echo lpic1111 |passwd john2 --stdin
Changing password for user john2.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# grep "^j" /etc/shadow
jane:$1$eG0/KGQY$ZJ1.ltYtVw0sv.C5OrqUu/:13691:0:99999:7:::
john2:$1$grkxo6ie$J2muvoTpwo3dZAYYTDYNu.:13691:0:180:7:29::
```

The leading `1` indicates an MD5 password, and the salt is a variable length field of up to 8 characters ending with the next `$` sign. The encrypted password is the remaining string of 22 characters.

Tools for users and groups

You have already seen several commands that manipulate the account and group files and their shadows. Here you learn about:

- Group administrators
- Editing commands for password and group files
- Conversion programs

Group administrators

In some circumstances you may want users other than root to be able to administer one or more groups by adding or removing group members. Listing 12 shows how root can add user jane as an administrator of group john2, and then jane, in turn, can add user ian as a member.

Listing 12. Adding group administrators and members

```
[root@pinguino ~]# gpasswd -A jane john2
[root@pinguino ~]# su - jane
[jane@pinguino ~]$ gpasswd -a ian john2
Adding user ian to group john2
[jane@pinguino ~]$ id ian;id jane
uid=500(ian) gid=500(ian) groups=500(ian),505(john2)
uid=504(jane) gid=504(jane) groups=504(jane)
```

You may be surprised to note that, although jane is an administrator of group john2, she is not a member of it. An examination of the structure of `/etc/gshadow` shows why. The `/etc/gshadow` file contains four fields for each entry as shown in Table 8. Note that the third field is a comma-separated list of administrators for the group.

Table 8. Fields in <code>/etc/gshadow</code>	
Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The field used to contain the encrypted password if the group has a password. If there is no password, you may see 'x', '!' or '!!' here.
Admins	A comma-separated list of group administrators.
Members	A comma-separated list of group members.

As you can see, the administrator list and the member list are two distinct fields. The `-A` option of `gpasswd` allows the root user to add administrators to a group, while the `-M` option allows root to add members. The `-a` (note lower case) option allows an administrator to add a member, while the `-d` option allows an administrator to remove a member. Additional options allow a group password to be removed. See the man pages for details.

Editing commands for password and group files

Although not listed in the LPI objectives, you should also be aware of the `vipw` command for safely editing `/etc/passwd` and `visgr` for safely editing `/etc/group`. The

commands will lock the necessary files while you make changes using the `vi` editor. If you make changes to `/etc/passwd`, then `vipw` will prompt you to see if you also need to update `/etc/shadow`. Similarly, if you update `/etc/group` using `vigr`, you will be prompted to update `/etc/gshadow`. If you need to remove group administrators, you may need to use `vigr`, as `gpasswd` only allows addition of administrators.

Conversion programs

Four other related commands are also not listed in the LPI objectives. They are `pwconv`, `pwunconv`, `grpconv`, and `grpunconv`. They are used for converting between shadowed and non-shadowed password and group files. You may never need these, but be aware of their existence. See the man pages for details.

Limited and special-purpose accounts

By convention, system users usually have an id of less than 100, with root having id 0. Normal users start automatic numbering from the `UID_MIN` value set in `/etc/login.defs`, with this value commonly being set at 500 or 1000.

Besides regular user accounts and the root account on your system, you will usually have several special-purpose accounts, for daemons such as FTP, SSH, mail, news, and so on. Listing 13 shows some entries from `/etc/passwd` for these.

Listing 13. Limited and special-purpose accounts

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
```

Such accounts frequently control files but should not be accessed by normal login. Therefore, they usually have a login shell specified as `/sbin/nologin`, or `/bin/false` so that login attempts will fail.

Section 3. Environment tuning

This section covers material for topic 1.111.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to tune the user environment, including these tasks:

- Set and unset environment variables
- Maintain skeleton directories for new user accounts
- Set command search paths

Set and unset environment variables

When you create a new user, you usually initialize many variable according to your local needs. These are usually set in the profiles that you provide for new users, such as `.bash_profile` and `.bashrc`, or in the system-wide profiles `/etc/profile` and `/etc/bashrc`. Listing 14 shows an example of how the `PS1` system prompt is set in `/etc/profile` on an Ubuntu 7.04 system. The first `if` statement checks whether the `PS1` variable is set, indicating an interactive shell, since a non-interactive shell doesn't need a prompt. The second `if` statement checks whether the `BASH` environment variable is set. If so, it sets a complex prompt and sources (note the dot) `/etc/bash.bashrc`. If the `BASH` variable is not set, then a check is made for root (`id=0`), and the prompt is set to `#` or `$` accordingly.

Listing 14. Setting environment variables

```
if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
```

The tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) has detailed information about the commands used for setting and unsetting environment variables, as well as information about how and when the various profiles are used.

When customizing environments for users, be aware of two major points:

1. `/etc/profile` is read only at login time, and so it is not executed when each new shell is created.

2. Functions and aliases are not inherited by new shells. Therefore, you will usually set these and your environment variables in `/etc/bashrc`, or in the user's own profiles.

In addition to the system profiles, `/etc/profile` and `/etc/bashrc`, the Linux Standard Base (LSB) specifies that additional scripts may be placed in the directory `/etc/profile.d`. These scripts are sourced when an interactive login shell is created. They provide a convenient way of separating customization for different programs. Listing 15 shows an example.

Listing 15. `/etc/profile.d/vim.sh` on Fedora 7

```
[if [ -n "$BASH_VERSION" -o -n "$KSH_VERSION" -o -n "$ZSH_VERSION" ];
then
  [ -x //usr/bin/id ] || return
  [ `//usr/bin/id -u` -le 100 ] && return
  # for bash and zsh, only if no alias is already set
  alias vi >/dev/null 2>&1 || alias vi=vim
fi
```

Remember that you should usually `export` any variables that you set in a profile; otherwise, they will not be available to commands that run in a new shell.

Maintain skeleton directories for new users

You learned in the section [Add and remove users and groups](#) that you can create or populate a new home directory for the user. The source for this new directory is the subtree rooted at `/etc/skel`. Listing 16 shows the files in this subtree for a Fedora 7 system. Note that most files start with a period (dot), so you need the `-a` option to list them. The `-R` options lists subdirectories recursively, and the `-L` option follows any symbolic links.

Listing 16. `/etc/skel` on Fedora 7

```
[ian@lyrebird ~]$ ls -aRL /etc/skel
/etc/skel:
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .xemacs

/etc/skel/.xemacs:
.  ..  init.el
```

In addition to `.bash_logout`, `.bash_profile`, and `.bashrc`, which you might expect for the Bash shell, note that this example includes profile information for the emacs and xemacs editors. See the tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) if you need to review the functions of the various profile files.

Listing 17 shows `/etc/skel/.bashrc` from the above system. This file might be different on a different release or different distribution, but it gives you an idea of how the default user setup can be done.

Listing 17. `/etc/skel/.bashrc` on Fedora 7

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
```

As you can see, the global `/etc/bashrc` is sourced, then any user specific instructions can be added. Listing 18 shows the part of `/etc/bashrc` in which the `.sh` scripts from `/etc/profile.d` are sourced.

Listing 18. Sourcing `.sh` scripts from `/etc/profile.d`

```
for i in /etc/profile.d/*.sh; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i
```

Note that the variable, `i`, is unset after the loop.

Set command search paths

Your default profiles often include `PATH` variables for local functions or for products that you may have installed. You can set these in the skeleton files in `/etc/skel`, modify `/etc/profile`, `/etc/bashrc`, or create a file in `/etc/profile.d` if your system uses that. If you do modify the system files, be sure to check that your changes are intact after any system updates. Listing 19 shows how to add a new directory, `/opt/productxyz/bin`, to either the front or rear of your existing `PATH`.

Listing 19. Adding a path directory

```
PATH="$PATH${PATH:+:}/opt/productxyz/bin"
PATH="/opt/productxyz/bin${PATH:+:}$PATH"
```

Although not strictly required, the expression `${PATH:+:}` inserts a path separator (colon) only if the `PATH` variable is unset or null.

Section 4. System log files

This section covers material for topic 1.111.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to configure and manage system logs, including these tasks:

- Manage the type and level of information logged
- Rotate and archive log files automatically
- Scan log files for notable activity
- Monitor log files
- Track down problems reported in log files

Manage the type and level of information logged

The system logging facility on a Linux system provides system logging and kernel message trapping. Logging can be done on a local system or sent to a remote system, and the level of logging can be finely controlled through the `/etc/syslog.conf` configuration file. Logging is performed by the `syslogd` daemon, which normally receives input through the `/dev/log` socket, as shown in Listing 20.

Listing 20. `/dev/log` is a socket

```
ian@pinguino:~$ ls -l /dev/log
srw-rw-rw- 1 root root 0 2007-07-05 15:42 /dev/log
```

For local logging, the main file is usually `/var/log/messages`, but many other files are used in most installations, and you can customize these extensively. For example, you may want a separate log for messages from the mail system.

The `syslog.conf` configuration file

The `syslog.conf` file is the main configuration file for the `syslogd` daemon. Logging is based on a combination of facility and priority. The defined facilities are `auth` (or `security`), `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `syslog`, `user`, `uucp`, and `local0` through `local7`. The keyword `auth` should be used instead of `security`, and the keyword `mark` is for internal use.

The priorities (in ascending order) are:

1. debug
2. info
3. notice
4. warning (or warn)
5. err (or error)
6. crit
7. alert
8. emerg (or panic)

The parenthesized keywords (warn, error, and panic) are now deprecated.

Entries in `syslog.conf` specify logging rules. Each rule has a selector field and an action field, which are separated by one or more spaces or tabs. The selector field identifies the facility and the priorities that the rule applies to, and the action field identifies the logging action for the facility and priorities. The default behavior is to take the action for the specified level and for all higher levels, although it is possible to limit logging to specific levels. Each selector consists of a facility and a priority separated by a period (dot). Multiple facilities for a given action can be specified by separating them with a comma. Multiple facility/priority pairs for a given action can be specified by separating them with a semi-colon. Listing 21 shows an example of a simple `syslog.conf`.

Listing 21. Example `syslog.conf`

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                               /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages

# The authpriv file has restricted access.
authpriv.*                             /var/log/secure

# Log all the mail messages in one place.
mail.*
-/var/log/maillog

# Log cron stuff
cron.*                                  /var/log/cron
```

```
# Everybody gets emergency messages
*.emerg                                     *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                             /var/log/spooler

# Save boot messages also to boot.log
local7.*
/var/log/boot.log
```

Notes:

- As with many configuration files, lines starting with # and blank lines are ignored.
- An * may be used to refer to all facilities or all priorities.
- The special priority keyword `none` indicates that no logging for this facility should be done with this action.
- The hyphen before a file name (such as `-/var/log/maillog`, in this example) indicates that the log file should not be synchronized after every write. You might lose information after a system crash, but you might gain performance by doing this.

The actions are generically referred to as "logfiles," although they do not have to be real files. Table 9 describe the possible logfiles.

Table 9. Actions in syslog.conf	
Action	Purpose
Regular File	Specify the full pathname, beginning with a slash (/). Prefix it with a hyphen (-) to omit syncing the file after each log entry. This may cause information loss if a crash occurs, but may improve performance.
Named Pipes	A fifo or named pipe can be used as a destination for log messages by putting a pipe symbol () before the filename. You must create the fifo using the <code>mkfifo</code> command before starting (or restarting) <code>syslogd</code> . Fifos are sometimes used for debugging.
Terminal and Console	A terminal such as <code>/dev/console</code> .
Remote Machine	To forward messages to another host, put an at (@) sign before the hostname. Note that messages are not forwarded from the receiving host.
List of Users	A comma-separated list of users to receive a message (if the user is

	logged in). The root user is frequently included here.
Everyone logged on	Specify an asterisk (*) to have everyone logged on notified using the wall command.

You may prefix ! to a priority to indicate that the action should not apply to this level and higher. Similarly you may prefix it with = to indicate that the rule applies only to this level or with != to indicate that the rule applies to all except this level. Listing 22 shows some examples, and the man page for syslog.conf has many more examples.

Listing 22. More syslog.conf examples

```
# Store all kernel messages in /var/log/kernel.
# Send critical and higher ones to remote host pinguino and to the
console
# Finally, Send info, notice and warning messages to
/var/log/kernel-info
#
kern.*                /var/log/kernel
kern.crit             @pinguino
kern.crit             /dev/console
kern.info;kern.!err  /var/log/kernel-info

# Store all mail messages except info priority in /var/log/mail.
mail.*;mail.!=info   /var/log/mail
```

Rotate and archive log files automatically

With the amount of logging that is possible, you need to be able to control the size of log files. This is done using the `logrotate` command, which is usually run as a cron job. Cron jobs are covered later in this tutorial in the section [Scheduling jobs](#). The general idea behind the `logrotate` command is that log files are periodically backed up and a new log is started. Several generations of log are kept, and when a log ages to the last generation, it may be archived. For example, it might be mailed to an archival user.

You use the `/etc/logrotate.conf` configuration file to specify how your log rotating and archiving should happen. You can specify different frequencies, such as daily, weekly, or monthly, for different log files, and you can control the number of generations to keep and when or whether to mail copies to an archival user. Listing 23 shows a sample `/etc/logrotate.conf` file.

Listing 23. Sample /etc/logrotate.conf

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

The logrotate.conf file has global options at the beginning. These are the defaults if nothing more specific is specified elsewhere. In this example, log files are rotated weekly, and four weeks worth of backups are kept. Once a log file is rotated, a new one is automatically created in place of the old one. Note that the logrotate.conf file may include specifications from other files. Here, all the files in /etc/logrotate.d are included.

This example also includes specific rules for /var/log/wtmp and /var/log/btmp, which are rotated monthly. No error message is issued if the files are missing. A new file is created, and only one backup is kept.

In this example, when a backup reaches the last generation, it is deleted because there is no specification of what else to do with it.

Note: The files /var/log/wtmp and /var/log/btmp record successful and unsuccessful login attempts, respectively. Unlike most log files, these are not clear text files. You may examine them using the `last` or `lastb` commands. See the man pages for these commands for details.

Log files may also be backed up when they reach a specific size, and commands may be scripted to run either prior to or after the backup operation. Listing 24 shows a more complex example.

Listing 24. Another logrotate configuration example

```
/var/log/messages {
```

```

rotate 5
mail logsave@pinguino
size 100k
postrotate
    /usr/bin/killall -HUP syslogd
endscript
}

```

In this example, `/var/log/messages` is rotated after it reaches 100KB in size. Five backups are kept, and when the oldest backup ages out, it is mailed to `logsave@pinguino`. The `postrotate` introduces a script that restarts the `syslogd` daemon after the rotation is complete, by sending it the HUP signal. The `endscript` statement is required to terminate the script and is also required if a `prerotate` script is present. See the `logrotate` man page for more complete information.

Scan log files for notable activity

Log files entries are usually time stamped and contain the hostname of the reporting process, along with the process name. Listing 25 shows a few lines from `/var/log/messages`, containing entries from `gconfd`, `ntpd`, `init`, and `yum`.

Listing 25. Sample log file entries

```

Jul  5 15:28:24 lyrebird gconfd (root-2832): Exiting
Jul  5 15:31:06 lyrebird ntpd[2063]: synchronized to 87.98.219.90,
stratum 2
Jul  5 15:31:06 lyrebird ntpd[2063]: kernel time sync status change 0001
Jul  5 15:31:24 lyrebird init: Trying to re-exec init
Jul  5 15:31:24 lyrebird yum: Updated: libselinux.i386 2.0.14-2.fc7
Jul  5 15:31:24 lyrebird yum: Updated: libsemanage.i386 2.0.3-4.fc7
Jul  5 15:31:25 lyrebird yum: Updated: cups-libs.i386 1.2.11-2.fc7
Jul  5 15:31:25 lyrebird yum: Updated: libXfont.i386 1.2.9-2.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager.i386 0.6.5-7.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager-glib.i386
0.6.5-7.fc7

```

You can scan log files using a pager, such as `less`, or search for specific entries (such as kernel messages from host `lyrebird`) using `grep` as shown in Listing 26.

Listing 26. Scanning log files

```

[root@lyrebird ~]# less /var/log/messages
[root@lyrebird ~]# grep "lyrebird kernel" /var/log/messages | tail -n 9
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HCI socket layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP ver 2.8
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM TTY layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM ver 1.8

```

```
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HIDP (Human Interface
Emulation) ver 1.2
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized drm 1.1.0 20060810
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized i915 1.6.0 20060119
on minor 0
```

Monitor log files

Occasionally you may need to monitor log files for events. For example, you might be trying to catch an infrequently occurring event at the time it happens. In such a case, you can use the `tail` command with the `-f` option to *follow* the log file. Listing 27 shows an example.

Listing 27. Following log file updates

```
[root@lyrebird ~]# tail -n 1 -f /var/log/messages
Jul  6 15:16:26 lyrebird syslogd 1.4.2: restart.
Jul  6 15:16:26 lyrebird kernel: klogd 1.4.2, log source = /proc/kmsg
started.
Jul  6 15:19:35 lyrebird yum: Updated: samba-common.i386 3.0.25b-2.fc7
Jul  6 15:19:35 lyrebird yum: Updated: procps.i386 3.2.7-14.fc7
Jul  6 15:19:36 lyrebird yum: Updated: samba-client.i386 3.0.25b-2.fc7
Jul  6 15:19:37 lyrebird yum: Updated: libsmbclient.i386 3.0.25b-2.fc7
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Received signal 15, shutting
down cleanly
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Exiting
Jul  6 15:19:57 lyrebird yum: Updated: bluez-gnome.i386 0.8-1.fc7
```

Track down problems reported in log files

When you find problems in log files, you will want to note the time, the hostname, and the process that generated the problem. If the message identifies the problem specifically enough for you to resolve it, you are done. If not, you might need to update `syslog.conf` to specify that more messages be logged for the appropriate facility. For example, you might need to show informational messages instead of warning messages or even debug level messages. Your application may have additional facilities that you can use.

Finally, if you need to put marks in the log file to help you know what messages were logged at what stage of your debugging activity, you can use the `logger` command from a terminal window or shell script to send a message of your choice to the syslog daemon for logging according to the rules in `syslog.conf`.

Section 5. Scheduling jobs

This section covers material for topic 1.111.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use the `cron` or `anacron` commands to run jobs at regular intervals
- Use the `at` command to run jobs at a specific time
- Manage cron and at jobs
- Configure user access to the cron and at services

In the previous section, you learned about the `logrotate` command and saw the need to run it periodically. You will see the same need to run commands regularly in the next two sections on backup and network time services. These are only some of the many administrative tasks that have to be done frequently and regularly. In this section, you learn about the tools that are used to automate periodic job scheduling and also the tools used to run a job at some specific time.

Run jobs at regular intervals

Running jobs at regular intervals is managed by the `cron` facility, which consists of the `crond` daemon and a set of tables describing what work is to be done and with what frequency. The daemon wakes up every minute and checks the crontabs to determine what needs to be done. Users manage crontabs using the `crontab` command. The `crond` daemon is usually started by the `init` process at system startup.

To keep things simple, let's suppose that you want to run the command shown in Listing 28 on a regular basis. This command doesn't actually do anything except report the day and the time, but it illustrates how to use `crontab` to set up cron jobs, and we'll know when it was run from the output. Setting up crontab entries requires a string with escaped shell metacharacters, so it is best done with simple commands and parameters, so in this example, the `echo` command will be run from within a script `/home/ian/mycrontab.sh`, which takes no parameters. This saves some careful work with escape characters.

Listing 28. A simple command example.

```
[ian@lyrebird ~]$ cat mycrontest.sh
#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
[ian@lyrebird ~]$ ./mycrontest.sh
It is now 18:37:42 on Friday
```

Creating a crontab

To create a crontab, you use the `crontab` command with the `-e` (for "edit") option. This will open the `vi` editor unless you have specified another editor in the `EDITOR` or `VISUAL` environment variable.

Each crontab entry contains six fields:

1. Minute
2. Hour
3. Day of the month
4. Month of the year
5. Day of the week
6. String to be executed by `sh`

Minutes and hours range from 0-59 and 0-12, respectively, while day or month and month of year range from 1-31 and 1-12, respectively. Day of week ranges from 0-6 with 0 being Sunday. Day of week may also be specified as `sun`, `mon`, `tue`, and so on. The sixth field is everything after the fifth field, is interpreted as a string to pass to `sh`. A percent sign (%) will be translated to a newline, so if you want a % or any other special character, precede it with a backslash (\). The line up to the first % is passed to the shell, while any line(s) after the % are passed as standard input.

The various time-related fields can specify an individual value, a range of values, such as 0-10 or `sun-wed`, or a comma-separated list of individual values and ranges. So a somewhat artificial crontab entry for our example command might be as shown in Listing 29.

Listing 29. A simple crontab example.

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
```

In this example, our command is executed at the 0th, 20th, and 40th minutes (every 20 minutes), for the hours between 10 P.M. and midnight on Fridays and Saturdays during July. See the man page for `crontab(5)` for details on additional ways to specify times.

What about the output?

You may be wondering what happens to any output from the command. Most

commands designed for use with the cron facility will log output using the syslog facility that you learned about in the previous section. However any output that is directed to stdout will be mailed to the user. Listing 30 shows the output you might receive from our example command.

Listing 30. Mailed cron output

```
From ian@lyrebird.raleigh.ibm.com Fri Jul 6 23:00:02 2007
Date: Fri, 6 Jul 2007 23:00:01 -0400
From: root@lyrebird.raleigh.ibm.com (Cron Daemon)
To: ian@lyrebird.raleigh.ibm.com
Subject: Cron <ian@lyrebird> /home/ian/mycronetest.sh
Content-Type: text/plain; charset=UTF-8
Auto-Submitted: auto-generated
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/home/ian>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=ian>
X-Cron-Env: <USER=ian>

It is now 23:00:01 on Friday
```

Where is my crontab?

The crontab that you created with the `crontab` command is stored in `/etc/spool/cron` under the name of the user who created it. So the above crontab is stored in `/etc/spool/cron/ian`. Given this, you will not be surprised to learn that the `crontab` command, like the `passwd` command you learned about earlier, is an `suid` program that runs with root authority.

`/etc/crontab`

In addition to the user crontab files in `/var/spool/cron`, `cron` also checks `/etc/crontab` and files in the `/etc/cron.d` directory. These system crontabs have one additional field between the fifth time entry (day) and the command. This additional field specifies the user for whom the command should be run, normally `root`. A `/etc/crontab` might look like the example in Listing 31.

Listing 31. `/etc/crontab`

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

In this example, the real work is done by the `run-parts` command, which runs

scripts from `/etc/cron.hourly`, `/etc/cron.daily`, and so on; `/etc/crontab` simply controls the timing of the recurring jobs. Note that the commands here all run as root. Note also that the crontab can include shell variables assignments that will be set before the commands are run.

Anacron

The cron facility works well for systems that run continuously. For systems that may be turned off much of the time, such as laptops, another facility, the *anacron* (for "anachronistic cron") can handle scheduling of the jobs usually done daily, weekly, or monthly by the cron facility. Anacron does not handle hourly jobs.

Anacron keeps timestamp files in `/var/spool/anacron` to record when jobs are run. When anacron runs, it checks to see if the required number of days has passed since the job was last run and runs it if necessary. The table of jobs for anacron is stored in `/etc/anacrontab`, which has a slightly different format than `/etc/crontab`. As with `/etc/crontab`, `/etc/anacrontab` may contain environment settings. Each job has four fields.

1. period
2. delay
3. job-identifier
4. command

The period is a number of days, but may be specified as `@monthly` to ensure that a job runs only once per month, regardless of the number of days in the month. The delay is the number of minutes to wait after the job is due to run before actually starting it. You can use this to prevent a flood of jobs when a system first starts. The job identifier can contain any non-blank character except slashes (`/`).

Both `/etc/crontab` and `/etc/anacrontab` are updated by direct editing. You do not use the `crontab` command to update these files or files in the `/etc/cron.d` directory.

Run jobs at specific times

Sometimes you may need to run a job just once, rather than regularly. For this you use the `at` command. The commands to be run are read from a file specified with the `-f` option, or from stdin if `-f` is not used. The `-m` option sends mail to the user even if there is no stdout from the command. The `-v` option will display the time at which the job will run before reading the job. The time is also displayed in the output. Listing 32 shows an example of running the `mycrontest.sh` script that you used earlier. Listing 33 shows the output that is mailed back to the user after the job runs.

Notice that it is somewhat more compact than the corresponding output from the cron job.

Listing 32. Using the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh -v 10:25
Sat Jul 7 10:25:00 2007

job 5 at Sat Jul 7 10:25:00 2007
```

Listing 33. Job output from at

```
From ian@lyrebird.raleigh.ibm.com Sat Jul 7 10:25:00 2007
Date: Sat, 7 Jul 2007 10:25:00 -0400
From: Ian Shields <ian@lyrebird.raleigh.ibm.com>
Subject: Output from your job 5
To: ian@lyrebird.raleigh.ibm.com

It is now 10:25:00 on Saturday
```

Time specifications can be quite complex. Listing 34 shows a few examples. See the man page for `at` or the file `/usr/share/doc/at/timespec` or a file such as `/usr/share/doc/at-3.1.10/timespec`, where 3.1.10 in this example is the version of the `at` package.

Listing 34. Time values with the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh 10pm tomorrow
job 14 at Sun Jul 8 22:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 tuesday
job 15 at Tue Jul 10 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 july 11
job 16 at Wed Jul 11 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 next week
job 17 at Sat Jul 14 02:00:00 2007
```

The `at` command also has a `-q` option. Increasing the queue increases the nice value for the job. There is also a `batch` command, which is similar to the `at` command except that jobs are run only when the system load is low enough. See the man pages for more details on these features.

Manage scheduled jobs

Listing scheduled jobs

You can manage your cron and `at` jobs. Use the `crontab` command with the `-l` option to list your crontab, and use the `atq` command to display the jobs you have queued using the `at` command, as shown in Listing 35.

Listing 35. Displaying scheduled jobs

```
[ian@lyrebird ~]$ crontab -l
0,20,40 22-23 * 7 fri-sat /home/ian/mycronstest.sh
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
```

If you want to review the actual command scheduled for execution by `at`, you can use the `at` command with the `-c` option and the job number. You will notice that most of the environment that was active at the time the `at` command was issued is saved with the scheduled job. Listing 36 shows part of the output for job 15.

Listing 36. Using `at -c` with a job number

```
#!/bin/sh
# atrun uid=500 gid=500
# mail ian 0
umask 2
HOSTNAME=lyrebird.raleigh.ibm.com; export HOSTNAME
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=9.67.219.151\ 3210\ 22; export SSH_CLIENT
SSH_TTY=/dev/pts/5; export SSH_TTY
USER=ian; export USER
...
HOME=/home/ian; export HOME
LOGNAME=ian; export LOGNAME
...
cd /home/ian || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
${SHELL:-/bin/sh} << `(dd if=/dev/urandom count=200 bs=1 \
 2>/dev/null|LC_ALL=C tr -d -c '[:alnum:]')`

#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
```

Note that the contents of our script file have been copied in as a here-document that will be executed by the shell specified by the `SHELL` variable or `/bin/sh` if the `SHELL` variable is not set. See the tutorial [LPI exam 101 prep, Topic 103: GNU and UNIX commands](#) if you need to review here-documents.

Deleting scheduled jobs

You can delete all scheduled cron jobs using the `crontab` command with the `-r` option as illustrated in Listing 37.

Listing 37. Displaying and deleting cron jobs

```
[ian@lyrebird ~]$ crontab -l
```

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycronetest.sh
[ian@lyrebird ~]$ crontab -r
[ian@lyrebird ~]$ crontab -l
no crontab for ian
```

To delete system cron or anacron jobs, edit `/etc/crontab`, `/etc/anacrontab`, or edit or delete files in the `/etc/cron.d` directory.

You can delete one or more jobs that were scheduled with the `at` command by using the `atrm` command with the job number. Multiple jobs should be separated by spaces. Listing 38 shows an example.

Listing 38. Displaying and removing jobs with `atq` and `atrm`

```
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
[ian@lyrebird ~]$ atrm 16 14 15
[ian@lyrebird ~]$ atq
17      Sat Jul 14 02:00:00 2007 a ian
```

Configure user access to job scheduling

If the file `/etc/cron.allow` exists, any non-root user must be listed in it in order to use `crontab` and the cron facility. If `/etc/cron.allow` does not exist, but `/etc/cron.deny` does exist, a non-root user who is listed in it cannot use `crontab` or the cron facility. If neither of these files exists, only the super user will be allowed to use this command. An empty `/etc/cron.deny` file allows all users to use the cron facility and is the default.

The corresponding `/etc/at.allow` and `/etc/at.deny` files have similar effects for the `at` facility.

Section 6. Data backup

This section covers material for topic 1.111.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Plan a backup strategy

- Dump a raw device to a file or restore a raw device from a file
- Perform partial and manual backups
- Verify the integrity of backup files
- Restore filesystems partially or fully from backups

Plan a backup strategy

Having a good backup is a necessary part of system administration, but deciding what to back up and when and how can be complex. Databases, such as customer orders or inventory, are usually critical to a business and many include specialized backup and recovery tools that are beyond the scope of this tutorial. At the other extreme, some files are temporary in nature and no backup is needed at all. In this section, we focus on system files and user data and discuss some of the considerations, methods, and tools for backup of such data.

There are three general approaches to backup:

1. A *full* backup is a complete backup, usually of a whole filesystem, directory, or group of related files. This takes the longest time to create, so it is usually used with one of the other two approaches.
2. A *differential* or *cumulative* backup is a backup of all things that have changed since the last full backup. Recovery requires the last full backup plus the latest differential backup.
3. An *incremental* backup is a backup of only those changes since the last incremental backup. Recovery requires the last full backup plus all of the incremental backups (in order) since the last full backup.

What to back up

When deciding what to back up, you should consider how volatile the data is. This will help you determine how often it should be backed up. Similarly, critical data should be backed up more often than non-critical data. Your operating system will probably be relatively easy to rebuild, particularly if you use a common image for several systems, although the files that customize each system would be more important to back up.

For programming staff, it may be sufficient to keep backups of repositories such as CVS repositories, while individual programmers' sandboxes may be less important. Depending on how important mail is to your operation, it may suffice to have infrequent mail backups, or it may be necessary to be able to recover mail to the

most recent date possible. You may want to keep backups of system cron files, but may not be so concerned about scheduled jobs for individual users.

The Filesystem Hierarchy Standard provides a classification of data that may help you with your backup choices. For details, see the tutorial [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#).

Once you have decided what to back up, you need to decide how often to do a full backup and whether to do differential or incremental backups in between those full backups. Having made those decisions, the following suggestions will help you choose appropriate tools.

Automating backups

In the previous section, you learned how to schedule jobs, and the cron facility is ideal for helping to automate the scheduling of your backups. However, backups are frequently made to removable media, particularly tape, so operator intervention is probably going to be needed. You should create and use backup scripts to ensure that the backup process is as automatic and repeatable as possible.

Dump and restore raw devices

One way to make a full backup of a filesystem is to make an image of the partition on which it resides. A *raw device*, such as `/dev/hda1` or `/dev/sda2`, can be opened and read as a sequential file. Similarly, it can be written from a backup as a sequential file. This requires no knowledge on the part of the backup tool as to the filesystem layout, but does require that the restore be done to space that is at least as large as the original. Some tools that handle raw devices are *filesystem aware*, meaning that they understand one or more of the Linux filesystems. These utilities can dump from a raw device but do not dump unused parts of the partition. They may or may not require restoration to the same or larger sized partition. The `dd` command is an example of the first type, while the `dump` command is an example of the second type that is specific to the `ext2` and `ext3` filesystems.

The `dd` command

In its simplest form, the `dd` command copies an input file to an output file, where either file may be a raw device. For backing up a raw device, such as `/dev/hda1` or `/dev/sda2`, the input file would be a raw device. Ideally, the filesystem on the device should be unmounted, or at least mounted read only, to ensure that data does not change during the backup. Listing 39 shows an example.

Listing 39. Backup a partition using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 of=backup-1
```

```
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 49.3103 s, 21.2 MB/s
```

The `if` and `of` parameters specify the input and output files respectively. In this example, the input file is a raw device, `dev/sda3`, and the output file is a file, `backup-1`, in the root user's home directory. To dump the file to tape or floppy disk, you would specify something like `of=/dev/fd0` or `of=/dev/st0`.

Note that 1,044,610,560 bytes of data was copied and the output file is indeed that large, even though only about 3% of this particular partition is actually used. Unless you are copying to a tape with hardware compression, you will probably want to compress the data. Listing 40 shows one way to accomplish this, along with the output of `ls` and `df` commands, which show you the file sizes and the usage percentage of the filesystem on `/dev/sda3`.

Listing 40. Backup with compression using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 | gzip > backup-2
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 117.723 s, 8.9 MB/s
[root@lyrebird ~]# ls -l backup-[12]
-rw-r--r-- 1 root root 1044610560 2007-07-08 15:17 backup-1
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
[root@lyrebird ~]# df -h /dev/sda3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       972M   28M  944M   3% /grubfile
```

The `gzip` compression reduced the file size to about 20% of the uncompressed size. However, unused blocks may contain arbitrary data, so even the compressed backup may be much larger than the total data on the partition.

If you divide the size by the number of records processed by `dd`, you will see that `dd` is writing 512-byte blocks of data. When copying to a raw output device such as tape, this can result in a very inefficient operation, so `dd` can read or write data in much larger blocks. Specify the `obs` option to change the output size or the `ibs` option to specify the input block size. You can also specify just `bs` to set both input and output block sizes to a common value.

If you need multiple tapes or other removable storage to store your backup, you will need to break it into smaller pieces using a utility such as `split`.

If you need to skip blocks such as disk or tape labels, you can do so with `dd`. See the man page for examples.

Besides just copying data, the `dd` command can do several conversions, such as between ASCII and EBCDIC, between big-endian and little-endian, or between variable-length data records and fixed-length data records. Obviously these

conversions are likely to be useful when copying real files rather than raw devices. Again, see the man page for details.

The dump command

The `dump` command can be used for full, differential, or incremental backups on ext2 or ext3 filesystems. Listing 41 shows an example.

Listing 41. Backup with compression using dump

```
[root@lyrebird ~]# dump -0 -f backup-4 -j -u /dev/sda3
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-4
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzip)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 12285 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:47:48 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-4
DUMP: Volume 1 completed at: Sun Jul  8 16:47:57 2007
DUMP: Volume 1 took 0:00:09
DUMP: Volume 1 transfer rate: 819 kB/s
DUMP: Volume 1 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: 12260 blocks (11.97MB) on 1 volume(s)
DUMP: finished in 9 seconds, throughput 1362 kBytes/sec
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Date this dump completed: Sun Jul  8 16:47:57 2007
DUMP: Average transfer rate: 819 kB/s
DUMP: Wrote 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[2-4]
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
-rw-r--r-- 1 root root 266932272 2007-07-08 15:44 backup-3
-rw-r--r-- 1 root root  7554939 2007-07-08 16:47 backup-4
```

In this example, `-0` specifies the *dump level* which is an integer, historically from 0 to 9, where 0 specifies a full dump. The `-f` option specifies the output file, which may be a raw device. Specify `-` to direct the output to stdout. The `-j` option specifies compression, with a default level of 2, using bzip compression. You can use the `-z` option to specify zlib compression if you prefer. The `-u` option causes the record of dump information, normally `/etc/dumpdates`, to be updated. Any parameters after the options represent a file or list of files, where the file may also be a raw device, as in this example. Notice how much smaller the backup is when the backup program is aware of the filesystem structure and can avoid the saving of unused blocks on the device.

If output is to a device such as tape, the `dump` command will prompt for another volume as each volume is filled. You can also provide multiple file names separated by commas. For example, if you wanted an unattended dump that required two tapes, you could load the tapes on `/dev/st0` and `/dev/st1`, schedule the `dump` command specifying both tapes as output, and go home to sleep.

When you specify a dump level greater than 0, an incremental dump is performed of all files that are new or have changed since the last dump at a lower level was taken. So a dump at level 1 will be a differential dump, even if a dump at level 2 or higher has been taken in the meantime. Listing 42 shows the result of updating the time stamp of an existing file on /dev/sda3 and creating a new file, then taking a dump at level 2. After that, another new file is created and a dump at level 1 is taken. The information from /etc/dumpdates is also shown. For brevity, part of the second dump output has been omitted.

Listing 42. Backup with compression using dump

```
[root@lyrebird ~]# dump -2 -f backup-5 -j -u /dev/sda3
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-5
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzlib)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 91 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:55:47 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-5
DUMP: Volume 1 completed at: Sun Jul  8 16:55:47 2007
DUMP: 90 blocks (0.09MB) on 1 volume(s)
DUMP: finished in less than a second
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date this dump completed: Sun Jul  8 16:55:47 2007
DUMP: Average transfer rate: 0 kB/s
DUMP: Wrote 90kB uncompressed, 15kB compressed, 6.000:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# echo "This data is even newer" >/grubfile/newerfile
[root@lyrebird ~]# dump -1 -f backup-6 -j -u -A backup-6-toc /dev/sda3
DUMP: Date of this level 1 dump: Sun Jul  8 17:08:18 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-6
...
DUMP: Wrote 100kB uncompressed, 16kB compressed, 6.250:1
DUMP: Archiving dump to backup-6-toc
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[4-6]
-rw-r--r-- 1 root root 7554939 2007-07-08 16:47 backup-4
-rw-r--r-- 1 root root  16198 2007-07-08 16:55 backup-5
-rw-r--r-- 1 root root  16560 2007-07-08 17:08 backup-6
[root@lyrebird ~]# cat /etc/dumpdates
/dev/sda3 0 Sun Jul  8 16:47:47 2007 -0400
/dev/sda3 2 Sun Jul  8 16:55:46 2007 -0400
/dev/sda3 1 Sun Jul  8 17:08:18 2007 -0400
```

Notice that backup-6 is, indeed, larger than backup 5. The level 1 dump illustrates the use of the `-A` option to create a table of contents that can be used to determine if a file is on an archive without actually mounting the archive. This is particularly useful with tape or other removable archive volumes. You will see these examples again when we discuss restoring data later in this section.

The `dump` command can dump files or subdirectories, but you cannot update

/etc/dumpdates and only level 0, of full dump, is supported.

Listing 43 illustrates the `dump` command dumping a directory, `/usr/include/bits`, and its contents to floppy disk. In this case, the dump will not fit on a single floppy, so a new volume is required. The prompt and response are shown in bold.

Listing 43. Backup a directory to multiple volumes using dump

```
[root@lyrebird ~]# dump -0 -f /dev/fd0 /usr/include/bits
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Dumping /dev/sdb9 (/ (dir usr/include/bits)) to /dev/fd0
DUMP: Label: /
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 2790 blocks.
DUMP: Volume 1 started with block 1 at: Mon Jul  9 16:03:30 2007
DUMP: dumping (Pass III) [directories]
DUMP: End of tape detected
DUMP: Closing /dev/fd0
DUMP: Volume 1 completed at: Mon Jul  9 16:04:49 2007
DUMP: Volume 1 1470 blocks (1.44MB)
DUMP: Volume 1 took 0:01:19
DUMP: Volume 1 transfer rate: 18 kB/s
DUMP: Change Volumes: Mount volume #2
DUMP: Is the new volume mounted and ready to go?: ("yes" or "no") y
DUMP: Volume 2 started with block 1441 at: Mon Jul  9 16:05:10 2007
DUMP: Volume 2 begins with blocks from inode 2
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing /dev/fd0
DUMP: Volume 2 completed at: Mon Jul  9 16:06:28 2007
DUMP: Volume 2 1410 blocks (1.38MB)
DUMP: Volume 2 took 0:01:18
DUMP: Volume 2 transfer rate: 18 kB/s
DUMP: 2850 blocks (2.78MB) on 2 volume(s)
DUMP: finished in 109 seconds, throughput 26 kBytes/sec
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Date this dump completed: Mon Jul  9 16:06:28 2007
DUMP: Average transfer rate: 18 kB/s
DUMP: DUMP IS DONE
```

If you back up to tape, remember that the tape will usually be rewound after each job. Devices with a name like `/dev/st0` or `/dev/st1` automatically rewind. The corresponding non-rewind equivalent devices are `/dev/nst0` and `/dev/nst1`. In any event, you can always use the `mt` command to perform magnetic tape operations such as forward spacing over files and records, back spacing, rewinding, and writing EOF marks. See the man pages for `mt` and `st` for additional information.

If you select the dump levels judiciously, you can minimize the number of archives you need to restore to any particular level. See the man pages for `dump` for a suggestion based on the Towers of Hanoi puzzle.

As with the `dd` command, there are many options that are not covered in this brief introduction. See the man pages for more details.

Partial and manual backups

So far, you have learned about tools that work well for backing up whole filesystems. Sometimes your backup needs to target selected files or subdirectories without backing up the whole filesystem. For example, you might need a weekly backup of most of your system, but daily backups of your mail files. Two other programs, `cpio` and `tar`, are more commonly used for this purpose. Both can write archives to files or to devices such as tape or floppy disk, and both can restore from such archives. Of the two, `tar` is more commonly used today, possibly because it handles complete directories better, and GNU `tar` supports both `gzip` and `bzip` compression.

Using `cpio`

The `cpio` command operates in *copy-out* mode to create an archive, *copy-in* mode to restore an archive, or *copy-pass* mode to copy a set of files from one location to another. You use the `-o` or `--create` option for copy-out mode, the `-i` or `--extract` option for copy-in mode, and the `-p` or `--pass-through` option for copy-pass mode. Input is a list of files provided on `stdin`. Output is either to `stdout` or to a device or file specified with the `-f` or `--file` option.

Listing 44 shows how to generate a list of files using the `find` command. Note the use of the `-print0` option on `find` to generate null-terminate strings for file names, and the corresponding `--null` option on `cpio` to read this format. This will correctly handle file names that have embedded blank or newline characters.

Listing 44. Back up a home directory using `cpio`

```
[root@lyrebird ~]# find ~ian -depth -print0 | cpio --null -o
>backup-cpio-1
18855 blocks
```

If you'd like to see the files listed as they are archived, add the `-v` option to `cpio`.

As with other commands that can archive to tape, the block size may be specified. For details on this and other options, see the man page.

Using `tar`

The `tar` (originally from *Tape ARchive*) creates an archive file, or *tarfile* or *tarball*, from a set of input files or directories; it also restores files from such an archive. If a directory is given as input to `tar`, all files and subdirectories are automatically included, which makes `tar` very convenient for archiving subtrees of your directory structure.

As with the other archiving commands we have discussed, output can be to a file, a

device such as tape or diskette, or stdout. The output location is specified with the `-f` option. Other common options are `-c` to create an archive, `-x` to extract an archive, `-v` for verbose output, which lists the files being processed, `-z` to use gzip compression, and `-j` to use bzip2 compression. Most `tar` options have a short form using a single hyphen and a long form using a pair of hyphens. The short forms are illustrated here. See the man pages for the long form and for additional options.

Listing 45 shows how to create a backup of the system cron jobs using `tar`.

Listing 45. Backup of system cron jobs using tar

```
[root@lyrebird ~]# tar -czvf backup-tar-1 /etc/*crontab /etc/cron.d
tar: Removing leading `/' from member names
/etc/anacrontab
/etc/crontab
/etc/cron.d/
/etc/cron.d/sa-update
/etc/cron.d/smolt
```

In the first line of output, you are told that `tar` will remove the leading slash (/) from member names. This allows files to be restored to some other location for verification before replacing system files. It is a good idea to avoid mixing absolute path names with relative path names when creating an archive, since all will be relative when restoring from the archive.

The `tar` command can append additional files to an archive using the `-r` or `--append` option. This may cause multiple copies of a file in the archive. In such a case, the *last* one will be restored during a restore operation. You can use the `--occurrence` option to select a specific file among multiples. If the archive is on a regular filesystem instead of tape, you may use the `-u` or `--update` option to update an archive. This works like appending to an archive, except that the time stamps of the files in the archive are compared with those on the filesystem, and only files that have been modified since the archived version are appended. As mentioned, this does not work for tape archives.

As with the other commands you have studied here, there are many options that are not covered in this brief introduction. See the man or info pages for more details.

Backup file integrity

Backup file integrity is extremely important. There is no point in having a backup if it is bad. A good backup strategy also involves checking your backups.

The first step to ensuring backup integrity is to ensure that you have properly captured the data you are backing up. If the filesystem is unmounted or mounted read only, this is usually straightforward as the data you are backing up cannot

change during your backup. If you must back up filesystems, directories, or files that are subject to modification while you are taking the backup, you should verify that no changes have been made during your backup. If changes were made, you will need to have a strategy for capturing them, either by repeating the backup, or perhaps by replacing or superseding the affected files in your backup. Needless to say, this will also affect your restore procedures.

Assuming you took good backups, you will periodically need to verify your backups. One way is to restore the backup to a spare volume and verify that it matches what you backed up. This is easiest to do right before you allow updates on the filesystem you are backing up. If you back up to media such as CD or DVD, you may be able to use the `diff` command as part of your backup procedure to ensure that your backup is good. Remember that even good backups can deteriorate in storage, so you should check periodically, even if you do verify at the time of backup. Keeping digests using programs such as `md5sum` or `sha1sum` is also a good check on the integrity of a backup file.

Restore filesystems from backups

A counterpart to backing up files is the ability to restore them when needed. Occasionally you will want to restore an entire filesystem, but it is far more common to need to restore only specific files or perhaps a set of directories. Almost always you will restore to some temporary space and verify that what you have restored is indeed what you want and is consistent with the current state of your system before actually making the restored files live.

A related issue is the need to verify that the items you want happen to be on a particular backup, as often happens when a user needs access to a version of a file that was modified or perhaps deleted "sometime in the last week or two." With these thoughts in mind, let's look at some of the restoration options.

Restoring a dd archive

Recall that the `dd` command was not filesystem aware, so you will need to restore a dump of a partition to find out what is on it. Listing 46 shows how to restore the partition that was dumped back in Listing 39 to a partition, `/dev/sdc7`, that was specially created on a removable USB drive just for this purpose.

Listing 46. Restoring a partition using dd

```
[root@lyrebird ~]# dd if=backup-1 of=/dev/sdc7
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 44.0084 s, 23.7 MB/s
```

Recall that we added some files to the filesystem on `/dev/sda3` after this backup was taken. If you mount the newly restore partition and compare it with the original, you will see that this is indeed the case, as shown in Listing 47. Note that the file whose timestamp was updated using `touch` is not shown here, as you would expect.

Listing 47. Comparing the restored partition with current state

```
[root@lyrebird ~]# mount /dev/sdc7 /mnt/temp-dd/
[root@lyrebird ~]# diff -rq /grubfile/ /mnt/temp-dd/
Only in /grubfile/: newerfile
Only in /grubfile/: newfile
```

Restoring a dump archive using restore

Recall that our final use of `dump` was a differential backup and that we created a table of contents. Listing 48 shows how to use `restore` to check the files in the archive created by `dump`, using the archive itself (`backup-5`) or the table of contents (`backup-6-toc`).

Listing 48. Checking the contents of archives

```
[root@lyrebird ~]# restore -t -f backup-5
Dump tape is compressed.
Dump   date: Sun Jul  8 16:55:46 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 2 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
      2      .
100481      ./ibshome
100482      ./ibshome/index.html
      16      ./newfile
[root@lyrebird ~]# restore -t -A backup-6-toc
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
Starting inode numbers by volume:
  Volume 1: 2
      2      .
100481      ./ibshome
100482      ./ibshome/index.html
      16      ./newfile
      17      ./newerfile
```

The `restore` command can also compare the contents of an archive with the contents of the filesystem using the `-C` option. In Listing 49 we updated `newerfile` and then compared the backup with the filesystem.

Listing 49. Comparing an archive with a filesystem using restore

```
[root@lyrebird ~]# echo "something different" >/grubfile/newerfile
[root@lyrebird ~]# restore -C -f backup-6
Dump tape is compressed.
```

```

Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
fileys = /grubfile
./newerfile: size has changed.
Some files were modified!  1 compare errors

```

The `restore` command can restore interactively or automatically. Listing 50 shows how to restore `newerfile` to root's home directory (so you could examine it before replacing the updated file if needed), then replace the updated file with the backup copy. This example illustrates interactive restoration.

Listing 50. Restoring a file using restore

```

[root@lyrebird ~]# restore -i -f backup-6
Dump tape is compressed.
restore > ?
Available commands are:
  ls [arg] - list directory
  cd arg - change directory
  pwd - print current directory
  add [arg] - add `arg' to list of files to be extracted
  delete [arg] - delete `arg' from list of files to be extracted
  extract - extract requested files
  setmodes - set modes of requested directories
  quit - immediately exit program
  what - list dump header information
  verbose - toggle verbose flag (useful with ``ls'')
  prompt - toggle the prompt display
  help or `?' - print this list
If no `arg' is supplied, the current directory is used
restore > ls new*
newerfile
newfile
restore > add newerfile
restore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume # (none if no more volumes): 1
set owner/mode for '.'? [yn] y
restore > q
[root@lyrebird ~]# mv -f newerfile /grubfile

```

Restoring a cpio archive

The `cpio` command in copy-in mode (option `-i` or `--extract`) can list the contents of an archive or restore selected files. When you list the files, specifying the `--absolute-filenames` option reduces the number of extraneous messages that `cpio` will otherwise issue as it strips any leading `/` characters from each path that has one. Partial output from listing our previous archive is shown in Listing 51.

Listing 51. Restoring selected files using cpio

```

[root@lyrebird ~]# cpio -id --list --absolute-filenames <backup-cpio-1

```

```

/home/ian/.gstreamer-0.10/registry.i686.xml
/home/ian/.gstreamer-0.10
/home/ian/.Trash/gnome-terminal.desktop
/home/ian/.Trash
/home/ian/.bash_profile

```

Listing 52 shows how to restore all the files with "samp" in their path name or file name. The output has been piped through `uniq` to reduce the number of "Removing leading '/' ..." messages. You must specify the `-d` option to create directories; otherwise, all files are created in the current directory. Furthermore, `cpio` will not replace any newer files on the filesystem with archive copies unless you specify the `-u` or `--unconditional` option.

Listing 52. Restoring selected files using cpio

```

[root@lyrebird ~]# cpio -ivd "*samp*" < backup-cpio-1 2>&1 |uniq
cpio: Removing leading `/' from member names
home/ian/crontab.samp
cpio: Removing leading `/' from member names
home/ian/sample.file
cpio: Removing leading `/' from member names
18855 blocks

```

Restoring a tar archive

The `tar` command can also compare archives with the current filesystem as well as restore files from archives. Use the `-d`, `--compare`, or `--diff` option to perform comparisons. The output will show files whose contents differ as well as files whose time stamps differ. Listing 53 shows verbose output (using option `-v`), from a comparison of the file created earlier and the files in `/etc` after `/etc/crontab` has been touched to alter its time stamp. The option `directory /` instructs `tar` to perform the comparison starting from the root directory rather than the current directory.

Listing 53. Comparing archives and files using tar

```

[root@lyrebird ~]# touch /etc/crontab
[root@lyrebird ~]# tar --diff -vf backup-tar-1 --directory /
etc/anacrontab
etc/crontab
etc/crontab: Mod time differs
etc/cron.d/
etc/cron.d/sa-update
etc/cron.d/smolt

```

Listing 54 shows how to extract just `/etc/crontab` and `/etc/anacrontab` into the current directory.

Listing 54. Extracting archive files using tar

```

[root@lyrebird ~]# tar -xzvf backup-tar-1 "*tab"

```

```
etc/anacrontab
etc/crontab
```

Note that `tar`, in contrast to `cpio` creates the directory hierarchy for you automatically.

The next section of this tutorial shows you how to maintain system time.

Section 7. System time

This section covers material for topic 1.111.6 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

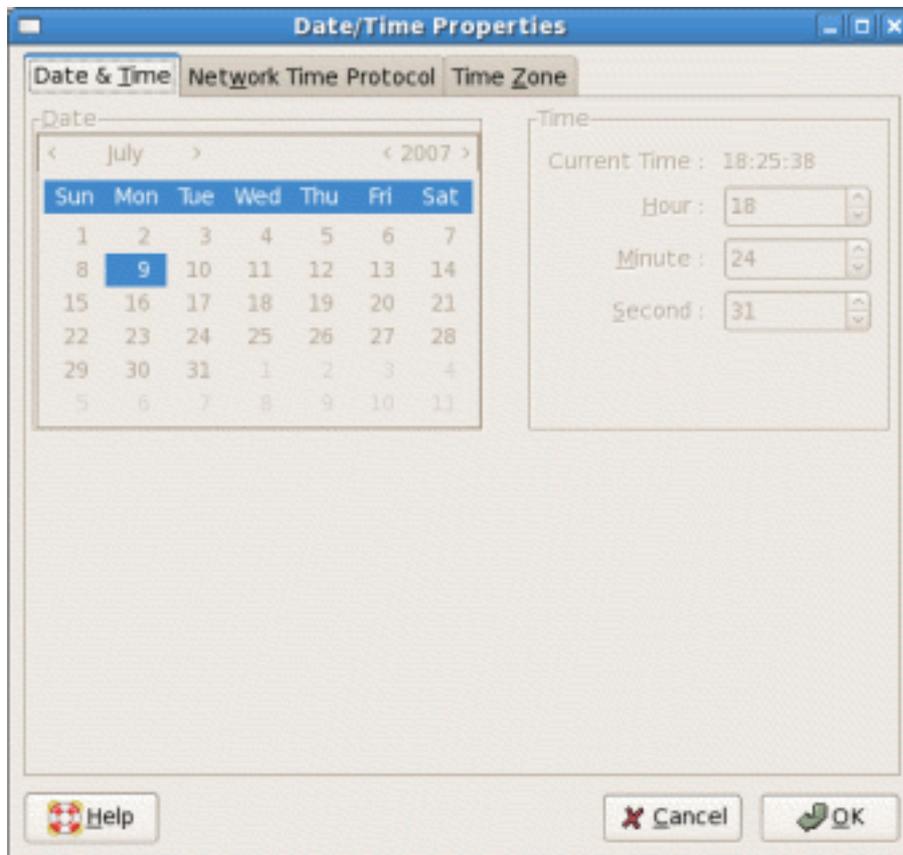
- Set the system date and time
- Set the BIOS clock to the correct UTC time
- Configure your time zone
- Configure the Network Time Protocol (NTP) service, including correcting for clock drift

Set the system date and time

System time on a Linux system is very important. You saw earlier how the cron and anacron facilities do things based on time, so they need an accurate time to base decisions on. Most of the backup and restore tools discussed in the previous section, along with development tools such as `make`, also depend on reliable time measurements. Most computers built since around 1980 include some kind of clock mechanism, and most built since 1984 or so have a persistent clock mechanism that keeps time even if the computer is turned off.

If you installed a Linux system graphically, you probably set the clock and chose a time zone suitable for your needs. You may have elected to use the Network Time Protocol (NTP) to set your clock, and you may or may not have elected to keep the system clock using Coordinated Universal Time or UTC. If you subsequently went to set the clock using graphical tools on a Fedora or Red Hat or similar system, you may have seen a dialog box like that in Figure 3.

Figure 3. Updating the date and time



Surprise! You can't actually set the clock yourself using this dialog. In this section you learn more about the difference between local clocks and NTP and how to set your system time.

No matter whether you live in New York, Budapest, Nakhodka, Ulan Bator, Bangkok, or Canberra, most of your Linux time computations are related to Coordinated Universal Time or UTC. If you run a dedicated Linux system, it is customary to keep the hardware clock set to UTC, but if you also boot another operating system such as Windows, you may need to set the hardware clock to local time. It really doesn't matter as far as Linux is concerned, except that there happen to be two different methods of keeping track of time zones internally in Linux, and if they don't agree, you can wind up with some odd time stamps on FAT filesystems, among other things. Listing 55 shows you how to use the `date` command to display the current date and time. The display is always in local time, even if your hardware clock keeps UTC time.

Listing 55. Displaying the current date and time

```
[root@lyrebird ~]# date;date -u
Mon Jul  9 22:40:01 EDT 2007
```

The `date` command supports a wide variety of possible output formats, some of which you already saw back in [Listing 28](#). See the man page for `date` if you'd like to learn more about the various date formats.

If you need to set the date, you can do this by providing a date and time as an argument. The required format is historical and is somewhat odd even to Americans and truly odd to the rest of the world. You must specify at least month, day, hour, and minute in MMDDhhmm format, and you may also append a two- or four-digit year (CCYY or YY) and optionally a period (.) followed by a two-digit number of seconds. Listing 56 shows an example that alters the system date by a little over a minute.

Listing 56. Setting the system date and time

```
[root@lyrebird ~]# date; date 0709221407;date
Mon Jul  9 23:12:37 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
```

Set the BIOS clock to UTC time

Your Linux system, along with most other current operating systems, actually has two clocks. The first is the hardware clock, sometimes called the Real Time Clock, RTC, or BIOS clock, which is usually tied to an oscillating quartz crystal that is accurate to within a few seconds per day. It is subject to variations such as ambient temperature. The second is the internal software clock, which is driven by counting system interrupts. It is subject to variations caused by high system load and interrupt latency. Nevertheless, your system typically reads the hardware clock at startup and from then on uses the software clock. The `date` command that you just learned about sets the software clock, not the hardware clock.

If you use the Network Time Protocol (NTP), you may possibly set the hardware clock when you first install the system and never worry about it again. If not, this part of the tutorial will show you how to display and set the hardware clock time.

You can use the `hwclock` command to display the current value of the hardware clock. Listing 57 shows the current value of both the system and hardware clocks.

Listing 57. System and hardware clock values

```
[root@lyrebird ~]# date;hwclock
Mon Jul  9 22:16:11 EDT 2007
Mon 09 Jul 2007 11:14:49 PM EDT -0.071616 seconds
```

Notice that the two values are different. You can synchronize the hardware clock

from the system clock using the `-w` or `--systohc` option of `hwclock`, and you can synchronize the system clock from the hardware clock using the `-s` or `--hctosys` option, as shown in Listing 58.

Listing 58. Setting the system clock from the hardware clock

```
[root@lyrebird ~]# date;hwclock;hwclock -s;date
Mon Jul  9 22:20:23 EDT 2007
Mon 09 Jul 2007 11:19:01 PM EDT  -0.414881 seconds
Mon Jul  9 23:19:02 EDT 2007
```

You may specify either the `--utc` or the `--localtime` option to have the system clock kept in UTC or local time. If no value is specified, the value is taken from the third line of `/etc/adjtime`.

The Linux kernel has a mode that copies the system time to the hardware clock every 11 minutes. This is off by default, but is turned on by NTP. Running anything that set the time the old fashioned way, such as `hwclock --hctosys`, turns it off, so it's a good idea to just let NTP do its work if you are using NTP. See the man page for `adjtimex` to find out how to check whether the clock is being updated every 11 minutes or not. You may need to install the `adjtimex` package as it is not always installed by default.

The `hwclock` command keeps track of changes made to the hardware clock in order to compensate for inaccuracies in the clock frequency. The necessary data points are kept in `/etc/adjtime`, which is an ASCII file. If you are not using the Network Time Protocol, you can use the `adjtimex` command to compensate for clock drift. Otherwise, the hardware clock will be adjusted approximately every 11 minutes by NTP. Besides showing whether your hardware clock is in local or UTC time, the first value in `/etc/adjtime` shows the amount of hardware clock drift per day (in seconds). Listing 59 shows two examples.

Listing 59. /etc/adjtime showing clock drift and local or UTC time.

```
[root@lyrebird ~]# cat /etc/adjtime
0.000990 1184019960 0.000000
1184019960
LOCAL
root@pinguino:~# cat /etc/adjtime
-0.003247 1182889954 0.000000
1182889954
LOCAL
```

Note that both these systems keep the hardware clock in local time, but the clock drifts are different — 0.000990 on lyrebird and -0.003247 on pinguino.

Configure your time zone

Your time zone is a measure of how far your local time differs from UTC. Information on available time zones that can be configured is kept in `/usr/share/zoneinfo`. Traditionally, `/etc/localtime` was a link to one of the time zone files in this directory tree, for example, `/usr/share/zoneinfo/Eire` or `/usr/share/zoneinfo/Australia/Hobart`. On modern systems it is much more likely to be a copy of the appropriate time zone data file since the `/usr/share` filesystem may not be mounted when the local time zone information is needed early in the boot process.

Similarly, another file, `/etc/timezone` was traditionally a link to `/etc/default/init` and was used to set the time zone environment variable `TZ`, and several locale-related environment variables. The file may or may not exist on your system. If it does, it may simply contain the name of the current time zone. You may also find time zone information in `/etc/sysconfig/clock`. Listing 60 shows these files from a Ubuntu 7.04 and a Fedora 7 system.

Listing 60. Time zone information in `/etc`

```
root@pinguino:~# cat /etc/timezone
America/New_York

[root@lyrebird ~]# cat /etc/sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
# The timezone of the system is defined by the contents of
/etc/localtime.
ZONE="America/New York"
UTC=false
ARC=false
```

Some systems such as Debian and Ubuntu have a `tzconfig` command to set the time zone. Others such as Fedora use `system-config-date` to set the time zone and to indicate whether the clock uses UTC or not. Listing 61 illustrates the use of the `tzconfig` command to display the current time zone.

Listing 61. Setting time zone with `tzconfig`

```
root@pinguino:~# tzconfig
Your current time zone is set to America/New_York
Do you want to change that? [n]:
Your time zone will not be changed
```

Configure the Network Time Protocol

The *Network Time Protocol (NTP)* is a protocol to synchronize computer clocks over a network. Synchronization is usually to UTC.

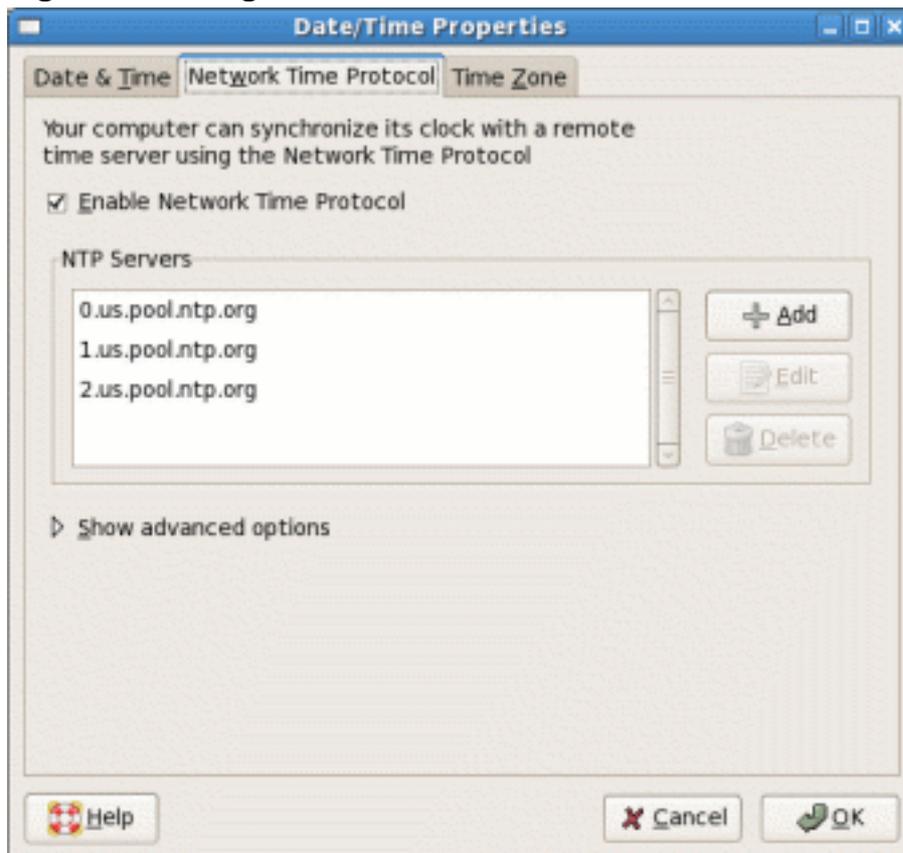
NTP version 3 is an Internet draft standard, formalized in RFC 1305. The current development version, NTP version 4 is a significant revision, which has not been formalized in an RFC. RFC 4330 describes Simple NTP (SNTP) version 4.

Time synchronization is accomplished by sending messages to *time servers*. The time returned is adjusted by an offset of half the round-trip delay. The accuracy of the time is therefore dependent on the network latency and the extent to which the latency is the same in both directions. The shorter the path to a time server, the more accurate the time is likely to be. See [Resources](#) for more detailed information than this simplistic description can provide.

There is a huge number of computers on the Internet, so time servers are organized into *strata*. A relatively small number of stratum 1 servers maintain very accurate time from a source such as an atomic clock. A larger number of stratum 2 servers get their time from stratum 1 servers and make it available to an even larger number of stratum 3 servers, and so on. To ease the load on time servers, a large number of volunteers donate time services through pool.ntp.org (see [Resources](#) for a link). Round robin DNS servers accomplish NTP load balancing by distributing NTP server requests among a pool of available servers.

If you use a graphical interface, you might be able to set your NTP time servers using a dialog similar to that in Figure 4. The fact that this system has enabled automatic time updates using NTP is why the dialog in Figure 3 did not allow the date and time to be changed.

Figure 4. Setting NTP servers



NTP configuration information is kept in `/etc/ntp.conf`, so you can also edit that file and then restart the `ntpd` daemon after you save it. Listing 62 shows an example `/etc/ntp.conf` file using the time servers from Figure 4.

Listing 62. Setting time zone with `tzconfig`

```
[root@lyrebird ~]# cat /etc/ntp.conf
# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#broadcast 192.168.1.255 key 42          # broadcast server
#broadcastclient          # broadcast client
#broadcast 224.0.1.1 key 42            # multicast server
#multicastclient 224.0.1.1            # multicast client
#manycastserver 239.255.254.254        # manycast server
#manycastclient 239.255.254.254 key 42 # manycast client

# Undisciplined Local Clock. This is a fake driver intended for backup
# and when no outside source of synchronized time is available.
#server 127.127.1.0 # local clock
#fudge 127.127.1.0 stratum 10

# Drift file. Put this in a directory which the daemon can write to.
# No symbolic links allowed, either, since the daemon updates the file
# by creating a temporary in the same directory and then rename()'ing
# it to the file.
driftfile /var/lib/ntp/drift

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8
server 0.us.pool.ntp.org
restrict 0.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 1.us.pool.ntp.org
restrict 1.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 2.us.pool.ntp.org
restrict 2.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
```

If you are using the `pool.ntp.org` time servers, these may be anywhere in the world. You will usually get better time by restricting your servers as in this example where `us.pool.ntp.org` is used, resulting in only U.S. servers being chosen. See [Resources](#)

for more information on the ntp.pool.org project.

NTP commands

You can use the `ntpdate` command to set your system time from an NTP time server as shown in Listing 63.

Listing 63. Setting system time from an NTP server using `ntpdate`

```
[root@lyrebird ~]# ntpdate 0.us.pool.ntp.org
10 Jul 10:27:39 ntpdate[15308]: adjust time server 66.199.242.154 offset
-0.007271 sec
```

Because the servers operate in round robin mode, the next time you run this command you will probably see a different server. Listing 64 shows the first few DNS responses for `0.us.ntp.pool.org` a few moments after the above `ntpdate` command was run.

Listing 64. Round robin NTP server pool

```
[root@lyrebird ~]# dig 0.pool.ntp.org +noall +answer | head -n 5
0.pool.ntp.org. 1062 IN A 217.116.227.3
0.pool.ntp.org. 1062 IN A 24.215.0.24
0.pool.ntp.org. 1062 IN A 62.66.254.154
0.pool.ntp.org. 1062 IN A 76.168.30.201
0.pool.ntp.org. 1062 IN A 81.169.139.140
```

The `ntpdate` command is now deprecated as the same function can be done using `ntpq` with the `-q` option, as shown in Listing 65.

Listing 65. Setting system time using `ntpd -q`

```
[root@lyrebird ~]# ntpd -q
ntpd: time slew -0.014406s
```

Note that the `ntpd` command uses the time server information from `/etc/ntp.conf`, or a configuration file provided on the command line. See the man page for more information and for information about other options for `ntpd`. Be aware also that if the `ntpd` daemon is running, `ntpd -q` will quietly exit, leaving a failure message in `/var/log/messages`.

Another related command is the `ntpq` command, which allows you to query the NTP daemon. See the man page for more details.

This brings us to the end of this tutorial. We have covered a lot of material on system administration. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- See the [Partimage homepage](#) for information on Partimage, a filesystem-aware partition dump and restore tool.
- "[/etc: Host-specific system configuration](#)" describes the Linux Standard Base (LSB) requirements for /etc.
- The [Network Time Protocol Project](#) produces a reference implementation of the NTP protocol, and implementation documentation.
- The [Network Time Synchronization Project](#) maintains an extensive array of documentation and background information, including briefing slides, on network time protocols.
- The [pool.ntp.org project](#) is a big virtual cluster of timeservers striving to provide reliable easy to use NTP service for millions of clients without putting a strain on the big popular timeservers.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content.](#)
- Get involved in the [developerWorks community](#) through our developer blogs, forums, podcasts, and community topics in our new [developerWorks spaces](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 111: Administrative tasks

Junior Level Administration (LPIC-1) topic 111

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

10 Jul 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this sixth in a [series of nine tutorials](#), Ian introduces you to administrative tasks. By the end of this tutorial, you will know how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *intermediate level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active intermediate level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the four junior and intermediate certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts.
Topic 111	LPI exam 102 prep: Administrative tasks	(This tutorial.) Learn how to manage user and group accounts and tune user and system environments, configure and use system log files, automate system administration tasks by scheduling jobs to run at another time, back up your system, and maintain system time. See the detailed objectives below.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Administrative tasks," the sixth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.111.1 User and group accounts	Weight 4	Add, remove, suspend, and change user accounts. Manage user and group information in password and group databases, including shadow databases. Create and manage special purpose and limited accounts.
1.111.2 Tune user and system environments	Weight 3	Modify global and user profiles. Set environment variables and maintain skeleton directories for new user accounts. Set command search paths.
1.111.3 Configure and use system log files to meet administrative and security needs	Weight 3	Configure and manage system logs, including the type and level of logged information. Scan and monitor log files for

		notable activity and track down noted problems. Rotate and archive log files.
1.111.4 Automate system administration tasks by scheduling jobs to run in the future	Weight 4	Use the <code>cron</code> or <code>anacron</code> commands to run jobs at regular intervals, and use the <code>at</code> command to run jobs at a specific time.
1.111.5 Maintain an effective data backup strategy	Weight 3	Plan a backup strategy and back up filesystems automatically to various media.
1.111.6 Maintain system time	Weight 4	Maintain the system time and time zone, and synchronize the clock via NTP. Set the BIOS clock to the correct time in UTC, and configure NTP, including correcting for clock drift.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be thoroughly familiar with the material from the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, which covers basic concepts of users, groups, and file permissions.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. User and group accounts

This section covers material for topic 1.111.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Add, modify, and remove users and groups
- Suspend and change user accounts
- Manage user and group information in the password databases and group databases
- Use the correct tools to manage shadow password databases and group databases
- Create and manage limited and special-purpose accounts

As you learned in the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, Linux is a multi-user system where each user belongs to one *primary* group and possibly to additional groups. Ownership of files in Linux is closely related to user ids and groups. Recall that you can log in as one user and become another user using the `su` or `sudo -s` commands, and that you can use the `whoami` command to check your current effective id and the `groups` command to find out what groups you belong to. In this section, you learn how to create, delete, and manage users and groups. You also learn about the files in `/etc`, where user and group information is stored.

Add and remove users and groups

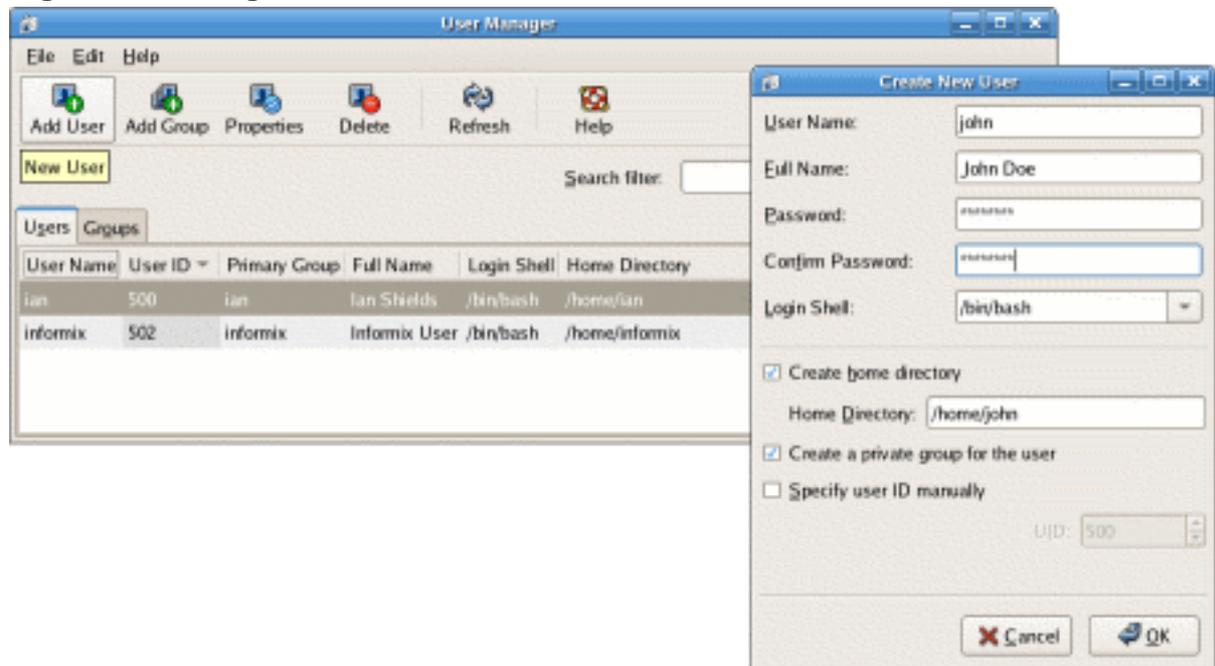
You add a user to a Linux system using the `useradd` command, and you delete a user using the `userdel` command. Similarly, you add or delete groups using the `groupadd` and `groupdel` commands.

Adding a user or group

Modern Linux desktops usually have graphical interfaces for user and group administration. The graphical interface is usually accessed through menu options for system administration. These interfaces do vary considerably, so the one on your system may not look much like the example here, but the underlying concepts and commands remain similar.

Let's start by adding a user to a Fedora Core 5 system graphically, and then examine the underlying commands. In the case of Fedora Core 5 with GNOME desktop, use **System > Administration > Users and Groups**, then click the **Add User** button.

Figure 1 depicts the User Manager panel with the Create New User panel showing basic information for a new user named 'john'. The full name of the user, John Doe, and a password have been entered. The panel provides a default login shell of `/bin/bash`. On Fedora systems, the default is to create a new group with the same name as the user, 'john' in this case, and a home directory of `/home/john`.

Figure 1. Adding a user

Listing 1 shows the use of the `id` command to display basic information about the new user. As you can see, john has user number 503 and a matching group, john, with group number 503. This is the only group of which john is a member.

Listing 1. Displaying user id information

```
[root@pinguino ~]# id john
uid=503(john) gid=503(john) groups=503(john)
```

To accomplish the same task from the command line, you use the `groupadd` and `useradd` commands to create the group and user, then use the `passwd` command to set the password for the newly created user. All of these commands require root authority. The basic use of these commands to add another user, jane, is illustrated in Listing 2.

Listing 2. Adding user jane

```
[root@pinguino ~]# groupadd jane
[root@pinguino ~]# useradd -c "Jane Doe" -g jane -m jane
[root@pinguino ~]# passwd jane
Changing password for user jane.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# id jane
uid=504(jane) gid=504(jane) groups=504(jane)
[root@pinguino ~]# ls -ld /home/jane
drwx----- 3 jane jane 4096 Jun 25 18:22 /home/jane
```

In these two examples, both the user id and the group id have values greater than 500. Be aware that some newer systems start user ids at 1000 rather than 500. These values normally signify ordinary users, while values below 500 (or 1000 if the system starts ordinary users at 1000) are reserved for *system users*. [System users](#) are covered later in this section. The actual cutoff points are set in `/etc/login.defs` as `UID_MIN` and `GID_MIN`.

In Listing 2 above, the `groupadd` command has a single parameter, `jane`, the name of the group to be added. Group names must begin with a lower case letter or an underscore, and usually contain only these along with hyphens or dashes. Options you may specify are shown in Table 3.

Option	Purpose
-f	Exit with success status if the group already exists. This is handy for scripting when you do not need to check if a group exists before attempting to create it.
-g	Specifies the group id manually. The default is to use the smallest value that is at least <code>GID_MIN</code> and also greater than the id of any existing group. Group ids are normally unique and must be non-negative
-o	Permits a group to have a non-unique id.
-K	Can be used to override defaults from <code>/etc/login.defs</code> .

In Listing 2 above, the `useradd` command has a single parameter, `jane`, the name of the user to be added, along with the `-c`, `-g`, and `-m` options. Common options for the `useradd` command are shown in Table 4.

Option	Purpose
-b --base-dir	The default base directory in which user home directories are created. This is usually <code>/home</code> , and the user's home directory is <code>/home/\$USER</code> .
-c --comment	A text string describing the id, such as the user's full name.
-d --home	Provides a specific directory name for the home directory.
-e	The date on which the account will

--expiredate	expire or be disabled in the form YYYY-MM_DD.
-g --gid	The name or number of the initial login group for the user. The group must exist, which is why group jane was created before user jane in Listing 2.
-G --groups	A comma-separated list of additional groups to which the user belongs.
-K	Can be used to override defaults from /etc/login.defs.
-m --create-home	Create the user's home directory if it does not exist. Copy the skeleton files and any directories from /etc/skel to the home directory.
-o --non-unique	Permits a user to have a non-unique id.
-p --password	The encrypted password. If a password is not specified, the default is to disable the account. You will usually use the <code>passwd</code> command in a subsequent step rather than generating an encrypted password and specifying it on the <code>useradd</code> command.
-s --shell	The name of the user's login shell if different from the default login shell.
-u --uid	The non-negative numerical userid, which must be unique if <code>-o</code> is not specified. The default is to use the smallest value that is at least <code>UID_MIN</code> and also greater than the id of any existing user.

Notes:

1. Some systems, including Fedora and Red Hat distributions, have extensions to the user-creation commands. For example, the default Fedora and Red Hat behavior is to create a new group for a user, and the `-n` option can be used on the `useradd` command to disable this function. Be aware of such possible system differences and refer to the man pages on your system when in doubt.
2. On SUSE systems, use YaST or YaST2 to access graphical user and group administration interfaces.
3. Graphical interfaces may perform additional tasks such as creating the user's mail file in `/var/spool/mail`.

Deleting a user or group

Deleting a user or group is much simpler than adding one, because there are fewer options. In fact, the `groupdel` command to delete a group requires only the group name; it has no options. You cannot delete any group that is the primary group of a user. If you use a graphical interface for deleting users and groups, the functions are very similar to the commands shown here.

Use the `userdel` command to delete a user. The `-r`, or `--remove` option requests removal of the user's home directory and anything it contains, along with the user's mail spool. When you delete a user, a group with the same name as the user will also be deleted if `USERGROUPS_ENAB` is set to `yes` in `/etc/login.defs`, but this will be done only if the group is not the primary group of another user.

In Listing 3 you see an example of deleting groups when multiple users share the same primary group. Here, another user, `jane2`, has previously been added to the system with the same group as `jane`.

Listing 3. Deleting users and groups

```
root@pinguino:~# groupdel jane
groupdel: cannot remove user's primary group.
root@pinguino:~# userdel -r jane
userdel: Cannot remove group jane which is a primary group for another
user.
root@pinguino:~# userdel -r jane2
root@pinguino:~# groupdel jane
```

Notes:

1. There is a `userdel` option, `-f` or `--force`, which can be used to delete users and their group. This option is dangerous, so you should use it only as a last resort. Read the man page carefully before you do.
2. Be aware that if you delete a user or group, and there are files that belong to that user or group on your filesystem, then the files are not automatically deleted or assigned to another user or group.

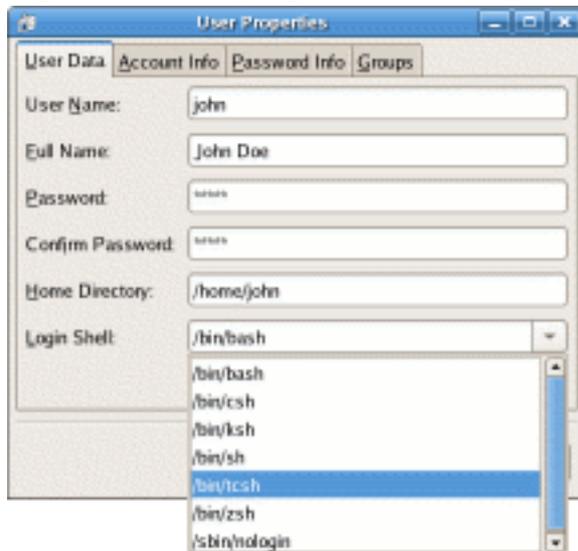
Suspend and change accounts

Now that you can create or delete a user id or a group, you may also find a need to modify one.

Modifying user accounts

Suppose user john wishes to have the tcsh shell as his default. From a graphical interface you will usually find a way to either edit a user (or group), or to examine the properties of the object. Figure 2 shows the properties dialog for the user john that we created earlier on a Fedora Core 5 system.

Figure 2. Modifying a user account



From the command line, you can use the `usermod` command to modify a user account. You can use most of the options that you use with `useradd`, except that you cannot create or populate a new home directory for the user. If you need to change the name of the user, specify the `-l` or `--login` option with the new name. You will probably want to rename the home directory to match the user id. You may also need to rename other items such as mail spool files. Finally, if the login shell is changed, some of the associated profile files may need to be altered. Listing 4 shows an example of the things you might need to do to change user john to john2 with `/bin/tcsh` as the default shell and renamed home directory `/home/john2`.

Listing 4. Modifying a user

```
[root@pinguino ~]# usermod -l john2 -s /bin/tcsh -d /home/john2 john
[root@pinguino ~]# ls -d ~john2
ls: /home/john2: No such file or directory
[root@pinguino ~]# mv /home/john /home/john2
[root@pinguino ~]# ls -d ~john2
/home/john2
```

Notes:

1. If you need to modify a user's additional groups, you must specify the complete list of additional groups. There is no command to simply add or delete a single group for a user.

2. There are restrictions on changing the name or id of a user who is logged in or who has running processes. Check the man pages for details.
3. If you change a user number, you may want to change files and directories owned by that user to match the new number.

Modifying groups

Not surprisingly, the `groupmod` command is used to modify group information. You can change the group number with the `-g` option, and the name with the `-n` option.

Listing 5. Renaming a group

```
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# groupmod -n john2 john
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

Notice in Listing 5 that the group name for the home directory of john2 magically changed when we used `groupmod` to change the group name. Are you surprised? Because groups are represented in the filesystem inodes by their number rather than by their name, this is not surprising. However, if you change a group's number, you should update any users for which that group is the primary group, and you may also want to update the files and directories belonging to that group to match the new number (in the same way as noted above for changing a user number). Listing 6 shows how to change the group number for john2 to 505, update the user account, and make appropriate changes to all the affected files in the `/home` filesystem. You probably want renumbering users and groups if at all possible.

Listing 6. Renumbering a group

```
[root@pinguino ~]# groupmod -g 505 john2
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# id john2
uid=503(john2) gid=503 groups=503
[root@pinguino ~]# usermod -g john2 john2
[root@pinguino ~]# id john2
uid=503(john2) gid=505(john2) groups=505(john2)
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# find /home -gid 503 -exec chgrp john2 {} \;
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

User and group passwords

You have already seen the `passwd` command, which is used to change a user password. The password is (or should be) unique to the user and may be changed by user. The root user may change any user's password as we have already seen.

Groups may also have passwords, and the `gpasswd` command is used to set them. Having a group password allows users to join a group temporarily with the `newgrp` command, if they know the group password. Of course, having multiple people knowing a password is somewhat problematic, so you will have to weigh the advantages of adding a user to a group using `usermod`, versus the security issue of having too many people knowing the group password.

Suspending or locking accounts

If you need to prevent a user from logging in, you can *suspend* or *lock* the account using the `-L` option of the `usermod` command. To *unlock* the account, use the `-U` option. Listing 7 shows how to lock account `john2` and what happens if `john2` attempts to log in to the system. Note that when the `john2` account is unlocked, the same password is restored.

Listing 7. Locking an account

```
[root@pinguino ~]# usermod -L john2
[root@pinguino ~]# ssh john2@pinguino
john2@pinguino's password:
Permission denied, please try again.
```

You may have noticed back in [Figure 2](#) that there were several tabs on the dialog box with additional user properties. We briefly mentioned the use of the `passwd` command for setting user passwords, but both it and the `usermod` command can perform many tasks related to user accounts, as can another command, the `chage` command. Some of these options are shown in Table 5. Refer to the appropriate man pages for more details on these and other options.

Table 5. Commands and options for changing user accounts			
	Option for command		Purpose
Usermod	Passwd	Chage	
-L	-l	N/A	Lock or suspend the account.
-U	-u	N/A	Unlock the account.
N/A	-d	N/A	Disable the account by setting it passwordless.

-e	-f	-E	Set the expiration date for an account.
N/A	-n	-m	The minimum password lifetime in days.
N/A	-x	-M	The maximum password lifetime in days.
N/A	-w	-W	The number of days of warning before a password must be changed.
-f	-i	-l	The number of days after a password expires until the account is disabled.
N/A	-S	-l	Output a short message about the current account status.

Manage user and group databases

The primary repositories for user and group information are four files in `/etc`.

`/etc/passwd`

is the *password* file containing basic information about users

`/etc/shadow`

is the *shadow password* file containing encrypted passwords

`/etc/group`

is the *group* file containing basic information about groups and which users belong to them

/etc/gshadow

is the *shadow group* file containing encrypted group passwords

These files are updated by the commands you have already seen in this tutorial and you will meet some more commands for working with them after we discuss the files themselves. All of these files are plain text files. In general, you should not edit them directly. Use the tools provided for updating them so they are properly locked and kept synchronized.

You will note that the passwd and group files are both *shadowed*. This is for security reasons. The passwd and group files themselves must be world readable, but the encrypted passwords should not be world readable. Therefore, the shadow files contain the encrypted passwords, and these files are only readable by root. The necessary authentication access is provided by an suid program that has root authority, but can be run by anyone. Make sure that your system has the permissions set appropriately. Listing 8 shows an example.

Listing 8. User and group database permissions

```
[ian@pinguino ~]$ ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow
-rw-r--r-- 1 root root 701 Jun 26 19:04 /etc/group
-r----- 1 root root 580 Jun 26 19:04 /etc/gshadow
-rw-r--r-- 1 root root 1939 Jun 26 19:43 /etc/passwd
-r----- 1 root root 1324 Jun 26 19:50 /etc/shadow
```

Note: Although it is still technically possible to run without shadowed password and group files, this is almost never done and is not recommended.

The /etc/passwd file

The /etc/passwd file contains one line for each user in the system. Some example lines are shown in Listing 9.

Listing 9. /etc/password entries

```
root:x:0:0:root:/root:/bin/bash
jane:x:504:504:Jane Doe:/home/jane:/bin/bash
john2:x:503:505:John Doe:/home/john2:/bin/tcsh
```

Each line contains seven fields separated by colons (:), as shown in Table 6.

Table 6. Fields in /etc/passwd	
Field	Purpose
Username	The name used to log in to the system.

	For example, john2.
Password	The encrypted password. When using shadow passwords, it contains a single x character.
User id (UID)	The number used to represent this user name in the system. For example, 503 for user john2.
Group id (GID)	The number used to represent this user's primary group in the system. For example, 505 for user john2.
Comment (GECOS)	An optional field used to describe the user. For example, "John Doe". The field may contain multiple comma-separated entries. It is also used by programs such as <code>finger</code> . The GECOS name is historic. See details in <code>man 5 passwd</code> .
Home	The absolute path the user's home directory. For example, <code>/home/john2</code>
Shell	The program automatically launched when a user logs in to the system. This is usually an interactive shell such as <code>/bin/bash</code> or <code>/bin/tcsh</code> , but may be any program, not necessarily an interactive shell.

The `/etc/group` file

The `/etc/group` file contains one line for each group in the system. Some example lines are shown in Listing 10.

Listing 10. `/etc/group` entries

```
root:x:0:root
jane:x:504:john2
john2:x:505:
```

Each line contains four fields separated by colons (:), as shown in Table 7.

Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The encrypted password. When using shadow group passwords, it contains a single x character.

Group id (GID)	The number used to represent this group in the system. For example, 505 for group john2.
Members	A comma-separated list of group members, excepting those members for whom this is the primary group.

Shadow files

The file `/etc/shadow` should only be readable by root. It contains encrypted passwords, along with password and account expiration information. See the man page (`man 5 shadow`) for information on the field layout. Passwords may be encrypted using DES, but are more usually encrypted using MD5. The DES algorithm uses the low order 7 bits of the first 8 characters of the user password as a 56-bit key, while the MD5 algorithm uses the whole password. In either case, passwords are *salted* so that two otherwise identical passwords do not generate the same encrypted value. Listing 11 shows how to set identical passwords for users jane and john2, and then shows the resulting encoded MD5 passwords in `/etc/shadow`.

Listing 11. Passwords in `/etc/shadow`

```
[root@pinguino ~]# echo lpic1111 |passwd jane --stdin
Changing password for user jane.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# echo lpic1111 |passwd john2 --stdin
Changing password for user john2.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# grep "^j" /etc/shadow
jane:$1$eG0/KGQY$ZJ1.ltYtVw0sv.C5OrqUu/:13691:0:99999:7:::
john2:$1$grkxo6ie$J2muvoTpwo3dZAYYTDYNu.:13691:0:180:7:29::
```

The leading `1` indicates an MD5 password, and the salt is a variable length field of up to 8 characters ending with the next `$` sign. The encrypted password is the remaining string of 22 characters.

Tools for users and groups

You have already seen several commands that manipulate the account and group files and their shadows. Here you learn about:

- Group administrators
- Editing commands for password and group files
- Conversion programs

Group administrators

In some circumstances you may want users other than root to be able to administer one or more groups by adding or removing group members. Listing 12 shows how root can add user jane as an administrator of group john2, and then jane, in turn, can add user ian as a member.

Listing 12. Adding group administrators and members

```
[root@pinguino ~]# gpasswd -A jane john2
[root@pinguino ~]# su - jane
[jane@pinguino ~]$ gpasswd -a ian john2
Adding user ian to group john2
[jane@pinguino ~]$ id ian;id jane
uid=500(ian) gid=500(ian) groups=500(ian),505(john2)
uid=504(jane) gid=504(jane) groups=504(jane)
```

You may be surprised to note that, although jane is an administrator of group john2, she is not a member of it. An examination of the structure of `/etc/gshadow` shows why. The `/etc/gshadow` file contains four fields for each entry as shown in Table 8. Note that the third field is a comma-separated list of administrators for the group.

Table 8. Fields in <code>/etc/gshadow</code>	
Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The field used to contain the encrypted password if the group has a password. If there is no password, you may see 'x', '!' or '!!' here.
Admins	A comma-separated list of group administrators.
Members	A comma-separated list of group members.

As you can see, the administrator list and the member list are two distinct fields. The `-A` option of `gpasswd` allows the root user to add administrators to a group, while the `-M` option allows root to add members. The `-a` (note lower case) option allows an administrator to add a member, while the `-d` option allows an administrator to remove a member. Additional options allow a group password to be removed. See the man pages for details.

Editing commands for password and group files

Although not listed in the LPI objectives, you should also be aware of the `vipw` command for safely editing `/etc/passwd` and `visgr` for safely editing `/etc/group`. The

commands will lock the necessary files while you make changes using the `vi` editor. If you make changes to `/etc/passwd`, then `vipw` will prompt you to see if you also need to update `/etc/shadow`. Similarly, if you update `/etc/group` using `vigr`, you will be prompted to update `/etc/gshadow`. If you need to remove group administrators, you may need to use `vigr`, as `gpasswd` only allows addition of administrators.

Conversion programs

Four other related commands are also not listed in the LPI objectives. They are `pwconv`, `pwunconv`, `grpconv`, and `grpunconv`. They are used for converting between shadowed and non-shadowed password and group files. You may never need these, but be aware of their existence. See the man pages for details.

Limited and special-purpose accounts

By convention, system users usually have an id of less than 100, with root having id 0. Normal users start automatic numbering from the `UID_MIN` value set in `/etc/login.defs`, with this value commonly being set at 500 or 1000.

Besides regular user accounts and the root account on your system, you will usually have several special-purpose accounts, for daemons such as FTP, SSH, mail, news, and so on. Listing 13 shows some entries from `/etc/passwd` for these.

Listing 13. Limited and special-purpose accounts

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
```

Such accounts frequently control files but should not be accessed by normal login. Therefore, they usually have a login shell specified as `/sbin/nologin`, or `/bin/false` so that login attempts will fail.

Section 3. Environment tuning

This section covers material for topic 1.111.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to tune the user environment, including these tasks:

- Set and unset environment variables
- Maintain skeleton directories for new user accounts
- Set command search paths

Set and unset environment variables

When you create a new user, you usually initialize many variable according to your local needs. These are usually set in the profiles that you provide for new users, such as `.bash_profile` and `.bashrc`, or in the system-wide profiles `/etc/profile` and `/etc/bashrc`. Listing 14 shows an example of how the `PS1` system prompt is set in `/etc/profile` on an Ubuntu 7.04 system. The first `if` statement checks whether the `PS1` variable is set, indicating an interactive shell, since a non-interactive shell doesn't need a prompt. The second `if` statement checks whether the `BASH` environment variable is set. If so, it sets a complex prompt and sources (note the dot) `/etc/bash.bashrc`. If the `BASH` variable is not set, then a check is made for root (`id=0`), and the prompt is set to `#` or `$` accordingly.

Listing 14. Setting environment variables

```
if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
```

The tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) has detailed information about the commands used for setting and unsetting environment variables, as well as information about how and when the various profiles are used.

When customizing environments for users, be aware of two major points:

1. `/etc/profile` is read only at login time, and so it is not executed when each new shell is created.

2. Functions and aliases are not inherited by new shells. Therefore, you will usually set these and your environment variables in `/etc/bashrc`, or in the user's own profiles.

In addition to the system profiles, `/etc/profile` and `/etc/bashrc`, the Linux Standard Base (LSB) specifies that additional scripts may be placed in the directory `/etc/profile.d`. These scripts are sourced when an interactive login shell is created. They provide a convenient way of separating customization for different programs. Listing 15 shows an example.

Listing 15. `/etc/profile.d/vim.sh` on Fedora 7

```
[if [ -n "$BASH_VERSION" -o -n "$KSH_VERSION" -o -n "$ZSH_VERSION" ];
then
  [ -x //usr/bin/id ] || return
  [ `//usr/bin/id -u` -le 100 ] && return
  # for bash and zsh, only if no alias is already set
  alias vi >/dev/null 2>&1 || alias vi=vim
fi
```

Remember that you should usually `export` any variables that you set in a profile; otherwise, they will not be available to commands that run in a new shell.

Maintain skeleton directories for new users

You learned in the section [Add and remove users and groups](#) that you can create or populate a new home directory for the user. The source for this new directory is the subtree rooted at `/etc/skel`. Listing 16 shows the files in this subtree for a Fedora 7 system. Note that most files start with a period (dot), so you need the `-a` option to list them. The `-R` options lists subdirectories recursively, and the `-L` option follows any symbolic links.

Listing 16. `/etc/skel` on Fedora 7

```
[ian@lyrebird ~]$ ls -aRL /etc/skel
/etc/skel:
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .xemacs

/etc/skel/.xemacs:
.  ..  init.el
```

In addition to `.bash_logout`, `.bash_profile`, and `.bashrc`, which you might expect for the Bash shell, note that this example includes profile information for the emacs and xemacs editors. See the tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) if you need to review the functions of the various profile files.

Listing 17 shows `/etc/skel/.bashrc` from the above system. This file might be different on a different release or different distribution, but it gives you an idea of how the default user setup can be done.

Listing 17. `/etc/skel/.bashrc` on Fedora 7

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
```

As you can see, the global `/etc/bashrc` is sourced, then any user specific instructions can be added. Listing 18 shows the part of `/etc/bashrc` in which the `.sh` scripts from `/etc/profile.d` are sourced.

Listing 18. Sourcing `.sh` scripts from `/etc/profile.d`

```
for i in /etc/profile.d/*.sh; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i
```

Note that the variable, `i`, is unset after the loop.

Set command search paths

Your default profiles often include `PATH` variables for local functions or for products that you may have installed. You can set these in the skeleton files in `/etc/skel`, modify `/etc/profile`, `/etc/bashrc`, or create a file in `/etc/profile.d` if your system uses that. If you do modify the system files, be sure to check that your changes are intact after any system updates. Listing 19 shows how to add a new directory, `/opt/productxyz/bin`, to either the front or rear of your existing `PATH`.

Listing 19. Adding a path directory

```
PATH="$PATH${PATH:+:}/opt/productxyz/bin"
PATH="/opt/productxyz/bin${PATH:+:}$PATH"
```

Although not strictly required, the expression `${PATH:+:}` inserts a path separator (colon) only if the `PATH` variable is unset or null.

Section 4. System log files

This section covers material for topic 1.111.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to configure and manage system logs, including these tasks:

- Manage the type and level of information logged
- Rotate and archive log files automatically
- Scan log files for notable activity
- Monitor log files
- Track down problems reported in log files

Manage the type and level of information logged

The system logging facility on a Linux system provides system logging and kernel message trapping. Logging can be done on a local system or sent to a remote system, and the level of logging can be finely controlled through the `/etc/syslog.conf` configuration file. Logging is performed by the `syslogd` daemon, which normally receives input through the `/dev/log` socket, as shown in Listing 20.

Listing 20. `/dev/log` is a socket

```
ian@pinguino:~$ ls -l /dev/log
srw-rw-rw- 1 root root 0 2007-07-05 15:42 /dev/log
```

For local logging, the main file is usually `/var/log/messages`, but many other files are used in most installations, and you can customize these extensively. For example, you may want a separate log for messages from the mail system.

The `syslog.conf` configuration file

The `syslog.conf` file is the main configuration file for the `syslogd` daemon. Logging is based on a combination of facility and priority. The defined facilities are `auth` (or `security`), `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `syslog`, `user`, `uucp`, and `local0` through `local7`. The keyword `auth` should be used instead of `security`, and the keyword `mark` is for internal use.

The priorities (in ascending order) are:

1. debug
2. info
3. notice
4. warning (or warn)
5. err (or error)
6. crit
7. alert
8. emerg (or panic)

The parenthesized keywords (warn, error, and panic) are now deprecated.

Entries in `syslog.conf` specify logging rules. Each rule has a selector field and an action field, which are separated by one or more spaces or tabs. The selector field identifies the facility and the priorities that the rule applies to, and the action field identifies the logging action for the facility and priorities. The default behavior is to take the action for the specified level and for all higher levels, although it is possible to limit logging to specific levels. Each selector consists of a facility and a priority separated by a period (dot). Multiple facilities for a given action can be specified by separating them with a comma. Multiple facility/priority pairs for a given action can be specified by separating them with a semi-colon. Listing 21 shows an example of a simple `syslog.conf`.

Listing 21. Example `syslog.conf`

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                               /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages

# The authpriv file has restricted access.
authpriv.*                             /var/log/secure

# Log all the mail messages in one place.
mail.*
-/var/log/maillog

# Log cron stuff
cron.*                                  /var/log/cron
```

```
# Everybody gets emergency messages
*.emerg                                     *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                             /var/log/spooler

# Save boot messages also to boot.log
local7.*
/var/log/boot.log
```

Notes:

- As with many configuration files, lines starting with # and blank lines are ignored.
- An * may be used to refer to all facilities or all priorities.
- The special priority keyword `none` indicates that no logging for this facility should be done with this action.
- The hyphen before a file name (such as `-/var/log/maillog`, in this example) indicates that the log file should not be synchronized after every write. You might lose information after a system crash, but you might gain performance by doing this.

The actions are generically referred to as "logfiles," although they do not have to be real files. Table 9 describe the possible logfiles.

Table 9. Actions in syslog.conf	
Action	Purpose
Regular File	Specify the full pathname, beginning with a slash (/). Prefix it with a hyphen (-) to omit syncing the file after each log entry. This may cause information loss if a crash occurs, but may improve performance.
Named Pipes	A fifo or named pipe can be used as a destination for log messages by putting a pipe symbol () before the filename. You must create the fifo using the <code>mkfifo</code> command before starting (or restarting) <code>syslogd</code> . Fifos are sometimes used for debugging.
Terminal and Console	A terminal such as <code>/dev/console</code> .
Remote Machine	To forward messages to another host, put an at (@) sign before the hostname. Note that messages are not forwarded from the receiving host.
List of Users	A comma-separated list of users to receive a message (if the user is

	logged in). The root user is frequently included here.
Everyone logged on	Specify an asterisk (*) to have everyone logged on notified using the wall command.

You may prefix ! to a priority to indicate that the action should not apply to this level and higher. Similarly you may prefix it with = to indicate that the rule applies only to this level or with != to indicate that the rule applies to all except this level. Listing 22 shows some examples, and the man page for syslog.conf has many more examples.

Listing 22. More syslog.conf examples

```
# Store all kernel messages in /var/log/kernel.
# Send critical and higher ones to remote host pinguino and to the
console
# Finally, Send info, notice and warning messages to
/var/log/kernel-info
#
kern.*                /var/log/kernel
kern.crit             @pinguino
kern.crit             /dev/console
kern.info;kern.!err  /var/log/kernel-info

# Store all mail messages except info priority in /var/log/mail.
mail.*;mail.!=info   /var/log/mail
```

Rotate and archive log files automatically

With the amount of logging that is possible, you need to be able to control the size of log files. This is done using the `logrotate` command, which is usually run as a cron job. Cron jobs are covered later in this tutorial in the section [Scheduling jobs](#). The general idea behind the `logrotate` command is that log files are periodically backed up and a new log is started. Several generations of log are kept, and when a log ages to the last generation, it may be archived. For example, it might be mailed to an archival user.

You use the `/etc/logrotate.conf` configuration file to specify how your log rotating and archiving should happen. You can specify different frequencies, such as daily, weekly, or monthly, for different log files, and you can control the number of generations to keep and when or whether to mail copies to an archival user. Listing 23 shows a sample `/etc/logrotate.conf` file.

Listing 23. Sample /etc/logrotate.conf

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

The `logrotate.conf` file has global options at the beginning. These are the defaults if nothing more specific is specified elsewhere. In this example, log files are rotated weekly, and four weeks worth of backups are kept. Once a log file is rotated, a new one is automatically created in place of the old one. Note that the `logrotate.conf` file may include specifications from other files. Here, all the files in `/etc/logrotate.d` are included.

This example also includes specific rules for `/var/log/wtmp` and `/var/log/btmp`, which are rotated monthly. No error message is issued if the files are missing. A new file is created, and only one backup is kept.

In this example, when a backup reaches the last generation, it is deleted because there is no specification of what else to do with it.

Note: The files `/var/log/wtmp` and `/var/log/btmp` record successful and unsuccessful login attempts, respectively. Unlike most log files, these are not clear text files. You may examine them using the `last` or `lastb` commands. See the man pages for these commands for details.

Log files may also be backed up when they reach a specific size, and commands may be scripted to run either prior to or after the backup operation. Listing 24 shows a more complex example.

Listing 24. Another logrotate configuration example

```
/var/log/messages {
```

```

rotate 5
mail logsave@pinguino
size 100k
postrotate
    /usr/bin/killall -HUP syslogd
endscript
}

```

In this example, `/var/log/messages` is rotated after it reaches 100KB in size. Five backups are kept, and when the oldest backup ages out, it is mailed to `logsave@pinguino`. The `postrotate` introduces a script that restarts the `syslogd` daemon after the rotation is complete, by sending it the HUP signal. The `endscript` statement is required to terminate the script and is also required if a `prerotate` script is present. See the `logrotate` man page for more complete information.

Scan log files for notable activity

Log files entries are usually time stamped and contain the hostname of the reporting process, along with the process name. Listing 25 shows a few lines from `/var/log/messages`, containing entries from `gconfd`, `ntpd`, `init`, and `yum`.

Listing 25. Sample log file entries

```

Jul  5 15:28:24 lyrebird gconfd (root-2832): Exiting
Jul  5 15:31:06 lyrebird ntpd[2063]: synchronized to 87.98.219.90,
stratum 2
Jul  5 15:31:06 lyrebird ntpd[2063]: kernel time sync status change 0001
Jul  5 15:31:24 lyrebird init: Trying to re-exec init
Jul  5 15:31:24 lyrebird yum: Updated: libselinux.i386 2.0.14-2.fc7
Jul  5 15:31:24 lyrebird yum: Updated: libsemanage.i386 2.0.3-4.fc7
Jul  5 15:31:25 lyrebird yum: Updated: cups-libs.i386 1.2.11-2.fc7
Jul  5 15:31:25 lyrebird yum: Updated: libXfont.i386 1.2.9-2.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager.i386 0.6.5-7.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager-glib.i386
0.6.5-7.fc7

```

You can scan log files using a pager, such as `less`, or search for specific entries (such as kernel messages from host `lyrebird`) using `grep` as shown in Listing 26.

Listing 26. Scanning log files

```

[root@lyrebird ~]# less /var/log/messages
[root@lyrebird ~]# grep "lyrebird kernel" /var/log/messages | tail -n 9
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HCI socket layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP ver 2.8
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM TTY layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM ver 1.8

```

```
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HIDP (Human Interface
Emulation) ver 1.2
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized drm 1.1.0 20060810
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized i915 1.6.0 20060119
on minor 0
```

Monitor log files

Occasionally you may need to monitor log files for events. For example, you might be trying to catch an infrequently occurring event at the time it happens. In such a case, you can use the `tail` command with the `-f` option to *follow* the log file. Listing 27 shows an example.

Listing 27. Following log file updates

```
[root@lyrebird ~]# tail -n 1 -f /var/log/messages
Jul  6 15:16:26 lyrebird syslogd 1.4.2: restart.
Jul  6 15:16:26 lyrebird kernel: klogd 1.4.2, log source = /proc/kmsg
started.
Jul  6 15:19:35 lyrebird yum: Updated: samba-common.i386 3.0.25b-2.fc7
Jul  6 15:19:35 lyrebird yum: Updated: procps.i386 3.2.7-14.fc7
Jul  6 15:19:36 lyrebird yum: Updated: samba-client.i386 3.0.25b-2.fc7
Jul  6 15:19:37 lyrebird yum: Updated: libsmbclient.i386 3.0.25b-2.fc7
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Received signal 15, shutting
down cleanly
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Exiting
Jul  6 15:19:57 lyrebird yum: Updated: bluez-gnome.i386 0.8-1.fc7
```

Track down problems reported in log files

When you find problems in log files, you will want to note the time, the hostname, and the process that generated the problem. If the message identifies the problem specifically enough for you to resolve it, you are done. If not, you might need to update `syslog.conf` to specify that more messages be logged for the appropriate facility. For example, you might need to show informational messages instead of warning messages or even debug level messages. Your application may have additional facilities that you can use.

Finally, if you need to put marks in the log file to help you know what messages were logged at what stage of your debugging activity, you can use the `logger` command from a terminal window or shell script to send a message of your choice to the syslog daemon for logging according to the rules in `syslog.conf`.

Section 5. Scheduling jobs

This section covers material for topic 1.111.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use the `cron` or `anacron` commands to run jobs at regular intervals
- Use the `at` command to run jobs at a specific time
- Manage cron and at jobs
- Configure user access to the cron and at services

In the previous section, you learned about the `logrotate` command and saw the need to run it periodically. You will see the same need to run commands regularly in the next two sections on backup and network time services. These are only some of the many administrative tasks that have to be done frequently and regularly. In this section, you learn about the tools that are used to automate periodic job scheduling and also the tools used to run a job at some specific time.

Run jobs at regular intervals

Running jobs at regular intervals is managed by the *cron* facility, which consists of the `crond` daemon and a set of tables describing what work is to be done and with what frequency. The daemon wakes up every minute and checks the crontabs to determine what needs to be done. Users manage crontabs using the `crontab` command. The `crond` daemon is usually started by the `init` process at system startup.

To keep things simple, let's suppose that you want to run the command shown in Listing 28 on a regular basis. This command doesn't actually do anything except report the day and the time, but it illustrates how to use `crontab` to set up cron jobs, and we'll know when it was run from the output. Setting up crontab entries requires a string with escaped shell metacharacters, so it is best done with simple commands and parameters, so in this example, the `echo` command will be run from within a script `/home/ian/mycrontab.sh`, which takes no parameters. This saves some careful work with escape characters.

Listing 28. A simple command example.

```
[ian@lyrebird ~]$ cat mycrontest.sh
#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
[ian@lyrebird ~]$ ./mycrontest.sh
It is now 18:37:42 on Friday
```

Creating a crontab

To create a crontab, you use the `crontab` command with the `-e` (for "edit") option. This will open the `vi` editor unless you have specified another editor in the `EDITOR` or `VISUAL` environment variable.

Each crontab entry contains six fields:

1. Minute
2. Hour
3. Day of the month
4. Month of the year
5. Day of the week
6. String to be executed by `sh`

Minutes and hours range from 0-59 and 0-12, respectively, while day or month and month of year range from 1-31 and 1-12, respectively. Day of week ranges from 0-6 with 0 being Sunday. Day of week may also be specified as `sun`, `mon`, `tue`, and so on. The sixth field is everything after the fifth field, is interpreted as a string to pass to `sh`. A percent sign (%) will be translated to a newline, so if you want a % or any other special character, precede it with a backslash (\). The line up to the first % is passed to the shell, while any line(s) after the % are passed as standard input.

The various time-related fields can specify an individual value, a range of values, such as 0-10 or `sun-wed`, or a comma-separated list of individual values and ranges. So a somewhat artificial crontab entry for our example command might be as shown in Listing 29.

Listing 29. A simple crontab example.

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
```

In this example, our command is executed at the 0th, 20th, and 40th minutes (every 20 minutes), for the hours between 10 P.M. and midnight on Fridays and Saturdays during July. See the man page for `crontab(5)` for details on additional ways to specify times.

What about the output?

You may be wondering what happens to any output from the command. Most

commands designed for use with the cron facility will log output using the syslog facility that you learned about in the previous section. However any output that is directed to stdout will be mailed to the user. Listing 30 shows the output you might receive from our example command.

Listing 30. Mailed cron output

```
From ian@lyrebird.raleigh.ibm.com Fri Jul 6 23:00:02 2007
Date: Fri, 6 Jul 2007 23:00:01 -0400
From: root@lyrebird.raleigh.ibm.com (Cron Daemon)
To: ian@lyrebird.raleigh.ibm.com
Subject: Cron <ian@lyrebird> /home/ian/mycronetest.sh
Content-Type: text/plain; charset=UTF-8
Auto-Submitted: auto-generated
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/home/ian>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=ian>
X-Cron-Env: <USER=ian>

It is now 23:00:01 on Friday
```

Where is my crontab?

The crontab that you created with the `crontab` command is stored in `/etc/spool/cron` under the name of the user who created it. So the above crontab is stored in `/etc/spool/cron/ian`. Given this, you will not be surprised to learn that the `crontab` command, like the `passwd` command you learned about earlier, is an `suid` program that runs with root authority.

`/etc/crontab`

In addition to the user crontab files in `/var/spool/cron`, `cron` also checks `/etc/crontab` and files in the `/etc/cron.d` directory. These system crontabs have one additional field between the fifth time entry (day) and the command. This additional field specifies the user for whom the command should be run, normally `root`. A `/etc/crontab` might look like the example in Listing 31.

Listing 31. `/etc/crontab`

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

In this example, the real work is done by the `run-parts` command, which runs

scripts from `/etc/cron.hourly`, `/etc/cron.daily`, and so on; `/etc/crontab` simply controls the timing of the recurring jobs. Note that the commands here all run as root. Note also that the crontab can include shell variables assignments that will be set before the commands are run.

Anacron

The cron facility works well for systems that run continuously. For systems that may be turned off much of the time, such as laptops, another facility, the *anacron* (for "anachronistic cron") can handle scheduling of the jobs usually done daily, weekly, or monthly by the cron facility. Anacron does not handle hourly jobs.

Anacron keeps timestamp files in `/var/spool/anacron` to record when jobs are run. When anacron runs, it checks to see if the required number of days has passed since the job was last run and runs it if necessary. The table of jobs for anacron is stored in `/etc/anacrontab`, which has a slightly different format than `/etc/crontab`. As with `/etc/crontab`, `/etc/anacrontab` may contain environment settings. Each job has four fields.

1. period
2. delay
3. job-identifier
4. command

The period is a number of days, but may be specified as `@monthly` to ensure that a job runs only once per month, regardless of the number of days in the month. The delay is the number of minutes to wait after the job is due to run before actually starting it. You can use this to prevent a flood of jobs when a system first starts. The job identifier can contain any non-blank character except slashes (`/`).

Both `/etc/crontab` and `/etc/anacrontab` are updated by direct editing. You do not use the `crontab` command to update these files or files in the `/etc/cron.d` directory.

Run jobs at specific times

Sometimes you may need to run a job just once, rather than regularly. For this you use the `at` command. The commands to be run are read from a file specified with the `-f` option, or from stdin if `-f` is not used. The `-m` option sends mail to the user even if there is no stdout from the command. The `-v` option will display the time at which the job will run before reading the job. The time is also displayed in the output. Listing 32 shows an example of running the `mycrontest.sh` script that you used earlier. Listing 33 shows the output that is mailed back to the user after the job runs.

Notice that it is somewhat more compact than the corresponding output from the cron job.

Listing 32. Using the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh -v 10:25
Sat Jul 7 10:25:00 2007

job 5 at Sat Jul 7 10:25:00 2007
```

Listing 33. Job output from at

```
From ian@lyrebird.raleigh.ibm.com Sat Jul 7 10:25:00 2007
Date: Sat, 7 Jul 2007 10:25:00 -0400
From: Ian Shields <ian@lyrebird.raleigh.ibm.com>
Subject: Output from your job 5
To: ian@lyrebird.raleigh.ibm.com

It is now 10:25:00 on Saturday
```

Time specifications can be quite complex. Listing 34 shows a few examples. See the man page for `at` or the file `/usr/share/doc/at/timespec` or a file such as `/usr/share/doc/at-3.1.10/timespec`, where 3.1.10 in this example is the version of the `at` package.

Listing 34. Time values with the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh 10pm tomorrow
job 14 at Sun Jul 8 22:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 tuesday
job 15 at Tue Jul 10 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 july 11
job 16 at Wed Jul 11 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 next week
job 17 at Sat Jul 14 02:00:00 2007
```

The `at` command also has a `-q` option. Increasing the queue increases the nice value for the job. There is also a `batch` command, which is similar to the `at` command except that jobs are run only when the system load is low enough. See the man pages for more details on these features.

Manage scheduled jobs

Listing scheduled jobs

You can manage your cron and `at` jobs. Use the `crontab` command with the `-l` option to list your crontab, and use the `atq` command to display the jobs you have queued using the `at` command, as shown in Listing 35.

Listing 35. Displaying scheduled jobs

```
[ian@lyrebird ~]$ crontab -l
0,20,40 22-23 * 7 fri-sat /home/ian/mycronstest.sh
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
```

If you want to review the actual command scheduled for execution by `at`, you can use the `at` command with the `-c` option and the job number. You will notice that most of the environment that was active at the time the `at` command was issued is saved with the scheduled job. Listing 36 shows part of the output for job 15.

Listing 36. Using `at -c` with a job number

```
#!/bin/sh
# atrun uid=500 gid=500
# mail ian 0
umask 2
HOSTNAME=lyrebird.raleigh.ibm.com; export HOSTNAME
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=9.67.219.151\ 3210\ 22; export SSH_CLIENT
SSH_TTY=/dev/pts/5; export SSH_TTY
USER=ian; export USER
...
HOME=/home/ian; export HOME
LOGNAME=ian; export LOGNAME
...
cd /home/ian || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
${SHELL:-/bin/sh} << `(dd if=/dev/urandom count=200 bs=1 \
  2>/dev/null|LC_ALL=C tr -d -c '[:alnum:]')`

#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
```

Note that the contents of our script file have been copied in as a here-document that will be executed by the shell specified by the `SHELL` variable or `/bin/sh` if the `SHELL` variable is not set. See the tutorial [LPI exam 101 prep, Topic 103: GNU and UNIX commands](#) if you need to review here-documents.

Deleting scheduled jobs

You can delete all scheduled cron jobs using the `crontab` command with the `-r` option as illustrated in Listing 37.

Listing 37. Displaying and deleting cron jobs

```
[ian@lyrebird ~]$ crontab -l
```

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycronetest.sh
[ian@lyrebird ~]$ crontab -r
[ian@lyrebird ~]$ crontab -l
no crontab for ian
```

To delete system cron or anacron jobs, edit `/etc/crontab`, `/etc/anacrontab`, or edit or delete files in the `/etc/cron.d` directory.

You can delete one or more jobs that were scheduled with the `at` command by using the `atrm` command with the job number. Multiple jobs should be separated by spaces. Listing 38 shows an example.

Listing 38. Displaying and removing jobs with `atq` and `atrm`

```
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
[ian@lyrebird ~]$ atrm 16 14 15
[ian@lyrebird ~]$ atq
17      Sat Jul 14 02:00:00 2007 a ian
```

Configure user access to job scheduling

If the file `/etc/cron.allow` exists, any non-root user must be listed in it in order to use `crontab` and the cron facility. If `/etc/cron.allow` does not exist, but `/etc/cron.deny` does exist, a non-root user who is listed in it cannot use `crontab` or the cron facility. If neither of these files exists, only the super user will be allowed to use this command. An empty `/etc/cron.deny` file allows all users to use the cron facility and is the default.

The corresponding `/etc/at.allow` and `/etc/at.deny` files have similar effects for the `at` facility.

Section 6. Data backup

This section covers material for topic 1.111.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Plan a backup strategy

- Dump a raw device to a file or restore a raw device from a file
- Perform partial and manual backups
- Verify the integrity of backup files
- Restore filesystems partially or fully from backups

Plan a backup strategy

Having a good backup is a necessary part of system administration, but deciding what to back up and when and how can be complex. Databases, such as customer orders or inventory, are usually critical to a business and many include specialized backup and recovery tools that are beyond the scope of this tutorial. At the other extreme, some files are temporary in nature and no backup is needed at all. In this section, we focus on system files and user data and discuss some of the considerations, methods, and tools for backup of such data.

There are three general approaches to backup:

1. A *full* backup is a complete backup, usually of a whole filesystem, directory, or group of related files. This takes the longest time to create, so it is usually used with one of the other two approaches.
2. A *differential* or *cumulative* backup is a backup of all things that have changed since the last full backup. Recovery requires the last full backup plus the latest differential backup.
3. An *incremental* backup is a backup of only those changes since the last incremental backup. Recovery requires the last full backup plus all of the incremental backups (in order) since the last full backup.

What to back up

When deciding what to back up, you should consider how volatile the data is. This will help you determine how often it should be backed up. Similarly, critical data should be backed up more often than non-critical data. Your operating system will probably be relatively easy to rebuild, particularly if you use a common image for several systems, although the files that customize each system would be more important to back up.

For programming staff, it may be sufficient to keep backups of repositories such as CVS repositories, while individual programmers' sandboxes may be less important. Depending on how important mail is to your operation, it may suffice to have infrequent mail backups, or it may be necessary to be able to recover mail to the

most recent date possible. You may want to keep backups of system cron files, but may not be so concerned about scheduled jobs for individual users.

The Filesystem Hierarchy Standard provides a classification of data that may help you with your backup choices. For details, see the tutorial [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#).

Once you have decided what to back up, you need to decide how often to do a full backup and whether to do differential or incremental backups in between those full backups. Having made those decisions, the following suggestions will help you choose appropriate tools.

Automating backups

In the previous section, you learned how to schedule jobs, and the cron facility is ideal for helping to automate the scheduling of your backups. However, backups are frequently made to removable media, particularly tape, so operator intervention is probably going to be needed. You should create and use backup scripts to ensure that the backup process is as automatic and repeatable as possible.

Dump and restore raw devices

One way to make a full backup of a filesystem is to make an image of the partition on which it resides. A *raw device*, such as `/dev/hda1` or `/dev/sda2`, can be opened and read as a sequential file. Similarly, it can be written from a backup as a sequential file. This requires no knowledge on the part of the backup tool as to the filesystem layout, but does require that the restore be done to space that is at least as large as the original. Some tools that handle raw devices are *filesystem aware*, meaning that they understand one or more of the Linux filesystems. These utilities can dump from a raw device but do not dump unused parts of the partition. They may or may not require restoration to the same or larger sized partition. The `dd` command is an example of the first type, while the `dump` command is an example of the second type that is specific to the `ext2` and `ext3` filesystems.

The `dd` command

In its simplest form, the `dd` command copies an input file to an output file, where either file may be a raw device. For backing up a raw device, such as `/dev/hda1` or `/dev/sda2`, the input file would be a raw device. Ideally, the filesystem on the device should be unmounted, or at least mounted read only, to ensure that data does not change during the backup. Listing 39 shows an example.

Listing 39. Backup a partition using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 of=backup-1
```

```
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 49.3103 s, 21.2 MB/s
```

The `if` and `of` parameters specify the input and output files respectively. In this example, the input file is a raw device, `dev/sda3`, and the output file is a file, `backup-1`, in the root user's home directory. To dump the file to tape or floppy disk, you would specify something like `of=/dev/fd0` or `of=/dev/st0`.

Note that 1,044,610,560 bytes of data was copied and the output file is indeed that large, even though only about 3% of this particular partition is actually used. Unless you are copying to a tape with hardware compression, you will probably want to compress the data. Listing 40 shows one way to accomplish this, along with the output of `ls` and `df` commands, which show you the file sizes and the usage percentage of the filesystem on `/dev/sda3`.

Listing 40. Backup with compression using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 | gzip > backup-2
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 117.723 s, 8.9 MB/s
[root@lyrebird ~]# ls -l backup-[12]
-rw-r--r-- 1 root root 1044610560 2007-07-08 15:17 backup-1
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
[root@lyrebird ~]# df -h /dev/sda3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        972M   28M  944M   3% /grubfile
```

The `gzip` compression reduced the file size to about 20% of the uncompressed size. However, unused blocks may contain arbitrary data, so even the compressed backup may be much larger than the total data on the partition.

If you divide the size by the number of records processed by `dd`, you will see that `dd` is writing 512-byte blocks of data. When copying to a raw output device such as tape, this can result in a very inefficient operation, so `dd` can read or write data in much larger blocks. Specify the `obs` option to change the output size or the `ibs` option to specify the input block size. You can also specify just `bs` to set both input and output block sizes to a common value.

If you need multiple tapes or other removable storage to store your backup, you will need to break it into smaller pieces using a utility such as `split`.

If you need to skip blocks such as disk or tape labels, you can do so with `dd`. See the man page for examples.

Besides just copying data, the `dd` command can do several conversions, such as between ASCII and EBCDIC, between big-endian and little-endian, or between variable-length data records and fixed-length data records. Obviously these

conversions are likely to be useful when copying real files rather than raw devices. Again, see the man page for details.

The dump command

The `dump` command can be used for full, differential, or incremental backups on ext2 or ext3 filesystems. Listing 41 shows an example.

Listing 41. Backup with compression using dump

```
[root@lyrebird ~]# dump -0 -f backup-4 -j -u /dev/sda3
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-4
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzlib)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 12285 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:47:48 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-4
DUMP: Volume 1 completed at: Sun Jul  8 16:47:57 2007
DUMP: Volume 1 took 0:00:09
DUMP: Volume 1 transfer rate: 819 kB/s
DUMP: Volume 1 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: 12260 blocks (11.97MB) on 1 volume(s)
DUMP: finished in 9 seconds, throughput 1362 kBytes/sec
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Date this dump completed: Sun Jul  8 16:47:57 2007
DUMP: Average transfer rate: 819 kB/s
DUMP: Wrote 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[2-4]
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
-rw-r--r-- 1 root root 266932272 2007-07-08 15:44 backup-3
-rw-r--r-- 1 root root  7554939 2007-07-08 16:47 backup-4
```

In this example, `-0` specifies the *dump level* which is an integer, historically from 0 to 9, where 0 specifies a full dump. The `-f` option specifies the output file, which may be a raw device. Specify `-` to direct the output to stdout. The `-j` option specifies compression, with a default level of 2, using bzlib compression. You can use the `-z` option to specify zlib compression if you prefer. The `-u` option causes the record of dump information, normally `/etc/dumpdates`, to be updated. Any parameters after the options represent a file or list of files, where the file may also be a raw device, as in this example. Notice how much smaller the backup is when the backup program is aware of the filesystem structure and can avoid the saving of unused blocks on the device.

If output is to a device such as tape, the `dump` command will prompt for another volume as each volume is filled. You can also provide multiple file names separated by commas. For example, if you wanted an unattended dump that required two tapes, you could load the tapes on `/dev/st0` and `/dev/st1`, schedule the `dump` command specifying both tapes as output, and go home to sleep.

When you specify a dump level greater than 0, an incremental dump is performed of all files that are new or have changed since the last dump at a lower level was taken. So a dump at level 1 will be a differential dump, even if a dump at level 2 or higher has been taken in the meantime. Listing 42 shows the result of updating the time stamp of an existing file on /dev/sda3 and creating a new file, then taking a dump at level 2. After that, another new file is created and a dump at level 1 is taken. The information from /etc/dumpdates is also shown. For brevity, part of the second dump output has been omitted.

Listing 42. Backup with compression using dump

```
[root@lyrebird ~]# dump -2 -f backup-5 -j -u /dev/sda3
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-5
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzlib)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 91 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:55:47 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-5
DUMP: Volume 1 completed at: Sun Jul  8 16:55:47 2007
DUMP: 90 blocks (0.09MB) on 1 volume(s)
DUMP: finished in less than a second
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date this dump completed: Sun Jul  8 16:55:47 2007
DUMP: Average transfer rate: 0 kB/s
DUMP: Wrote 90kB uncompressed, 15kB compressed, 6.000:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# echo "This data is even newer" >/grubfile/newerfile
[root@lyrebird ~]# dump -1 -f backup-6 -j -u -A backup-6-toc /dev/sda3
DUMP: Date of this level 1 dump: Sun Jul  8 17:08:18 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-6
...
DUMP: Wrote 100kB uncompressed, 16kB compressed, 6.250:1
DUMP: Archiving dump to backup-6-toc
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[4-6]
-rw-r--r-- 1 root root 7554939 2007-07-08 16:47 backup-4
-rw-r--r-- 1 root root  16198 2007-07-08 16:55 backup-5
-rw-r--r-- 1 root root  16560 2007-07-08 17:08 backup-6
[root@lyrebird ~]# cat /etc/dumpdates
/dev/sda3 0 Sun Jul  8 16:47:47 2007 -0400
/dev/sda3 2 Sun Jul  8 16:55:46 2007 -0400
/dev/sda3 1 Sun Jul  8 17:08:18 2007 -0400
```

Notice that backup-6 is, indeed, larger than backup 5. The level 1 dump illustrates the use of the `-A` option to create a table of contents that can be used to determine if a file is on an archive without actually mounting the archive. This is particularly useful with tape or other removable archive volumes. You will see these examples again when we discuss restoring data later in this section.

The `dump` command can dump files or subdirectories, but you cannot update

/etc/dumpdates and only level 0, of full dump, is supported.

Listing 43 illustrates the `dump` command dumping a directory, `/usr/include/bits`, and its contents to floppy disk. In this case, the dump will not fit on a single floppy, so a new volume is required. The prompt and response are shown in bold.

Listing 43. Backup a directory to multiple volumes using dump

```
[root@lyrebird ~]# dump -0 -f /dev/fd0 /usr/include/bits
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Dumping /dev/sdb9 (/ (dir usr/include/bits)) to /dev/fd0
DUMP: Label: /
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 2790 blocks.
DUMP: Volume 1 started with block 1 at: Mon Jul  9 16:03:30 2007
DUMP: dumping (Pass III) [directories]
DUMP: End of tape detected
DUMP: Closing /dev/fd0
DUMP: Volume 1 completed at: Mon Jul  9 16:04:49 2007
DUMP: Volume 1 1470 blocks (1.44MB)
DUMP: Volume 1 took 0:01:19
DUMP: Volume 1 transfer rate: 18 kB/s
DUMP: Change Volumes: Mount volume #2
DUMP: Is the new volume mounted and ready to go?: ("yes" or "no") y
DUMP: Volume 2 started with block 1441 at: Mon Jul  9 16:05:10 2007
DUMP: Volume 2 begins with blocks from inode 2
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing /dev/fd0
DUMP: Volume 2 completed at: Mon Jul  9 16:06:28 2007
DUMP: Volume 2 1410 blocks (1.38MB)
DUMP: Volume 2 took 0:01:18
DUMP: Volume 2 transfer rate: 18 kB/s
DUMP: 2850 blocks (2.78MB) on 2 volume(s)
DUMP: finished in 109 seconds, throughput 26 kBytes/sec
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Date this dump completed: Mon Jul  9 16:06:28 2007
DUMP: Average transfer rate: 18 kB/s
DUMP: DUMP IS DONE
```

If you back up to tape, remember that the tape will usually be rewound after each job. Devices with a name like `/dev/st0` or `/dev/st1` automatically rewind. The corresponding non-rewind equivalent devices are `/dev/nst0` and `/dev/nst1`. In any event, you can always use the `mt` command to perform magnetic tape operations such as forward spacing over files and records, back spacing, rewinding, and writing EOF marks. See the man pages for `mt` and `st` for additional information.

If you select the dump levels judiciously, you can minimize the number of archives you need to restore to any particular level. See the man pages for `dump` for a suggestion based on the Towers of Hanoi puzzle.

As with the `dd` command, there are many options that are not covered in this brief introduction. See the man pages for more details.

Partial and manual backups

So far, you have learned about tools that work well for backing up whole filesystems. Sometimes your backup needs to target selected files or subdirectories without backing up the whole filesystem. For example, you might need a weekly backup of most of your system, but daily backups of your mail files. Two other programs, `cpio` and `tar`, are more commonly used for this purpose. Both can write archives to files or to devices such as tape or floppy disk, and both can restore from such archives. Of the two, `tar` is more commonly used today, possibly because it handles complete directories better, and GNU `tar` supports both `gzip` and `bzip` compression.

Using `cpio`

The `cpio` command operates in *copy-out* mode to create an archive, *copy-in* mode to restore an archive, or *copy-pass* mode to copy a set of files from one location to another. You use the `-o` or `--create` option for copy-out mode, the `-i` or `--extract` option for copy-in mode, and the `-p` or `--pass-through` option for copy-pass mode. Input is a list of files provided on `stdin`. Output is either to `stdout` or to a device or file specified with the `-f` or `--file` option.

Listing 44 shows how to generate a list of files using the `find` command. Note the use of the `-print0` option on `find` to generate null-terminate strings for file names, and the corresponding `--null` option on `cpio` to read this format. This will correctly handle file names that have embedded blank or newline characters.

Listing 44. Back up a home directory using `cpio`

```
[root@lyrebird ~]# find ~ian -depth -print0 | cpio --null -o
>backup-cpio-1
18855 blocks
```

If you'd like to see the files listed as they are archived, add the `-v` option to `cpio`.

As with other commands that can archive to tape, the block size may be specified. For details on this and other options, see the man page.

Using `tar`

The `tar` (originally from *Tape ARchive*) creates an archive file, or *tarfile* or *tarball*, from a set of input files or directories; it also restores files from such an archive. If a directory is given as input to `tar`, all files and subdirectories are automatically included, which makes `tar` very convenient for archiving subtrees of your directory structure.

As with the other archiving commands we have discussed, output can be to a file, a

device such as tape or diskette, or stdout. The output location is specified with the `-f` option. Other common options are `-c` to create an archive, `-x` to extract an archive, `-v` for verbose output, which lists the files being processed, `-z` to use gzip compression, and `-j` to use bzip2 compression. Most `tar` options have a short form using a single hyphen and a long form using a pair of hyphens. The short forms are illustrated here. See the man pages for the long form and for additional options.

Listing 45 shows how to create a backup of the system cron jobs using `tar`.

Listing 45. Backup of system cron jobs using tar

```
[root@lyrebird ~]# tar -czvf backup-tar-1 /etc/*crontab /etc/cron.d
tar: Removing leading `/' from member names
/etc/anacrontab
/etc/crontab
/etc/cron.d/
/etc/cron.d/sa-update
/etc/cron.d/smolt
```

In the first line of output, you are told that `tar` will remove the leading slash (/) from member names. This allows files to be restored to some other location for verification before replacing system files. It is a good idea to avoid mixing absolute path names with relative path names when creating an archive, since all will be relative when restoring from the archive.

The `tar` command can append additional files to an archive using the `-r` or `--append` option. This may cause multiple copies of a file in the archive. In such a case, the *last* one will be restored during a restore operation. You can use the `--occurrence` option to select a specific file among multiples. If the archive is on a regular filesystem instead of tape, you may use the `-u` or `--update` option to update an archive. This works like appending to an archive, except that the time stamps of the files in the archive are compared with those on the filesystem, and only files that have been modified since the archived version are appended. As mentioned, this does not work for tape archives.

As with the other commands you have studied here, there are many options that are not covered in this brief introduction. See the man or info pages for more details.

Backup file integrity

Backup file integrity is extremely important. There is no point in having a backup if it is bad. A good backup strategy also involves checking your backups.

The first step to ensuring backup integrity is to ensure that you have properly captured the data you are backing up. If the filesystem is unmounted or mounted read only, this is usually straightforward as the data you are backing up cannot

change during your backup. If you must back up filesystems, directories, or files that are subject to modification while you are taking the backup, you should verify that no changes have been made during your backup. If changes were made, you will need to have a strategy for capturing them, either by repeating the backup, or perhaps by replacing or superseding the affected files in your backup. Needless to say, this will also affect your restore procedures.

Assuming you took good backups, you will periodically need to verify your backups. One way is to restore the backup to a spare volume and verify that it matches what you backed up. This is easiest to do right before you allow updates on the filesystem you are backing up. If you back up to media such as CD or DVD, you may be able to use the `diff` command as part of your backup procedure to ensure that your backup is good. Remember that even good backups can deteriorate in storage, so you should check periodically, even if you do verify at the time of backup. Keeping digests using programs such as `md5sum` or `sha1sum` is also a good check on the integrity of a backup file.

Restore filesystems from backups

A counterpart to backing up files is the ability to restore them when needed. Occasionally you will want to restore an entire filesystem, but it is far more common to need to restore only specific files or perhaps a set of directories. Almost always you will restore to some temporary space and verify that what you have restored is indeed what you want and is consistent with the current state of your system before actually making the restored files live.

A related issue is the need to verify that the items you want happen to be on a particular backup, as often happens when a user needs access to a version of a file that was modified or perhaps deleted "sometime in the last week or two." With these thoughts in mind, let's look at some of the restoration options.

Restoring a dd archive

Recall that the `dd` command was not filesystem aware, so you will need to restore a dump of a partition to find out what is on it. Listing 46 shows how to restore the partition that was dumped back in Listing 39 to a partition, `/dev/sdc7`, that was specially created on a removable USB drive just for this purpose.

Listing 46. Restoring a partition using dd

```
[root@lyrebird ~]# dd if=backup-1 of=/dev/sdc7
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 44.0084 s, 23.7 MB/s
```

Recall that we added some files to the filesystem on `/dev/sda3` after this backup was taken. If you mount the newly restore partition and compare it with the original, you will see that this is indeed the case, as shown in Listing 47. Note that the file whose timestamp was updated using `touch` is not shown here, as you would expect.

Listing 47. Comparing the restored partition with current state

```
[root@lyrebird ~]# mount /dev/sdc7 /mnt/temp-dd/
[root@lyrebird ~]# diff -rq /grubfile/ /mnt/temp-dd/
Only in /grubfile/: newerfile
Only in /grubfile/: newfile
```

Restoring a dump archive using restore

Recall that our final use of `dump` was a differential backup and that we created a table of contents. Listing 48 shows how to use `restore` to check the files in the archive created by `dump`, using the archive itself (`backup-5`) or the table of contents (`backup-6-toc`).

Listing 48. Checking the contents of archives

```
[root@lyrebird ~]# restore -t -f backup-5
Dump tape is compressed.
Dump   date: Sun Jul  8 16:55:46 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 2 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
[root@lyrebird ~]# restore -t -A backup-6-toc
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
Starting inode numbers by volume:
Volume 1: 2
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
      17     ./newerfile
```

The `restore` command can also compare the contents of an archive with the contents of the filesystem using the `-C` option. In Listing 49 we updated `newerfile` and then compared the backup with the filesystem.

Listing 49. Comparing an archive with a filesystem using restore

```
[root@lyrebird ~]# echo "something different" >/grubfile/newerfile
[root@lyrebird ~]# restore -C -f backup-6
Dump tape is compressed.
```

```
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
fileSYS = /grubfile
./newerfile: size has changed.
Some files were modified!  1 compare errors
```

The `restore` command can restore interactively or automatically. Listing 50 shows how to restore `newerfile` to root's home directory (so you could examine it before replacing the updated file if needed), then replace the updated file with the backup copy. This example illustrates interactive restoration.

Listing 50. Restoring a file using restore

```
[root@lyrebird ~]# restore -i -f backup-6
Dump tape is compressed.
restore > ?
Available commands are:
  ls [arg] - list directory
  cd arg - change directory
  pwd - print current directory
  add [arg] - add `arg' to list of files to be extracted
  delete [arg] - delete `arg' from list of files to be extracted
  extract - extract requested files
  setmodes - set modes of requested directories
  quit - immediately exit program
  what - list dump header information
  verbose - toggle verbose flag (useful with ``ls'')
  prompt - toggle the prompt display
  help or `?' - print this list
If no `arg' is supplied, the current directory is used
restore > ls new*
newerfile
newfile
restore > add newerfile
restore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume # (none if no more volumes): 1
set owner/mode for '.'? [yn] y
restore > q
[root@lyrebird ~]# mv -f newerfile /grubfile
```

Restoring a cpio archive

The `cpio` command in copy-in mode (option `-i` or `--extract`) can list the contents of an archive or restore selected files. When you list the files, specifying the `--absolute-filenames` option reduces the number of extraneous messages that `cpio` will otherwise issue as it strips any leading `/` characters from each path that has one. Partial output from listing our previous archive is shown in Listing 51.

Listing 51. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio -id --list --absolute-filenames <backup-cpio-1
```

```
/home/ian/.gstreamer-0.10/registry.i686.xml
/home/ian/.gstreamer-0.10
/home/ian/.Trash/gnome-terminal.desktop
/home/ian/.Trash
/home/ian/.bash_profile
```

Listing 52 shows how to restore all the files with "samp" in their path name or file name. The output has been piped through `uniq` to reduce the number of "Removing leading '/' ..." messages. You must specify the `-d` option to create directories; otherwise, all files are created in the current directory. Furthermore, `cpio` will not replace any newer files on the filesystem with archive copies unless you specify the `-u` or `--unconditional` option.

Listing 52. Restoring selected files using `cpio`

```
[root@lyrebird ~]# cpio -ivd "*samp*" < backup-cpio-1 2>&1 |uniq
cpio: Removing leading `/' from member names
home/ian/crontab.samp
cpio: Removing leading `/' from member names
home/ian/sample.file
cpio: Removing leading `/' from member names
18855 blocks
```

Restoring a tar archive

The `tar` command can also compare archives with the current filesystem as well as restore files from archives. Use the `-d`, `--compare`, or `--diff` option to perform comparisons. The output will show files whose contents differ as well as files whose time stamps differ. Listing 53 shows verbose output (using option `-v`), from a comparison of the file created earlier and the files in `/etc` after `/etc/crontab` has been touched to alter its time stamp. The option `directory /` instructs `tar` to perform the comparison starting from the root directory rather than the current directory.

Listing 53. Comparing archives and files using `tar`

```
[root@lyrebird ~]# touch /etc/crontab
[root@lyrebird ~]# tar --diff -vf backup-tar-1 --directory /
etc/anacrontab
etc/crontab
etc/crontab: Mod time differs
etc/cron.d/
etc/cron.d/sa-update
etc/cron.d/smolt
```

Listing 54 shows how to extract just `/etc/crontab` and `/etc/anacrontab` into the current directory.

Listing 54. Extracting archive files using `tar`

```
[root@lyrebird ~]# tar -xzvf backup-tar-1 "*tab"
```

```
etc/anacrontab
etc/crontab
```

Note that `tar`, in contrast to `cpio` creates the directory hierarchy for you automatically.

The next section of this tutorial shows you how to maintain system time.

Section 7. System time

This section covers material for topic 1.111.6 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

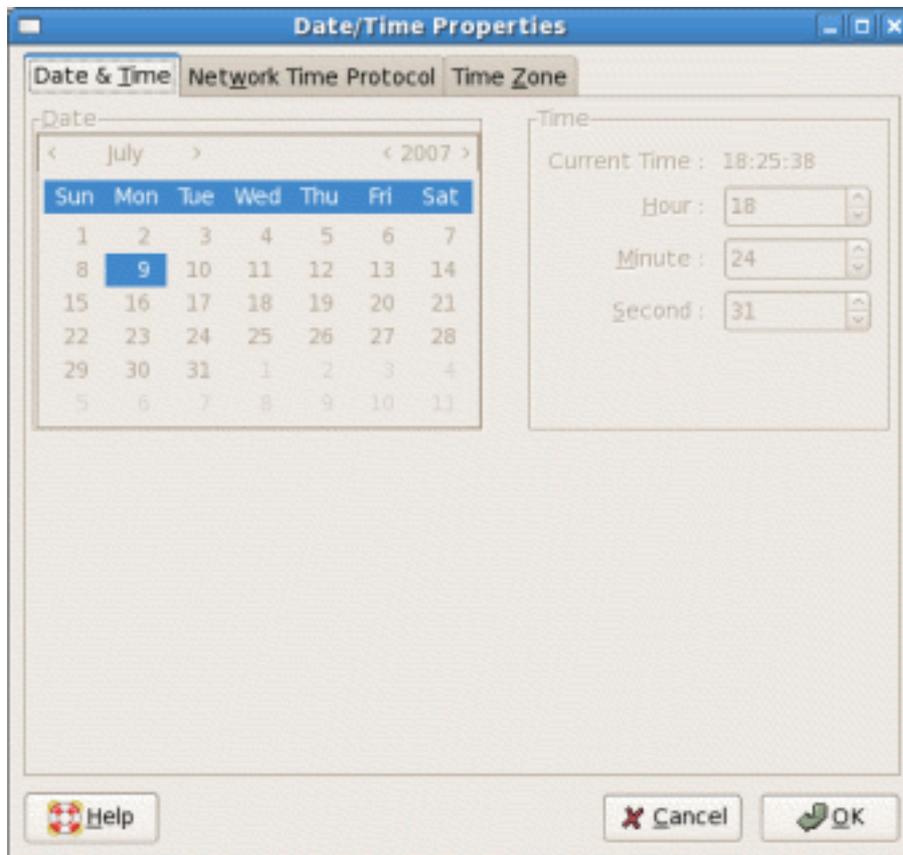
- Set the system date and time
- Set the BIOS clock to the correct UTC time
- Configure your time zone
- Configure the Network Time Protocol (NTP) service, including correcting for clock drift

Set the system date and time

System time on a Linux system is very important. You saw earlier how the cron and anacron facilities do things based on time, so they need an accurate time to base decisions on. Most of the backup and restore tools discussed in the previous section, along with development tools such as `make`, also depend on reliable time measurements. Most computers built since around 1980 include some kind of clock mechanism, and most built since 1984 or so have a persistent clock mechanism that keeps time even if the computer is turned off.

If you installed a Linux system graphically, you probably set the clock and chose a time zone suitable for your needs. You may have elected to use the Network Time Protocol (NTP) to set your clock, and you may or may not have elected to keep the system clock using Coordinated Universal Time or UTC. If you subsequently went to set the clock using graphical tools on a Fedora or Red Hat or similar system, you may have seen a dialog box like that in Figure 3.

Figure 3. Updating the date and time



Surprise! You can't actually set the clock yourself using this dialog. In this section you learn more about the difference between local clocks and NTP and how to set your system time.

No matter whether you live in New York, Budapest, Nakhodka, Ulan Bator, Bangkok, or Canberra, most of your Linux time computations are related to Coordinated Universal Time or UTC. If you run a dedicated Linux system, it is customary to keep the hardware clock set to UTC, but if you also boot another operating system such as Windows, you may need to set the hardware clock to local time. It really doesn't matter as far as Linux is concerned, except that there happen to be two different methods of keeping track of time zones internally in Linux, and if they don't agree, you can wind up with some odd time stamps on FAT filesystems, among other things. Listing 55 shows you how to use the `date` command to display the current date and time. The display is always in local time, even if your hardware clock keeps UTC time.

Listing 55. Displaying the current date and time

```
[root@lyrebird ~]# date;date -u
Mon Jul 9 22:40:01 EDT 2007
```

The `date` command supports a wide variety of possible output formats, some of which you already saw back in [Listing 28](#). See the man page for `date` if you'd like to learn more about the various date formats.

If you need to set the date, you can do this by providing a date and time as an argument. The required format is historical and is somewhat odd even to Americans and truly odd to the rest of the world. You must specify at least month, day, hour, and minute in MMDDhhmm format, and you may also append a two- or four-digit year (CCYY or YY) and optionally a period (.) followed by a two-digit number of seconds. Listing 56 shows an example that alters the system date by a little over a minute.

Listing 56. Setting the system date and time

```
[root@lyrebird ~]# date; date 0709221407;date
Mon Jul  9 23:12:37 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
```

Set the BIOS clock to UTC time

Your Linux system, along with most other current operating systems, actually has two clocks. The first is the hardware clock, sometimes called the Real Time Clock, RTC, or BIOS clock, which is usually tied to an oscillating quartz crystal that is accurate to within a few seconds per day. It is subject to variations such as ambient temperature. The second is the internal software clock, which is driven by counting system interrupts. It is subject to variations caused by high system load and interrupt latency. Nevertheless, your system typically reads the hardware clock at startup and from then on uses the software clock. The `date` command that you just learned about sets the software clock, not the hardware clock.

If you use the Network Time Protocol (NTP), you may possibly set the hardware clock when you first install the system and never worry about it again. If not, this part of the tutorial will show you how to display and set the hardware clock time.

You can use the `hwclock` command to display the current value of the hardware clock. Listing 57 shows the current value of both the system and hardware clocks.

Listing 57. System and hardware clock values

```
[root@lyrebird ~]# date;hwclock
Mon Jul  9 22:16:11 EDT 2007
Mon 09 Jul 2007 11:14:49 PM EDT -0.071616 seconds
```

Notice that the two values are different. You can synchronize the hardware clock

from the system clock using the `-w` or `--systohc` option of `hwclock`, and you can synchronize the system clock from the hardware clock using the `-s` or `--hctosys` option, as shown in Listing 58.

Listing 58. Setting the system clock from the hardware clock

```
[root@lyrebird ~]# date;hwclock;hwclock -s;date
Mon Jul  9 22:20:23 EDT 2007
Mon 09 Jul 2007 11:19:01 PM EDT  -0.414881 seconds
Mon Jul  9 23:19:02 EDT 2007
```

You may specify either the `--utc` or the `--localtime` option to have the system clock kept in UTC or local time. If no value is specified, the value is taken from the third line of `/etc/adjtime`.

The Linux kernel has a mode that copies the system time to the hardware clock every 11 minutes. This is off by default, but is turned on by NTP. Running anything that set the time the old fashioned way, such as `hwclock --hctosys`, turns it off, so it's a good idea to just let NTP do its work if you are using NTP. See the man page for `adjtimex` to find out how to check whether the clock is being updated every 11 minutes or not. You may need to install the `adjtimex` package as it is not always installed by default.

The `hwclock` command keeps track of changes made to the hardware clock in order to compensate for inaccuracies in the clock frequency. The necessary data points are kept in `/etc/adjtime`, which is an ASCII file. If you are not using the Network Time Protocol, you can use the `adjtimex` command to compensate for clock drift. Otherwise, the hardware clock will be adjusted approximately every 11 minutes by NTP. Besides showing whether your hardware clock is in local or UTC time, the first value in `/etc/adjtime` shows the amount of hardware clock drift per day (in seconds). Listing 59 shows two examples.

Listing 59. /etc/adjtime showing clock drift and local or UTC time.

```
[root@lyrebird ~]# cat /etc/adjtime
0.000990 1184019960 0.000000
1184019960
LOCAL
root@pinguino:~# cat /etc/adjtime
-0.003247 1182889954 0.000000
1182889954
LOCAL
```

Note that both these systems keep the hardware clock in local time, but the clock drifts are different — 0.000990 on lyrebird and -0.003247 on pinguino.

Configure your time zone

Your time zone is a measure of how far your local time differs from UTC. Information on available time zones that can be configured is kept in `/usr/share/zoneinfo`. Traditionally, `/etc/localtime` was a link to one of the time zone files in this directory tree, for example, `/usr/share/zoneinfo/Eire` or `/usr/share/zoneinfo/Australia/Hobart`. On modern systems it is much more likely to be a copy of the appropriate time zone data file since the `/usr/share` filesystem may not be mounted when the local time zone information is needed early in the boot process.

Similarly, another file, `/etc/timezone` was traditionally a link to `/etc/default/init` and was used to set the time zone environment variable `TZ`, and several locale-related environment variables. The file may or may not exist on your system. If it does, it may simply contain the name of the current time zone. You may also find time zone information in `/etc/sysconfig/clock`. Listing 60 shows these files from a Ubuntu 7.04 and a Fedora 7 system.

Listing 60. Time zone information in `/etc`

```
root@pinguino:~# cat /etc/timezone
America/New_York

[root@lyrebird ~]# cat /etc/sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
# The timezone of the system is defined by the contents of
/etc/localtime.
ZONE="America/New York"
UTC=false
ARC=false
```

Some systems such as Debian and Ubuntu have a `tzconfig` command to set the time zone. Others such as Fedora use `system-config-date` to set the time zone and to indicate whether the clock uses UTC or not. Listing 61 illustrates the use of the `tzconfig` command to display the current time zone.

Listing 61. Setting time zone with `tzconfig`

```
root@pinguino:~# tzconfig
Your current time zone is set to America/New_York
Do you want to change that? [n]:
Your time zone will not be changed
```

Configure the Network Time Protocol

The *Network Time Protocol (NTP)* is a protocol to synchronize computer clocks over a network. Synchronization is usually to UTC.

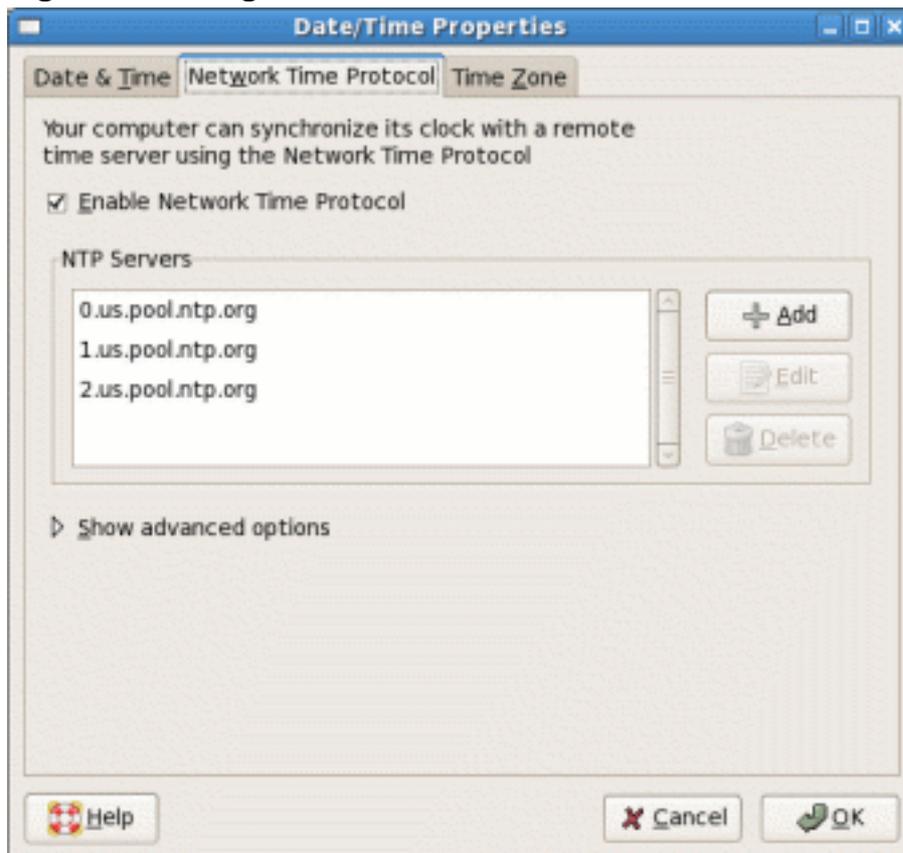
NTP version 3 is an Internet draft standard, formalized in RFC 1305. The current development version, NTP version 4 is a significant revision, which has not been formalized in an RFC. RFC 4330 describes Simple NTP (SNTP) version 4.

Time synchronization is accomplished by sending messages to *time servers*. The time returned is adjusted by an offset of half the round-trip delay. The accuracy of the time is therefore dependent on the network latency and the extent to which the latency is the same in both directions. The shorter the path to a time server, the more accurate the time is likely to be. See [Resources](#) for more detailed information than this simplistic description can provide.

There is a huge number of computers on the Internet, so time servers are organized into *strata*. A relatively small number of stratum 1 servers maintain very accurate time from a source such as an atomic clock. A larger number of stratum 2 servers get their time from stratum 1 servers and make it available to an even larger number of stratum 3 servers, and so on. To ease the load on time servers, a large number of volunteers donate time services through pool.ntp.org (see [Resources](#) for a link). Round robin DNS servers accomplish NTP load balancing by distributing NTP server requests among a pool of available servers.

If you use a graphical interface, you might be able to set your NTP time servers using a dialog similar to that in Figure 4. The fact that this system has enabled automatic time updates using NTP is why the dialog in Figure 3 did not allow the date and time to be changed.

Figure 4. Setting NTP servers



NTP configuration information is kept in `/etc/ntp.conf`, so you can also edit that file and then restart the `ntpd` daemon after you save it. Listing 62 shows an example `/etc/ntp.conf` file using the time servers from Figure 4.

Listing 62. Setting time zone with `tzconfig`

```
[root@lyrebird ~]# cat /etc/ntp.conf
# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#broadcast 192.168.1.255 key 42          # broadcast server
#broadcastclient          # broadcast client
#broadcast 224.0.1.1 key 42            # multicast server
#multicastclient 224.0.1.1            # multicast client
#manycastserver 239.255.254.254        # manycast server
#manycastclient 239.255.254.254 key 42 # manycast client

# Undisciplined Local Clock. This is a fake driver intended for backup
# and when no outside source of synchronized time is available.
#server 127.127.1.0 # local clock
#fudge 127.127.1.0 stratum 10

# Drift file. Put this in a directory which the daemon can write to.
# No symbolic links allowed, either, since the daemon updates the file
# by creating a temporary in the same directory and then rename()'ing
# it to the file.
driftfile /var/lib/ntp/drift

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8
server 0.us.pool.ntp.org
restrict 0.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 1.us.pool.ntp.org
restrict 1.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 2.us.pool.ntp.org
restrict 2.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
```

If you are using the `pool.ntp.org` time servers, these may be anywhere in the world. You will usually get better time by restricting your servers as in this example where `us.pool.ntp.org` is used, resulting in only U.S. servers being chosen. See [Resources](#)

for more information on the ntp.pool.org project.

NTP commands

You can use the `ntpdate` command to set your system time from an NTP time server as shown in Listing 63.

Listing 63. Setting system time from an NTP server using ntpdate

```
[root@lyrebird ~]# ntpdate 0.us.pool.ntp.org
10 Jul 10:27:39 ntpdate[15308]: adjust time server 66.199.242.154 offset
-0.007271 sec
```

Because the servers operate in round robin mode, the next time you run this command you will probably see a different server. Listing 64 shows the first few DNS responses for `0.us.ntp.pool.org` a few moments after the above `ntpdate` command was run.

Listing 64. Round robin NTP server pool

```
[root@lyrebird ~]# dig 0.pool.ntp.org +noall +answer | head -n 5
0.pool.ntp.org. 1062 IN A 217.116.227.3
0.pool.ntp.org. 1062 IN A 24.215.0.24
0.pool.ntp.org. 1062 IN A 62.66.254.154
0.pool.ntp.org. 1062 IN A 76.168.30.201
0.pool.ntp.org. 1062 IN A 81.169.139.140
```

The `ntpdate` command is now deprecated as the same function can be done using `ntpq` with the `-q` option, as shown in Listing 65.

Listing 65. Setting system time using ntpd -q

```
[root@lyrebird ~]# ntpd -q
ntpd: time slew -0.014406s
```

Note that the `ntpd` command uses the time server information from `/etc/ntp.conf`, or a configuration file provided on the command line. See the man page for more information and for information about other options for `ntpd`. Be aware also that if the `ntpd` daemon is running, `ntpd -q` will quietly exit, leaving a failure message in `/var/log/messages`.

Another related command is the `ntpq` command, which allows you to query the NTP daemon. See the man page for more details.

This brings us to the end of this tutorial. We have covered a lot of material on system administration. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- See the [Partimage homepage](#) for information on Partimage, a filesystem-aware partition dump and restore tool.
- "[/etc: Host-specific system configuration](#)" describes the Linux Standard Base (LSB) requirements for /etc.
- The [Network Time Protocol Project](#) produces a reference implementation of the NTP protocol, and implementation documentation.
- The [Network Time Synchronization Project](#) maintains an extensive array of documentation and background information, including briefing slides, on network time protocols.
- The [pool.ntp.org project](#) is a big virtual cluster of timeservers striving to provide reliable easy to use NTP service for millions of clients without putting a strain on the big popular timeservers.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content.](#)
- Get involved in the [developerWorks community](#) through our developer blogs, forums, podcasts, and community topics in our new [developerWorks spaces](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 111: Administrative tasks

Junior Level Administration (LPIC-1) topic 111

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

10 Jul 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this sixth in a [series of nine tutorials](#), Ian introduces you to administrative tasks. By the end of this tutorial, you will know how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *intermediate level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active intermediate level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the four junior and intermediate certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts.
Topic 111	LPI exam 102 prep: Administrative tasks	(This tutorial.) Learn how to manage user and group accounts and tune user and system environments, configure and use system log files, automate system administration tasks by scheduling jobs to run at another time, back up your system, and maintain system time. See the detailed objectives below.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Administrative tasks," the sixth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Administrative tasks: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
1.111.1 User and group accounts	Weight 4	Add, remove, suspend, and change user accounts. Manage user and group information in password and group databases, including shadow databases. Create and manage special purpose and limited accounts.
1.111.2 Tune user and system environments	Weight 3	Modify global and user profiles. Set environment variables and maintain skeleton directories for new user accounts. Set command search paths.
1.111.3 Configure and use system log files to meet administrative and security needs	Weight 3	Configure and manage system logs, including the type and level of logged information. Scan and monitor log files for

		notable activity and track down noted problems. Rotate and archive log files.
1.111.4 Automate system administration tasks by scheduling jobs to run in the future	Weight 4	Use the <code>cron</code> or <code>anacron</code> commands to run jobs at regular intervals, and use the <code>at</code> command to run jobs at a specific time.
1.111.5 Maintain an effective data backup strategy	Weight 3	Plan a backup strategy and back up filesystems automatically to various media.
1.111.6 Maintain system time	Weight 4	Maintain the system time and time zone, and synchronize the clock via NTP. Set the BIOS clock to the correct time in UTC, and configure NTP, including correcting for clock drift.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be thoroughly familiar with the material from the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, which covers basic concepts of users, groups, and file permissions.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. User and group accounts

This section covers material for topic 1.111.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Add, modify, and remove users and groups
- Suspend and change user accounts
- Manage user and group information in the password databases and group databases
- Use the correct tools to manage shadow password databases and group databases
- Create and manage limited and special-purpose accounts

As you learned in the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, Linux is a multi-user system where each user belongs to one *primary* group and possibly to additional groups. Ownership of files in Linux is closely related to user ids and groups. Recall that you can log in as one user and become another user using the `su` or `sudo -s` commands, and that you can use the `whoami` command to check your current effective id and the `groups` command to find out what groups you belong to. In this section, you learn how to create, delete, and manage users and groups. You also learn about the files in `/etc`, where user and group information is stored.

Add and remove users and groups

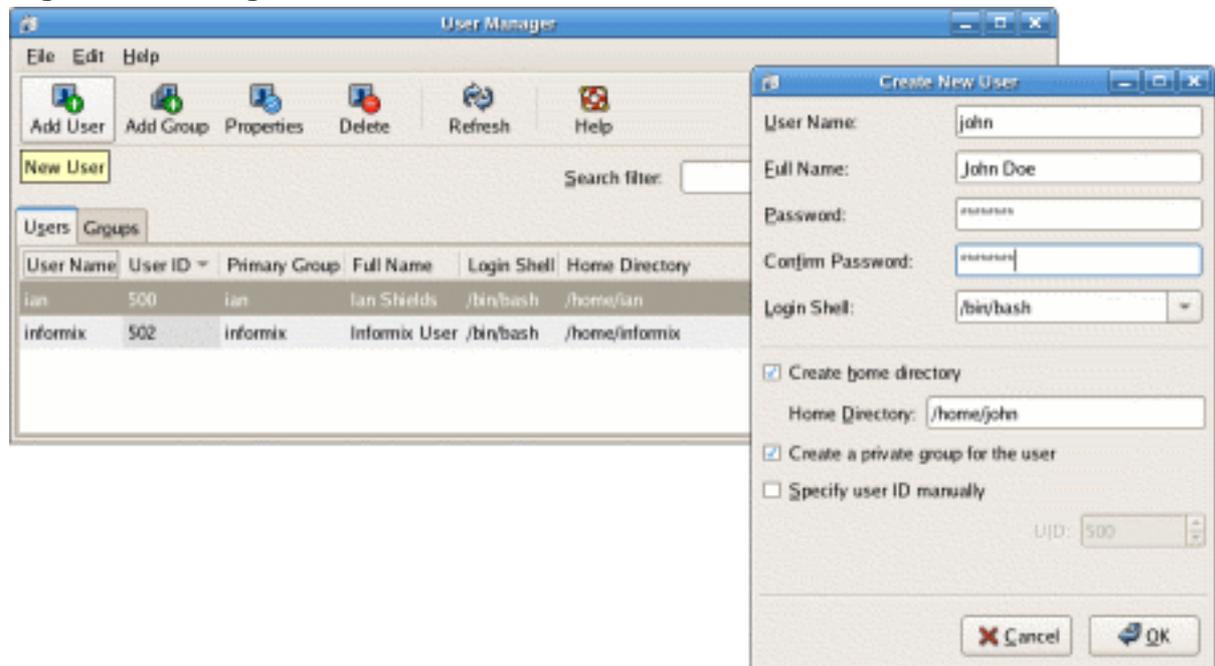
You add a user to a Linux system using the `useradd` command, and you delete a user using the `userdel` command. Similarly, you add or delete groups using the `groupadd` and `groupdel` commands.

Adding a user or group

Modern Linux desktops usually have graphical interfaces for user and group administration. The graphical interface is usually accessed through menu options for system administration. These interfaces do vary considerably, so the one on your system may not look much like the example here, but the underlying concepts and commands remain similar.

Let's start by adding a user to a Fedora Core 5 system graphically, and then examine the underlying commands. In the case of Fedora Core 5 with GNOME desktop, use **System > Administration > Users and Groups**, then click the **Add User** button.

Figure 1 depicts the User Manager panel with the Create New User panel showing basic information for a new user named 'john'. The full name of the user, John Doe, and a password have been entered. The panel provides a default login shell of `/bin/bash`. On Fedora systems, the default is to create a new group with the same name as the user, 'john' in this case, and a home directory of `/home/john`.

Figure 1. Adding a user

Listing 1 shows the use of the `id` command to display basic information about the new user. As you can see, john has user number 503 and a matching group, john, with group number 503. This is the only group of which john is a member.

Listing 1. Displaying user id information

```
[root@pinguino ~]# id john
uid=503(john) gid=503(john) groups=503(john)
```

To accomplish the same task from the command line, you use the `groupadd` and `useradd` commands to create the group and user, then use the `passwd` command to set the password for the newly created user. All of these commands require root authority. The basic use of these commands to add another user, jane, is illustrated in Listing 2.

Listing 2. Adding user jane

```
[root@pinguino ~]# groupadd jane
[root@pinguino ~]# useradd -c "Jane Doe" -g jane -m jane
[root@pinguino ~]# passwd jane
Changing password for user jane.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# id jane
uid=504(jane) gid=504(jane) groups=504(jane)
[root@pinguino ~]# ls -ld /home/jane
drwx----- 3 jane jane 4096 Jun 25 18:22 /home/jane
```

In these two examples, both the user id and the group id have values greater than 500. Be aware that some newer systems start user ids at 1000 rather than 500. These values normally signify ordinary users, while values below 500 (or 1000 if the system starts ordinary users at 1000) are reserved for *system users*. [System users](#) are covered later in this section. The actual cutoff points are set in `/etc/login.defs` as `UID_MIN` and `GID_MIN`.

In Listing 2 above, the `groupadd` command has a single parameter, `jane`, the name of the group to be added. Group names must begin with a lower case letter or an underscore, and usually contain only these along with hyphens or dashes. Options you may specify are shown in Table 3.

Option	Purpose
-f	Exit with success status if the group already exists. This is handy for scripting when you do not need to check if a group exists before attempting to create it.
-g	Specifies the group id manually. The default is to use the smallest value that is at least <code>GID_MIN</code> and also greater than the id of any existing group. Group ids are normally unique and must be non-negative
-o	Permits a group to have a non-unique id.
-K	Can be used to override defaults from <code>/etc/login.defs</code> .

In Listing 2 above, the `useradd` command has a single parameter, `jane`, the name of the user to be added, along with the `-c`, `-g`, and `-m` options. Common options for the `useradd` command are shown in Table 4.

Option	Purpose
-b --base-dir	The default base directory in which user home directories are created. This is usually <code>/home</code> , and the user's home directory is <code>/home/\$USER</code> .
-c --comment	A text string describing the id, such as the user's full name.
-d --home	Provides a specific directory name for the home directory.
-e	The date on which the account will

--expiredate	expire or be disabled in the form YYYY-MM_DD.
-g --gid	The name or number of the initial login group for the user. The group must exist, which is why group jane was created before user jane in Listing 2.
-G --groups	A comma-separated list of additional groups to which the user belongs.
-K	Can be used to override defaults from /etc/login.defs.
-m --create-home	Create the user's home directory if it does not exist. Copy the skeleton files and any directories from /etc/skel to the home directory.
-o --non-unique	Permits a user to have a non-unique id.
-p --password	The encrypted password. If a password is not specified, the default is to disable the account. You will usually use the <code>passwd</code> command in a subsequent step rather than generating an encrypted password and specifying it on the <code>useradd</code> command.
-s --shell	The name of the user's login shell if different from the default login shell.
-u --uid	The non-negative numerical userid, which must be unique if <code>-o</code> is not specified. The default is to use the smallest value that is at least <code>UID_MIN</code> and also greater than the id of any existing user.

Notes:

1. Some systems, including Fedora and Red Hat distributions, have extensions to the user-creation commands. For example, the default Fedora and Red Hat behavior is to create a new group for a user, and the `-n` option can be used on the `useradd` command to disable this function. Be aware of such possible system differences and refer to the man pages on your system when in doubt.
2. On SUSE systems, use YaST or YaST2 to access graphical user and group administration interfaces.
3. Graphical interfaces may perform additional tasks such as creating the user's mail file in `/var/spool/mail`.

Deleting a user or group

Deleting a user or group is much simpler than adding one, because there are fewer options. In fact, the `groupdel` command to delete a group requires only the group name; it has no options. You cannot delete any group that is the primary group of a user. If you use a graphical interface for deleting users and groups, the functions are very similar to the commands shown here.

Use the `userdel` command to delete a user. The `-r`, or `--remove` option requests removal of the user's home directory and anything it contains, along with the user's mail spool. When you delete a user, a group with the same name as the user will also be deleted if `USERGROUPS_ENAB` is set to `yes` in `/etc/login.defs`, but this will be done only if the group is not the primary group of another user.

In Listing 3 you see an example of deleting groups when multiple users share the same primary group. Here, another user, `jane2`, has previously been added to the system with the same group as `jane`.

Listing 3. Deleting users and groups

```
root@pinguino:~# groupdel jane
groupdel: cannot remove user's primary group.
root@pinguino:~# userdel -r jane
userdel: Cannot remove group jane which is a primary group for another
user.
root@pinguino:~# userdel -r jane2
root@pinguino:~# groupdel jane
```

Notes:

1. There is a `userdel` option, `-f` or `--force`, which can be used to delete users and their group. This option is dangerous, so you should use it only as a last resort. Read the man page carefully before you do.
2. Be aware that if you delete a user or group, and there are files that belong to that user or group on your filesystem, then the files are not automatically deleted or assigned to another user or group.

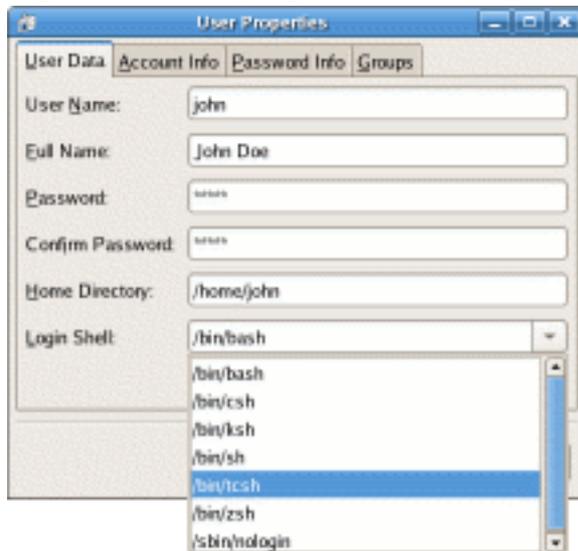
Suspend and change accounts

Now that you can create or delete a user id or a group, you may also find a need to modify one.

Modifying user accounts

Suppose user john wishes to have the tcsh shell as his default. From a graphical interface you will usually find a way to either edit a user (or group), or to examine the properties of the object. Figure 2 shows the properties dialog for the user john that we created earlier on a Fedora Core 5 system.

Figure 2. Modifying a user account



From the command line, you can use the `usermod` command to modify a user account. You can use most of the options that you use with `useradd`, except that you cannot create or populate a new home directory for the user. If you need to change the name of the user, specify the `-l` or `--login` option with the new name. You will probably want to rename the home directory to match the user id. You may also need to rename other items such as mail spool files. Finally, if the login shell is changed, some of the associated profile files may need to be altered. Listing 4 shows an example of the things you might need to do to change user john to john2 with `/bin/tcsh` as the default shell and renamed home directory `/home/john2`.

Listing 4. Modifying a user

```
[root@pinguino ~]# usermod -l john2 -s /bin/tcsh -d /home/john2 john
[root@pinguino ~]# ls -d ~john2
ls: /home/john2: No such file or directory
[root@pinguino ~]# mv /home/john /home/john2
[root@pinguino ~]# ls -d ~john2
/home/john2
```

Notes:

1. If you need to modify a user's additional groups, you must specify the complete list of additional groups. There is no command to simply add or delete a single group for a user.

2. There are restrictions on changing the name or id of a user who is logged in or who has running processes. Check the man pages for details.
3. If you change a user number, you may want to change files and directories owned by that user to match the new number.

Modifying groups

Not surprisingly, the `groupmod` command is used to modify group information. You can change the group number with the `-g` option, and the name with the `-n` option.

Listing 5. Renaming a group

```
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# groupmod -n john2 john
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

Notice in Listing 5 that the group name for the home directory of `john2` magically changed when we used `groupmod` to change the group name. Are you surprised? Because groups are represented in the filesystem inodes by their number rather than by their name, this is not surprising. However, if you change a group's number, you should update any users for which that group is the primary group, and you may also want to update the files and directories belonging to that group to match the new number (in the same way as noted above for changing a user number). Listing 6 shows how to change the group number for `john2` to 505, update the user account, and make appropriate changes to all the affected files in the `/home` filesystem. You probably want renumbering users and groups if at all possible.

Listing 6. Renumbering a group

```
[root@pinguino ~]# groupmod -g 505 john2
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# id john2
uid=503(john2) gid=503 groups=503
[root@pinguino ~]# usermod -g john2 john2
[root@pinguino ~]# id john2
uid=503(john2) gid=505(john2) groups=505(john2)
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# find /home -gid 503 -exec chgrp john2 {} \;
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

User and group passwords

You have already seen the `passwd` command, which is used to change a user password. The password is (or should be) unique to the user and may be changed by user. The root user may change any user's password as we have already seen.

Groups may also have passwords, and the `gpasswd` command is used to set them. Having a group password allows users to join a group temporarily with the `newgrp` command, if they know the group password. Of course, having multiple people knowing a password is somewhat problematic, so you will have to weigh the advantages of adding a user to a group using `usermod`, versus the security issue of having too many people knowing the group password.

Suspending or locking accounts

If you need to prevent a user from logging in, you can *suspend* or *lock* the account using the `-L` option of the `usermod` command. To *unlock* the account, use the `-U` option. Listing 7 shows how to lock account `john2` and what happens if `john2` attempts to log in to the system. Note that when the `john2` account is unlocked, the same password is restored.

Listing 7. Locking an account

```
[root@pinguino ~]# usermod -L john2
[root@pinguino ~]# ssh john2@pinguino
john2@pinguino's password:
Permission denied, please try again.
```

You may have noticed back in [Figure 2](#) that there were several tabs on the dialog box with additional user properties. We briefly mentioned the use of the `passwd` command for setting user passwords, but both it and the `usermod` command can perform many tasks related to user accounts, as can another command, the `chage` command. Some of these options are shown in Table 5. Refer to the appropriate man pages for more details on these and other options.

Table 5. Commands and options for changing user accounts			
	Option for command		Purpose
Usermod	Passwd	Chage	
-L	-l	N/A	Lock or suspend the account.
-U	-u	N/A	Unlock the account.
N/A	-d	N/A	Disable the account by setting it passwordless.

-e	-f	-E	Set the expiration date for an account.
N/A	-n	-m	The minimum password lifetime in days.
N/A	-x	-M	The maximum password lifetime in days.
N/A	-w	-W	The number of days of warning before a password must be changed.
-f	-i	-l	The number of days after a password expires until the account is disabled.
N/A	-S	-l	Output a short message about the current account status.

Manage user and group databases

The primary repositories for user and group information are four files in `/etc`.

`/etc/passwd`

is the *password* file containing basic information about users

`/etc/shadow`

is the *shadow password* file containing encrypted passwords

`/etc/group`

is the *group* file containing basic information about groups and which users belong to them

/etc/gshadow

is the *shadow group* file containing encrypted group passwords

These files are updated by the commands you have already seen in this tutorial and you will meet some more commands for working with them after we discuss the files themselves. All of these files are plain text files. In general, you should not edit them directly. Use the tools provided for updating them so they are properly locked and kept synchronized.

You will note that the passwd and group files are both *shadowed*. This is for security reasons. The passwd and group files themselves must be world readable, but the encrypted passwords should not be world readable. Therefore, the shadow files contain the encrypted passwords, and these files are only readable by root. The necessary authentication access is provided by an suid program that has root authority, but can be run by anyone. Make sure that your system has the permissions set appropriately. Listing 8 shows an example.

Listing 8. User and group database permissions

```
[ian@pinguino ~]$ ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow
-rw-r--r-- 1 root root 701 Jun 26 19:04 /etc/group
-r----- 1 root root 580 Jun 26 19:04 /etc/gshadow
-rw-r--r-- 1 root root 1939 Jun 26 19:43 /etc/passwd
-r----- 1 root root 1324 Jun 26 19:50 /etc/shadow
```

Note: Although it is still technically possible to run without shadowed password and group files, this is almost never done and is not recommended.

The /etc/passwd file

The /etc/passwd file contains one line for each user in the system. Some example lines are shown in Listing 9.

Listing 9. /etc/password entries

```
root:x:0:0:root:/root:/bin/bash
jane:x:504:504:Jane Doe:/home/jane:/bin/bash
john2:x:503:505:John Doe:/home/john2:/bin/tcsh
```

Each line contains seven fields separated by colons (:), as shown in Table 6.

Table 6. Fields in /etc/passwd	
Field	Purpose
Username	The name used to log in to the system.

	For example, john2.
Password	The encrypted password. When using shadow passwords, it contains a single x character.
User id (UID)	The number used to represent this user name in the system. For example, 503 for user john2.
Group id (GID)	The number used to represent this user's primary group in the system. For example, 505 for user john2.
Comment (GECOS)	An optional field used to describe the user. For example, "John Doe". The field may contain multiple comma-separated entries. It is also used by programs such as <code>finger</code> . The GECOS name is historic. See details in <code>man 5 passwd</code> .
Home	The absolute path the user's home directory. For example, <code>/home/john2</code>
Shell	The program automatically launched when a user logs in to the system. This is usually an interactive shell such as <code>/bin/bash</code> or <code>/bin/tcsh</code> , but may be any program, not necessarily an interactive shell.

The `/etc/group` file

The `/etc/group` file contains one line for each group in the system. Some example lines are shown in Listing 10.

Listing 10. `/etc/group` entries

```
root:x:0:root
jane:x:504:john2
john2:x:505:
```

Each line contains four fields separated by colons (:), as shown in Table 7.

Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The encrypted password. When using shadow group passwords, it contains a single x character.

Group id (GID)	The number used to represent this group in the system. For example, 505 for group john2.
Members	A comma-separated list of group members, excepting those members for whom this is the primary group.

Shadow files

The file `/etc/shadow` should only be readable by root. It contains encrypted passwords, along with password and account expiration information. See the man page (`man 5 shadow`) for information on the field layout. Passwords may be encrypted using DES, but are more usually encrypted using MD5. The DES algorithm uses the low order 7 bits of the first 8 characters of the user password as a 56-bit key, while the MD5 algorithm uses the whole password. In either case, passwords are *salted* so that two otherwise identical passwords do not generate the same encrypted value. Listing 11 shows how to set identical passwords for users jane and john2, and then shows the resulting encoded MD5 passwords in `/etc/shadow`.

Listing 11. Passwords in `/etc/shadow`

```
[root@pinguino ~]# echo lpic1111 |passwd jane --stdin
Changing password for user jane.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# echo lpic1111 |passwd john2 --stdin
Changing password for user john2.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# grep "^j" /etc/shadow
jane:$1$eG0/KGQY$ZJ1.ltYtVw0sv.C5OrqUu/:13691:0:99999:7:::
john2:$1$grkxo6ie$J2muvoTpwo3dZAYYTDYNu.:13691:0:180:7:29::
```

The leading `1` indicates an MD5 password, and the salt is a variable length field of up to 8 characters ending with the next `$` sign. The encrypted password is the remaining string of 22 characters.

Tools for users and groups

You have already seen several commands that manipulate the account and group files and their shadows. Here you learn about:

- Group administrators
- Editing commands for password and group files
- Conversion programs

Group administrators

In some circumstances you may want users other than root to be able to administer one or more groups by adding or removing group members. Listing 12 shows how root can add user jane as an administrator of group john2, and then jane, in turn, can add user ian as a member.

Listing 12. Adding group administrators and members

```
[root@pinguino ~]# gpasswd -A jane john2
[root@pinguino ~]# su - jane
[jane@pinguino ~]$ gpasswd -a ian john2
Adding user ian to group john2
[jane@pinguino ~]$ id ian;id jane
uid=500(ian) gid=500(ian) groups=500(ian),505(john2)
uid=504(jane) gid=504(jane) groups=504(jane)
```

You may be surprised to note that, although jane is an administrator of group john2, she is not a member of it. An examination of the structure of `/etc/gshadow` shows why. The `/etc/gshadow` file contains four fields for each entry as shown in Table 8. Note that the third field is a comma-separated list of administrators for the group.

Table 8. Fields in <code>/etc/gshadow</code>	
Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The field used to contain the encrypted password if the group has a password. If there is no password, you may see 'x', '!' or '!!' here.
Admins	A comma-separated list of group administrators.
Members	A comma-separated list of group members.

As you can see, the administrator list and the member list are two distinct fields. The `-A` option of `gpasswd` allows the root user to add administrators to a group, while the `-M` option allows root to add members. The `-a` (note lower case) option allows an administrator to add a member, while the `-d` option allows an administrator to remove a member. Additional options allow a group password to be removed. See the man pages for details.

Editing commands for password and group files

Although not listed in the LPI objectives, you should also be aware of the `vipw` command for safely editing `/etc/passwd` and `visgr` for safely editing `/etc/group`. The

commands will lock the necessary files while you make changes using the `vi` editor. If you make changes to `/etc/passwd`, then `vipw` will prompt you to see if you also need to update `/etc/shadow`. Similarly, if you update `/etc/group` using `vigr`, you will be prompted to update `/etc/gshadow`. If you need to remove group administrators, you may need to use `vigr`, as `gpasswd` only allows addition of administrators.

Conversion programs

Four other related commands are also not listed in the LPI objectives. They are `pwconv`, `pwunconv`, `grpconv`, and `grpunconv`. They are used for converting between shadowed and non-shadowed password and group files. You may never need these, but be aware of their existence. See the man pages for details.

Limited and special-purpose accounts

By convention, system users usually have an id of less than 100, with root having id 0. Normal users start automatic numbering from the `UID_MIN` value set in `/etc/login.defs`, with this value commonly being set at 500 or 1000.

Besides regular user accounts and the root account on your system, you will usually have several special-purpose accounts, for daemons such as FTP, SSH, mail, news, and so on. Listing 13 shows some entries from `/etc/passwd` for these.

Listing 13. Limited and special-purpose accounts

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
```

Such accounts frequently control files but should not be accessed by normal login. Therefore, they usually have a login shell specified as `/sbin/nologin`, or `/bin/false` so that login attempts will fail.

Section 3. Environment tuning

This section covers material for topic 1.111.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to tune the user environment, including these tasks:

- Set and unset environment variables
- Maintain skeleton directories for new user accounts
- Set command search paths

Set and unset environment variables

When you create a new user, you usually initialize many variable according to your local needs. These are usually set in the profiles that you provide for new users, such as `.bash_profile` and `.bashrc`, or in the system-wide profiles `/etc/profile` and `/etc/bashrc`. Listing 14 shows an example of how the `PS1` system prompt is set in `/etc/profile` on an Ubuntu 7.04 system. The first `if` statement checks whether the `PS1` variable is set, indicating an interactive shell, since a non-interactive shell doesn't need a prompt. The second `if` statement checks whether the `BASH` environment variable is set. If so, it sets a complex prompt and sources (note the dot) `/etc/bash.bashrc`. If the `BASH` variable is not set, then a check is made for root (`id=0`), and the prompt is set to `#` or `$` accordingly.

Listing 14. Setting environment variables

```
if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
```

The tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) has detailed information about the commands used for setting and unsetting environment variables, as well as information about how and when the various profiles are used.

When customizing environments for users, be aware of two major points:

1. `/etc/profile` is read only at login time, and so it is not executed when each new shell is created.

2. Functions and aliases are not inherited by new shells. Therefore, you will usually set these and your environment variables in `/etc/bashrc`, or in the user's own profiles.

In addition to the system profiles, `/etc/profile` and `/etc/bashrc`, the Linux Standard Base (LSB) specifies that additional scripts may be placed in the directory `/etc/profile.d`. These scripts are sourced when an interactive login shell is created. They provide a convenient way of separating customization for different programs. Listing 15 shows an example.

Listing 15. `/etc/profile.d/vim.sh` on Fedora 7

```
[if [ -n "$BASH_VERSION" -o -n "$KSH_VERSION" -o -n "$ZSH_VERSION" ];
then
  [ -x //usr/bin/id ] || return
  [ `//usr/bin/id -u` -le 100 ] && return
  # for bash and zsh, only if no alias is already set
  alias vi >/dev/null 2>&1 || alias vi=vim
fi
```

Remember that you should usually `export` any variables that you set in a profile; otherwise, they will not be available to commands that run in a new shell.

Maintain skeleton directories for new users

You learned in the section [Add and remove users and groups](#) that you can create or populate a new home directory for the user. The source for this new directory is the subtree rooted at `/etc/skel`. Listing 16 shows the files in this subtree for a Fedora 7 system. Note that most files start with a period (dot), so you need the `-a` option to list them. The `-R` options lists subdirectories recursively, and the `-L` option follows any symbolic links.

Listing 16. `/etc/skel` on Fedora 7

```
[ian@lyrebird ~]$ ls -aRL /etc/skel
/etc/skel:
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .xemacs

/etc/skel/.xemacs:
.  ..  init.el
```

In addition to `.bash_logout`, `.bash_profile`, and `.bashrc`, which you might expect for the Bash shell, note that this example includes profile information for the emacs and xemacs editors. See the tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) if you need to review the functions of the various profile files.

Listing 17 shows `/etc/skel/.bashrc` from the above system. This file might be different on a different release or different distribution, but it gives you an idea of how the default user setup can be done.

Listing 17. `/etc/skel/.bashrc` on Fedora 7

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
```

As you can see, the global `/etc/bashrc` is sourced, then any user specific instructions can be added. Listing 18 shows the part of `/etc/bashrc` in which the `.sh` scripts from `/etc/profile.d` are sourced.

Listing 18. Sourcing `.sh` scripts from `/etc/profile.d`

```
for i in /etc/profile.d/*.sh; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i
```

Note that the variable, `i`, is unset after the loop.

Set command search paths

Your default profiles often include `PATH` variables for local functions or for products that you may have installed. You can set these in the skeleton files in `/etc/skel`, modify `/etc/profile`, `/etc/bashrc`, or create a file in `/etc/profile.d` if your system uses that. If you do modify the system files, be sure to check that your changes are intact after any system updates. Listing 19 shows how to add a new directory, `/opt/productxyz/bin`, to either the front or rear of your existing `PATH`.

Listing 19. Adding a path directory

```
PATH="$PATH${PATH:+:}/opt/productxyz/bin"
PATH="/opt/productxyz/bin${PATH:+:}$PATH"
```

Although not strictly required, the expression `${PATH:+:}` inserts a path separator (colon) only if the `PATH` variable is unset or null.

Section 4. System log files

This section covers material for topic 1.111.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to configure and manage system logs, including these tasks:

- Manage the type and level of information logged
- Rotate and archive log files automatically
- Scan log files for notable activity
- Monitor log files
- Track down problems reported in log files

Manage the type and level of information logged

The system logging facility on a Linux system provides system logging and kernel message trapping. Logging can be done on a local system or sent to a remote system, and the level of logging can be finely controlled through the `/etc/syslog.conf` configuration file. Logging is performed by the `syslogd` daemon, which normally receives input through the `/dev/log` socket, as shown in Listing 20.

Listing 20. `/dev/log` is a socket

```
ian@pinguino:~$ ls -l /dev/log
srw-rw-rw- 1 root root 0 2007-07-05 15:42 /dev/log
```

For local logging, the main file is usually `/var/log/messages`, but many other files are used in most installations, and you can customize these extensively. For example, you may want a separate log for messages from the mail system.

The `syslog.conf` configuration file

The `syslog.conf` file is the main configuration file for the `syslogd` daemon. Logging is based on a combination of facility and priority. The defined facilities are `auth` (or `security`), `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `syslog`, `user`, `uucp`, and `local0` through `local7`. The keyword `auth` should be used instead of `security`, and the keyword `mark` is for internal use.

The priorities (in ascending order) are:

1. debug
2. info
3. notice
4. warning (or warn)
5. err (or error)
6. crit
7. alert
8. emerg (or panic)

The parenthesized keywords (warn, error, and panic) are now deprecated.

Entries in `syslog.conf` specify logging rules. Each rule has a selector field and an action field, which are separated by one or more spaces or tabs. The selector field identifies the facility and the priorities that the rule applies to, and the action field identifies the logging action for the facility and priorities. The default behavior is to take the action for the specified level and for all higher levels, although it is possible to limit logging to specific levels. Each selector consists of a facility and a priority separated by a period (dot). Multiple facilities for a given action can be specified by separating them with a comma. Multiple facility/priority pairs for a given action can be specified by separating them with a semi-colon. Listing 21 shows an example of a simple `syslog.conf`.

Listing 21. Example `syslog.conf`

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                               /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages

# The authpriv file has restricted access.
authpriv.*                             /var/log/secure

# Log all the mail messages in one place.
mail.*
-/var/log/maillog

# Log cron stuff
cron.*                                  /var/log/cron
```

```
# Everybody gets emergency messages
*.emerg                                     *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                             /var/log/spooler

# Save boot messages also to boot.log
local7.*
/var/log/boot.log
```

Notes:

- As with many configuration files, lines starting with # and blank lines are ignored.
- An * may be used to refer to all facilities or all priorities.
- The special priority keyword `none` indicates that no logging for this facility should be done with this action.
- The hyphen before a file name (such as `-/var/log/maillog`, in this example) indicates that the log file should not be synchronized after every write. You might lose information after a system crash, but you might gain performance by doing this.

The actions are generically referred to as "logfiles," although they do not have to be real files. Table 9 describe the possible logfiles.

Table 9. Actions in syslog.conf	
Action	Purpose
Regular File	Specify the full pathname, beginning with a slash (/). Prefix it with a hyphen (-) to omit syncing the file after each log entry. This may cause information loss if a crash occurs, but may improve performance.
Named Pipes	A fifo or named pipe can be used as a destination for log messages by putting a pipe symbol () before the filename. You must create the fifo using the <code>mkfifo</code> command before starting (or restarting) <code>syslogd</code> . Fifos are sometimes used for debugging.
Terminal and Console	A terminal such as <code>/dev/console</code> .
Remote Machine	To forward messages to another host, put an at (@) sign before the hostname. Note that messages are not forwarded from the receiving host.
List of Users	A comma-separated list of users to receive a message (if the user is

	logged in). The root user is frequently included here.
Everyone logged on	Specify an asterisk (*) to have everyone logged on notified using the wall command.

You may prefix ! to a priority to indicate that the action should not apply to this level and higher. Similarly you may prefix it with = to indicate that the rule applies only to this level or with != to indicate that the rule applies to all except this level. Listing 22 shows some examples, and the man page for syslog.conf has many more examples.

Listing 22. More syslog.conf examples

```
# Store all kernel messages in /var/log/kernel.
# Send critical and higher ones to remote host pinguino and to the
console
# Finally, Send info, notice and warning messages to
/var/log/kernel-info
#
kern.*                /var/log/kernel
kern.crit             @pinguino
kern.crit             /dev/console
kern.info;kern.!err  /var/log/kernel-info

# Store all mail messages except info priority in /var/log/mail.
mail.*;mail.!=info   /var/log/mail
```

Rotate and archive log files automatically

With the amount of logging that is possible, you need to be able to control the size of log files. This is done using the `logrotate` command, which is usually run as a cron job. Cron jobs are covered later in this tutorial in the section [Scheduling jobs](#). The general idea behind the `logrotate` command is that log files are periodically backed up and a new log is started. Several generations of log are kept, and when a log ages to the last generation, it may be archived. For example, it might be mailed to an archival user.

You use the `/etc/logrotate.conf` configuration file to specify how your log rotating and archiving should happen. You can specify different frequencies, such as daily, weekly, or monthly, for different log files, and you can control the number of generations to keep and when or whether to mail copies to an archival user. Listing 23 shows a sample `/etc/logrotate.conf` file.

Listing 23. Sample /etc/logrotate.conf

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

The `logrotate.conf` file has global options at the beginning. These are the defaults if nothing more specific is specified elsewhere. In this example, log files are rotated weekly, and four weeks worth of backups are kept. Once a log file is rotated, a new one is automatically created in place of the old one. Note that the `logrotate.conf` file may include specifications from other files. Here, all the files in `/etc/logrotate.d` are included.

This example also includes specific rules for `/var/log/wtmp` and `/var/log/btmp`, which are rotated monthly. No error message is issued if the files are missing. A new file is created, and only one backup is kept.

In this example, when a backup reaches the last generation, it is deleted because there is no specification of what else to do with it.

Note: The files `/var/log/wtmp` and `/var/log/btmp` record successful and unsuccessful login attempts, respectively. Unlike most log files, these are not clear text files. You may examine them using the `last` or `lastb` commands. See the man pages for these commands for details.

Log files may also be backed up when they reach a specific size, and commands may be scripted to run either prior to or after the backup operation. Listing 24 shows a more complex example.

Listing 24. Another logrotate configuration example

```
/var/log/messages {
```

```

rotate 5
mail logsave@pinguino
size 100k
postrotate
    /usr/bin/killall -HUP syslogd
endscript
}

```

In this example, `/var/log/messages` is rotated after it reaches 100KB in size. Five backups are kept, and when the oldest backup ages out, it is mailed to `logsave@pinguino`. The `postrotate` introduces a script that restarts the `syslogd` daemon after the rotation is complete, by sending it the HUP signal. The `endscript` statement is required to terminate the script and is also required if a `prerotate` script is present. See the `logrotate` man page for more complete information.

Scan log files for notable activity

Log files entries are usually time stamped and contain the hostname of the reporting process, along with the process name. Listing 25 shows a few lines from `/var/log/messages`, containing entries from `gconfd`, `ntpd`, `init`, and `yum`.

Listing 25. Sample log file entries

```

Jul  5 15:28:24 lyrebird gconfd (root-2832): Exiting
Jul  5 15:31:06 lyrebird ntpd[2063]: synchronized to 87.98.219.90,
stratum 2
Jul  5 15:31:06 lyrebird ntpd[2063]: kernel time sync status change 0001
Jul  5 15:31:24 lyrebird init: Trying to re-exec init
Jul  5 15:31:24 lyrebird yum: Updated: libselinux.i386 2.0.14-2.fc7
Jul  5 15:31:24 lyrebird yum: Updated: libsemanage.i386 2.0.3-4.fc7
Jul  5 15:31:25 lyrebird yum: Updated: cups-libs.i386 1.2.11-2.fc7
Jul  5 15:31:25 lyrebird yum: Updated: libXfont.i386 1.2.9-2.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager.i386 0.6.5-7.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager-glib.i386
0.6.5-7.fc7

```

You can scan log files using a pager, such as `less`, or search for specific entries (such as kernel messages from host `lyrebird`) using `grep` as shown in Listing 26.

Listing 26. Scanning log files

```

[root@lyrebird ~]# less /var/log/messages
[root@lyrebird ~]# grep "lyrebird kernel" /var/log/messages | tail -n 9
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HCI socket layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP ver 2.8
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM TTY layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM ver 1.8

```

```
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HIDP (Human Interface
Emulation) ver 1.2
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized drm 1.1.0 20060810
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized i915 1.6.0 20060119
on minor 0
```

Monitor log files

Occasionally you may need to monitor log files for events. For example, you might be trying to catch an infrequently occurring event at the time it happens. In such a case, you can use the `tail` command with the `-f` option to *follow* the log file. Listing 27 shows an example.

Listing 27. Following log file updates

```
[root@lyrebird ~]# tail -n 1 -f /var/log/messages
Jul  6 15:16:26 lyrebird syslogd 1.4.2: restart.
Jul  6 15:16:26 lyrebird kernel: klogd 1.4.2, log source = /proc/kmsg
started.
Jul  6 15:19:35 lyrebird yum: Updated: samba-common.i386 3.0.25b-2.fc7
Jul  6 15:19:35 lyrebird yum: Updated: procps.i386 3.2.7-14.fc7
Jul  6 15:19:36 lyrebird yum: Updated: samba-client.i386 3.0.25b-2.fc7
Jul  6 15:19:37 lyrebird yum: Updated: libsmbclient.i386 3.0.25b-2.fc7
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Received signal 15, shutting
down cleanly
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Exiting
Jul  6 15:19:57 lyrebird yum: Updated: bluez-gnome.i386 0.8-1.fc7
```

Track down problems reported in log files

When you find problems in log files, you will want to note the time, the hostname, and the process that generated the problem. If the message identifies the problem specifically enough for you to resolve it, you are done. If not, you might need to update `syslog.conf` to specify that more messages be logged for the appropriate facility. For example, you might need to show informational messages instead of warning messages or even debug level messages. Your application may have additional facilities that you can use.

Finally, if you need to put marks in the log file to help you know what messages were logged at what stage of your debugging activity, you can use the `logger` command from a terminal window or shell script to send a message of your choice to the syslog daemon for logging according to the rules in `syslog.conf`.

Section 5. Scheduling jobs

This section covers material for topic 1.111.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use the `cron` or `anacron` commands to run jobs at regular intervals
- Use the `at` command to run jobs at a specific time
- Manage cron and at jobs
- Configure user access to the cron and at services

In the previous section, you learned about the `logrotate` command and saw the need to run it periodically. You will see the same need to run commands regularly in the next two sections on backup and network time services. These are only some of the many administrative tasks that have to be done frequently and regularly. In this section, you learn about the tools that are used to automate periodic job scheduling and also the tools used to run a job at some specific time.

Run jobs at regular intervals

Running jobs at regular intervals is managed by the `cron` facility, which consists of the `crond` daemon and a set of tables describing what work is to be done and with what frequency. The daemon wakes up every minute and checks the crontabs to determine what needs to be done. Users manage crontabs using the `crontab` command. The `crond` daemon is usually started by the `init` process at system startup.

To keep things simple, let's suppose that you want to run the command shown in Listing 28 on a regular basis. This command doesn't actually do anything except report the day and the time, but it illustrates how to use `crontab` to set up cron jobs, and we'll know when it was run from the output. Setting up crontab entries requires a string with escaped shell metacharacters, so it is best done with simple commands and parameters, so in this example, the `echo` command will be run from within a script `/home/ian/mycrontab.sh`, which takes no parameters. This saves some careful work with escape characters.

Listing 28. A simple command example.

```
[ian@lyrebird ~]$ cat mycrontest.sh
#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
[ian@lyrebird ~]$ ./mycrontest.sh
It is now 18:37:42 on Friday
```

Creating a crontab

To create a crontab, you use the `crontab` command with the `-e` (for "edit") option. This will open the `vi` editor unless you have specified another editor in the `EDITOR` or `VISUAL` environment variable.

Each crontab entry contains six fields:

1. Minute
2. Hour
3. Day of the month
4. Month of the year
5. Day of the week
6. String to be executed by `sh`

Minutes and hours range from 0-59 and 0-12, respectively, while day or month and month of year range from 1-31 and 1-12, respectively. Day of week ranges from 0-6 with 0 being Sunday. Day of week may also be specified as `sun`, `mon`, `tue`, and so on. The sixth field is everything after the fifth field, is interpreted as a string to pass to `sh`. A percent sign (%) will be translated to a newline, so if you want a % or any other special character, precede it with a backslash (\). The line up to the first % is passed to the shell, while any line(s) after the % are passed as standard input.

The various time-related fields can specify an individual value, a range of values, such as 0-10 or `sun-wed`, or a comma-separated list of individual values and ranges. So a somewhat artificial crontab entry for our example command might be as shown in Listing 29.

Listing 29. A simple crontab example.

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
```

In this example, our command is executed at the 0th, 20th, and 40th minutes (every 20 minutes), for the hours between 10 P.M. and midnight on Fridays and Saturdays during July. See the man page for `crontab(5)` for details on additional ways to specify times.

What about the output?

You may be wondering what happens to any output from the command. Most

commands designed for use with the cron facility will log output using the syslog facility that you learned about in the previous section. However any output that is directed to stdout will be mailed to the user. Listing 30 shows the output you might receive from our example command.

Listing 30. Mailed cron output

```
From ian@lyrebird.raleigh.ibm.com Fri Jul 6 23:00:02 2007
Date: Fri, 6 Jul 2007 23:00:01 -0400
From: root@lyrebird.raleigh.ibm.com (Cron Daemon)
To: ian@lyrebird.raleigh.ibm.com
Subject: Cron <ian@lyrebird> /home/ian/mycronetest.sh
Content-Type: text/plain; charset=UTF-8
Auto-Submitted: auto-generated
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/home/ian>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=ian>
X-Cron-Env: <USER=ian>

It is now 23:00:01 on Friday
```

Where is my crontab?

The crontab that you created with the `crontab` command is stored in `/etc/spool/cron` under the name of the user who created it. So the above crontab is stored in `/etc/spool/cron/ian`. Given this, you will not be surprised to learn that the `crontab` command, like the `passwd` command you learned about earlier, is an `suid` program that runs with root authority.

`/etc/crontab`

In addition to the user crontab files in `/var/spool/cron`, `cron` also checks `/etc/crontab` and files in the `/etc/cron.d` directory. These system crontabs have one additional field between the fifth time entry (day) and the command. This additional field specifies the user for whom the command should be run, normally `root`. A `/etc/crontab` might look like the example in Listing 31.

Listing 31. `/etc/crontab`

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

In this example, the real work is done by the `run-parts` command, which runs

scripts from `/etc/cron.hourly`, `/etc/cron.daily`, and so on; `/etc/crontab` simply controls the timing of the recurring jobs. Note that the commands here all run as root. Note also that the crontab can include shell variables assignments that will be set before the commands are run.

Anacron

The cron facility works well for systems that run continuously. For systems that may be turned off much of the time, such as laptops, another facility, the *anacron* (for "anachronistic cron") can handle scheduling of the jobs usually done daily, weekly, or monthly by the cron facility. Anacron does not handle hourly jobs.

Anacron keeps timestamp files in `/var/spool/anacron` to record when jobs are run. When anacron runs, it checks to see if the required number of days has passed since the job was last run and runs it if necessary. The table of jobs for anacron is stored in `/etc/anacrontab`, which has a slightly different format than `/etc/crontab`. As with `/etc/crontab`, `/etc/anacrontab` may contain environment settings. Each job has four fields.

1. period
2. delay
3. job-identifier
4. command

The period is a number of days, but may be specified as `@monthly` to ensure that a job runs only once per month, regardless of the number of days in the month. The delay is the number of minutes to wait after the job is due to run before actually starting it. You can use this to prevent a flood of jobs when a system first starts. The job identifier can contain any non-blank character except slashes (`/`).

Both `/etc/crontab` and `/etc/anacrontab` are updated by direct editing. You do not use the `crontab` command to update these files or files in the `/etc/cron.d` directory.

Run jobs at specific times

Sometimes you may need to run a job just once, rather than regularly. For this you use the `at` command. The commands to be run are read from a file specified with the `-f` option, or from stdin if `-f` is not used. The `-m` option sends mail to the user even if there is no stdout from the command. The `-v` option will display the time at which the job will run before reading the job. The time is also displayed in the output. Listing 32 shows an example of running the `mycrontest.sh` script that you used earlier. Listing 33 shows the output that is mailed back to the user after the job runs.

Notice that it is somewhat more compact than the corresponding output from the cron job.

Listing 32. Using the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh -v 10:25
Sat Jul 7 10:25:00 2007

job 5 at Sat Jul 7 10:25:00 2007
```

Listing 33. Job output from at

```
From ian@lyrebird.raleigh.ibm.com Sat Jul 7 10:25:00 2007
Date: Sat, 7 Jul 2007 10:25:00 -0400
From: Ian Shields <ian@lyrebird.raleigh.ibm.com>
Subject: Output from your job 5
To: ian@lyrebird.raleigh.ibm.com

It is now 10:25:00 on Saturday
```

Time specifications can be quite complex. Listing 34 shows a few examples. See the man page for `at` or the file `/usr/share/doc/at/timespec` or a file such as `/usr/share/doc/at-3.1.10/timespec`, where 3.1.10 in this example is the version of the `at` package.

Listing 34. Time values with the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh 10pm tomorrow
job 14 at Sun Jul 8 22:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 tuesday
job 15 at Tue Jul 10 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 july 11
job 16 at Wed Jul 11 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 next week
job 17 at Sat Jul 14 02:00:00 2007
```

The `at` command also has a `-q` option. Increasing the queue increases the nice value for the job. There is also a `batch` command, which is similar to the `at` command except that jobs are run only when the system load is low enough. See the man pages for more details on these features.

Manage scheduled jobs

Listing scheduled jobs

You can manage your cron and `at` jobs. Use the `crontab` command with the `-l` option to list your crontab, and use the `atq` command to display the jobs you have queued using the `at` command, as shown in Listing 35.

Listing 35. Displaying scheduled jobs

```
[ian@lyrebird ~]$ crontab -l
0,20,40 22-23 * 7 fri-sat /home/ian/mycronstest.sh
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
```

If you want to review the actual command scheduled for execution by `at`, you can use the `at` command with the `-c` option and the job number. You will notice that most of the environment that was active at the time the `at` command was issued is saved with the scheduled job. Listing 36 shows part of the output for job 15.

Listing 36. Using `at -c` with a job number

```
#!/bin/sh
# atrun uid=500 gid=500
# mail ian 0
umask 2
HOSTNAME=lyrebird.raleigh.ibm.com; export HOSTNAME
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=9.67.219.151\ 3210\ 22; export SSH_CLIENT
SSH_TTY=/dev/pts/5; export SSH_TTY
USER=ian; export USER
...
HOME=/home/ian; export HOME
LOGNAME=ian; export LOGNAME
...
cd /home/ian || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
${SHELL:-/bin/sh} << `(dd if=/dev/urandom count=200 bs=1 \
  2>/dev/null|LC_ALL=C tr -d -c '[:alnum:]')`

#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
```

Note that the contents of our script file have been copied in as a here-document that will be executed by the shell specified by the `SHELL` variable or `/bin/sh` if the `SHELL` variable is not set. See the tutorial [LPI exam 101 prep, Topic 103: GNU and UNIX commands](#) if you need to review here-documents.

Deleting scheduled jobs

You can delete all scheduled cron jobs using the `crontab` command with the `-r` option as illustrated in Listing 37.

Listing 37. Displaying and deleting cron jobs

```
[ian@lyrebird ~]$ crontab -l
```

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycronetest.sh
[ian@lyrebird ~]$ crontab -r
[ian@lyrebird ~]$ crontab -l
no crontab for ian
```

To delete system cron or anacron jobs, edit `/etc/crontab`, `/etc/anacrontab`, or edit or delete files in the `/etc/cron.d` directory.

You can delete one or more jobs that were scheduled with the `at` command by using the `atrm` command with the job number. Multiple jobs should be separated by spaces. Listing 38 shows an example.

Listing 38. Displaying and removing jobs with `atq` and `atrm`

```
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
[ian@lyrebird ~]$ atrm 16 14 15
[ian@lyrebird ~]$ atq
17      Sat Jul 14 02:00:00 2007 a ian
```

Configure user access to job scheduling

If the file `/etc/cron.allow` exists, any non-root user must be listed in it in order to use `crontab` and the cron facility. If `/etc/cron.allow` does not exist, but `/etc/cron.deny` does exist, a non-root user who is listed in it cannot use `crontab` or the cron facility. If neither of these files exists, only the super user will be allowed to use this command. An empty `/etc/cron.deny` file allows all users to use the cron facility and is the default.

The corresponding `/etc/at.allow` and `/etc/at.deny` files have similar effects for the `at` facility.

Section 6. Data backup

This section covers material for topic 1.111.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Plan a backup strategy

- Dump a raw device to a file or restore a raw device from a file
- Perform partial and manual backups
- Verify the integrity of backup files
- Restore filesystems partially or fully from backups

Plan a backup strategy

Having a good backup is a necessary part of system administration, but deciding what to back up and when and how can be complex. Databases, such as customer orders or inventory, are usually critical to a business and many include specialized backup and recovery tools that are beyond the scope of this tutorial. At the other extreme, some files are temporary in nature and no backup is needed at all. In this section, we focus on system files and user data and discuss some of the considerations, methods, and tools for backup of such data.

There are three general approaches to backup:

1. A *full* backup is a complete backup, usually of a whole filesystem, directory, or group of related files. This takes the longest time to create, so it is usually used with one of the other two approaches.
2. A *differential* or *cumulative* backup is a backup of all things that have changed since the last full backup. Recovery requires the last full backup plus the latest differential backup.
3. An *incremental* backup is a backup of only those changes since the last incremental backup. Recovery requires the last full backup plus all of the incremental backups (in order) since the last full backup.

What to back up

When deciding what to back up, you should consider how volatile the data is. This will help you determine how often it should be backed up. Similarly, critical data should be backed up more often than non-critical data. Your operating system will probably be relatively easy to rebuild, particularly if you use a common image for several systems, although the files that customize each system would be more important to back up.

For programming staff, it may be sufficient to keep backups of repositories such as CVS repositories, while individual programmers' sandboxes may be less important. Depending on how important mail is to your operation, it may suffice to have infrequent mail backups, or it may be necessary to be able to recover mail to the

most recent date possible. You may want to keep backups of system cron files, but may not be so concerned about scheduled jobs for individual users.

The Filesystem Hierarchy Standard provides a classification of data that may help you with your backup choices. For details, see the tutorial [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#).

Once you have decided what to back up, you need to decide how often to do a full backup and whether to do differential or incremental backups in between those full backups. Having made those decisions, the following suggestions will help you choose appropriate tools.

Automating backups

In the previous section, you learned how to schedule jobs, and the cron facility is ideal for helping to automate the scheduling of your backups. However, backups are frequently made to removable media, particularly tape, so operator intervention is probably going to be needed. You should create and use backup scripts to ensure that the backup process is as automatic and repeatable as possible.

Dump and restore raw devices

One way to make a full backup of a filesystem is to make an image of the partition on which it resides. A *raw device*, such as `/dev/hda1` or `/dev/sda2`, can be opened and read as a sequential file. Similarly, it can be written from a backup as a sequential file. This requires no knowledge on the part of the backup tool as to the filesystem layout, but does require that the restore be done to space that is at least as large as the original. Some tools that handle raw devices are *filesystem aware*, meaning that they understand one or more of the Linux filesystems. These utilities can dump from a raw device but do not dump unused parts of the partition. They may or may not require restoration to the same or larger sized partition. The `dd` command is an example of the first type, while the `dump` command is an example of the second type that is specific to the `ext2` and `ext3` filesystems.

The `dd` command

In its simplest form, the `dd` command copies an input file to an output file, where either file may be a raw device. For backing up a raw device, such as `/dev/hda1` or `/dev/sda2`, the input file would be a raw device. Ideally, the filesystem on the device should be unmounted, or at least mounted read only, to ensure that data does not change during the backup. Listing 39 shows an example.

Listing 39. Backup a partition using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 of=backup-1
```

```
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 49.3103 s, 21.2 MB/s
```

The `if` and `of` parameters specify the input and output files respectively. In this example, the input file is a raw device, `dev/sda3`, and the output file is a file, `backup-1`, in the root user's home directory. To dump the file to tape or floppy disk, you would specify something like `of=/dev/fd0` or `of=/dev/st0`.

Note that 1,044,610,560 bytes of data was copied and the output file is indeed that large, even though only about 3% of this particular partition is actually used. Unless you are copying to a tape with hardware compression, you will probably want to compress the data. Listing 40 shows one way to accomplish this, along with the output of `ls` and `df` commands, which show you the file sizes and the usage percentage of the filesystem on `/dev/sda3`.

Listing 40. Backup with compression using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 | gzip > backup-2
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 117.723 s, 8.9 MB/s
[root@lyrebird ~]# ls -l backup-[12]
-rw-r--r-- 1 root root 1044610560 2007-07-08 15:17 backup-1
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
[root@lyrebird ~]# df -h /dev/sda3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       972M   28M  944M   3% /grubfile
```

The `gzip` compression reduced the file size to about 20% of the uncompressed size. However, unused blocks may contain arbitrary data, so even the compressed backup may be much larger than the total data on the partition.

If you divide the size by the number of records processed by `dd`, you will see that `dd` is writing 512-byte blocks of data. When copying to a raw output device such as tape, this can result in a very inefficient operation, so `dd` can read or write data in much larger blocks. Specify the `obs` option to change the output size or the `ibs` option to specify the input block size. You can also specify just `bs` to set both input and output block sizes to a common value.

If you need multiple tapes or other removable storage to store your backup, you will need to break it into smaller pieces using a utility such as `split`.

If you need to skip blocks such as disk or tape labels, you can do so with `dd`. See the man page for examples.

Besides just copying data, the `dd` command can do several conversions, such as between ASCII and EBCDIC, between big-endian and little-endian, or between variable-length data records and fixed-length data records. Obviously these

conversions are likely to be useful when copying real files rather than raw devices. Again, see the man page for details.

The dump command

The `dump` command can be used for full, differential, or incremental backups on `ext2` or `ext3` filesystems. Listing 41 shows an example.

Listing 41. Backup with compression using dump

```
[root@lyrebird ~]# dump -0 -f backup-4 -j -u /dev/sda3
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-4
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzip)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 12285 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:47:48 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-4
DUMP: Volume 1 completed at: Sun Jul  8 16:47:57 2007
DUMP: Volume 1 took 0:00:09
DUMP: Volume 1 transfer rate: 819 kB/s
DUMP: Volume 1 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: 12260 blocks (11.97MB) on 1 volume(s)
DUMP: finished in 9 seconds, throughput 1362 kBytes/sec
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Date this dump completed: Sun Jul  8 16:47:57 2007
DUMP: Average transfer rate: 819 kB/s
DUMP: Wrote 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[2-4]
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
-rw-r--r-- 1 root root 266932272 2007-07-08 15:44 backup-3
-rw-r--r-- 1 root root  7554939 2007-07-08 16:47 backup-4
```

In this example, `-0` specifies the *dump level* which is an integer, historically from 0 to 9, where 0 specifies a full dump. The `-f` option specifies the output file, which may be a raw device. Specify `-` to direct the output to stdout. The `-j` option specifies compression, with a default level of 2, using bzip compression. You can use the `-z` option to specify zlib compression if you prefer. The `-u` option causes the record of dump information, normally `/etc/dumpdates`, to be updated. Any parameters after the options represent a file or list of files, where the file may also be a raw device, as in this example. Notice how much smaller the backup is when the backup program is aware of the filesystem structure and can avoid the saving of unused blocks on the device.

If output is to a device such as tape, the `dump` command will prompt for another volume as each volume is filled. You can also provide multiple file names separated by commas. For example, if you wanted an unattended dump that required two tapes, you could load the tapes on `/dev/st0` and `/dev/st1`, schedule the `dump` command specifying both tapes as output, and go home to sleep.

When you specify a dump level greater than 0, an incremental dump is performed of all files that are new or have changed since the last dump at a lower level was taken. So a dump at level 1 will be a differential dump, even if a dump at level 2 or higher has been taken in the meantime. Listing 42 shows the result of updating the time stamp of an existing file on /dev/sda3 and creating a new file, then taking a dump at level 2. After that, another new file is created and a dump at level 1 is taken. The information from /etc/dumpdates is also shown. For brevity, part of the second dump output has been omitted.

Listing 42. Backup with compression using dump

```
[root@lyrebird ~]# dump -2 -f backup-5 -j -u /dev/sda3
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-5
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzlib)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 91 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:55:47 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-5
DUMP: Volume 1 completed at: Sun Jul  8 16:55:47 2007
DUMP: 90 blocks (0.09MB) on 1 volume(s)
DUMP: finished in less than a second
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date this dump completed: Sun Jul  8 16:55:47 2007
DUMP: Average transfer rate: 0 kB/s
DUMP: Wrote 90kB uncompressed, 15kB compressed, 6.000:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# echo "This data is even newer" >/grubfile/newerfile
[root@lyrebird ~]# dump -1 -f backup-6 -j -u -A backup-6-toc /dev/sda3
DUMP: Date of this level 1 dump: Sun Jul  8 17:08:18 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-6
...
DUMP: Wrote 100kB uncompressed, 16kB compressed, 6.250:1
DUMP: Archiving dump to backup-6-toc
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[4-6]
-rw-r--r-- 1 root root 7554939 2007-07-08 16:47 backup-4
-rw-r--r-- 1 root root  16198 2007-07-08 16:55 backup-5
-rw-r--r-- 1 root root  16560 2007-07-08 17:08 backup-6
[root@lyrebird ~]# cat /etc/dumpdates
/dev/sda3 0 Sun Jul  8 16:47:47 2007 -0400
/dev/sda3 2 Sun Jul  8 16:55:46 2007 -0400
/dev/sda3 1 Sun Jul  8 17:08:18 2007 -0400
```

Notice that backup-6 is, indeed, larger than backup 5. The level 1 dump illustrates the use of the `-A` option to create a table of contents that can be used to determine if a file is on an archive without actually mounting the archive. This is particularly useful with tape or other removable archive volumes. You will see these examples again when we discuss restoring data later in this section.

The `dump` command can dump files or subdirectories, but you cannot update

/etc/dumpdates and only level 0, of full dump, is supported.

Listing 43 illustrates the `dump` command dumping a directory, `/usr/include/bits`, and its contents to floppy disk. In this case, the dump will not fit on a single floppy, so a new volume is required. The prompt and response are shown in bold.

Listing 43. Backup a directory to multiple volumes using dump

```
[root@lyrebird ~]# dump -0 -f /dev/fd0 /usr/include/bits
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Dumping /dev/sdb9 (/ (dir usr/include/bits)) to /dev/fd0
DUMP: Label: /
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 2790 blocks.
DUMP: Volume 1 started with block 1 at: Mon Jul  9 16:03:30 2007
DUMP: dumping (Pass III) [directories]
DUMP: End of tape detected
DUMP: Closing /dev/fd0
DUMP: Volume 1 completed at: Mon Jul  9 16:04:49 2007
DUMP: Volume 1 1470 blocks (1.44MB)
DUMP: Volume 1 took 0:01:19
DUMP: Volume 1 transfer rate: 18 kB/s
DUMP: Change Volumes: Mount volume #2
DUMP: Is the new volume mounted and ready to go?: ("yes" or "no") y
DUMP: Volume 2 started with block 1441 at: Mon Jul  9 16:05:10 2007
DUMP: Volume 2 begins with blocks from inode 2
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing /dev/fd0
DUMP: Volume 2 completed at: Mon Jul  9 16:06:28 2007
DUMP: Volume 2 1410 blocks (1.38MB)
DUMP: Volume 2 took 0:01:18
DUMP: Volume 2 transfer rate: 18 kB/s
DUMP: 2850 blocks (2.78MB) on 2 volume(s)
DUMP: finished in 109 seconds, throughput 26 kBytes/sec
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Date this dump completed: Mon Jul  9 16:06:28 2007
DUMP: Average transfer rate: 18 kB/s
DUMP: DUMP IS DONE
```

If you back up to tape, remember that the tape will usually be rewound after each job. Devices with a name like `/dev/st0` or `/dev/st1` automatically rewind. The corresponding non-rewind equivalent devices are `/dev/nst0` and `/dev/nst1`. In any event, you can always use the `mt` command to perform magnetic tape operations such as forward spacing over files and records, back spacing, rewinding, and writing EOF marks. See the man pages for `mt` and `st` for additional information.

If you select the dump levels judiciously, you can minimize the number of archives you need to restore to any particular level. See the man pages for `dump` for a suggestion based on the Towers of Hanoi puzzle.

As with the `dd` command, there are many options that are not covered in this brief introduction. See the man pages for more details.

Partial and manual backups

So far, you have learned about tools that work well for backing up whole filesystems. Sometimes your backup needs to target selected files or subdirectories without backing up the whole filesystem. For example, you might need a weekly backup of most of your system, but daily backups of your mail files. Two other programs, `cpio` and `tar`, are more commonly used for this purpose. Both can write archives to files or to devices such as tape or floppy disk, and both can restore from such archives. Of the two, `tar` is more commonly used today, possibly because it handles complete directories better, and GNU `tar` supports both `gzip` and `bzip` compression.

Using `cpio`

The `cpio` command operates in *copy-out* mode to create an archive, *copy-in* mode to restore an archive, or *copy-pass* mode to copy a set of files from one location to another. You use the `-o` or `--create` option for copy-out mode, the `-i` or `--extract` option for copy-in mode, and the `-p` or `--pass-through` option for copy-pass mode. Input is a list of files provided on `stdin`. Output is either to `stdout` or to a device or file specified with the `-f` or `--file` option.

Listing 44 shows how to generate a list of files using the `find` command. Note the use of the `-print0` option on `find` to generate null-terminate strings for file names, and the corresponding `--null` option on `cpio` to read this format. This will correctly handle file names that have embedded blank or newline characters.

Listing 44. Back up a home directory using `cpio`

```
[root@lyrebird ~]# find ~ian -depth -print0 | cpio --null -o
>backup-cpio-1
18855 blocks
```

If you'd like to see the files listed as they are archived, add the `-v` option to `cpio`.

As with other commands that can archive to tape, the block size may be specified. For details on this and other options, see the man page.

Using `tar`

The `tar` (originally from *Tape ARchive*) creates an archive file, or *tarfile* or *tarball*, from a set of input files or directories; it also restores files from such an archive. If a directory is given as input to `tar`, all files and subdirectories are automatically included, which makes `tar` very convenient for archiving subtrees of your directory structure.

As with the other archiving commands we have discussed, output can be to a file, a

device such as tape or diskette, or stdout. The output location is specified with the `-f` option. Other common options are `-c` to create an archive, `-x` to extract an archive, `-v` for verbose output, which lists the files being processed, `-z` to use gzip compression, and `-j` to use bzip2 compression. Most `tar` options have a short form using a single hyphen and a long form using a pair of hyphens. The short forms are illustrated here. See the man pages for the long form and for additional options.

Listing 45 shows how to create a backup of the system cron jobs using `tar`.

Listing 45. Backup of system cron jobs using tar

```
[root@lyrebird ~]# tar -czvf backup-tar-1 /etc/*crontab /etc/cron.d
tar: Removing leading `/' from member names
/etc/anacrontab
/etc/crontab
/etc/cron.d/
/etc/cron.d/sa-update
/etc/cron.d/smolt
```

In the first line of output, you are told that `tar` will remove the leading slash (/) from member names. This allows files to be restored to some other location for verification before replacing system files. It is a good idea to avoid mixing absolute path names with relative path names when creating an archive, since all will be relative when restoring from the archive.

The `tar` command can append additional files to an archive using the `-r` or `--append` option. This may cause multiple copies of a file in the archive. In such a case, the *last* one will be restored during a restore operation. You can use the `--occurrence` option to select a specific file among multiples. If the archive is on a regular filesystem instead of tape, you may use the `-u` or `--update` option to update an archive. This works like appending to an archive, except that the time stamps of the files in the archive are compared with those on the filesystem, and only files that have been modified since the archived version are appended. As mentioned, this does not work for tape archives.

As with the other commands you have studied here, there are many options that are not covered in this brief introduction. See the man or info pages for more details.

Backup file integrity

Backup file integrity is extremely important. There is no point in having a backup if it is bad. A good backup strategy also involves checking your backups.

The first step to ensuring backup integrity is to ensure that you have properly captured the data you are backing up. If the filesystem is unmounted or mounted read only, this is usually straightforward as the data you are backing up cannot

change during your backup. If you must back up filesystems, directories, or files that are subject to modification while you are taking the backup, you should verify that no changes have been made during your backup. If changes were made, you will need to have a strategy for capturing them, either by repeating the backup, or perhaps by replacing or superseding the affected files in your backup. Needless to say, this will also affect your restore procedures.

Assuming you took good backups, you will periodically need to verify your backups. One way is to restore the backup to a spare volume and verify that it matches what you backed up. This is easiest to do right before you allow updates on the filesystem you are backing up. If you back up to media such as CD or DVD, you may be able to use the `diff` command as part of your backup procedure to ensure that your backup is good. Remember that even good backups can deteriorate in storage, so you should check periodically, even if you do verify at the time of backup. Keeping digests using programs such as `md5sum` or `sha1sum` is also a good check on the integrity of a backup file.

Restore filesystems from backups

A counterpart to backing up files is the ability to restore them when needed. Occasionally you will want to restore an entire filesystem, but it is far more common to need to restore only specific files or perhaps a set of directories. Almost always you will restore to some temporary space and verify that what you have restored is indeed what you want and is consistent with the current state of your system before actually making the restored files live.

A related issue is the need to verify that the items you want happen to be on a particular backup, as often happens when a user needs access to a version of a file that was modified or perhaps deleted "sometime in the last week or two." With these thoughts in mind, let's look at some of the restoration options.

Restoring a dd archive

Recall that the `dd` command was not filesystem aware, so you will need to restore a dump of a partition to find out what is on it. Listing 46 shows how to restore the partition that was dumped back in Listing 39 to a partition, `/dev/sdc7`, that was specially created on a removable USB drive just for this purpose.

Listing 46. Restoring a partition using dd

```
[root@lyrebird ~]# dd if=backup-1 of=/dev/sdc7
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 44.0084 s, 23.7 MB/s
```

Recall that we added some files to the filesystem on `/dev/sda3` after this backup was taken. If you mount the newly restore partition and compare it with the original, you will see that this is indeed the case, as shown in Listing 47. Note that the file whose timestamp was updated using `touch` is not shown here, as you would expect.

Listing 47. Comparing the restored partition with current state

```
[root@lyrebird ~]# mount /dev/sdc7 /mnt/temp-dd/
[root@lyrebird ~]# diff -rq /grubfile/ /mnt/temp-dd/
Only in /grubfile/: newerfile
Only in /grubfile/: newfile
```

Restoring a dump archive using restore

Recall that our final use of `dump` was a differential backup and that we created a table of contents. Listing 48 shows how to use `restore` to check the files in the archive created by `dump`, using the archive itself (`backup-5`) or the table of contents (`backup-6-toc`).

Listing 48. Checking the contents of archives

```
[root@lyrebird ~]# restore -t -f backup-5
Dump tape is compressed.
Dump   date: Sun Jul  8 16:55:46 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 2 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
[root@lyrebird ~]# restore -t -A backup-6-toc
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
Starting inode numbers by volume:
  Volume 1: 2
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
      17     ./newerfile
```

The `restore` command can also compare the contents of an archive with the contents of the filesystem using the `-C` option. In Listing 49 we updated `newerfile` and then compared the backup with the filesystem.

Listing 49. Comparing an archive with a filesystem using restore

```
[root@lyrebird ~]# echo "something different" >/grubfile/newerfile
[root@lyrebird ~]# restore -C -f backup-6
Dump tape is compressed.
```

```
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
fileSYS = /grubfile
./newerfile: size has changed.
Some files were modified!  1 compare errors
```

The `restore` command can restore interactively or automatically. Listing 50 shows how to restore `newerfile` to root's home directory (so you could examine it before replacing the updated file if needed), then replace the updated file with the backup copy. This example illustrates interactive restoration.

Listing 50. Restoring a file using restore

```
[root@lyrebird ~]# restore -i -f backup-6
Dump tape is compressed.
restore > ?
Available commands are:
  ls [arg] - list directory
  cd arg - change directory
  pwd - print current directory
  add [arg] - add `arg' to list of files to be extracted
  delete [arg] - delete `arg' from list of files to be extracted
  extract - extract requested files
  setmodes - set modes of requested directories
  quit - immediately exit program
  what - list dump header information
  verbose - toggle verbose flag (useful with ``ls'')
  prompt - toggle the prompt display
  help or `?' - print this list
If no `arg' is supplied, the current directory is used
restore > ls new*
newerfile
newfile
restore > add newerfile
restore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume # (none if no more volumes): 1
set owner/mode for '.'? [yn] y
restore > q
[root@lyrebird ~]# mv -f newerfile /grubfile
```

Restoring a cpio archive

The `cpio` command in copy-in mode (option `-i` or `--extract`) can list the contents of an archive or restore selected files. When you list the files, specifying the `--absolute-filenames` option reduces the number of extraneous messages that `cpio` will otherwise issue as it strips any leading `/` characters from each path that has one. Partial output from listing our previous archive is shown in Listing 51.

Listing 51. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio -id --list --absolute-filenames <backup-cpio-1
```

```
/home/ian/.gstreamer-0.10/registry.i686.xml
/home/ian/.gstreamer-0.10
/home/ian/.Trash/gnome-terminal.desktop
/home/ian/.Trash
/home/ian/.bash_profile
```

Listing 52 shows how to restore all the files with "samp" in their path name or file name. The output has been piped through `uniq` to reduce the number of "Removing leading '/' ..." messages. You must specify the `-d` option to create directories; otherwise, all files are created in the current directory. Furthermore, `cpio` will not replace any newer files on the filesystem with archive copies unless you specify the `-u` or `--unconditional` option.

Listing 52. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio -ivd "*samp*" < backup-cpio-1 2>&1 |uniq
cpio: Removing leading `/' from member names
home/ian/crontab.samp
cpio: Removing leading `/' from member names
home/ian/sample.file
cpio: Removing leading `/' from member names
18855 blocks
```

Restoring a tar archive

The `tar` command can also compare archives with the current filesystem as well as restore files from archives. Use the `-d`, `--compare`, or `--diff` option to perform comparisons. The output will show files whose contents differ as well as files whose time stamps differ. Listing 53 shows verbose output (using option `-v`), from a comparison of the file created earlier and the files in `/etc` after `/etc/crontab` has been touched to alter its time stamp. The option `directory /` instructs `tar` to perform the comparison starting from the root directory rather than the current directory.

Listing 53. Comparing archives and files using tar

```
[root@lyrebird ~]# touch /etc/crontab
[root@lyrebird ~]# tar --diff -vf backup-tar-1 --directory /
etc/anacrontab
etc/crontab
etc/crontab: Mod time differs
etc/cron.d/
etc/cron.d/sa-update
etc/cron.d/smolt
```

Listing 54 shows how to extract just `/etc/crontab` and `/etc/anacrontab` into the current directory.

Listing 54. Extracting archive files using tar

```
[root@lyrebird ~]# tar -xzvf backup-tar-1 "*tab"
```

```
etc/anacrontab
etc/crontab
```

Note that `tar`, in contrast to `cpio` creates the directory hierarchy for you automatically.

The next section of this tutorial shows you how to maintain system time.

Section 7. System time

This section covers material for topic 1.111.6 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

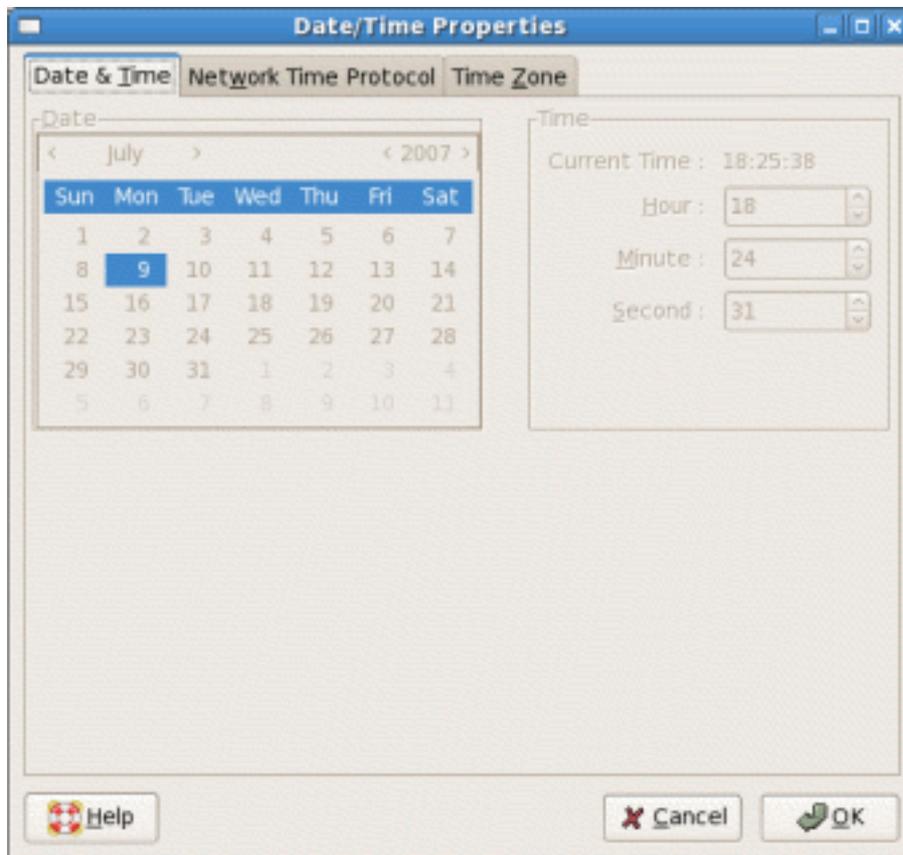
- Set the system date and time
- Set the BIOS clock to the correct UTC time
- Configure your time zone
- Configure the Network Time Protocol (NTP) service, including correcting for clock drift

Set the system date and time

System time on a Linux system is very important. You saw earlier how the cron and anacron facilities do things based on time, so they need an accurate time to base decisions on. Most of the backup and restore tools discussed in the previous section, along with development tools such as `make`, also depend on reliable time measurements. Most computers built since around 1980 include some kind of clock mechanism, and most built since 1984 or so have a persistent clock mechanism that keeps time even if the computer is turned off.

If you installed a Linux system graphically, you probably set the clock and chose a time zone suitable for your needs. You may have elected to use the Network Time Protocol (NTP) to set your clock, and you may or may not have elected to keep the system clock using Coordinated Universal Time or UTC. If you subsequently went to set the clock using graphical tools on a Fedora or Red Hat or similar system, you may have seen a dialog box like that in Figure 3.

Figure 3. Updating the date and time



Surprise! You can't actually set the clock yourself using this dialog. In this section you learn more about the difference between local clocks and NTP and how to set your system time.

No matter whether you live in New York, Budapest, Nakhodka, Ulan Bator, Bangkok, or Canberra, most of your Linux time computations are related to Coordinated Universal Time or UTC. If you run a dedicated Linux system, it is customary to keep the hardware clock set to UTC, but if you also boot another operating system such as Windows, you may need to set the hardware clock to local time. It really doesn't matter as far as Linux is concerned, except that there happen to be two different methods of keeping track of time zones internally in Linux, and if they don't agree, you can wind up with some odd time stamps on FAT filesystems, among other things. Listing 55 shows you how to use the `date` command to display the current date and time. The display is always in local time, even if your hardware clock keeps UTC time.

Listing 55. Displaying the current date and time

```
[root@lyrebird ~]# date;date -u
Mon Jul  9 22:40:01 EDT 2007
```

The `date` command supports a wide variety of possible output formats, some of which you already saw back in [Listing 28](#). See the man page for `date` if you'd like to learn more about the various date formats.

If you need to set the date, you can do this by providing a date and time as an argument. The required format is historical and is somewhat odd even to Americans and truly odd to the rest of the world. You must specify at least month, day, hour, and minute in MMDDhhmm format, and you may also append a two- or four-digit year (CCYY or YY) and optionally a period (.) followed by a two-digit number of seconds. Listing 56 shows an example that alters the system date by a little over a minute.

Listing 56. Setting the system date and time

```
[root@lyrebird ~]# date; date 0709221407;date
Mon Jul  9 23:12:37 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
```

Set the BIOS clock to UTC time

Your Linux system, along with most other current operating systems, actually has two clocks. The first is the hardware clock, sometimes called the Real Time Clock, RTC, or BIOS clock, which is usually tied to an oscillating quartz crystal that is accurate to within a few seconds per day. It is subject to variations such as ambient temperature. The second is the internal software clock, which is driven by counting system interrupts. It is subject to variations caused by high system load and interrupt latency. Nevertheless, your system typically reads the hardware clock at startup and from then on uses the software clock. The `date` command that you just learned about sets the software clock, not the hardware clock.

If you use the Network Time Protocol (NTP), you may possibly set the hardware clock when you first install the system and never worry about it again. If not, this part of the tutorial will show you how to display and set the hardware clock time.

You can use the `hwclock` command to display the current value of the hardware clock. Listing 57 shows the current value of both the system and hardware clocks.

Listing 57. System and hardware clock values

```
[root@lyrebird ~]# date;hwclock
Mon Jul  9 22:16:11 EDT 2007
Mon 09 Jul 2007 11:14:49 PM EDT -0.071616 seconds
```

Notice that the two values are different. You can synchronize the hardware clock

from the system clock using the `-w` or `--systohc` option of `hwclock`, and you can synchronize the system clock from the hardware clock using the `-s` or `--hctosys` option, as shown in Listing 58.

Listing 58. Setting the system clock from the hardware clock

```
[root@lyrebird ~]# date;hwclock;hwclock -s;date
Mon Jul  9 22:20:23 EDT 2007
Mon 09 Jul 2007 11:19:01 PM EDT  -0.414881 seconds
Mon Jul  9 23:19:02 EDT 2007
```

You may specify either the `--utc` or the `--localtime` option to have the system clock kept in UTC or local time. If no value is specified, the value is taken from the third line of `/etc/adjtime`.

The Linux kernel has a mode that copies the system time to the hardware clock every 11 minutes. This is off by default, but is turned on by NTP. Running anything that set the time the old fashioned way, such as `hwclock --hctosys`, turns it off, so it's a good idea to just let NTP do its work if you are using NTP. See the man page for `adjtimex` to find out how to check whether the clock is being updated every 11 minutes or not. You may need to install the `adjtimex` package as it is not always installed by default.

The `hwclock` command keeps track of changes made to the hardware clock in order to compensate for inaccuracies in the clock frequency. The necessary data points are kept in `/etc/adjtime`, which is an ASCII file. If you are not using the Network Time Protocol, you can use the `adjtimex` command to compensate for clock drift. Otherwise, the hardware clock will be adjusted approximately every 11 minutes by NTP. Besides showing whether your hardware clock is in local or UTC time, the first value in `/etc/adjtime` shows the amount of hardware clock drift per day (in seconds). Listing 59 shows two examples.

Listing 59. /etc/adjtime showing clock drift and local or UTC time.

```
[root@lyrebird ~]# cat /etc/adjtime
0.000990 1184019960 0.000000
1184019960
LOCAL
root@pinguino:~# cat /etc/adjtime
-0.003247 1182889954 0.000000
1182889954
LOCAL
```

Note that both these systems keep the hardware clock in local time, but the clock drifts are different — 0.000990 on lyrebird and -0.003247 on pinguino.

Configure your time zone

Your time zone is a measure of how far your local time differs from UTC. Information on available time zones that can be configured is kept in `/usr/share/zoneinfo`. Traditionally, `/etc/localtime` was a link to one of the time zone files in this directory tree, for example, `/usr/share/zoneinfo/Eire` or `/usr/share/zoneinfo/Australia/Hobart`. On modern systems it is much more likely to be a copy of the appropriate time zone data file since the `/usr/share` filesystem may not be mounted when the local time zone information is needed early in the boot process.

Similarly, another file, `/etc/timezone` was traditionally a link to `/etc/default/init` and was used to set the time zone environment variable `TZ`, and several locale-related environment variables. The file may or may not exist on your system. If it does, it may simply contain the name of the current time zone. You may also find time zone information in `/etc/sysconfig/clock`. Listing 60 shows these files from a Ubuntu 7.04 and a Fedora 7 system.

Listing 60. Time zone information in `/etc`

```
root@pinguino:~# cat /etc/timezone
America/New_York

[root@lyrebird ~]# cat /etc/sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
# The timezone of the system is defined by the contents of
/etc/localtime.
ZONE="America/New York"
UTC=false
ARC=false
```

Some systems such as Debian and Ubuntu have a `tzconfig` command to set the time zone. Others such as Fedora use `system-config-date` to set the time zone and to indicate whether the clock uses UTC or not. Listing 61 illustrates the use of the `tzconfig` command to display the current time zone.

Listing 61. Setting time zone with `tzconfig`

```
root@pinguino:~# tzconfig
Your current time zone is set to America/New_York
Do you want to change that? [n]:
Your time zone will not be changed
```

Configure the Network Time Protocol

The *Network Time Protocol (NTP)* is a protocol to synchronize computer clocks over a network. Synchronization is usually to UTC.

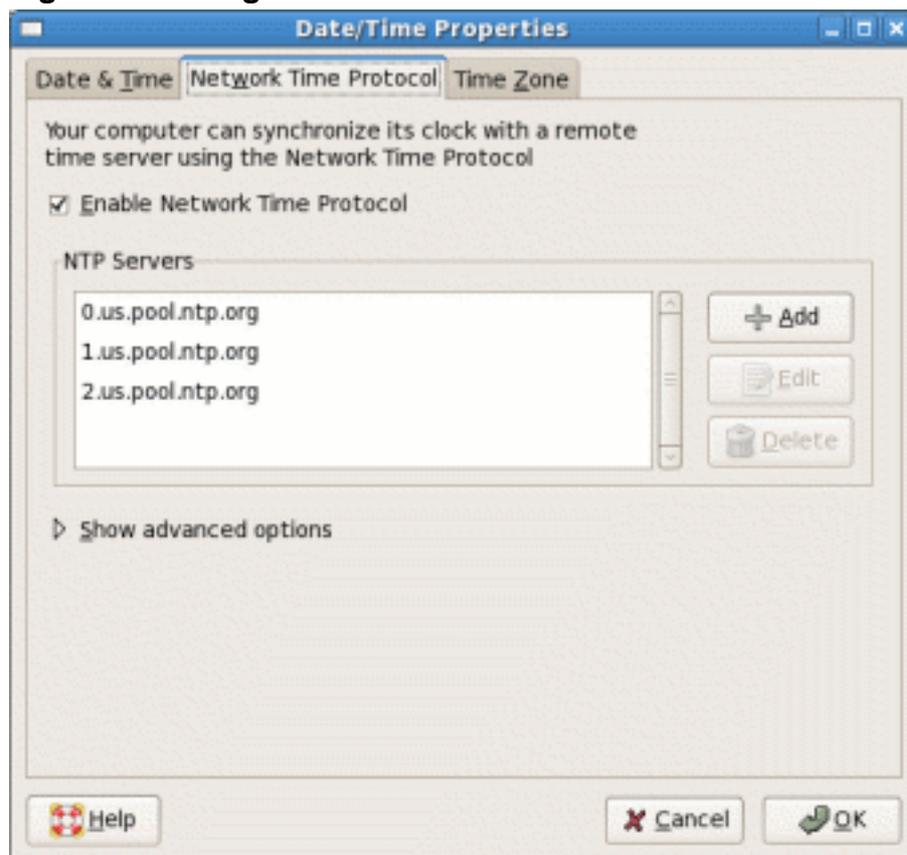
NTP version 3 is an Internet draft standard, formalized in RFC 1305. The current development version, NTP version 4 is a significant revision, which has not been formalized in an RFC. RFC 4330 describes Simple NTP (SNTP) version 4.

Time synchronization is accomplished by sending messages to *time servers*. The time returned is adjusted by an offset of half the round-trip delay. The accuracy of the time is therefore dependent on the network latency and the extent to which the latency is the same in both directions. The shorter the path to a time server, the more accurate the time is likely to be. See [Resources](#) for more detailed information than this simplistic description can provide.

There is a huge number of computers on the Internet, so time servers are organized into *strata*. A relatively small number of stratum 1 servers maintain very accurate time from a source such as an atomic clock. A larger number of stratum 2 servers get their time from stratum 1 servers and make it available to an even larger number of stratum 3 servers, and so on. To ease the load on time servers, a large number of volunteers donate time services through pool.ntp.org (see [Resources](#) for a link). Round robin DNS servers accomplish NTP load balancing by distributing NTP server requests among a pool of available servers.

If you use a graphical interface, you might be able to set your NTP time servers using a dialog similar to that in Figure 4. The fact that this system has enabled automatic time updates using NTP is why the dialog in Figure 3 did not allow the date and time to be changed.

Figure 4. Setting NTP servers



NTP configuration information is kept in `/etc/ntp.conf`, so you can also edit that file and then restart the `ntpd` daemon after you save it. Listing 62 shows an example `/etc/ntp.conf` file using the time servers from Figure 4.

Listing 62. Setting time zone with `tzconfig`

```
[root@lyrebird ~]# cat /etc/ntp.conf
# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#broadcast 192.168.1.255 key 42          # broadcast server
#broadcastclient          # broadcast client
#broadcast 224.0.1.1 key 42            # multicast server
#multicastclient 224.0.1.1            # multicast client
#manycastserver 239.255.254.254        # manycast server
#manycastclient 239.255.254.254 key 42 # manycast client

# Undisciplined Local Clock. This is a fake driver intended for backup
# and when no outside source of synchronized time is available.
#server 127.127.1.0 # local clock
#fudge 127.127.1.0 stratum 10

# Drift file. Put this in a directory which the daemon can write to.
# No symbolic links allowed, either, since the daemon updates the file
# by creating a temporary in the same directory and then rename()'ing
# it to the file.
driftfile /var/lib/ntp/drift

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8
server 0.us.pool.ntp.org
restrict 0.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 1.us.pool.ntp.org
restrict 1.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 2.us.pool.ntp.org
restrict 2.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
```

If you are using the `pool.ntp.org` time servers, these may be anywhere in the world. You will usually get better time by restricting your servers as in this example where `us.pool.ntp.org` is used, resulting in only U.S. servers being chosen. See [Resources](#)

for more information on the ntp.pool.org project.

NTP commands

You can use the `ntpdate` command to set your system time from an NTP time server as shown in Listing 63.

Listing 63. Setting system time from an NTP server using ntpdate

```
[root@lyrebird ~]# ntpdate 0.us.pool.ntp.org
10 Jul 10:27:39 ntpdate[15308]: adjust time server 66.199.242.154 offset
-0.007271 sec
```

Because the servers operate in round robin mode, the next time you run this command you will probably see a different server. Listing 64 shows the first few DNS responses for `0.us.ntp.pool.org` a few moments after the above `ntpdate` command was run.

Listing 64. Round robin NTP server pool

```
[root@lyrebird ~]# dig 0.pool.ntp.org +noall +answer | head -n 5
0.pool.ntp.org. 1062 IN A 217.116.227.3
0.pool.ntp.org. 1062 IN A 24.215.0.24
0.pool.ntp.org. 1062 IN A 62.66.254.154
0.pool.ntp.org. 1062 IN A 76.168.30.201
0.pool.ntp.org. 1062 IN A 81.169.139.140
```

The `ntpdate` command is now deprecated as the same function can be done using `ntpq` with the `-q` option, as shown in Listing 65.

Listing 65. Setting system time using ntpd -q

```
[root@lyrebird ~]# ntpd -q
ntpd: time slew -0.014406s
```

Note that the `ntpd` command uses the time server information from `/etc/ntp.conf`, or a configuration file provided on the command line. See the man page for more information and for information about other options for `ntpd`. Be aware also that if the `ntpd` daemon is running, `ntpd -q` will quietly exit, leaving a failure message in `/var/log/messages`.

Another related command is the `ntpq` command, which allows you to query the NTP daemon. See the man page for more details.

This brings us to the end of this tutorial. We have covered a lot of material on system administration. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- See the [Partimage homepage](#) for information on Partimage, a filesystem-aware partition dump and restore tool.
- "[/etc: Host-specific system configuration](#)" describes the Linux Standard Base (LSB) requirements for /etc.
- The [Network Time Protocol Project](#) produces a reference implementation of the NTP protocol, and implementation documentation.
- The [Network Time Synchronization Project](#) maintains an extensive array of documentation and background information, including briefing slides, on network time protocols.
- The [pool.ntp.org project](#) is a big virtual cluster of timeservers striving to provide reliable easy to use NTP service for millions of clients without putting a strain on the big popular timeservers.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content.](#)
- Get involved in the [developerWorks community](#) through our developer blogs, forums, podcasts, and community topics in our new [developerWorks spaces](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 111: Administrative tasks

Junior Level Administration (LPIC-1) topic 111

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

10 Jul 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this sixth in a [series of nine tutorials](#), Ian introduces you to administrative tasks. By the end of this tutorial, you will know how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *intermediate level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active intermediate level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the four junior and intermediate certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts.
Topic 111	LPI exam 102 prep: Administrative tasks	(This tutorial.) Learn how to manage user and group accounts and tune user and system environments, configure and use system log files, automate system administration tasks by scheduling jobs to run at another time, back up your system, and maintain system time. See the detailed objectives below.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Administrative tasks," the sixth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.111.1 User and group accounts	Weight 4	Add, remove, suspend, and change user accounts. Manage user and group information in password and group databases, including shadow databases. Create and manage special purpose and limited accounts.
1.111.2 Tune user and system environments	Weight 3	Modify global and user profiles. Set environment variables and maintain skeleton directories for new user accounts. Set command search paths.
1.111.3 Configure and use system log files to meet administrative and security needs	Weight 3	Configure and manage system logs, including the type and level of logged information. Scan and monitor log files for

		notable activity and track down noted problems. Rotate and archive log files.
1.111.4 Automate system administration tasks by scheduling jobs to run in the future	Weight 4	Use the <code>cron</code> or <code>anacron</code> commands to run jobs at regular intervals, and use the <code>at</code> command to run jobs at a specific time.
1.111.5 Maintain an effective data backup strategy	Weight 3	Plan a backup strategy and back up filesystems automatically to various media.
1.111.6 Maintain system time	Weight 4	Maintain the system time and time zone, and synchronize the clock via NTP. Set the BIOS clock to the correct time in UTC, and configure NTP, including correcting for clock drift.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be thoroughly familiar with the material from the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, which covers basic concepts of users, groups, and file permissions.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. User and group accounts

This section covers material for topic 1.111.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Add, modify, and remove users and groups
- Suspend and change user accounts
- Manage user and group information in the password databases and group databases
- Use the correct tools to manage shadow password databases and group databases
- Create and manage limited and special-purpose accounts

As you learned in the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, Linux is a multi-user system where each user belongs to one *primary* group and possibly to additional groups. Ownership of files in Linux is closely related to user ids and groups. Recall that you can log in as one user and become another user using the `su` or `sudo -s` commands, and that you can use the `whoami` command to check your current effective id and the `groups` command to find out what groups you belong to. In this section, you learn how to create, delete, and manage users and groups. You also learn about the files in `/etc`, where user and group information is stored.

Add and remove users and groups

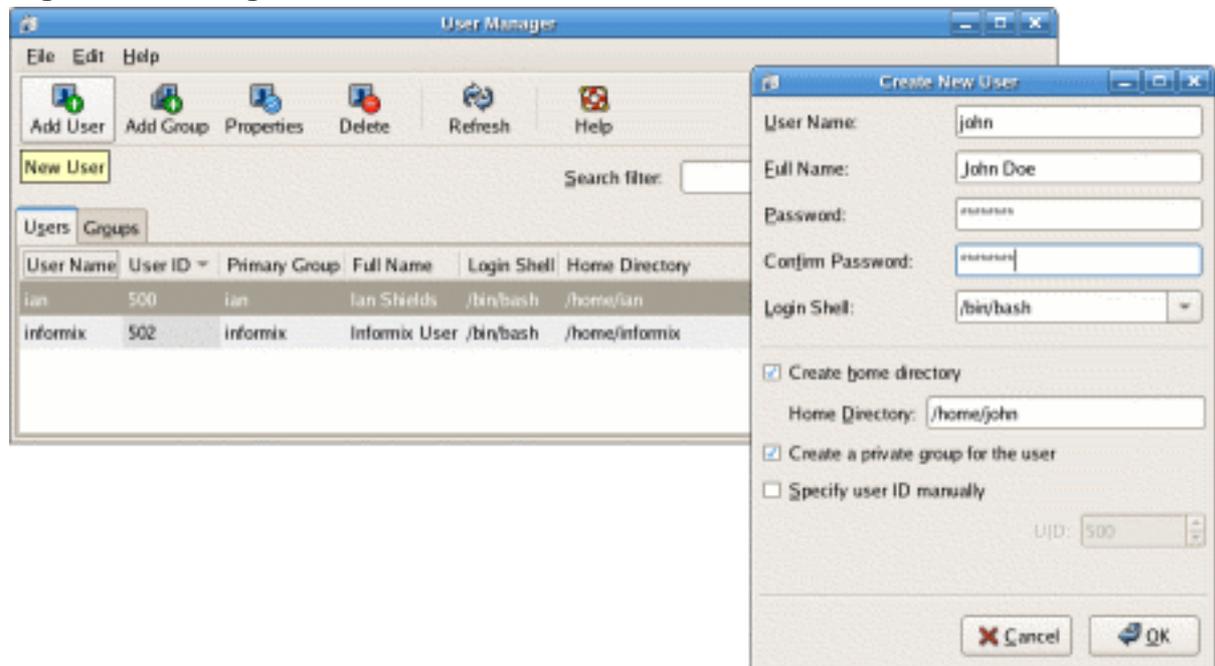
You add a user to a Linux system using the `useradd` command, and you delete a user using the `userdel` command. Similarly, you add or delete groups using the `groupadd` and `groupdel` commands.

Adding a user or group

Modern Linux desktops usually have graphical interfaces for user and group administration. The graphical interface is usually accessed through menu options for system administration. These interfaces do vary considerably, so the one on your system may not look much like the example here, but the underlying concepts and commands remain similar.

Let's start by adding a user to a Fedora Core 5 system graphically, and then examine the underlying commands. In the case of Fedora Core 5 with GNOME desktop, use **System > Administration > Users and Groups**, then click the **Add User** button.

Figure 1 depicts the User Manager panel with the Create New User panel showing basic information for a new user named 'john'. The full name of the user, John Doe, and a password have been entered. The panel provides a default login shell of `/bin/bash`. On Fedora systems, the default is to create a new group with the same name as the user, 'john' in this case, and a home directory of `/home/john`.

Figure 1. Adding a user

Listing 1 shows the use of the `id` command to display basic information about the new user. As you can see, john has user number 503 and a matching group, john, with group number 503. This is the only group of which john is a member.

Listing 1. Displaying user id information

```
[root@pinguino ~]# id john
uid=503(john) gid=503(john) groups=503(john)
```

To accomplish the same task from the command line, you use the `groupadd` and `useradd` commands to create the group and user, then use the `passwd` command to set the password for the newly created user. All of these commands require root authority. The basic use of these commands to add another user, jane, is illustrated in Listing 2.

Listing 2. Adding user jane

```
[root@pinguino ~]# groupadd jane
[root@pinguino ~]# useradd -c "Jane Doe" -g jane -m jane
[root@pinguino ~]# passwd jane
Changing password for user jane.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# id jane
uid=504(jane) gid=504(jane) groups=504(jane)
[root@pinguino ~]# ls -ld /home/jane
drwx----- 3 jane jane 4096 Jun 25 18:22 /home/jane
```

In these two examples, both the user id and the group id have values greater than 500. Be aware that some newer systems start user ids at 1000 rather than 500. These values normally signify ordinary users, while values below 500 (or 1000 if the system starts ordinary users at 1000) are reserved for *system users*. [System users](#) are covered later in this section. The actual cutoff points are set in `/etc/login.defs` as `UID_MIN` and `GID_MIN`.

In Listing 2 above, the `groupadd` command has a single parameter, `jane`, the name of the group to be added. Group names must begin with a lower case letter or an underscore, and usually contain only these along with hyphens or dashes. Options you may specify are shown in Table 3.

Option	Purpose
-f	Exit with success status if the group already exists. This is handy for scripting when you do not need to check if a group exists before attempting to create it.
-g	Specifies the group id manually. The default is to use the smallest value that is at least <code>GID_MIN</code> and also greater than the id of any existing group. Group ids are normally unique and must be non-negative
-o	Permits a group to have a non-unique id.
-K	Can be used to override defaults from <code>/etc/login.defs</code> .

In Listing 2 above, the `useradd` command has a single parameter, `jane`, the name of the user to be added, along with the `-c`, `-g`, and `-m` options. Common options for the `useradd` command are shown in Table 4.

Option	Purpose
-b --base-dir	The default base directory in which user home directories are created. This is usually <code>/home</code> , and the user's home directory is <code>/home/\$USER</code> .
-c --comment	A text string describing the id, such as the user's full name.
-d --home	Provides a specific directory name for the home directory.
-e	The date on which the account will

--expiredate	expire or be disabled in the form YYYY-MM_DD.
-g --gid	The name or number of the initial login group for the user. The group must exist, which is why group jane was created before user jane in Listing 2.
-G --groups	A comma-separated list of additional groups to which the user belongs.
-K	Can be used to override defaults from /etc/login.defs.
-m --create-home	Create the user's home directory if it does not exist. Copy the skeleton files and any directories from /etc/skel to the home directory.
-o --non-unique	Permits a user to have a non-unique id.
-p --password	The encrypted password. If a password is not specified, the default is to disable the account. You will usually use the <code>passwd</code> command in a subsequent step rather than generating an encrypted password and specifying it on the <code>useradd</code> command.
-s --shell	The name of the user's login shell if different from the default login shell.
-u --uid	The non-negative numerical userid, which must be unique if <code>-o</code> is not specified. The default is to use the smallest value that is at least <code>UID_MIN</code> and also greater than the id of any existing user.

Notes:

1. Some systems, including Fedora and Red Hat distributions, have extensions to the user-creation commands. For example, the default Fedora and Red Hat behavior is to create a new group for a user, and the `-n` option can be used on the `useradd` command to disable this function. Be aware of such possible system differences and refer to the man pages on your system when in doubt.
2. On SUSE systems, use YaST or YaST2 to access graphical user and group administration interfaces.
3. Graphical interfaces may perform additional tasks such as creating the user's mail file in `/var/spool/mail`.

Deleting a user or group

Deleting a user or group is much simpler than adding one, because there are fewer options. In fact, the `groupdel` command to delete a group requires only the group name; it has no options. You cannot delete any group that is the primary group of a user. If you use a graphical interface for deleting users and groups, the functions are very similar to the commands shown here.

Use the `userdel` command to delete a user. The `-r`, or `--remove` option requests removal of the user's home directory and anything it contains, along with the user's mail spool. When you delete a user, a group with the same name as the user will also be deleted if `USERGROUPS_ENAB` is set to `yes` in `/etc/login.defs`, but this will be done only if the group is not the primary group of another user.

In Listing 3 you see an example of deleting groups when multiple users share the same primary group. Here, another user, `jane2`, has previously been added to the system with the same group as `jane`.

Listing 3. Deleting users and groups

```
root@pinguino:~# groupdel jane
groupdel: cannot remove user's primary group.
root@pinguino:~# userdel -r jane
userdel: Cannot remove group jane which is a primary group for another
user.
root@pinguino:~# userdel -r jane2
root@pinguino:~# groupdel jane
```

Notes:

1. There is a `userdel` option, `-f` or `--force`, which can be used to delete users and their group. This option is dangerous, so you should use it only as a last resort. Read the man page carefully before you do.
2. Be aware that if you delete a user or group, and there are files that belong to that user or group on your filesystem, then the files are not automatically deleted or assigned to another user or group.

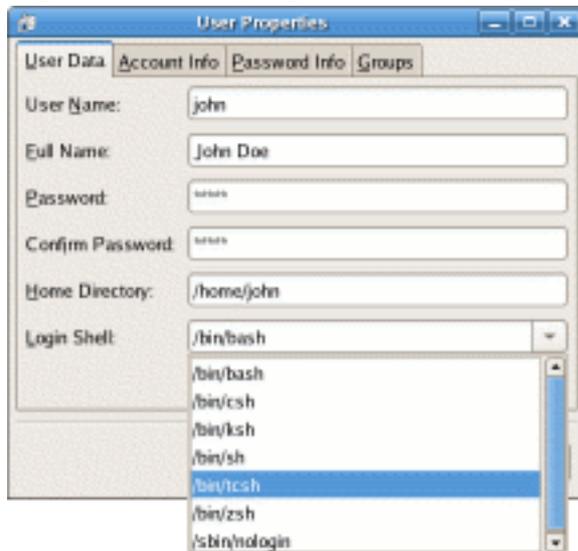
Suspend and change accounts

Now that you can create or delete a user id or a group, you may also find a need to modify one.

Modifying user accounts

Suppose user john wishes to have the tcsh shell as his default. From a graphical interface you will usually find a way to either edit a user (or group), or to examine the properties of the object. Figure 2 shows the properties dialog for the user john that we created earlier on a Fedora Core 5 system.

Figure 2. Modifying a user account



From the command line, you can use the `usermod` command to modify a user account. You can use most of the options that you use with `useradd`, except that you cannot create or populate a new home directory for the user. If you need to change the name of the user, specify the `-l` or `--login` option with the new name. You will probably want to rename the home directory to match the user id. You may also need to rename other items such as mail spool files. Finally, if the login shell is changed, some of the associated profile files may need to be altered. Listing 4 shows an example of the things you might need to do to change user john to john2 with `/bin/tcsh` as the default shell and renamed home directory `/home/john2`.

Listing 4. Modifying a user

```
[root@pinguino ~]# usermod -l john2 -s /bin/tcsh -d /home/john2 john
[root@pinguino ~]# ls -d ~john2
ls: /home/john2: No such file or directory
[root@pinguino ~]# mv /home/john /home/john2
[root@pinguino ~]# ls -d ~john2
/home/john2
```

Notes:

1. If you need to modify a user's additional groups, you must specify the complete list of additional groups. There is no command to simply add or delete a single group for a user.

2. There are restrictions on changing the name or id of a user who is logged in or who has running processes. Check the man pages for details.
3. If you change a user number, you may want to change files and directories owned by that user to match the new number.

Modifying groups

Not surprisingly, the `groupmod` command is used to modify group information. You can change the group number with the `-g` option, and the name with the `-n` option.

Listing 5. Renaming a group

```
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# groupmod -n john2 john
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

Notice in Listing 5 that the group name for the home directory of john2 magically changed when we used `groupmod` to change the group name. Are you surprised? Because groups are represented in the filesystem inodes by their number rather than by their name, this is not surprising. However, if you change a group's number, you should update any users for which that group is the primary group, and you may also want to update the files and directories belonging to that group to match the new number (in the same way as noted above for changing a user number). Listing 6 shows how to change the group number for john2 to 505, update the user account, and make appropriate changes to all the affected files in the `/home` filesystem. You probably want renumbering users and groups if at all possible.

Listing 6. Renumbering a group

```
[root@pinguino ~]# groupmod -g 505 john2
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# id john2
uid=503(john2) gid=503 groups=503
[root@pinguino ~]# usermod -g john2 john2
[root@pinguino ~]# id john2
uid=503(john2) gid=505(john2) groups=505(john2)
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# find /home -gid 503 -exec chgrp john2 {} \;
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

User and group passwords

You have already seen the `passwd` command, which is used to change a user password. The password is (or should be) unique to the user and may be changed by user. The root user may change any user's password as we have already seen.

Groups may also have passwords, and the `gpasswd` command is used to set them. Having a group password allows users to join a group temporarily with the `newgrp` command, if they know the group password. Of course, having multiple people knowing a password is somewhat problematic, so you will have to weigh the advantages of adding a user to a group using `usermod`, versus the security issue of having too many people knowing the group password.

Suspending or locking accounts

If you need to prevent a user from logging in, you can *suspend* or *lock* the account using the `-L` option of the `usermod` command. To *unlock* the account, use the `-U` option. Listing 7 shows how to lock account `john2` and what happens if `john2` attempts to log in to the system. Note that when the `john2` account is unlocked, the same password is restored.

Listing 7. Locking an account

```
[root@pinguino ~]# usermod -L john2
[root@pinguino ~]# ssh john2@pinguino
john2@pinguino's password:
Permission denied, please try again.
```

You may have noticed back in [Figure 2](#) that there were several tabs on the dialog box with additional user properties. We briefly mentioned the use of the `passwd` command for setting user passwords, but both it and the `usermod` command can perform many tasks related to user accounts, as can another command, the `chage` command. Some of these options are shown in Table 5. Refer to the appropriate man pages for more details on these and other options.

Table 5. Commands and options for changing user accounts			
	Option for command		Purpose
Usermod	Passwd	Chage	
-L	-l	N/A	Lock or suspend the account.
-U	-u	N/A	Unlock the account.
N/A	-d	N/A	Disable the account by setting it passwordless.

-e	-f	-E	Set the expiration date for an account.
N/A	-n	-m	The minimum password lifetime in days.
N/A	-x	-M	The maximum password lifetime in days.
N/A	-w	-W	The number of days of warning before a password must be changed.
-f	-i	-l	The number of days after a password expires until the account is disabled.
N/A	-S	-l	Output a short message about the current account status.

Manage user and group databases

The primary repositories for user and group information are four files in `/etc`.

`/etc/passwd`

is the *password* file containing basic information about users

`/etc/shadow`

is the *shadow password* file containing encrypted passwords

`/etc/group`

is the *group* file containing basic information about groups and which users belong to them

/etc/gshadow

is the *shadow group* file containing encrypted group passwords

These files are updated by the commands you have already seen in this tutorial and you will meet some more commands for working with them after we discuss the files themselves. All of these files are plain text files. In general, you should not edit them directly. Use the tools provided for updating them so they are properly locked and kept synchronized.

You will note that the passwd and group files are both *shadowed*. This is for security reasons. The passwd and group files themselves must be world readable, but the encrypted passwords should not be world readable. Therefore, the shadow files contain the encrypted passwords, and these files are only readable by root. The necessary authentication access is provided by an suid program that has root authority, but can be run by anyone. Make sure that your system has the permissions set appropriately. Listing 8 shows an example.

Listing 8. User and group database permissions

```
[ian@pinguino ~]$ ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow
-rw-r--r-- 1 root root 701 Jun 26 19:04 /etc/group
-r----- 1 root root 580 Jun 26 19:04 /etc/gshadow
-rw-r--r-- 1 root root 1939 Jun 26 19:43 /etc/passwd
-r----- 1 root root 1324 Jun 26 19:50 /etc/shadow
```

Note: Although it is still technically possible to run without shadowed password and group files, this is almost never done and is not recommended.

The /etc/passwd file

The /etc/passwd file contains one line for each user in the system. Some example lines are shown in Listing 9.

Listing 9. /etc/password entries

```
root:x:0:0:root:/root:/bin/bash
jane:x:504:504:Jane Doe:/home/jane:/bin/bash
john2:x:503:505:John Doe:/home/john2:/bin/tcsh
```

Each line contains seven fields separated by colons (:), as shown in Table 6.

Table 6. Fields in /etc/passwd	
Field	Purpose
Username	The name used to log in to the system.

	For example, john2.
Password	The encrypted password. When using shadow passwords, it contains a single x character.
User id (UID)	The number used to represent this user name in the system. For example, 503 for user john2.
Group id (GID)	The number used to represent this user's primary group in the system. For example, 505 for user john2.
Comment (GECOS)	An optional field used to describe the user. For example, "John Doe". The field may contain multiple comma-separated entries. It is also used by programs such as <code>finger</code> . The GECOS name is historic. See details in <code>man 5 passwd</code> .
Home	The absolute path the user's home directory. For example, <code>/home/john2</code>
Shell	The program automatically launched when a user logs in to the system. This is usually an interactive shell such as <code>/bin/bash</code> or <code>/bin/tcsh</code> , but may be any program, not necessarily an interactive shell.

The `/etc/group` file

The `/etc/group` file contains one line for each group in the system. Some example lines are shown in Listing 10.

Listing 10. `/etc/group` entries

```
root:x:0:root
jane:x:504:john2
john2:x:505:
```

Each line contains four fields separated by colons (:), as shown in Table 7.

Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The encrypted password. When using shadow group passwords, it contains a single x character.

Group id (GID)	The number used to represent this group in the system. For example, 505 for group john2.
Members	A comma-separated list of group members, excepting those members for whom this is the primary group.

Shadow files

The file `/etc/shadow` should only be readable by root. It contains encrypted passwords, along with password and account expiration information. See the man page (`man 5 shadow`) for information on the field layout. Passwords may be encrypted using DES, but are more usually encrypted using MD5. The DES algorithm uses the low order 7 bits of the first 8 characters of the user password as a 56-bit key, while the MD5 algorithm uses the whole password. In either case, passwords are *salted* so that two otherwise identical passwords do not generate the same encrypted value. Listing 11 shows how to set identical passwords for users jane and john2, and then shows the resulting encoded MD5 passwords in `/etc/shadow`.

Listing 11. Passwords in `/etc/shadow`

```
[root@pinguino ~]# echo lpic1111 |passwd jane --stdin
Changing password for user jane.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# echo lpic1111 |passwd john2 --stdin
Changing password for user john2.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# grep "^j" /etc/shadow
jane:$1$eG0/KGQY$ZJ1.ltYtVw0sv.C5OrqUu/:13691:0:99999:7:::
john2:$1$grkxo6ie$J2muvoTpwo3dZAYYTDYNu.:13691:0:180:7:29::
```

The leading `1` indicates an MD5 password, and the salt is a variable length field of up to 8 characters ending with the next `$` sign. The encrypted password is the remaining string of 22 characters.

Tools for users and groups

You have already seen several commands that manipulate the account and group files and their shadows. Here you learn about:

- Group administrators
- Editing commands for password and group files
- Conversion programs

Group administrators

In some circumstances you may want users other than root to be able to administer one or more groups by adding or removing group members. Listing 12 shows how root can add user jane as an administrator of group john2, and then jane, in turn, can add user ian as a member.

Listing 12. Adding group administrators and members

```
[root@pinguino ~]# gpasswd -A jane john2
[root@pinguino ~]# su - jane
[jane@pinguino ~]$ gpasswd -a ian john2
Adding user ian to group john2
[jane@pinguino ~]$ id ian;id jane
uid=500(ian) gid=500(ian) groups=500(ian),505(john2)
uid=504(jane) gid=504(jane) groups=504(jane)
```

You may be surprised to note that, although jane is an administrator of group john2, she is not a member of it. An examination of the structure of `/etc/gshadow` shows why. The `/etc/gshadow` file contains four fields for each entry as shown in Table 8. Note that the third field is a comma-separated list of administrators for the group.

Table 8. Fields in <code>/etc/gshadow</code>	
Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The field used to contain the encrypted password if the group has a password. If there is no password, you may see 'x', '!' or '!!' here.
Admins	A comma-separated list of group administrators.
Members	A comma-separated list of group members.

As you can see, the administrator list and the member list are two distinct fields. The `-A` option of `gpasswd` allows the root user to add administrators to a group, while the `-M` option allows root to add members. The `-a` (note lower case) option allows an administrator to add a member, while the `-d` option allows an administrator to remove a member. Additional options allow a group password to be removed. See the man pages for details.

Editing commands for password and group files

Although not listed in the LPI objectives, you should also be aware of the `vipw` command for safely editing `/etc/passwd` and `visgr` for safely editing `/etc/group`. The

commands will lock the necessary files while you make changes using the `vi` editor. If you make changes to `/etc/passwd`, then `vipw` will prompt you to see if you also need to update `/etc/shadow`. Similarly, if you update `/etc/group` using `vigr`, you will be prompted to update `/etc/gshadow`. If you need to remove group administrators, you may need to use `vigr`, as `gpasswd` only allows addition of administrators.

Conversion programs

Four other related commands are also not listed in the LPI objectives. They are `pwconv`, `pwunconv`, `grpconv`, and `grpunconv`. They are used for converting between shadowed and non-shadowed password and group files. You may never need these, but be aware of their existence. See the man pages for details.

Limited and special-purpose accounts

By convention, system users usually have an id of less than 100, with root having id 0. Normal users start automatic numbering from the `UID_MIN` value set in `/etc/login.defs`, with this value commonly being set at 500 or 1000.

Besides regular user accounts and the root account on your system, you will usually have several special-purpose accounts, for daemons such as FTP, SSH, mail, news, and so on. Listing 13 shows some entries from `/etc/passwd` for these.

Listing 13. Limited and special-purpose accounts

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
```

Such accounts frequently control files but should not be accessed by normal login. Therefore, they usually have a login shell specified as `/sbin/nologin`, or `/bin/false` so that login attempts will fail.

Section 3. Environment tuning

This section covers material for topic 1.111.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to tune the user environment, including these tasks:

- Set and unset environment variables
- Maintain skeleton directories for new user accounts
- Set command search paths

Set and unset environment variables

When you create a new user, you usually initialize many variable according to your local needs. These are usually set in the profiles that you provide for new users, such as `.bash_profile` and `.bashrc`, or in the system-wide profiles `/etc/profile` and `/etc/bashrc`. Listing 14 shows an example of how the `PS1` system prompt is set in `/etc/profile` on an Ubuntu 7.04 system. The first `if` statement checks whether the `PS1` variable is set, indicating an interactive shell, since a non-interactive shell doesn't need a prompt. The second `if` statement checks whether the `BASH` environment variable is set. If so, it sets a complex prompt and sources (note the dot) `/etc/bash.bashrc`. If the `BASH` variable is not set, then a check is made for root (`id=0`), and the prompt is set to `#` or `$` accordingly.

Listing 14. Setting environment variables

```
if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
```

The tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) has detailed information about the commands used for setting and unsetting environment variables, as well as information about how and when the various profiles are used.

When customizing environments for users, be aware of two major points:

1. `/etc/profile` is read only at login time, and so it is not executed when each new shell is created.

2. Functions and aliases are not inherited by new shells. Therefore, you will usually set these and your environment variables in `/etc/bashrc`, or in the user's own profiles.

In addition to the system profiles, `/etc/profile` and `/etc/bashrc`, the Linux Standard Base (LSB) specifies that additional scripts may be placed in the directory `/etc/profile.d`. These scripts are sourced when an interactive login shell is created. They provide a convenient way of separating customization for different programs. Listing 15 shows an example.

Listing 15. `/etc/profile.d/vim.sh` on Fedora 7

```
[if [ -n "$BASH_VERSION" -o -n "$KSH_VERSION" -o -n "$ZSH_VERSION" ];
then
  [ -x //usr/bin/id ] || return
  [ `//usr/bin/id -u` -le 100 ] && return
  # for bash and zsh, only if no alias is already set
  alias vi >/dev/null 2>&1 || alias vi=vim
fi
```

Remember that you should usually `export` any variables that you set in a profile; otherwise, they will not be available to commands that run in a new shell.

Maintain skeleton directories for new users

You learned in the section [Add and remove users and groups](#) that you can create or populate a new home directory for the user. The source for this new directory is the subtree rooted at `/etc/skel`. Listing 16 shows the files in this subtree for a Fedora 7 system. Note that most files start with a period (dot), so you need the `-a` option to list them. The `-R` options lists subdirectories recursively, and the `-L` option follows any symbolic links.

Listing 16. `/etc/skel` on Fedora 7

```
[ian@lyrebird ~]$ ls -aRL /etc/skel
/etc/skel:
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .xemacs

/etc/skel/.xemacs:
.  ..  init.el
```

In addition to `.bash_logout`, `.bash_profile`, and `.bashrc`, which you might expect for the Bash shell, note that this example includes profile information for the emacs and xemacs editors. See the tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) if you need to review the functions of the various profile files.

Listing 17 shows `/etc/skel/.bashrc` from the above system. This file might be different on a different release or different distribution, but it gives you an idea of how the default user setup can be done.

Listing 17. `/etc/skel/.bashrc` on Fedora 7

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
```

As you can see, the global `/etc/bashrc` is sourced, then any user specific instructions can be added. Listing 18 shows the part of `/etc/bashrc` in which the `.sh` scripts from `/etc/profile.d` are sourced.

Listing 18. Sourcing `.sh` scripts from `/etc/profile.d`

```
for i in /etc/profile.d/*.sh; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i
```

Note that the variable, `i`, is unset after the loop.

Set command search paths

Your default profiles often include `PATH` variables for local functions or for products that you may have installed. You can set these in the skeleton files in `/etc/skel`, modify `/etc/profile`, `/etc/bashrc`, or create a file in `/etc/profile.d` if your system uses that. If you do modify the system files, be sure to check that your changes are intact after any system updates. Listing 19 shows how to add a new directory, `/opt/productxyz/bin`, to either the front or rear of your existing `PATH`.

Listing 19. Adding a path directory

```
PATH="$PATH${PATH:+:}/opt/productxyz/bin"
PATH="/opt/productxyz/bin${PATH:+:}$PATH"
```

Although not strictly required, the expression `${PATH:+:}` inserts a path separator (colon) only if the `PATH` variable is unset or null.

Section 4. System log files

This section covers material for topic 1.111.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to configure and manage system logs, including these tasks:

- Manage the type and level of information logged
- Rotate and archive log files automatically
- Scan log files for notable activity
- Monitor log files
- Track down problems reported in log files

Manage the type and level of information logged

The system logging facility on a Linux system provides system logging and kernel message trapping. Logging can be done on a local system or sent to a remote system, and the level of logging can be finely controlled through the `/etc/syslog.conf` configuration file. Logging is performed by the `syslogd` daemon, which normally receives input through the `/dev/log` socket, as shown in Listing 20.

Listing 20. `/dev/log` is a socket

```
ian@pinguino:~$ ls -l /dev/log
srw-rw-rw- 1 root root 0 2007-07-05 15:42 /dev/log
```

For local logging, the main file is usually `/var/log/messages`, but many other files are used in most installations, and you can customize these extensively. For example, you may want a separate log for messages from the mail system.

The `syslog.conf` configuration file

The `syslog.conf` file is the main configuration file for the `syslogd` daemon. Logging is based on a combination of facility and priority. The defined facilities are `auth` (or `security`), `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `syslog`, `user`, `uucp`, and `local0` through `local7`. The keyword `auth` should be used instead of `security`, and the keyword `mark` is for internal use.

The priorities (in ascending order) are:

1. debug
2. info
3. notice
4. warning (or warn)
5. err (or error)
6. crit
7. alert
8. emerg (or panic)

The parenthesized keywords (warn, error, and panic) are now deprecated.

Entries in `syslog.conf` specify logging rules. Each rule has a selector field and an action field, which are separated by one or more spaces or tabs. The selector field identifies the facility and the priorities that the rule applies to, and the action field identifies the logging action for the facility and priorities. The default behavior is to take the action for the specified level and for all higher levels, although it is possible to limit logging to specific levels. Each selector consists of a facility and a priority separated by a period (dot). Multiple facilities for a given action can be specified by separating them with a comma. Multiple facility/priority pairs for a given action can be specified by separating them with a semi-colon. Listing 21 shows an example of a simple `syslog.conf`.

Listing 21. Example `syslog.conf`

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                               /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages

# The authpriv file has restricted access.
authpriv.*                             /var/log/secure

# Log all the mail messages in one place.
mail.*
-/var/log/maillog

# Log cron stuff
cron.*                                  /var/log/cron
```

```
# Everybody gets emergency messages
*.emerg                                     *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                             /var/log/spooler

# Save boot messages also to boot.log
local7.*
/var/log/boot.log
```

Notes:

- As with many configuration files, lines starting with # and blank lines are ignored.
- An * may be used to refer to all facilities or all priorities.
- The special priority keyword `none` indicates that no logging for this facility should be done with this action.
- The hyphen before a file name (such as `-/var/log/maillog`, in this example) indicates that the log file should not be synchronized after every write. You might lose information after a system crash, but you might gain performance by doing this.

The actions are generically referred to as "logfiles," although they do not have to be real files. Table 9 describe the possible logfiles.

Table 9. Actions in syslog.conf	
Action	Purpose
Regular File	Specify the full pathname, beginning with a slash (/). Prefix it with a hyphen (-) to omit syncing the file after each log entry. This may cause information loss if a crash occurs, but may improve performance.
Named Pipes	A fifo or named pipe can be used as a destination for log messages by putting a pipe symbol () before the filename. You must create the fifo using the <code>mkfifo</code> command before starting (or restarting) <code>syslogd</code> . Fifos are sometimes used for debugging.
Terminal and Console	A terminal such as <code>/dev/console</code> .
Remote Machine	To forward messages to another host, put an at (@) sign before the hostname. Note that messages are not forwarded from the receiving host.
List of Users	A comma-separated list of users to receive a message (if the user is

	logged in). The root user is frequently included here.
Everyone logged on	Specify an asterisk (*) to have everyone logged on notified using the wall command.

You may prefix ! to a priority to indicate that the action should not apply to this level and higher. Similarly you may prefix it with = to indicate that the rule applies only to this level or with != to indicate that the rule applies to all except this level. Listing 22 shows some examples, and the man page for syslog.conf has many more examples.

Listing 22. More syslog.conf examples

```
# Store all kernel messages in /var/log/kernel.
# Send critical and higher ones to remote host pinguino and to the
console
# Finally, Send info, notice and warning messages to
/var/log/kernel-info
#
kern.*                /var/log/kernel
kern.crit             @pinguino
kern.crit             /dev/console
kern.info;kern.!err  /var/log/kernel-info

# Store all mail messages except info priority in /var/log/mail.
mail.*;mail.!=info   /var/log/mail
```

Rotate and archive log files automatically

With the amount of logging that is possible, you need to be able to control the size of log files. This is done using the `logrotate` command, which is usually run as a cron job. Cron jobs are covered later in this tutorial in the section [Scheduling jobs](#). The general idea behind the `logrotate` command is that log files are periodically backed up and a new log is started. Several generations of log are kept, and when a log ages to the last generation, it may be archived. For example, it might be mailed to an archival user.

You use the `/etc/logrotate.conf` configuration file to specify how your log rotating and archiving should happen. You can specify different frequencies, such as daily, weekly, or monthly, for different log files, and you can control the number of generations to keep and when or whether to mail copies to an archival user. Listing 23 shows a sample `/etc/logrotate.conf` file.

Listing 23. Sample /etc/logrotate.conf

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

The `logrotate.conf` file has global options at the beginning. These are the defaults if nothing more specific is specified elsewhere. In this example, log files are rotated weekly, and four weeks worth of backups are kept. Once a log file is rotated, a new one is automatically created in place of the old one. Note that the `logrotate.conf` file may include specifications from other files. Here, all the files in `/etc/logrotate.d` are included.

This example also includes specific rules for `/var/log/wtmp` and `/var/log/btmp`, which are rotated monthly. No error message is issued if the files are missing. A new file is created, and only one backup is kept.

In this example, when a backup reaches the last generation, it is deleted because there is no specification of what else to do with it.

Note: The files `/var/log/wtmp` and `/var/log/btmp` record successful and unsuccessful login attempts, respectively. Unlike most log files, these are not clear text files. You may examine them using the `last` or `lastb` commands. See the man pages for these commands for details.

Log files may also be backed up when they reach a specific size, and commands may be scripted to run either prior to or after the backup operation. Listing 24 shows a more complex example.

Listing 24. Another logrotate configuration example

```
/var/log/messages {
```

```

rotate 5
mail logsave@pinguino
size 100k
  postrotate
    /usr/bin/killall -HUP syslogd
  endscrip
}

```

In this example, `/var/log/messages` is rotated after it reaches 100KB in size. Five backups are kept, and when the oldest backup ages out, it is mailed to `logsave@pinguino`. The `postrotate` introduces a script that restarts the `syslogd` daemon after the rotation is complete, by sending it the HUP signal. The `endscript` statement is required to terminate the script and is also required if a `prerotate` script is present. See the `logrotate` man page for more complete information.

Scan log files for notable activity

Log files entries are usually time stamped and contain the hostname of the reporting process, along with the process name. Listing 25 shows a few lines from `/var/log/messages`, containing entries from `gconfd`, `ntpd`, `init`, and `yum`.

Listing 25. Sample log file entries

```

Jul  5 15:28:24 lyrebird gconfd (root-2832): Exiting
Jul  5 15:31:06 lyrebird ntpd[2063]: synchronized to 87.98.219.90,
stratum 2
Jul  5 15:31:06 lyrebird ntpd[2063]: kernel time sync status change 0001
Jul  5 15:31:24 lyrebird init: Trying to re-exec init
Jul  5 15:31:24 lyrebird yum: Updated: libselinux.i386 2.0.14-2.fc7
Jul  5 15:31:24 lyrebird yum: Updated: libsemanage.i386 2.0.3-4.fc7
Jul  5 15:31:25 lyrebird yum: Updated: cups-libs.i386 1.2.11-2.fc7
Jul  5 15:31:25 lyrebird yum: Updated: libXfont.i386 1.2.9-2.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager.i386 0.6.5-7.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager-glib.i386
0.6.5-7.fc7

```

You can scan log files using a pager, such as `less`, or search for specific entries (such as kernel messages from host `lyrebird`) using `grep` as shown in Listing 26.

Listing 26. Scanning log files

```

[root@lyrebird ~]# less /var/log/messages
[root@lyrebird ~]# grep "lyrebird kernel" /var/log/messages | tail -n 9
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HCI socket layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP ver 2.8
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM TTY layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM ver 1.8

```

```
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HIDP (Human Interface
Emulation) ver 1.2
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized drm 1.1.0 20060810
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized i915 1.6.0 20060119
on minor 0
```

Monitor log files

Occasionally you may need to monitor log files for events. For example, you might be trying to catch an infrequently occurring event at the time it happens. In such a case, you can use the `tail` command with the `-f` option to *follow* the log file. Listing 27 shows an example.

Listing 27. Following log file updates

```
[root@lyrebird ~]# tail -n 1 -f /var/log/messages
Jul  6 15:16:26 lyrebird syslogd 1.4.2: restart.
Jul  6 15:16:26 lyrebird kernel: klogd 1.4.2, log source = /proc/kmsg
started.
Jul  6 15:19:35 lyrebird yum: Updated: samba-common.i386 3.0.25b-2.fc7
Jul  6 15:19:35 lyrebird yum: Updated: procps.i386 3.2.7-14.fc7
Jul  6 15:19:36 lyrebird yum: Updated: samba-client.i386 3.0.25b-2.fc7
Jul  6 15:19:37 lyrebird yum: Updated: libsmbclient.i386 3.0.25b-2.fc7
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Received signal 15, shutting
down cleanly
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Exiting
Jul  6 15:19:57 lyrebird yum: Updated: bluez-gnome.i386 0.8-1.fc7
```

Track down problems reported in log files

When you find problems in log files, you will want to note the time, the hostname, and the process that generated the problem. If the message identifies the problem specifically enough for you to resolve it, you are done. If not, you might need to update `syslog.conf` to specify that more messages be logged for the appropriate facility. For example, you might need to show informational messages instead of warning messages or even debug level messages. Your application may have additional facilities that you can use.

Finally, if you need to put marks in the log file to help you know what messages were logged at what stage of your debugging activity, you can use the `logger` command from a terminal window or shell script to send a message of your choice to the syslog daemon for logging according to the rules in `syslog.conf`.

Section 5. Scheduling jobs

This section covers material for topic 1.111.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use the `cron` or `anacron` commands to run jobs at regular intervals
- Use the `at` command to run jobs at a specific time
- Manage cron and at jobs
- Configure user access to the cron and at services

In the previous section, you learned about the `logrotate` command and saw the need to run it periodically. You will see the same need to run commands regularly in the next two sections on backup and network time services. These are only some of the many administrative tasks that have to be done frequently and regularly. In this section, you learn about the tools that are used to automate periodic job scheduling and also the tools used to run a job at some specific time.

Run jobs at regular intervals

Running jobs at regular intervals is managed by the `cron` facility, which consists of the `crond` daemon and a set of tables describing what work is to be done and with what frequency. The daemon wakes up every minute and checks the crontabs to determine what needs to be done. Users manage crontabs using the `crontab` command. The `crond` daemon is usually started by the `init` process at system startup.

To keep things simple, let's suppose that you want to run the command shown in Listing 28 on a regular basis. This command doesn't actually do anything except report the day and the time, but it illustrates how to use `crontab` to set up cron jobs, and we'll know when it was run from the output. Setting up crontab entries requires a string with escaped shell metacharacters, so it is best done with simple commands and parameters, so in this example, the `echo` command will be run from within a script `/home/ian/mycrontab.sh`, which takes no parameters. This saves some careful work with escape characters.

Listing 28. A simple command example.

```
[ian@lyrebird ~]$ cat mycrontest.sh
#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
[ian@lyrebird ~]$ ./mycrontest.sh
It is now 18:37:42 on Friday
```

Creating a crontab

To create a crontab, you use the `crontab` command with the `-e` (for "edit") option. This will open the `vi` editor unless you have specified another editor in the `EDITOR` or `VISUAL` environment variable.

Each crontab entry contains six fields:

1. Minute
2. Hour
3. Day of the month
4. Month of the year
5. Day of the week
6. String to be executed by `sh`

Minutes and hours range from 0-59 and 0-12, respectively, while day or month and month of year range from 1-31 and 1-12, respectively. Day of week ranges from 0-6 with 0 being Sunday. Day of week may also be specified as `sun`, `mon`, `tue`, and so on. The sixth field is everything after the fifth field, is interpreted as a string to pass to `sh`. A percent sign (%) will be translated to a newline, so if you want a % or any other special character, precede it with a backslash (\). The line up to the first % is passed to the shell, while any line(s) after the % are passed as standard input.

The various time-related fields can specify an individual value, a range of values, such as 0-10 or `sun-wed`, or a comma-separated list of individual values and ranges. So a somewhat artificial crontab entry for our example command might be as shown in Listing 29.

Listing 29. A simple crontab example.

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
```

In this example, our command is executed at the 0th, 20th, and 40th minutes (every 20 minutes), for the hours between 10 P.M. and midnight on Fridays and Saturdays during July. See the man page for `crontab(5)` for details on additional ways to specify times.

What about the output?

You may be wondering what happens to any output from the command. Most

commands designed for use with the cron facility will log output using the syslog facility that you learned about in the previous section. However any output that is directed to stdout will be mailed to the user. Listing 30 shows the output you might receive from our example command.

Listing 30. Mailed cron output

```
From ian@lyrebird.raleigh.ibm.com Fri Jul 6 23:00:02 2007
Date: Fri, 6 Jul 2007 23:00:01 -0400
From: root@lyrebird.raleigh.ibm.com (Cron Daemon)
To: ian@lyrebird.raleigh.ibm.com
Subject: Cron <ian@lyrebird> /home/ian/mycronetest.sh
Content-Type: text/plain; charset=UTF-8
Auto-Submitted: auto-generated
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/home/ian>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=ian>
X-Cron-Env: <USER=ian>

It is now 23:00:01 on Friday
```

Where is my crontab?

The crontab that you created with the `crontab` command is stored in `/etc/spool/cron` under the name of the user who created it. So the above crontab is stored in `/etc/spool/cron/ian`. Given this, you will not be surprised to learn that the `crontab` command, like the `passwd` command you learned about earlier, is an `suid` program that runs with root authority.

`/etc/crontab`

In addition to the user crontab files in `/var/spool/cron`, `cron` also checks `/etc/crontab` and files in the `/etc/cron.d` directory. These system crontabs have one additional field between the fifth time entry (day) and the command. This additional field specifies the user for whom the command should be run, normally `root`. A `/etc/crontab` might look like the example in Listing 31.

Listing 31. `/etc/crontab`

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

In this example, the real work is done by the `run-parts` command, which runs

scripts from `/etc/cron.hourly`, `/etc/cron.daily`, and so on; `/etc/crontab` simply controls the timing of the recurring jobs. Note that the commands here all run as root. Note also that the crontab can include shell variables assignments that will be set before the commands are run.

Anacron

The cron facility works well for systems that run continuously. For systems that may be turned off much of the time, such as laptops, another facility, the *anacron* (for "anachronistic cron") can handle scheduling of the jobs usually done daily, weekly, or monthly by the cron facility. Anacron does not handle hourly jobs.

Anacron keeps timestamp files in `/var/spool/anacron` to record when jobs are run. When anacron runs, it checks to see if the required number of days has passed since the job was last run and runs it if necessary. The table of jobs for anacron is stored in `/etc/anacrontab`, which has a slightly different format than `/etc/crontab`. As with `/etc/crontab`, `/etc/anacrontab` may contain environment settings. Each job has four fields.

1. period
2. delay
3. job-identifier
4. command

The period is a number of days, but may be specified as `@monthly` to ensure that a job runs only once per month, regardless of the number of days in the month. The delay is the number of minutes to wait after the job is due to run before actually starting it. You can use this to prevent a flood of jobs when a system first starts. The job identifier can contain any non-blank character except slashes (`/`).

Both `/etc/crontab` and `/etc/anacrontab` are updated by direct editing. You do not use the `crontab` command to update these files or files in the `/etc/cron.d` directory.

Run jobs at specific times

Sometimes you may need to run a job just once, rather than regularly. For this you use the `at` command. The commands to be run are read from a file specified with the `-f` option, or from stdin if `-f` is not used. The `-m` option sends mail to the user even if there is no stdout from the command. The `-v` option will display the time at which the job will run before reading the job. The time is also displayed in the output. Listing 32 shows an example of running the `mycrontest.sh` script that you used earlier. Listing 33 shows the output that is mailed back to the user after the job runs.

Notice that it is somewhat more compact than the corresponding output from the cron job.

Listing 32. Using the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh -v 10:25
Sat Jul 7 10:25:00 2007

job 5 at Sat Jul 7 10:25:00 2007
```

Listing 33. Job output from at

```
From ian@lyrebird.raleigh.ibm.com Sat Jul 7 10:25:00 2007
Date: Sat, 7 Jul 2007 10:25:00 -0400
From: Ian Shields <ian@lyrebird.raleigh.ibm.com>
Subject: Output from your job 5
To: ian@lyrebird.raleigh.ibm.com

It is now 10:25:00 on Saturday
```

Time specifications can be quite complex. Listing 34 shows a few examples. See the man page for `at` or the file `/usr/share/doc/at/timespec` or a file such as `/usr/share/doc/at-3.1.10/timespec`, where 3.1.10 in this example is the version of the `at` package.

Listing 34. Time values with the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh 10pm tomorrow
job 14 at Sun Jul 8 22:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 tuesday
job 15 at Tue Jul 10 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 july 11
job 16 at Wed Jul 11 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 next week
job 17 at Sat Jul 14 02:00:00 2007
```

The `at` command also has a `-q` option. Increasing the queue increases the nice value for the job. There is also a `batch` command, which is similar to the `at` command except that jobs are run only when the system load is low enough. See the man pages for more details on these features.

Manage scheduled jobs

Listing scheduled jobs

You can manage your cron and `at` jobs. Use the `crontab` command with the `-l` option to list your crontab, and use the `atq` command to display the jobs you have queued using the `at` command, as shown in Listing 35.

Listing 35. Displaying scheduled jobs

```
[ian@lyrebird ~]$ crontab -l
0,20,40 22-23 * 7 fri-sat /home/ian/mycronstest.sh
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
```

If you want to review the actual command scheduled for execution by `at`, you can use the `at` command with the `-c` option and the job number. You will notice that most of the environment that was active at the time the `at` command was issued is saved with the scheduled job. Listing 36 shows part of the output for job 15.

Listing 36. Using `at -c` with a job number

```
#!/bin/sh
# atrun uid=500 gid=500
# mail ian 0
umask 2
HOSTNAME=lyrebird.raleigh.ibm.com; export HOSTNAME
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=9.67.219.151\ 3210\ 22; export SSH_CLIENT
SSH_TTY=/dev/pts/5; export SSH_TTY
USER=ian; export USER
...
HOME=/home/ian; export HOME
LOGNAME=ian; export LOGNAME
...
cd /home/ian || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
${SHELL:-/bin/sh} << `(dd if=/dev/urandom count=200 bs=1 \
  2>/dev/null|LC_ALL=C tr -d -c '[:alnum:]')`

#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
```

Note that the contents of our script file have been copied in as a here-document that will be executed by the shell specified by the `SHELL` variable or `/bin/sh` if the `SHELL` variable is not set. See the tutorial [LPI exam 101 prep, Topic 103: GNU and UNIX commands](#) if you need to review here-documents.

Deleting scheduled jobs

You can delete all scheduled cron jobs using the `crontab` command with the `-r` option as illustrated in Listing 37.

Listing 37. Displaying and deleting cron jobs

```
[ian@lyrebird ~]$ crontab -l
```

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycronstest.sh
[ian@lyrebird ~]$ crontab -r
[ian@lyrebird ~]$ crontab -l
no crontab for ian
```

To delete system cron or anacron jobs, edit `/etc/crontab`, `/etc/anacrontab`, or edit or delete files in the `/etc/cron.d` directory.

You can delete one or more jobs that were scheduled with the `at` command by using the `atrm` command with the job number. Multiple jobs should be separated by spaces. Listing 38 shows an example.

Listing 38. Displaying and removing jobs with `atq` and `atrm`

```
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
[ian@lyrebird ~]$ atrm 16 14 15
[ian@lyrebird ~]$ atq
17      Sat Jul 14 02:00:00 2007 a ian
```

Configure user access to job scheduling

If the file `/etc/cron.allow` exists, any non-root user must be listed in it in order to use `crontab` and the cron facility. If `/etc/cron.allow` does not exist, but `/etc/cron.deny` does exist, a non-root user who is listed in it cannot use `crontab` or the cron facility. If neither of these files exists, only the super user will be allowed to use this command. An empty `/etc/cron.deny` file allows all users to use the cron facility and is the default.

The corresponding `/etc/at.allow` and `/etc/at.deny` files have similar effects for the `at` facility.

Section 6. Data backup

This section covers material for topic 1.111.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Plan a backup strategy

- Dump a raw device to a file or restore a raw device from a file
- Perform partial and manual backups
- Verify the integrity of backup files
- Restore filesystems partially or fully from backups

Plan a backup strategy

Having a good backup is a necessary part of system administration, but deciding what to back up and when and how can be complex. Databases, such as customer orders or inventory, are usually critical to a business and many include specialized backup and recovery tools that are beyond the scope of this tutorial. At the other extreme, some files are temporary in nature and no backup is needed at all. In this section, we focus on system files and user data and discuss some of the considerations, methods, and tools for backup of such data.

There are three general approaches to backup:

1. A *full* backup is a complete backup, usually of a whole filesystem, directory, or group of related files. This takes the longest time to create, so it is usually used with one of the other two approaches.
2. A *differential* or *cumulative* backup is a backup of all things that have changed since the last full backup. Recovery requires the last full backup plus the latest differential backup.
3. An *incremental* backup is a backup of only those changes since the last incremental backup. Recovery requires the last full backup plus all of the incremental backups (in order) since the last full backup.

What to back up

When deciding what to back up, you should consider how volatile the data is. This will help you determine how often it should be backed up. Similarly, critical data should be backed up more often than non-critical data. Your operating system will probably be relatively easy to rebuild, particularly if you use a common image for several systems, although the files that customize each system would be more important to back up.

For programming staff, it may be sufficient to keep backups of repositories such as CVS repositories, while individual programmers' sandboxes may be less important. Depending on how important mail is to your operation, it may suffice to have infrequent mail backups, or it may be necessary to be able to recover mail to the

most recent date possible. You may want to keep backups of system cron files, but may not be so concerned about scheduled jobs for individual users.

The Filesystem Hierarchy Standard provides a classification of data that may help you with your backup choices. For details, see the tutorial [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#).

Once you have decided what to back up, you need to decide how often to do a full backup and whether to do differential or incremental backups in between those full backups. Having made those decisions, the following suggestions will help you choose appropriate tools.

Automating backups

In the previous section, you learned how to schedule jobs, and the cron facility is ideal for helping to automate the scheduling of your backups. However, backups are frequently made to removable media, particularly tape, so operator intervention is probably going to be needed. You should create and use backup scripts to ensure that the backup process is as automatic and repeatable as possible.

Dump and restore raw devices

One way to make a full backup of a filesystem is to make an image of the partition on which it resides. A *raw device*, such as `/dev/hda1` or `/dev/sda2`, can be opened and read as a sequential file. Similarly, it can be written from a backup as a sequential file. This requires no knowledge on the part of the backup tool as to the filesystem layout, but does require that the restore be done to space that is at least as large as the original. Some tools that handle raw devices are *filesystem aware*, meaning that they understand one or more of the Linux filesystems. These utilities can dump from a raw device but do not dump unused parts of the partition. They may or may not require restoration to the same or larger sized partition. The `dd` command is an example of the first type, while the `dump` command is an example of the second type that is specific to the `ext2` and `ext3` filesystems.

The `dd` command

In its simplest form, the `dd` command copies an input file to an output file, where either file may be a raw device. For backing up a raw device, such as `/dev/hda1` or `/dev/sda2`, the input file would be a raw device. Ideally, the filesystem on the device should be unmounted, or at least mounted read only, to ensure that data does not change during the backup. Listing 39 shows an example.

Listing 39. Backup a partition using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 of=backup-1
```

```
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 49.3103 s, 21.2 MB/s
```

The `if` and `of` parameters specify the input and output files respectively. In this example, the input file is a raw device, `dev/sda3`, and the output file is a file, `backup-1`, in the root user's home directory. To dump the file to tape or floppy disk, you would specify something like `of=/dev/fd0` or `of=/dev/st0`.

Note that 1,044,610,560 bytes of data was copied and the output file is indeed that large, even though only about 3% of this particular partition is actually used. Unless you are copying to a tape with hardware compression, you will probably want to compress the data. Listing 40 shows one way to accomplish this, along with the output of `ls` and `df` commands, which show you the file sizes and the usage percentage of the filesystem on `/dev/sda3`.

Listing 40. Backup with compression using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 | gzip > backup-2
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 117.723 s, 8.9 MB/s
[root@lyrebird ~]# ls -l backup-[12]
-rw-r--r-- 1 root root 1044610560 2007-07-08 15:17 backup-1
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
[root@lyrebird ~]# df -h /dev/sda3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        972M   28M  944M   3% /grubfile
```

The `gzip` compression reduced the file size to about 20% of the uncompressed size. However, unused blocks may contain arbitrary data, so even the compressed backup may be much larger than the total data on the partition.

If you divide the size by the number of records processed by `dd`, you will see that `dd` is writing 512-byte blocks of data. When copying to a raw output device such as tape, this can result in a very inefficient operation, so `dd` can read or write data in much larger blocks. Specify the `obs` option to change the output size or the `ibs` option to specify the input block size. You can also specify just `bs` to set both input and output block sizes to a common value.

If you need multiple tapes or other removable storage to store your backup, you will need to break it into smaller pieces using a utility such as `split`.

If you need to skip blocks such as disk or tape labels, you can do so with `dd`. See the man page for examples.

Besides just copying data, the `dd` command can do several conversions, such as between ASCII and EBCDIC, between big-endian and little-endian, or between variable-length data records and fixed-length data records. Obviously these

conversions are likely to be useful when copying real files rather than raw devices. Again, see the man page for details.

The dump command

The `dump` command can be used for full, differential, or incremental backups on `ext2` or `ext3` filesystems. Listing 41 shows an example.

Listing 41. Backup with compression using dump

```
[root@lyrebird ~]# dump -0 -f backup-4 -j -u /dev/sda3
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-4
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzip)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 12285 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:47:48 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-4
DUMP: Volume 1 completed at: Sun Jul  8 16:47:57 2007
DUMP: Volume 1 took 0:00:09
DUMP: Volume 1 transfer rate: 819 kB/s
DUMP: Volume 1 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: 12260 blocks (11.97MB) on 1 volume(s)
DUMP: finished in 9 seconds, throughput 1362 kBytes/sec
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Date this dump completed: Sun Jul  8 16:47:57 2007
DUMP: Average transfer rate: 819 kB/s
DUMP: Wrote 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[2-4]
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
-rw-r--r-- 1 root root 266932272 2007-07-08 15:44 backup-3
-rw-r--r-- 1 root root  7554939 2007-07-08 16:47 backup-4
```

In this example, `-0` specifies the *dump level* which is an integer, historically from 0 to 9, where 0 specifies a full dump. The `-f` option specifies the output file, which may be a raw device. Specify `-` to direct the output to stdout. The `-j` option specifies compression, with a default level of 2, using bzip compression. You can use the `-z` option to specify zlib compression if you prefer. The `-u` option causes the record of dump information, normally `/etc/dumpdates`, to be updated. Any parameters after the options represent a file or list of files, where the file may also be a raw device, as in this example. Notice how much smaller the backup is when the backup program is aware of the filesystem structure and can avoid the saving of unused blocks on the device.

If output is to a device such as tape, the `dump` command will prompt for another volume as each volume is filled. You can also provide multiple file names separated by commas. For example, if you wanted an unattended dump that required two tapes, you could load the tapes on `/dev/st0` and `/dev/st1`, schedule the `dump` command specifying both tapes as output, and go home to sleep.

When you specify a dump level greater than 0, an incremental dump is performed of all files that are new or have changed since the last dump at a lower level was taken. So a dump at level 1 will be a differential dump, even if a dump at level 2 or higher has been taken in the meantime. Listing 42 shows the result of updating the time stamp of an existing file on /dev/sda3 and creating a new file, then taking a dump at level 2. After that, another new file is created and a dump at level 1 is taken. The information from /etc/dumpdates is also shown. For brevity, part of the second dump output has been omitted.

Listing 42. Backup with compression using dump

```
[root@lyrebird ~]# dump -2 -f backup-5 -j -u /dev/sda3
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-5
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzlib)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 91 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:55:47 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-5
DUMP: Volume 1 completed at: Sun Jul  8 16:55:47 2007
DUMP: 90 blocks (0.09MB) on 1 volume(s)
DUMP: finished in less than a second
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date this dump completed: Sun Jul  8 16:55:47 2007
DUMP: Average transfer rate: 0 kB/s
DUMP: Wrote 90kB uncompressed, 15kB compressed, 6.000:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# echo "This data is even newer" >/grubfile/newerfile
[root@lyrebird ~]# dump -1 -f backup-6 -j -u -A backup-6-toc /dev/sda3
DUMP: Date of this level 1 dump: Sun Jul  8 17:08:18 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-6
...
DUMP: Wrote 100kB uncompressed, 16kB compressed, 6.250:1
DUMP: Archiving dump to backup-6-toc
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[4-6]
-rw-r--r-- 1 root root 7554939 2007-07-08 16:47 backup-4
-rw-r--r-- 1 root root  16198 2007-07-08 16:55 backup-5
-rw-r--r-- 1 root root  16560 2007-07-08 17:08 backup-6
[root@lyrebird ~]# cat /etc/dumpdates
/dev/sda3 0 Sun Jul  8 16:47:47 2007 -0400
/dev/sda3 2 Sun Jul  8 16:55:46 2007 -0400
/dev/sda3 1 Sun Jul  8 17:08:18 2007 -0400
```

Notice that backup-6 is, indeed, larger than backup 5. The level 1 dump illustrates the use of the `-A` option to create a table of contents that can be used to determine if a file is on an archive without actually mounting the archive. This is particularly useful with tape or other removable archive volumes. You will see these examples again when we discuss restoring data later in this section.

The `dump` command can dump files or subdirectories, but you cannot update

/etc/dumpdates and only level 0, of full dump, is supported.

Listing 43 illustrates the `dump` command dumping a directory, `/usr/include/bits`, and its contents to floppy disk. In this case, the dump will not fit on a single floppy, so a new volume is required. The prompt and response are shown in bold.

Listing 43. Backup a directory to multiple volumes using dump

```
[root@lyrebird ~]# dump -0 -f /dev/fd0 /usr/include/bits
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Dumping /dev/sdb9 (/ (dir usr/include/bits)) to /dev/fd0
DUMP: Label: /
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 2790 blocks.
DUMP: Volume 1 started with block 1 at: Mon Jul  9 16:03:30 2007
DUMP: dumping (Pass III) [directories]
DUMP: End of tape detected
DUMP: Closing /dev/fd0
DUMP: Volume 1 completed at: Mon Jul  9 16:04:49 2007
DUMP: Volume 1 1470 blocks (1.44MB)
DUMP: Volume 1 took 0:01:19
DUMP: Volume 1 transfer rate: 18 kB/s
DUMP: Change Volumes: Mount volume #2
DUMP: Is the new volume mounted and ready to go?: ("yes" or "no") y
DUMP: Volume 2 started with block 1441 at: Mon Jul  9 16:05:10 2007
DUMP: Volume 2 begins with blocks from inode 2
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing /dev/fd0
DUMP: Volume 2 completed at: Mon Jul  9 16:06:28 2007
DUMP: Volume 2 1410 blocks (1.38MB)
DUMP: Volume 2 took 0:01:18
DUMP: Volume 2 transfer rate: 18 kB/s
DUMP: 2850 blocks (2.78MB) on 2 volume(s)
DUMP: finished in 109 seconds, throughput 26 kBytes/sec
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Date this dump completed: Mon Jul  9 16:06:28 2007
DUMP: Average transfer rate: 18 kB/s
DUMP: DUMP IS DONE
```

If you back up to tape, remember that the tape will usually be rewound after each job. Devices with a name like `/dev/st0` or `/dev/st1` automatically rewind. The corresponding non-rewind equivalent devices are `/dev/nst0` and `/dev/nst1`. In any event, you can always use the `mt` command to perform magnetic tape operations such as forward spacing over files and records, back spacing, rewinding, and writing EOF marks. See the man pages for `mt` and `st` for additional information.

If you select the dump levels judiciously, you can minimize the number of archives you need to restore to any particular level. See the man pages for `dump` for a suggestion based on the Towers of Hanoi puzzle.

As with the `dd` command, there are many options that are not covered in this brief introduction. See the man pages for more details.

Partial and manual backups

So far, you have learned about tools that work well for backing up whole filesystems. Sometimes your backup needs to target selected files or subdirectories without backing up the whole filesystem. For example, you might need a weekly backup of most of your system, but daily backups of your mail files. Two other programs, `cpio` and `tar`, are more commonly used for this purpose. Both can write archives to files or to devices such as tape or floppy disk, and both can restore from such archives. Of the two, `tar` is more commonly used today, possibly because it handles complete directories better, and GNU `tar` supports both `gzip` and `bzip` compression.

Using `cpio`

The `cpio` command operates in *copy-out* mode to create an archive, *copy-in* mode to restore an archive, or *copy-pass* mode to copy a set of files from one location to another. You use the `-o` or `--create` option for copy-out mode, the `-i` or `--extract` option for copy-in mode, and the `-p` or `--pass-through` option for copy-pass mode. Input is a list of files provided on `stdin`. Output is either to `stdout` or to a device or file specified with the `-f` or `--file` option.

Listing 44 shows how to generate a list of files using the `find` command. Note the use of the `-print0` option on `find` to generate null-terminate strings for file names, and the corresponding `--null` option on `cpio` to read this format. This will correctly handle file names that have embedded blank or newline characters.

Listing 44. Back up a home directory using `cpio`

```
[root@lyrebird ~]# find ~ian -depth -print0 | cpio --null -o
>backup-cpio-1
18855 blocks
```

If you'd like to see the files listed as they are archived, add the `-v` option to `cpio`.

As with other commands that can archive to tape, the block size may be specified. For details on this and other options, see the man page.

Using `tar`

The `tar` (originally from *Tape ARchive*) creates an archive file, or *tarfile* or *tarball*, from a set of input files or directories; it also restores files from such an archive. If a directory is given as input to `tar`, all files and subdirectories are automatically included, which makes `tar` very convenient for archiving subtrees of your directory structure.

As with the other archiving commands we have discussed, output can be to a file, a

device such as tape or diskette, or stdout. The output location is specified with the `-f` option. Other common options are `-c` to create an archive, `-x` to extract an archive, `-v` for verbose output, which lists the files being processed, `-z` to use gzip compression, and `-j` to use bzip2 compression. Most `tar` options have a short form using a single hyphen and a long form using a pair of hyphens. The short forms are illustrated here. See the man pages for the long form and for additional options.

Listing 45 shows how to create a backup of the system cron jobs using `tar`.

Listing 45. Backup of system cron jobs using tar

```
[root@lyrebird ~]# tar -czvf backup-tar-1 /etc/*crontab /etc/cron.d
tar: Removing leading `/' from member names
/etc/anacrontab
/etc/crontab
/etc/cron.d/
/etc/cron.d/sa-update
/etc/cron.d/smolt
```

In the first line of output, you are told that `tar` will remove the leading slash (/) from member names. This allows files to be restored to some other location for verification before replacing system files. It is a good idea to avoid mixing absolute path names with relative path names when creating an archive, since all will be relative when restoring from the archive.

The `tar` command can append additional files to an archive using the `-r` or `--append` option. This may cause multiple copies of a file in the archive. In such a case, the *last* one will be restored during a restore operation. You can use the `--occurrence` option to select a specific file among multiples. If the archive is on a regular filesystem instead of tape, you may use the `-u` or `--update` option to update an archive. This works like appending to an archive, except that the time stamps of the files in the archive are compared with those on the filesystem, and only files that have been modified since the archived version are appended. As mentioned, this does not work for tape archives.

As with the other commands you have studied here, there are many options that are not covered in this brief introduction. See the man or info pages for more details.

Backup file integrity

Backup file integrity is extremely important. There is no point in having a backup if it is bad. A good backup strategy also involves checking your backups.

The first step to ensuring backup integrity is to ensure that you have properly captured the data you are backing up. If the filesystem is unmounted or mounted read only, this is usually straightforward as the data you are backing up cannot

change during your backup. If you must back up filesystems, directories, or files that are subject to modification while you are taking the backup, you should verify that no changes have been made during your backup. If changes were made, you will need to have a strategy for capturing them, either by repeating the backup, or perhaps by replacing or superseding the affected files in your backup. Needless to say, this will also affect your restore procedures.

Assuming you took good backups, you will periodically need to verify your backups. One way is to restore the backup to a spare volume and verify that it matches what you backed up. This is easiest to do right before you allow updates on the filesystem you are backing up. If you back up to media such as CD or DVD, you may be able to use the `diff` command as part of your backup procedure to ensure that your backup is good. Remember that even good backups can deteriorate in storage, so you should check periodically, even if you do verify at the time of backup. Keeping digests using programs such as `md5sum` or `sha1sum` is also a good check on the integrity of a backup file.

Restore filesystems from backups

A counterpart to backing up files is the ability to restore them when needed. Occasionally you will want to restore an entire filesystem, but it is far more common to need to restore only specific files or perhaps a set of directories. Almost always you will restore to some temporary space and verify that what you have restored is indeed what you want and is consistent with the current state of your system before actually making the restored files live.

A related issue is the need to verify that the items you want happen to be on a particular backup, as often happens when a user needs access to a version of a file that was modified or perhaps deleted "sometime in the last week or two." With these thoughts in mind, let's look at some of the restoration options.

Restoring a dd archive

Recall that the `dd` command was not filesystem aware, so you will need to restore a dump of a partition to find out what is on it. Listing 46 shows how to restore the partition that was dumped back in Listing 39 to a partition, `/dev/sdc7`, that was specially created on a removable USB drive just for this purpose.

Listing 46. Restoring a partition using dd

```
[root@lyrebird ~]# dd if=backup-1 of=/dev/sdc7
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 44.0084 s, 23.7 MB/s
```

Recall that we added some files to the filesystem on `/dev/sda3` after this backup was taken. If you mount the newly restore partition and compare it with the original, you will see that this is indeed the case, as shown in Listing 47. Note that the file whose timestamp was updated using `touch` is not shown here, as you would expect.

Listing 47. Comparing the restored partition with current state

```
[root@lyrebird ~]# mount /dev/sdc7 /mnt/temp-dd/
[root@lyrebird ~]# diff -rq /grubfile/ /mnt/temp-dd/
Only in /grubfile/: newerfile
Only in /grubfile/: newfile
```

Restoring a dump archive using restore

Recall that our final use of `dump` was a differential backup and that we created a table of contents. Listing 48 shows how to use `restore` to check the files in the archive created by `dump`, using the archive itself (`backup-5`) or the table of contents (`backup-6-toc`).

Listing 48. Checking the contents of archives

```
[root@lyrebird ~]# restore -t -f backup-5
Dump tape is compressed.
Dump   date: Sun Jul  8 16:55:46 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 2 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
[root@lyrebird ~]# restore -t -A backup-6-toc
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
Starting inode numbers by volume:
Volume 1: 2
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
      17     ./newerfile
```

The `restore` command can also compare the contents of an archive with the contents of the filesystem using the `-C` option. In Listing 49 we updated `newerfile` and then compared the backup with the filesystem.

Listing 49. Comparing an archive with a filesystem using restore

```
[root@lyrebird ~]# echo "something different" >/grubfile/newerfile
[root@lyrebird ~]# restore -C -f backup-6
Dump tape is compressed.
```

```
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
fileSYS = /grubfile
./newerfile: size has changed.
Some files were modified!  1 compare errors
```

The `restore` command can restore interactively or automatically. Listing 50 shows how to restore `newerfile` to root's home directory (so you could examine it before replacing the updated file if needed), then replace the updated file with the backup copy. This example illustrates interactive restoration.

Listing 50. Restoring a file using restore

```
[root@lyrebird ~]# restore -i -f backup-6
Dump tape is compressed.
restore > ?
Available commands are:
  ls [arg] - list directory
  cd arg - change directory
  pwd - print current directory
  add [arg] - add `arg' to list of files to be extracted
  delete [arg] - delete `arg' from list of files to be extracted
  extract - extract requested files
  setmodes - set modes of requested directories
  quit - immediately exit program
  what - list dump header information
  verbose - toggle verbose flag (useful with ``ls'')
  prompt - toggle the prompt display
  help or `?' - print this list
If no `arg' is supplied, the current directory is used
restore > ls new*
newerfile
newfile
restore > add newerfile
restore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume # (none if no more volumes): 1
set owner/mode for '.'? [yn] y
restore > q
[root@lyrebird ~]# mv -f newerfile /grubfile
```

Restoring a cpio archive

The `cpio` command in copy-in mode (option `-i` or `--extract`) can list the contents of an archive or restore selected files. When you list the files, specifying the `--absolute-filenames` option reduces the number of extraneous messages that `cpio` will otherwise issue as it strips any leading `/` characters from each path that has one. Partial output from listing our previous archive is shown in Listing 51.

Listing 51. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio -id --list --absolute-filenames <backup-cpio-1
```

```
/home/ian/.gstreamer-0.10/registry.i686.xml
/home/ian/.gstreamer-0.10
/home/ian/.Trash/gnome-terminal.desktop
/home/ian/.Trash
/home/ian/.bash_profile
```

Listing 52 shows how to restore all the files with "samp" in their path name or file name. The output has been piped through `uniq` to reduce the number of "Removing leading '/' ..." messages. You must specify the `-d` option to create directories; otherwise, all files are created in the current directory. Furthermore, `cpio` will not replace any newer files on the filesystem with archive copies unless you specify the `-u` or `--unconditional` option.

Listing 52. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio -ivd "*samp*" < backup-cpio-1 2>&1 |uniq
cpio: Removing leading `/' from member names
home/ian/crontab.samp
cpio: Removing leading `/' from member names
home/ian/sample.file
cpio: Removing leading `/' from member names
18855 blocks
```

Restoring a tar archive

The `tar` command can also compare archives with the current filesystem as well as restore files from archives. Use the `-d`, `--compare`, or `--diff` option to perform comparisons. The output will show files whose contents differ as well as files whose time stamps differ. Listing 53 shows verbose output (using option `-v`), from a comparison of the file created earlier and the files in `/etc` after `/etc/crontab` has been touched to alter its time stamp. The option `directory /` instructs `tar` to perform the comparison starting from the root directory rather than the current directory.

Listing 53. Comparing archives and files using tar

```
[root@lyrebird ~]# touch /etc/crontab
[root@lyrebird ~]# tar --diff -vf backup-tar-1 --directory /
etc/anacrontab
etc/crontab
etc/crontab: Mod time differs
etc/cron.d/
etc/cron.d/sa-update
etc/cron.d/smolt
```

Listing 54 shows how to extract just `/etc/crontab` and `/etc/anacrontab` into the current directory.

Listing 54. Extracting archive files using tar

```
[root@lyrebird ~]# tar -xzvf backup-tar-1 "*tab"
```

```
etc/anacrontab
etc/crontab
```

Note that `tar`, in contrast to `cpio` creates the directory hierarchy for you automatically.

The next section of this tutorial shows you how to maintain system time.

Section 7. System time

This section covers material for topic 1.111.6 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

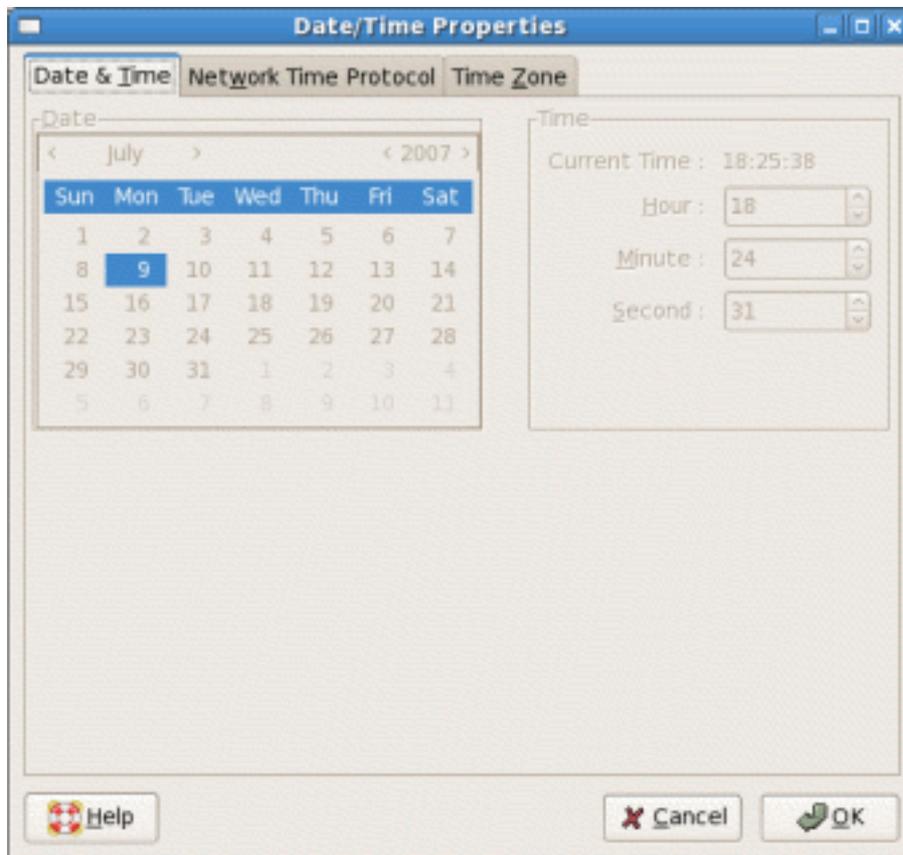
- Set the system date and time
- Set the BIOS clock to the correct UTC time
- Configure your time zone
- Configure the Network Time Protocol (NTP) service, including correcting for clock drift

Set the system date and time

System time on a Linux system is very important. You saw earlier how the cron and anacron facilities do things based on time, so they need an accurate time to base decisions on. Most of the backup and restore tools discussed in the previous section, along with development tools such as `make`, also depend on reliable time measurements. Most computers built since around 1980 include some kind of clock mechanism, and most built since 1984 or so have a persistent clock mechanism that keeps time even if the computer is turned off.

If you installed a Linux system graphically, you probably set the clock and chose a time zone suitable for your needs. You may have elected to use the Network Time Protocol (NTP) to set your clock, and you may or may not have elected to keep the system clock using Coordinated Universal Time or UTC. If you subsequently went to set the clock using graphical tools on a Fedora or Red Hat or similar system, you may have seen a dialog box like that in Figure 3.

Figure 3. Updating the date and time



Surprise! You can't actually set the clock yourself using this dialog. In this section you learn more about the difference between local clocks and NTP and how to set your system time.

No matter whether you live in New York, Budapest, Nakhodka, Ulan Bator, Bangkok, or Canberra, most of your Linux time computations are related to Coordinated Universal Time or UTC. If you run a dedicated Linux system, it is customary to keep the hardware clock set to UTC, but if you also boot another operating system such as Windows, you may need to set the hardware clock to local time. It really doesn't matter as far as Linux is concerned, except that there happen to be two different methods of keeping track of time zones internally in Linux, and if they don't agree, you can wind up with some odd time stamps on FAT filesystems, among other things. Listing 55 shows you how to use the `date` command to display the current date and time. The display is always in local time, even if your hardware clock keeps UTC time.

Listing 55. Displaying the current date and time

```
[root@lyrebird ~]# date;date -u
Mon Jul 9 22:40:01 EDT 2007
```

The `date` command supports a wide variety of possible output formats, some of which you already saw back in [Listing 28](#). See the man page for `date` if you'd like to learn more about the various date formats.

If you need to set the date, you can do this by providing a date and time as an argument. The required format is historical and is somewhat odd even to Americans and truly odd to the rest of the world. You must specify at least month, day, hour, and minute in MMDDhhmm format, and you may also append a two- or four-digit year (CCYY or YY) and optionally a period (.) followed by a two-digit number of seconds. Listing 56 shows an example that alters the system date by a little over a minute.

Listing 56. Setting the system date and time

```
[root@lyrebird ~]# date; date 0709221407;date
Mon Jul  9 23:12:37 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
```

Set the BIOS clock to UTC time

Your Linux system, along with most other current operating systems, actually has two clocks. The first is the hardware clock, sometimes called the Real Time Clock, RTC, or BIOS clock, which is usually tied to an oscillating quartz crystal that is accurate to within a few seconds per day. It is subject to variations such as ambient temperature. The second is the internal software clock, which is driven by counting system interrupts. It is subject to variations caused by high system load and interrupt latency. Nevertheless, your system typically reads the hardware clock at startup and from then on uses the software clock. The `date` command that you just learned about sets the software clock, not the hardware clock.

If you use the Network Time Protocol (NTP), you may possibly set the hardware clock when you first install the system and never worry about it again. If not, this part of the tutorial will show you how to display and set the hardware clock time.

You can use the `hwclock` command to display the current value of the hardware clock. Listing 57 shows the current value of both the system and hardware clocks.

Listing 57. System and hardware clock values

```
[root@lyrebird ~]# date;hwclock
Mon Jul  9 22:16:11 EDT 2007
Mon 09 Jul 2007 11:14:49 PM EDT -0.071616 seconds
```

Notice that the two values are different. You can synchronize the hardware clock

from the system clock using the `-w` or `--systohc` option of `hwclock`, and you can synchronize the system clock from the hardware clock using the `-s` or `--hctosys` option, as shown in Listing 58.

Listing 58. Setting the system clock from the hardware clock

```
[root@lyrebird ~]# date;hwclock;hwclock -s;date
Mon Jul  9 22:20:23 EDT 2007
Mon 09 Jul 2007 11:19:01 PM EDT  -0.414881 seconds
Mon Jul  9 23:19:02 EDT 2007
```

You may specify either the `--utc` or the `--localtime` option to have the system clock kept in UTC or local time. If no value is specified, the value is taken from the third line of `/etc/adjtime`.

The Linux kernel has a mode that copies the system time to the hardware clock every 11 minutes. This is off by default, but is turned on by NTP. Running anything that set the time the old fashioned way, such as `hwclock --hctosys`, turns it off, so it's a good idea to just let NTP do its work if you are using NTP. See the man page for `adjtimex` to find out how to check whether the clock is being updated every 11 minutes or not. You may need to install the `adjtimex` package as it is not always installed by default.

The `hwclock` command keeps track of changes made to the hardware clock in order to compensate for inaccuracies in the clock frequency. The necessary data points are kept in `/etc/adjtime`, which is an ASCII file. If you are not using the Network Time Protocol, you can use the `adjtimex` command to compensate for clock drift. Otherwise, the hardware clock will be adjusted approximately every 11 minutes by NTP. Besides showing whether your hardware clock is in local or UTC time, the first value in `/etc/adjtime` shows the amount of hardware clock drift per day (in seconds). Listing 59 shows two examples.

Listing 59. /etc/adjtime showing clock drift and local or UTC time.

```
[root@lyrebird ~]# cat /etc/adjtime
0.000990 1184019960 0.000000
1184019960
LOCAL
root@pinguino:~# cat /etc/adjtime
-0.003247 1182889954 0.000000
1182889954
LOCAL
```

Note that both these systems keep the hardware clock in local time, but the clock drifts are different — 0.000990 on lyrebird and -0.003247 on pinguino.

Configure your time zone

Your time zone is a measure of how far your local time differs from UTC. Information on available time zones that can be configured is kept in `/usr/share/zoneinfo`. Traditionally, `/etc/localtime` was a link to one of the time zone files in this directory tree, for example, `/usr/share/zoneinfo/Eire` or `/usr/share/zoneinfo/Australia/Hobart`. On modern systems it is much more likely to be a copy of the appropriate time zone data file since the `/usr/share` filesystem may not be mounted when the local time zone information is needed early in the boot process.

Similarly, another file, `/etc/timezone` was traditionally a link to `/etc/default/init` and was used to set the time zone environment variable `TZ`, and several locale-related environment variables. The file may or may not exist on your system. If it does, it may simply contain the name of the current time zone. You may also find time zone information in `/etc/sysconfig/clock`. Listing 60 shows these files from a Ubuntu 7.04 and a Fedora 7 system.

Listing 60. Time zone information in `/etc`

```
root@pinguino:~# cat /etc/timezone
America/New_York

[root@lyrebird ~]# cat /etc/sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
# The timezone of the system is defined by the contents of
/etc/localtime.
ZONE="America/New York"
UTC=false
ARC=false
```

Some systems such as Debian and Ubuntu have a `tzconfig` command to set the time zone. Others such as Fedora use `system-config-date` to set the time zone and to indicate whether the clock uses UTC or not. Listing 61 illustrates the use of the `tzconfig` command to display the current time zone.

Listing 61. Setting time zone with `tzconfig`

```
root@pinguino:~# tzconfig
Your current time zone is set to America/New_York
Do you want to change that? [n]:
Your time zone will not be changed
```

Configure the Network Time Protocol

The *Network Time Protocol (NTP)* is a protocol to synchronize computer clocks over a network. Synchronization is usually to UTC.

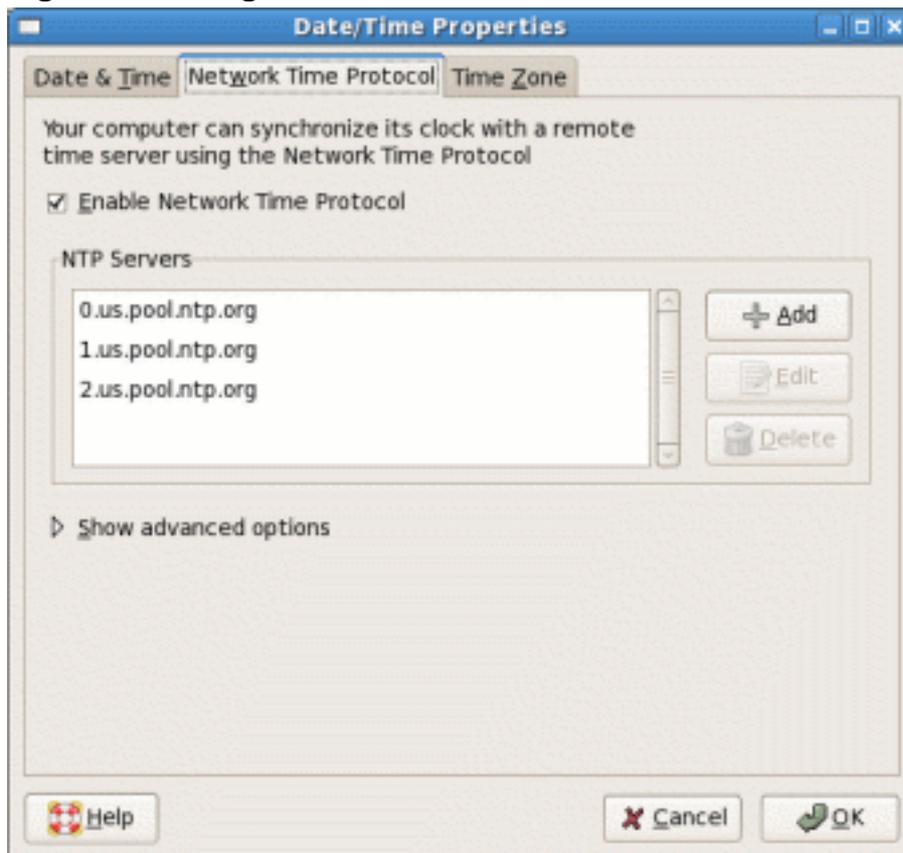
NTP version 3 is an Internet draft standard, formalized in RFC 1305. The current development version, NTP version 4 is a significant revision, which has not been formalized in an RFC. RFC 4330 describes Simple NTP (SNTP) version 4.

Time synchronization is accomplished by sending messages to *time servers*. The time returned is adjusted by an offset of half the round-trip delay. The accuracy of the time is therefore dependent on the network latency and the extent to which the latency is the same in both directions. The shorter the path to a time server, the more accurate the time is likely to be. See [Resources](#) for more detailed information than this simplistic description can provide.

There is a huge number of computers on the Internet, so time servers are organized into *strata*. A relatively small number of stratum 1 servers maintain very accurate time from a source such as an atomic clock. A larger number of stratum 2 servers get their time from stratum 1 servers and make it available to an even larger number of stratum 3 servers, and so on. To ease the load on time servers, a large number of volunteers donate time services through pool.ntp.org (see [Resources](#) for a link). Round robin DNS servers accomplish NTP load balancing by distributing NTP server requests among a pool of available servers.

If you use a graphical interface, you might be able to set your NTP time servers using a dialog similar to that in Figure 4. The fact that this system has enabled automatic time updates using NTP is why the dialog in Figure 3 did not allow the date and time to be changed.

Figure 4. Setting NTP servers



NTP configuration information is kept in `/etc/ntp.conf`, so you can also edit that file and then restart the `ntpd` daemon after you save it. Listing 62 shows an example `/etc/ntp.conf` file using the time servers from Figure 4.

Listing 62. Setting time zone with `tzconfig`

```
[root@lyrebird ~]# cat /etc/ntp.conf
# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#broadcast 192.168.1.255 key 42          # broadcast server
#broadcastclient          # broadcast client
#broadcast 224.0.1.1 key 42            # multicast server
#multicastclient 224.0.1.1            # multicast client
#manycastserver 239.255.254.254        # manycast server
#manycastclient 239.255.254.254 key 42 # manycast client

# Undisciplined Local Clock. This is a fake driver intended for backup
# and when no outside source of synchronized time is available.
#server 127.127.1.0 # local clock
#fudge 127.127.1.0 stratum 10

# Drift file. Put this in a directory which the daemon can write to.
# No symbolic links allowed, either, since the daemon updates the file
# by creating a temporary in the same directory and then rename()'ing
# it to the file.
driftfile /var/lib/ntp/drift

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8
server 0.us.pool.ntp.org
restrict 0.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 1.us.pool.ntp.org
restrict 1.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 2.us.pool.ntp.org
restrict 2.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
```

If you are using the `pool.ntp.org` time servers, these may be anywhere in the world. You will usually get better time by restricting your servers as in this example where `us.pool.ntp.org` is used, resulting in only U.S. servers being chosen. See [Resources](#)

for more information on the ntp.pool.org project.

NTP commands

You can use the `ntpdate` command to set your system time from an NTP time server as shown in Listing 63.

Listing 63. Setting system time from an NTP server using ntpdate

```
[root@lyrebird ~]# ntpdate 0.us.pool.ntp.org
10 Jul 10:27:39 ntpdate[15308]: adjust time server 66.199.242.154 offset
-0.007271 sec
```

Because the servers operate in round robin mode, the next time you run this command you will probably see a different server. Listing 64 shows the first few DNS responses for `0.us.ntp.pool.org` a few moments after the above `ntpdate` command was run.

Listing 64. Round robin NTP server pool

```
[root@lyrebird ~]# dig 0.pool.ntp.org +noall +answer | head -n 5
0.pool.ntp.org. 1062 IN A 217.116.227.3
0.pool.ntp.org. 1062 IN A 24.215.0.24
0.pool.ntp.org. 1062 IN A 62.66.254.154
0.pool.ntp.org. 1062 IN A 76.168.30.201
0.pool.ntp.org. 1062 IN A 81.169.139.140
```

The `ntpdate` command is now deprecated as the same function can be done using `ntpq` with the `-q` option, as shown in Listing 65.

Listing 65. Setting system time using ntpd -q

```
[root@lyrebird ~]# ntpd -q
ntpd: time slew -0.014406s
```

Note that the `ntpd` command uses the time server information from `/etc/ntp.conf`, or a configuration file provided on the command line. See the man page for more information and for information about other options for `ntpd`. Be aware also that if the `ntpd` daemon is running, `ntpd -q` will quietly exit, leaving a failure message in `/var/log/messages`.

Another related command is the `ntpq` command, which allows you to query the NTP daemon. See the man page for more details.

This brings us to the end of this tutorial. We have covered a lot of material on system administration. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- See the [Partimage homepage](#) for information on Partimage, a filesystem-aware partition dump and restore tool.
- "[/etc: Host-specific system configuration](#)" describes the Linux Standard Base (LSB) requirements for /etc.
- The [Network Time Protocol Project](#) produces a reference implementation of the NTP protocol, and implementation documentation.
- The [Network Time Synchronization Project](#) maintains an extensive array of documentation and background information, including briefing slides, on network time protocols.
- The [pool.ntp.org project](#) is a big virtual cluster of timeservers striving to provide reliable easy to use NTP service for millions of clients without putting a strain on the big popular timeservers.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content.](#)
- Get involved in the [developerWorks community](#) through our developer blogs, forums, podcasts, and community topics in our new [developerWorks spaces](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 102 prep, Topic 111: Administrative tasks

Junior Level Administration (LPIC-1) topic 111

Skill Level: Intermediate

[Ian Shields \(ishields@us.ibm.com\)](mailto:ishields@us.ibm.com)
Senior Programmer
IBM

10 Jul 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this sixth in a [series of nine tutorials](#), Ian introduces you to administrative tasks. By the end of this tutorial, you will know how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *intermediate level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active intermediate level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the four junior and intermediate certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics		
LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts.
Topic 111	LPI exam 102 prep: Administrative tasks	(This tutorial.) Learn how to manage user and group accounts and tune user and system environments, configure and use system log files, automate system administration tasks by scheduling jobs to run at another time, back up your system, and maintain system time. See the detailed objectives below.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Administrative tasks," the sixth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
1.111.1 User and group accounts	Weight 4	Add, remove, suspend, and change user accounts. Manage user and group information in password and group databases, including shadow databases. Create and manage special purpose and limited accounts.
1.111.2 Tune user and system environments	Weight 3	Modify global and user profiles. Set environment variables and maintain skeleton directories for new user accounts. Set command search paths.
1.111.3 Configure and use system log files to meet administrative and security needs	Weight 3	Configure and manage system logs, including the type and level of logged information. Scan and monitor log files for

		notable activity and track down noted problems. Rotate and archive log files.
1.111.4 Automate system administration tasks by scheduling jobs to run in the future	Weight 4	Use the <code>cron</code> or <code>anacron</code> commands to run jobs at regular intervals, and use the <code>at</code> command to run jobs at a specific time.
1.111.5 Maintain an effective data backup strategy	Weight 3	Plan a backup strategy and back up filesystems automatically to various media.
1.111.6 Maintain system time	Weight 4	Maintain the system time and time zone, and synchronize the clock via NTP. Set the BIOS clock to the correct time in UTC, and configure NTP, including correcting for clock drift.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be thoroughly familiar with the material from the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, which covers basic concepts of users, groups, and file permissions.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. User and group accounts

This section covers material for topic 1.111.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Add, modify, and remove users and groups
- Suspend and change user accounts
- Manage user and group information in the password databases and group databases
- Use the correct tools to manage shadow password databases and group databases
- Create and manage limited and special-purpose accounts

As you learned in the "[LPI exam 101 prep \(topic 104\) Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#)" tutorial, Linux is a multi-user system where each user belongs to one *primary* group and possibly to additional groups. Ownership of files in Linux is closely related to user ids and groups. Recall that you can log in as one user and become another user using the `su` or `sudo -s` commands, and that you can use the `whoami` command to check your current effective id and the `groups` command to find out what groups you belong to. In this section, you learn how to create, delete, and manage users and groups. You also learn about the files in `/etc`, where user and group information is stored.

Add and remove users and groups

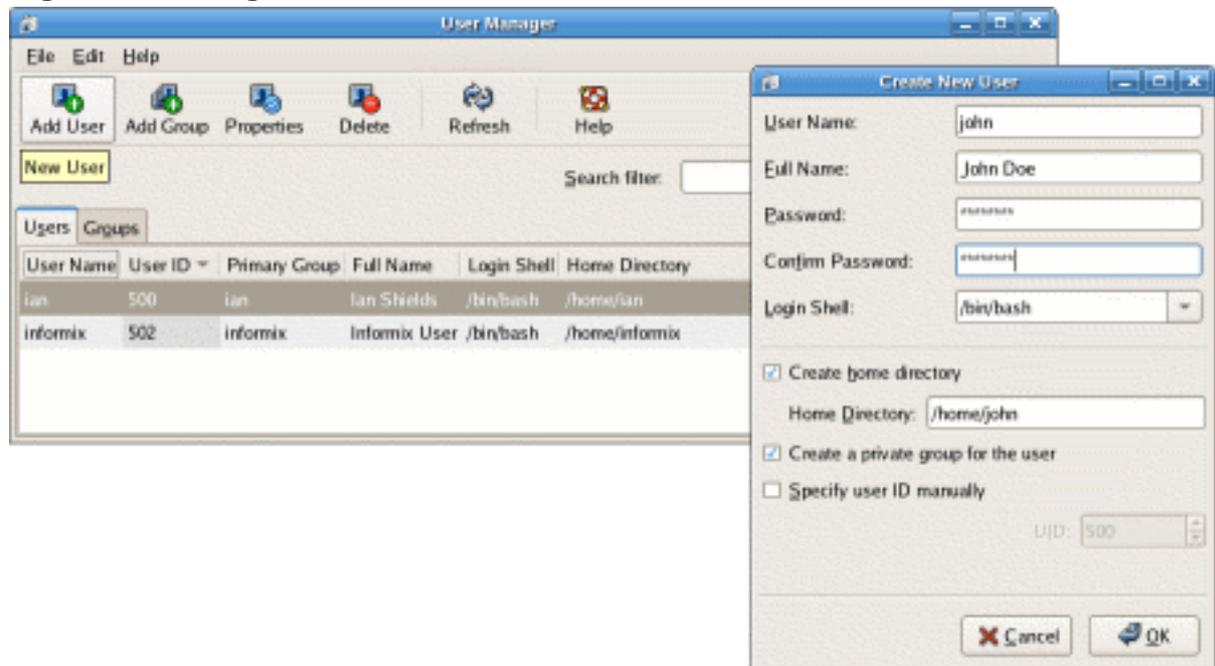
You add a user to a Linux system using the `useradd` command, and you delete a user using the `userdel` command. Similarly, you add or delete groups using the `groupadd` and `groupdel` commands.

Adding a user or group

Modern Linux desktops usually have graphical interfaces for user and group administration. The graphical interface is usually accessed through menu options for system administration. These interfaces do vary considerably, so the one on your system may not look much like the example here, but the underlying concepts and commands remain similar.

Let's start by adding a user to a Fedora Core 5 system graphically, and then examine the underlying commands. In the case of Fedora Core 5 with GNOME desktop, use **System > Administration > Users and Groups**, then click the **Add User** button.

Figure 1 depicts the User Manager panel with the Create New User panel showing basic information for a new user named 'john'. The full name of the user, John Doe, and a password have been entered. The panel provides a default login shell of `/bin/bash`. On Fedora systems, the default is to create a new group with the same name as the user, 'john' in this case, and a home directory of `/home/john`.

Figure 1. Adding a user

Listing 1 shows the use of the `id` command to display basic information about the new user. As you can see, john has user number 503 and a matching group, john, with group number 503. This is the only group of which john is a member.

Listing 1. Displaying user id information

```
[root@pinguino ~]# id john
uid=503(john) gid=503(john) groups=503(john)
```

To accomplish the same task from the command line, you use the `groupadd` and `useradd` commands to create the group and user, then use the `passwd` command to set the password for the newly created user. All of these commands require root authority. The basic use of these commands to add another user, jane, is illustrated in Listing 2.

Listing 2. Adding user jane

```
[root@pinguino ~]# groupadd jane
[root@pinguino ~]# useradd -c "Jane Doe" -g jane -m jane
[root@pinguino ~]# passwd jane
Changing password for user jane.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# id jane
uid=504(jane) gid=504(jane) groups=504(jane)
[root@pinguino ~]# ls -ld /home/jane
drwx----- 3 jane jane 4096 Jun 25 18:22 /home/jane
```

In these two examples, both the user id and the group id have values greater than 500. Be aware that some newer systems start user ids at 1000 rather than 500. These values normally signify ordinary users, while values below 500 (or 1000 if the system starts ordinary users at 1000) are reserved for *system users*. [System users](#) are covered later in this section. The actual cutoff points are set in `/etc/login.defs` as `UID_MIN` and `GID_MIN`.

In Listing 2 above, the `groupadd` command has a single parameter, `jane`, the name of the group to be added. Group names must begin with a lower case letter or an underscore, and usually contain only these along with hyphens or dashes. Options you may specify are shown in Table 3.

Option	Purpose
-f	Exit with success status if the group already exists. This is handy for scripting when you do not need to check if a group exists before attempting to create it.
-g	Specifies the group id manually. The default is to use the smallest value that is at least <code>GID_MIN</code> and also greater than the id of any existing group. Group ids are normally unique and must be non-negative
-o	Permits a group to have a non-unique id.
-K	Can be used to override defaults from <code>/etc/login.defs</code> .

In Listing 2 above, the `useradd` command has a single parameter, `jane`, the name of the user to be added, along with the `-c`, `-g`, and `-m` options. Common options for the `useradd` command are shown in Table 4.

Option	Purpose
-b --base-dir	The default base directory in which user home directories are created. This is usually <code>/home</code> , and the user's home directory is <code>/home/\$USER</code> .
-c --comment	A text string describing the id, such as the user's full name.
-d --home	Provides a specific directory name for the home directory.
-e	The date on which the account will

--expiredate	expire or be disabled in the form YYYY-MM_DD.
-g --gid	The name or number of the initial login group for the user. The group must exist, which is why group jane was created before user jane in Listing 2.
-G --groups	A comma-separated list of additional groups to which the user belongs.
-K	Can be used to override defaults from /etc/login.defs.
-m --create-home	Create the user's home directory if it does not exist. Copy the skeleton files and any directories from /etc/skel to the home directory.
-o --non-unique	Permits a user to have a non-unique id.
-p --password	The encrypted password. If a password is not specified, the default is to disable the account. You will usually use the <code>passwd</code> command in a subsequent step rather than generating an encrypted password and specifying it on the <code>useradd</code> command.
-s --shell	The name of the user's login shell if different from the default login shell.
-u --uid	The non-negative numerical userid, which must be unique if <code>-o</code> is not specified. The default is to use the smallest value that is at least <code>UID_MIN</code> and also greater than the id of any existing user.

Notes:

1. Some systems, including Fedora and Red Hat distributions, have extensions to the user-creation commands. For example, the default Fedora and Red Hat behavior is to create a new group for a user, and the `-n` option can be used on the `useradd` command to disable this function. Be aware of such possible system differences and refer to the man pages on your system when in doubt.
2. On SUSE systems, use YaST or YaST2 to access graphical user and group administration interfaces.
3. Graphical interfaces may perform additional tasks such as creating the user's mail file in `/var/spool/mail`.

Deleting a user or group

Deleting a user or group is much simpler than adding one, because there are fewer options. In fact, the `groupdel` command to delete a group requires only the group name; it has no options. You cannot delete any group that is the primary group of a user. If you use a graphical interface for deleting users and groups, the functions are very similar to the commands shown here.

Use the `userdel` command to delete a user. The `-r`, or `--remove` option requests removal of the user's home directory and anything it contains, along with the user's mail spool. When you delete a user, a group with the same name as the user will also be deleted if `USERGROUPS_ENAB` is set to `yes` in `/etc/login.defs`, but this will be done only if the group is not the primary group of another user.

In Listing 3 you see an example of deleting groups when multiple users share the same primary group. Here, another user, `jane2`, has previously been added to the system with the same group as `jane`.

Listing 3. Deleting users and groups

```
root@pinguino:~# groupdel jane
groupdel: cannot remove user's primary group.
root@pinguino:~# userdel -r jane
userdel: Cannot remove group jane which is a primary group for another
user.
root@pinguino:~# userdel -r jane2
root@pinguino:~# groupdel jane
```

Notes:

1. There is a `userdel` option, `-f` or `--force`, which can be used to delete users and their group. This option is dangerous, so you should use it only as a last resort. Read the man page carefully before you do.
2. Be aware that if you delete a user or group, and there are files that belong to that user or group on your filesystem, then the files are not automatically deleted or assigned to another user or group.

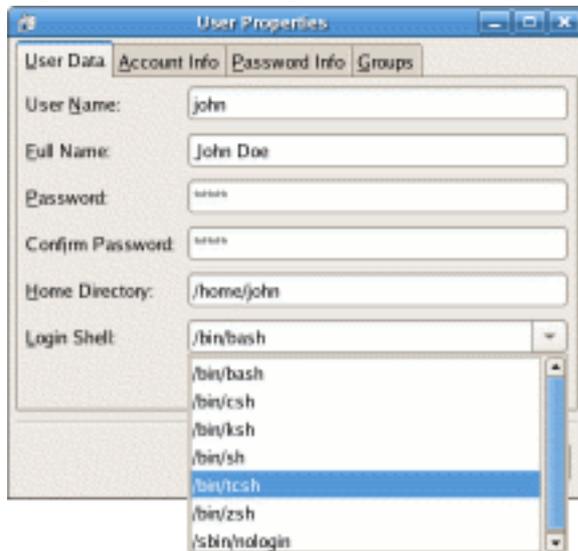
Suspend and change accounts

Now that you can create or delete a user id or a group, you may also find a need to modify one.

Modifying user accounts

Suppose user john wishes to have the tcsh shell as his default. From a graphical interface you will usually find a way to either edit a user (or group), or to examine the properties of the object. Figure 2 shows the properties dialog for the user john that we created earlier on a Fedora Core 5 system.

Figure 2. Modifying a user account



From the command line, you can use the `usermod` command to modify a user account. You can use most of the options that you use with `useradd`, except that you cannot create or populate a new home directory for the user. If you need to change the name of the user, specify the `-l` or `--login` option with the new name. You will probably want to rename the home directory to match the user id. You may also need to rename other items such as mail spool files. Finally, if the login shell is changed, some of the associated profile files may need to be altered. Listing 4 shows an example of the things you might need to do to change user john to john2 with `/bin/tcsh` as the default shell and renamed home directory `/home/john2`.

Listing 4. Modifying a user

```
[root@pinguino ~]# usermod -l john2 -s /bin/tcsh -d /home/john2 john
[root@pinguino ~]# ls -d ~john2
ls: /home/john2: No such file or directory
[root@pinguino ~]# mv /home/john /home/john2
[root@pinguino ~]# ls -d ~john2
/home/john2
```

Notes:

1. If you need to modify a user's additional groups, you must specify the complete list of additional groups. There is no command to simply add or delete a single group for a user.

2. There are restrictions on changing the name or id of a user who is logged in or who has running processes. Check the man pages for details.
3. If you change a user number, you may want to change files and directories owned by that user to match the new number.

Modifying groups

Not surprisingly, the `groupmod` command is used to modify group information. You can change the group number with the `-g` option, and the name with the `-n` option.

Listing 5. Renaming a group

```
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# groupmod -n john2 john
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

Notice in Listing 5 that the group name for the home directory of john2 magically changed when we used `groupmod` to change the group name. Are you surprised? Because groups are represented in the filesystem inodes by their number rather than by their name, this is not surprising. However, if you change a group's number, you should update any users for which that group is the primary group, and you may also want to update the files and directories belonging to that group to match the new number (in the same way as noted above for changing a user number). Listing 6 shows how to change the group number for john2 to 505, update the user account, and make appropriate changes to all the affected files in the `/home` filesystem. You probably want renumbering users and groups if at all possible.

Listing 6. Renumbering a group

```
[root@pinguino ~]# groupmod -g 505 john2
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# id john2
uid=503(john2) gid=503 groups=503
[root@pinguino ~]# usermod -g john2 john2
[root@pinguino ~]# id john2
uid=503(john2) gid=505(john2) groups=505(john2)
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# find /home -gid 503 -exec chgrp john2 {} \;
[root@pinguino ~]# ls -ld ~john2
drwx----- 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

User and group passwords

You have already seen the `passwd` command, which is used to change a user password. The password is (or should be) unique to the user and may be changed by user. The root user may change any user's password as we have already seen.

Groups may also have passwords, and the `gpasswd` command is used to set them. Having a group password allows users to join a group temporarily with the `newgrp` command, if they know the group password. Of course, having multiple people knowing a password is somewhat problematic, so you will have to weigh the advantages of adding a user to a group using `usermod`, versus the security issue of having too many people knowing the group password.

Suspending or locking accounts

If you need to prevent a user from logging in, you can *suspend* or *lock* the account using the `-L` option of the `usermod` command. To *unlock* the account, use the `-U` option. Listing 7 shows how to lock account `john2` and what happens if `john2` attempts to log in to the system. Note that when the `john2` account is unlocked, the same password is restored.

Listing 7. Locking an account

```
[root@pinguino ~]# usermod -L john2
[root@pinguino ~]# ssh john2@pinguino
john2@pinguino's password:
Permission denied, please try again.
```

You may have noticed back in [Figure 2](#) that there were several tabs on the dialog box with additional user properties. We briefly mentioned the use of the `passwd` command for setting user passwords, but both it and the `usermod` command can perform many tasks related to user accounts, as can another command, the `chage` command. Some of these options are shown in Table 5. Refer to the appropriate man pages for more details on these and other options.

Table 5. Commands and options for changing user accounts			
	Option for command		Purpose
Usermod	Passwd	Chage	
-L	-l	N/A	Lock or suspend the account.
-U	-u	N/A	Unlock the account.
N/A	-d	N/A	Disable the account by setting it passwordless.

-e	-f	-E	Set the expiration date for an account.
N/A	-n	-m	The minimum password lifetime in days.
N/A	-x	-M	The maximum password lifetime in days.
N/A	-w	-W	The number of days of warning before a password must be changed.
-f	-i	-l	The number of days after a password expires until the account is disabled.
N/A	-S	-l	Output a short message about the current account status.

Manage user and group databases

The primary repositories for user and group information are four files in `/etc`.

`/etc/passwd`

is the *password* file containing basic information about users

`/etc/shadow`

is the *shadow password* file containing encrypted passwords

`/etc/group`

is the *group* file containing basic information about groups and which users belong to them

/etc/gshadow

is the *shadow group* file containing encrypted group passwords

These files are updated by the commands you have already seen in this tutorial and you will meet some more commands for working with them after we discuss the files themselves. All of these files are plain text files. In general, you should not edit them directly. Use the tools provided for updating them so they are properly locked and kept synchronized.

You will note that the passwd and group files are both *shadowed*. This is for security reasons. The passwd and group files themselves must be world readable, but the encrypted passwords should not be world readable. Therefore, the shadow files contain the encrypted passwords, and these files are only readable by root. The necessary authentication access is provided by an suid program that has root authority, but can be run by anyone. Make sure that your system has the permissions set appropriately. Listing 8 shows an example.

Listing 8. User and group database permissions

```
[ian@pinguino ~]$ ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow
-rw-r--r-- 1 root root 701 Jun 26 19:04 /etc/group
-r----- 1 root root 580 Jun 26 19:04 /etc/gshadow
-rw-r--r-- 1 root root 1939 Jun 26 19:43 /etc/passwd
-r----- 1 root root 1324 Jun 26 19:50 /etc/shadow
```

Note: Although it is still technically possible to run without shadowed password and group files, this is almost never done and is not recommended.

The /etc/passwd file

The /etc/passwd file contains one line for each user in the system. Some example lines are shown in Listing 9.

Listing 9. /etc/password entries

```
root:x:0:0:root:/root:/bin/bash
jane:x:504:504:Jane Doe:/home/jane:/bin/bash
john2:x:503:505:John Doe:/home/john2:/bin/tcsh
```

Each line contains seven fields separated by colons (:), as shown in Table 6.

Table 6. Fields in /etc/passwd	
Field	Purpose
Username	The name used to log in to the system.

	For example, john2.
Password	The encrypted password. When using shadow passwords, it contains a single x character.
User id (UID)	The number used to represent this user name in the system. For example, 503 for user john2.
Group id (GID)	The number used to represent this user's primary group in the system. For example, 505 for user john2.
Comment (GECOS)	An optional field used to describe the user. For example, "John Doe". The field may contain multiple comma-separated entries. It is also used by programs such as <code>finger</code> . The GECOS name is historic. See details in <code>man 5 passwd</code> .
Home	The absolute path the user's home directory. For example, <code>/home/john2</code>
Shell	The program automatically launched when a user logs in to the system. This is usually an interactive shell such as <code>/bin/bash</code> or <code>/bin/tcsh</code> , but may be any program, not necessarily an interactive shell.

The `/etc/group` file

The `/etc/group` file contains one line for each group in the system. Some example lines are shown in Listing 10.

Listing 10. `/etc/group` entries

```
root:x:0:root
jane:x:504:john2
john2:x:505:
```

Each line contains four fields separated by colons (:), as shown in Table 7.

Table 7. Fields in <code>/etc/group</code>	
Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The encrypted password. When using shadow group passwords, it contains a single x character.

Group id (GID)	The number used to represent this group in the system. For example, 505 for group john2.
Members	A comma-separated list of group members, excepting those members for whom this is the primary group.

Shadow files

The file `/etc/shadow` should only be readable by root. It contains encrypted passwords, along with password and account expiration information. See the man page (`man 5 shadow`) for information on the field layout. Passwords may be encrypted using DES, but are more usually encrypted using MD5. The DES algorithm uses the low order 7 bits of the first 8 characters of the user password as a 56-bit key, while the MD5 algorithm uses the whole password. In either case, passwords are *salted* so that two otherwise identical passwords do not generate the same encrypted value. Listing 11 shows how to set identical passwords for users jane and john2, and then shows the resulting encoded MD5 passwords in `/etc/shadow`.

Listing 11. Passwords in `/etc/shadow`

```
[root@pinguino ~]# echo lpic1111 |passwd jane --stdin
Changing password for user jane.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# echo lpic1111 |passwd john2 --stdin
Changing password for user john2.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# grep "^j" /etc/shadow
jane:$1$eG0/KGQY$ZJ1.ltYtVw0sv.C5OrqUu/:13691:0:99999:7:::
john2:$1$grkxo6ie$J2muvoTpwo3dZAYYTDYNu.:13691:0:180:7:29::
```

The leading `1` indicates an MD5 password, and the salt is a variable length field of up to 8 characters ending with the next `$` sign. The encrypted password is the remaining string of 22 characters.

Tools for users and groups

You have already seen several commands that manipulate the account and group files and their shadows. Here you learn about:

- Group administrators
- Editing commands for password and group files
- Conversion programs

Group administrators

In some circumstances you may want users other than root to be able to administer one or more groups by adding or removing group members. Listing 12 shows how root can add user jane as an administrator of group john2, and then jane, in turn, can add user ian as a member.

Listing 12. Adding group administrators and members

```
[root@pinguino ~]# gpasswd -A jane john2
[root@pinguino ~]# su - jane
[jane@pinguino ~]$ gpasswd -a ian john2
Adding user ian to group john2
[jane@pinguino ~]$ id ian;id jane
uid=500(ian) gid=500(ian) groups=500(ian),505(john2)
uid=504(jane) gid=504(jane) groups=504(jane)
```

You may be surprised to note that, although jane is an administrator of group john2, she is not a member of it. An examination of the structure of `/etc/gshadow` shows why. The `/etc/gshadow` file contains four fields for each entry as shown in Table 8. Note that the third field is a comma-separated list of administrators for the group.

Table 8. Fields in <code>/etc/gshadow</code>	
Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The field used to contain the encrypted password if the group has a password. If there is no password, you may see 'x', '!' or '!!' here.
Admins	A comma-separated list of group administrators.
Members	A comma-separated list of group members.

As you can see, the administrator list and the member list are two distinct fields. The `-A` option of `gpasswd` allows the root user to add administrators to a group, while the `-M` option allows root to add members. The `-a` (note lower case) option allows an administrator to add a member, while the `-d` option allows an administrator to remove a member. Additional options allow a group password to be removed. See the man pages for details.

Editing commands for password and group files

Although not listed in the LPI objectives, you should also be aware of the `vipw` command for safely editing `/etc/passwd` and `visgr` for safely editing `/etc/group`. The

commands will lock the necessary files while you make changes using the `vi` editor. If you make changes to `/etc/passwd`, then `vipw` will prompt you to see if you also need to update `/etc/shadow`. Similarly, if you update `/etc/group` using `vigr`, you will be prompted to update `/etc/gshadow`. If you need to remove group administrators, you may need to use `vigr`, as `gpasswd` only allows addition of administrators.

Conversion programs

Four other related commands are also not listed in the LPI objectives. They are `pwconv`, `pwunconv`, `grpconv`, and `grpunconv`. They are used for converting between shadowed and non-shadowed password and group files. You may never need these, but be aware of their existence. See the man pages for details.

Limited and special-purpose accounts

By convention, system users usually have an id of less than 100, with root having id 0. Normal users start automatic numbering from the `UID_MIN` value set in `/etc/login.defs`, with this value commonly being set at 500 or 1000.

Besides regular user accounts and the root account on your system, you will usually have several special-purpose accounts, for daemons such as FTP, SSH, mail, news, and so on. Listing 13 shows some entries from `/etc/passwd` for these.

Listing 13. Limited and special-purpose accounts

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
```

Such accounts frequently control files but should not be accessed by normal login. Therefore, they usually have a login shell specified as `/sbin/nologin`, or `/bin/false` so that login attempts will fail.

Section 3. Environment tuning

This section covers material for topic 1.111.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to tune the user environment, including these tasks:

- Set and unset environment variables
- Maintain skeleton directories for new user accounts
- Set command search paths

Set and unset environment variables

When you create a new user, you usually initialize many variable according to your local needs. These are usually set in the profiles that you provide for new users, such as `.bash_profile` and `.bashrc`, or in the system-wide profiles `/etc/profile` and `/etc/bashrc`. Listing 14 shows an example of how the `PS1` system prompt is set in `/etc/profile` on an Ubuntu 7.04 system. The first `if` statement checks whether the `PS1` variable is set, indicating an interactive shell, since a non-interactive shell doesn't need a prompt. The second `if` statement checks whether the `BASH` environment variable is set. If so, it sets a complex prompt and sources (note the dot) `/etc/bash.bashrc`. If the `BASH` variable is not set, then a check is made for root (`id=0`), and the prompt is set to `#` or `$` accordingly.

Listing 14. Setting environment variables

```
if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
```

The tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) has detailed information about the commands used for setting and unsetting environment variables, as well as information about how and when the various profiles are used.

When customizing environments for users, be aware of two major points:

1. `/etc/profile` is read only at login time, and so it is not executed when each new shell is created.

2. Functions and aliases are not inherited by new shells. Therefore, you will usually set these and your environment variables in `/etc/bashrc`, or in the user's own profiles.

In addition to the system profiles, `/etc/profile` and `/etc/bashrc`, the Linux Standard Base (LSB) specifies that additional scripts may be placed in the directory `/etc/profile.d`. These scripts are sourced when an interactive login shell is created. They provide a convenient way of separating customization for different programs. Listing 15 shows an example.

Listing 15. `/etc/profile.d/vim.sh` on Fedora 7

```
[if [ -n "$BASH_VERSION" -o -n "$KSH_VERSION" -o -n "$ZSH_VERSION" ];
then
  [ -x //usr/bin/id ] || return
  [ `//usr/bin/id -u` -le 100 ] && return
  # for bash and zsh, only if no alias is already set
  alias vi >/dev/null 2>&1 || alias vi=vim
fi
```

Remember that you should usually `export` any variables that you set in a profile; otherwise, they will not be available to commands that run in a new shell.

Maintain skeleton directories for new users

You learned in the section [Add and remove users and groups](#) that you can create or populate a new home directory for the user. The source for this new directory is the subtree rooted at `/etc/skel`. Listing 16 shows the files in this subtree for a Fedora 7 system. Note that most files start with a period (dot), so you need the `-a` option to list them. The `-R` options lists subdirectories recursively, and the `-L` option follows any symbolic links.

Listing 16. `/etc/skel` on Fedora 7

```
[ian@lyrebird ~]$ ls -aRL /etc/skel
/etc/skel:
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .xemacs

/etc/skel/.xemacs:
.  ..  init.el
```

In addition to `.bash_logout`, `.bash_profile`, and `.bashrc`, which you might expect for the Bash shell, note that this example includes profile information for the emacs and xemacs editors. See the tutorial [LPI exam 102 prep: Shells, scripting, programming, and compiling](#) if you need to review the functions of the various profile files.

Listing 17 shows `/etc/skel/.bashrc` from the above system. This file might be different on a different release or different distribution, but it gives you an idea of how the default user setup can be done.

Listing 17. `/etc/skel/.bashrc` on Fedora 7

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
```

As you can see, the global `/etc/bashrc` is sourced, then any user specific instructions can be added. Listing 18 shows the part of `/etc/bashrc` in which the `.sh` scripts from `/etc/profile.d` are sourced.

Listing 18. Sourcing `.sh` scripts from `/etc/profile.d`

```
for i in /etc/profile.d/*.sh; do
    if [ -r "$i" ]; then
        . $i
    fi
done
unset i
```

Note that the variable, `i`, is unset after the loop.

Set command search paths

Your default profiles often include `PATH` variables for local functions or for products that you may have installed. You can set these in the skeleton files in `/etc/skel`, modify `/etc/profile`, `/etc/bashrc`, or create a file in `/etc/profile.d` if your system uses that. If you do modify the system files, be sure to check that your changes are intact after any system updates. Listing 19 shows how to add a new directory, `/opt/productxyz/bin`, to either the front or rear of your existing `PATH`.

Listing 19. Adding a path directory

```
PATH="$PATH${PATH:+:}/opt/productxyz/bin"
PATH="/opt/productxyz/bin${PATH:+:}$PATH"
```

Although not strictly required, the expression `${PATH:+:}` inserts a path separator (colon) only if the `PATH` variable is unset or null.

Section 4. System log files

This section covers material for topic 1.111.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to configure and manage system logs, including these tasks:

- Manage the type and level of information logged
- Rotate and archive log files automatically
- Scan log files for notable activity
- Monitor log files
- Track down problems reported in log files

Manage the type and level of information logged

The system logging facility on a Linux system provides system logging and kernel message trapping. Logging can be done on a local system or sent to a remote system, and the level of logging can be finely controlled through the `/etc/syslog.conf` configuration file. Logging is performed by the `syslogd` daemon, which normally receives input through the `/dev/log` socket, as shown in Listing 20.

Listing 20. `/dev/log` is a socket

```
ian@pinguino:~$ ls -l /dev/log
srw-rw-rw- 1 root root 0 2007-07-05 15:42 /dev/log
```

For local logging, the main file is usually `/var/log/messages`, but many other files are used in most installations, and you can customize these extensively. For example, you may want a separate log for messages from the mail system.

The `syslog.conf` configuration file

The `syslog.conf` file is the main configuration file for the `syslogd` daemon. Logging is based on a combination of facility and priority. The defined facilities are `auth` (or `security`), `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `syslog`, `user`, `uucp`, and `local0` through `local7`. The keyword `auth` should be used instead of `security`, and the keyword `mark` is for internal use.

The priorities (in ascending order) are:

1. debug
2. info
3. notice
4. warning (or warn)
5. err (or error)
6. crit
7. alert
8. emerg (or panic)

The parenthesized keywords (warn, error, and panic) are now deprecated.

Entries in `syslog.conf` specify logging rules. Each rule has a selector field and an action field, which are separated by one or more spaces or tabs. The selector field identifies the facility and the priorities that the rule applies to, and the action field identifies the logging action for the facility and priorities. The default behavior is to take the action for the specified level and for all higher levels, although it is possible to limit logging to specific levels. Each selector consists of a facility and a priority separated by a period (dot). Multiple facilities for a given action can be specified by separating them with a comma. Multiple facility/priority pairs for a given action can be specified by separating them with a semi-colon. Listing 21 shows an example of a simple `syslog.conf`.

Listing 21. Example `syslog.conf`

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                               /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages

# The authpriv file has restricted access.
authpriv.*                             /var/log/secure

# Log all the mail messages in one place.
mail.*
-/var/log/maillog

# Log cron stuff
cron.*                                  /var/log/cron
```

```
# Everybody gets emergency messages
*.emerg                                     *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                             /var/log/spooler

# Save boot messages also to boot.log
local7.*
/var/log/boot.log
```

Notes:

- As with many configuration files, lines starting with # and blank lines are ignored.
- An * may be used to refer to all facilities or all priorities.
- The special priority keyword `none` indicates that no logging for this facility should be done with this action.
- The hyphen before a file name (such as `-/var/log/maillog`, in this example) indicates that the log file should not be synchronized after every write. You might lose information after a system crash, but you might gain performance by doing this.

The actions are generically referred to as "logfiles," although they do not have to be real files. Table 9 describe the possible logfiles.

Table 9. Actions in syslog.conf	
Action	Purpose
Regular File	Specify the full pathname, beginning with a slash (/). Prefix it with a hyphen (-) to omit syncing the file after each log entry. This may cause information loss if a crash occurs, but may improve performance.
Named Pipes	A fifo or named pipe can be used as a destination for log messages by putting a pipe symbol () before the filename. You must create the fifo using the <code>mkfifo</code> command before starting (or restarting) <code>syslogd</code> . Fifos are sometimes used for debugging.
Terminal and Console	A terminal such as <code>/dev/console</code> .
Remote Machine	To forward messages to another host, put an at (@) sign before the hostname. Note that messages are not forwarded from the receiving host.
List of Users	A comma-separated list of users to receive a message (if the user is

	logged in). The root user is frequently included here.
Everyone logged on	Specify an asterisk (*) to have everyone logged on notified using the wall command.

You may prefix ! to a priority to indicate that the action should not apply to this level and higher. Similarly you may prefix it with = to indicate that the rule applies only to this level or with != to indicate that the rule applies to all except this level. Listing 22 shows some examples, and the man page for syslog.conf has many more examples.

Listing 22. More syslog.conf examples

```
# Store all kernel messages in /var/log/kernel.
# Send critical and higher ones to remote host pinguino and to the
console
# Finally, Send info, notice and warning messages to
/var/log/kernel-info
#
kern.*                /var/log/kernel
kern.crit             @pinguino
kern.crit             /dev/console
kern.info;kern.!err  /var/log/kernel-info

# Store all mail messages except info priority in /var/log/mail.
mail.*;mail.!=info   /var/log/mail
```

Rotate and archive log files automatically

With the amount of logging that is possible, you need to be able to control the size of log files. This is done using the `logrotate` command, which is usually run as a cron job. Cron jobs are covered later in this tutorial in the section [Scheduling jobs](#). The general idea behind the `logrotate` command is that log files are periodically backed up and a new log is started. Several generations of log are kept, and when a log ages to the last generation, it may be archived. For example, it might be mailed to an archival user.

You use the `/etc/logrotate.conf` configuration file to specify how your log rotating and archiving should happen. You can specify different frequencies, such as daily, weekly, or monthly, for different log files, and you can control the number of generations to keep and when or whether to mail copies to an archival user. Listing 23 shows a sample `/etc/logrotate.conf` file.

Listing 23. Sample /etc/logrotate.conf

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

The logrotate.conf file has global options at the beginning. These are the defaults if nothing more specific is specified elsewhere. In this example, log files are rotated weekly, and four weeks worth of backups are kept. Once a log file is rotated, a new one is automatically created in place of the old one. Note that the logrotate.conf file may include specifications from other files. Here, all the files in /etc/logrotate.d are included.

This example also includes specific rules for /var/log/wtmp and /var/log/btmp, which are rotated monthly. No error message is issued if the files are missing. A new file is created, and only one backup is kept.

In this example, when a backup reaches the last generation, it is deleted because there is no specification of what else to do with it.

Note: The files /var/log/wtmp and /var/log/btmp record successful and unsuccessful login attempts, respectively. Unlike most log files, these are not clear text files. You may examine them using the `last` or `lastb` commands. See the man pages for these commands for details.

Log files may also be backed up when they reach a specific size, and commands may be scripted to run either prior to or after the backup operation. Listing 24 shows a more complex example.

Listing 24. Another logrotate configuration example

```
/var/log/messages {
```

```

rotate 5
mail logsave@pinguino
size 100k
postrotate
    /usr/bin/killall -HUP syslogd
endscript
}

```

In this example, `/var/log/messages` is rotated after it reaches 100KB in size. Five backups are kept, and when the oldest backup ages out, it is mailed to `logsave@pinguino`. The `postrotate` introduces a script that restarts the `syslogd` daemon after the rotation is complete, by sending it the HUP signal. The `endscript` statement is required to terminate the script and is also required if a `prerotate` script is present. See the `logrotate` man page for more complete information.

Scan log files for notable activity

Log files entries are usually time stamped and contain the hostname of the reporting process, along with the process name. Listing 25 shows a few lines from `/var/log/messages`, containing entries from `gconfd`, `ntpd`, `init`, and `yum`.

Listing 25. Sample log file entries

```

Jul  5 15:28:24 lyrebird gconfd (root-2832): Exiting
Jul  5 15:31:06 lyrebird ntpd[2063]: synchronized to 87.98.219.90,
stratum 2
Jul  5 15:31:06 lyrebird ntpd[2063]: kernel time sync status change 0001
Jul  5 15:31:24 lyrebird init: Trying to re-exec init
Jul  5 15:31:24 lyrebird yum: Updated: libselinux.i386 2.0.14-2.fc7
Jul  5 15:31:24 lyrebird yum: Updated: libsemanage.i386 2.0.3-4.fc7
Jul  5 15:31:25 lyrebird yum: Updated: cups-libs.i386 1.2.11-2.fc7
Jul  5 15:31:25 lyrebird yum: Updated: libXfont.i386 1.2.9-2.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager.i386 0.6.5-7.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager-glib.i386
0.6.5-7.fc7

```

You can scan log files using a pager, such as `less`, or search for specific entries (such as kernel messages from host `lyrebird`) using `grep` as shown in Listing 26.

Listing 26. Scanning log files

```

[root@lyrebird ~]# less /var/log/messages
[root@lyrebird ~]# grep "lyrebird kernel" /var/log/messages | tail -n 9
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HCI socket layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP ver 2.8
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM TTY layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM ver 1.8

```

```
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HIDP (Human Interface
Emulation) ver 1.2
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized drm 1.1.0 20060810
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized i915 1.6.0 20060119
on minor 0
```

Monitor log files

Occasionally you may need to monitor log files for events. For example, you might be trying to catch an infrequently occurring event at the time it happens. In such a case, you can use the `tail` command with the `-f` option to *follow* the log file. Listing 27 shows an example.

Listing 27. Following log file updates

```
[root@lyrebird ~]# tail -n 1 -f /var/log/messages
Jul  6 15:16:26 lyrebird syslogd 1.4.2: restart.
Jul  6 15:16:26 lyrebird kernel: klogd 1.4.2, log source = /proc/kmsg
started.
Jul  6 15:19:35 lyrebird yum: Updated: samba-common.i386 3.0.25b-2.fc7
Jul  6 15:19:35 lyrebird yum: Updated: procps.i386 3.2.7-14.fc7
Jul  6 15:19:36 lyrebird yum: Updated: samba-client.i386 3.0.25b-2.fc7
Jul  6 15:19:37 lyrebird yum: Updated: libsmbclient.i386 3.0.25b-2.fc7
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Received signal 15, shutting
down cleanly
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Exiting
Jul  6 15:19:57 lyrebird yum: Updated: bluez-gnome.i386 0.8-1.fc7
```

Track down problems reported in log files

When you find problems in log files, you will want to note the time, the hostname, and the process that generated the problem. If the message identifies the problem specifically enough for you to resolve it, you are done. If not, you might need to update `syslog.conf` to specify that more messages be logged for the appropriate facility. For example, you might need to show informational messages instead of warning messages or even debug level messages. Your application may have additional facilities that you can use.

Finally, if you need to put marks in the log file to help you know what messages were logged at what stage of your debugging activity, you can use the `logger` command from a terminal window or shell script to send a message of your choice to the syslog daemon for logging according to the rules in `syslog.conf`.

Section 5. Scheduling jobs

This section covers material for topic 1.111.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use the `cron` or `anacron` commands to run jobs at regular intervals
- Use the `at` command to run jobs at a specific time
- Manage cron and at jobs
- Configure user access to the cron and at services

In the previous section, you learned about the `logrotate` command and saw the need to run it periodically. You will see the same need to run commands regularly in the next two sections on backup and network time services. These are only some of the many administrative tasks that have to be done frequently and regularly. In this section, you learn about the tools that are used to automate periodic job scheduling and also the tools used to run a job at some specific time.

Run jobs at regular intervals

Running jobs at regular intervals is managed by the `cron` facility, which consists of the `crond` daemon and a set of tables describing what work is to be done and with what frequency. The daemon wakes up every minute and checks the crontabs to determine what needs to be done. Users manage crontabs using the `crontab` command. The `crond` daemon is usually started by the `init` process at system startup.

To keep things simple, let's suppose that you want to run the command shown in Listing 28 on a regular basis. This command doesn't actually do anything except report the day and the time, but it illustrates how to use `crontab` to set up cron jobs, and we'll know when it was run from the output. Setting up crontab entries requires a string with escaped shell metacharacters, so it is best done with simple commands and parameters, so in this example, the `echo` command will be run from within a script `/home/ian/mycrontab.sh`, which takes no parameters. This saves some careful work with escape characters.

Listing 28. A simple command example.

```
[ian@lyrebird ~]$ cat mycrontest.sh
#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
[ian@lyrebird ~]$ ./mycrontest.sh
It is now 18:37:42 on Friday
```

Creating a crontab

To create a crontab, you use the `crontab` command with the `-e` (for "edit") option. This will open the `vi` editor unless you have specified another editor in the `EDITOR` or `VISUAL` environment variable.

Each crontab entry contains six fields:

1. Minute
2. Hour
3. Day of the month
4. Month of the year
5. Day of the week
6. String to be executed by `sh`

Minutes and hours range from 0-59 and 0-12, respectively, while day or month and month of year range from 1-31 and 1-12, respectively. Day of week ranges from 0-6 with 0 being Sunday. Day of week may also be specified as `sun`, `mon`, `tue`, and so on. The sixth field is everything after the fifth field, is interpreted as a string to pass to `sh`. A percent sign (%) will be translated to a newline, so if you want a % or any other special character, precede it with a backslash (\). The line up to the first % is passed to the shell, while any line(s) after the % are passed as standard input.

The various time-related fields can specify an individual value, a range of values, such as 0-10 or `sun-wed`, or a comma-separated list of individual values and ranges. So a somewhat artificial crontab entry for our example command might be as shown in Listing 29.

Listing 29. A simple crontab example.

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
```

In this example, our command is executed at the 0th, 20th, and 40th minutes (every 20 minutes), for the hours between 10 P.M. and midnight on Fridays and Saturdays during July. See the man page for `crontab(5)` for details on additional ways to specify times.

What about the output?

You may be wondering what happens to any output from the command. Most

commands designed for use with the cron facility will log output using the syslog facility that you learned about in the previous section. However any output that is directed to stdout will be mailed to the user. Listing 30 shows the output you might receive from our example command.

Listing 30. Mailed cron output

```
From ian@lyrebird.raleigh.ibm.com Fri Jul 6 23:00:02 2007
Date: Fri, 6 Jul 2007 23:00:01 -0400
From: root@lyrebird.raleigh.ibm.com (Cron Daemon)
To: ian@lyrebird.raleigh.ibm.com
Subject: Cron <ian@lyrebird> /home/ian/mycronetest.sh
Content-Type: text/plain; charset=UTF-8
Auto-Submitted: auto-generated
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/home/ian>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=ian>
X-Cron-Env: <USER=ian>

It is now 23:00:01 on Friday
```

Where is my crontab?

The crontab that you created with the `crontab` command is stored in `/etc/spool/cron` under the name of the user who created it. So the above crontab is stored in `/etc/spool/cron/ian`. Given this, you will not be surprised to learn that the `crontab` command, like the `passwd` command you learned about earlier, is an `suid` program that runs with root authority.

`/etc/crontab`

In addition to the user crontab files in `/var/spool/cron`, `cron` also checks `/etc/crontab` and files in the `/etc/cron.d` directory. These system crontabs have one additional field between the fifth time entry (day) and the command. This additional field specifies the user for whom the command should be run, normally `root`. A `/etc/crontab` might look like the example in Listing 31.

Listing 31. `/etc/crontab`

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

In this example, the real work is done by the `run-parts` command, which runs

scripts from `/etc/cron.hourly`, `/etc/cron.daily`, and so on; `/etc/crontab` simply controls the timing of the recurring jobs. Note that the commands here all run as root. Note also that the crontab can include shell variables assignments that will be set before the commands are run.

Anacron

The cron facility works well for systems that run continuously. For systems that may be turned off much of the time, such as laptops, another facility, the *anacron* (for "anachronistic cron") can handle scheduling of the jobs usually done daily, weekly, or monthly by the cron facility. Anacron does not handle hourly jobs.

Anacron keeps timestamp files in `/var/spool/anacron` to record when jobs are run. When anacron runs, it checks to see if the required number of days has passed since the job was last run and runs it if necessary. The table of jobs for anacron is stored in `/etc/anacrontab`, which has a slightly different format than `/etc/crontab`. As with `/etc/crontab`, `/etc/anacrontab` may contain environment settings. Each job has four fields.

1. period
2. delay
3. job-identifier
4. command

The period is a number of days, but may be specified as `@monthly` to ensure that a job runs only once per month, regardless of the number of days in the month. The delay is the number of minutes to wait after the job is due to run before actually starting it. You can use this to prevent a flood of jobs when a system first starts. The job identifier can contain any non-blank character except slashes (`/`).

Both `/etc/crontab` and `/etc/anacrontab` are updated by direct editing. You do not use the `crontab` command to update these files or files in the `/etc/cron.d` directory.

Run jobs at specific times

Sometimes you may need to run a job just once, rather than regularly. For this you use the `at` command. The commands to be run are read from a file specified with the `-f` option, or from stdin if `-f` is not used. The `-m` option sends mail to the user even if there is no stdout from the command. The `-v` option will display the time at which the job will run before reading the job. The time is also displayed in the output. Listing 32 shows an example of running the `mycrontest.sh` script that you used earlier. Listing 33 shows the output that is mailed back to the user after the job runs.

Notice that it is somewhat more compact than the corresponding output from the cron job.

Listing 32. Using the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh -v 10:25
Sat Jul 7 10:25:00 2007

job 5 at Sat Jul 7 10:25:00 2007
```

Listing 33. Job output from at

```
From ian@lyrebird.raleigh.ibm.com Sat Jul 7 10:25:00 2007
Date: Sat, 7 Jul 2007 10:25:00 -0400
From: Ian Shields <ian@lyrebird.raleigh.ibm.com>
Subject: Output from your job 5
To: ian@lyrebird.raleigh.ibm.com

It is now 10:25:00 on Saturday
```

Time specifications can be quite complex. Listing 34 shows a few examples. See the man page for `at` or the file `/usr/share/doc/at/timespec` or a file such as `/usr/share/doc/at-3.1.10/timespec`, where 3.1.10 in this example is the version of the `at` package.

Listing 34. Time values with the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh 10pm tomorrow
job 14 at Sun Jul 8 22:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 tuesday
job 15 at Tue Jul 10 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 july 11
job 16 at Wed Jul 11 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 next week
job 17 at Sat Jul 14 02:00:00 2007
```

The `at` command also has a `-q` option. Increasing the queue increases the nice value for the job. There is also a `batch` command, which is similar to the `at` command except that jobs are run only when the system load is low enough. See the man pages for more details on these features.

Manage scheduled jobs

Listing scheduled jobs

You can manage your cron and `at` jobs. Use the `crontab` command with the `-l` option to list your crontab, and use the `atq` command to display the jobs you have queued using the `at` command, as shown in Listing 35.

Listing 35. Displaying scheduled jobs

```
[ian@lyrebird ~]$ crontab -l
0,20,40 22-23 * 7 fri-sat /home/ian/mycronstest.sh
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
```

If you want to review the actual command scheduled for execution by `at`, you can use the `at` command with the `-c` option and the job number. You will notice that most of the environment that was active at the time the `at` command was issued is saved with the scheduled job. Listing 36 shows part of the output for job 15.

Listing 36. Using `at -c` with a job number

```
#!/bin/sh
# atrun uid=500 gid=500
# mail ian 0
umask 2
HOSTNAME=lyrebird.raleigh.ibm.com; export HOSTNAME
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=9.67.219.151\ 3210\ 22; export SSH_CLIENT
SSH_TTY=/dev/pts/5; export SSH_TTY
USER=ian; export USER
...
HOME=/home/ian; export HOME
LOGNAME=ian; export LOGNAME
...
cd /home/ian || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
${SHELL:-/bin/sh} << `(dd if=/dev/urandom count=200 bs=1 \
  2>/dev/null|LC_ALL=C tr -d -c '[:alnum:]')`

#!/bin/bash
echo "It is now $(date +%T) on $(date +%A)"
```

Note that the contents of our script file have been copied in as a here-document that will be executed by the shell specified by the `SHELL` variable or `/bin/sh` if the `SHELL` variable is not set. See the tutorial [LPI exam 101 prep, Topic 103: GNU and UNIX commands](#) if you need to review here-documents.

Deleting scheduled jobs

You can delete all scheduled cron jobs using the `crontab` command with the `-r` option as illustrated in Listing 37.

Listing 37. Displaying and deleting cron jobs

```
[ian@lyrebird ~]$ crontab -l
```

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycronetest.sh
[ian@lyrebird ~]$ crontab -r
[ian@lyrebird ~]$ crontab -l
no crontab for ian
```

To delete system cron or anacron jobs, edit `/etc/crontab`, `/etc/anacrontab`, or edit or delete files in the `/etc/cron.d` directory.

You can delete one or more jobs that were scheduled with the `at` command by using the `atrm` command with the job number. Multiple jobs should be separated by spaces. Listing 38 shows an example.

Listing 38. Displaying and removing jobs with `atq` and `atrm`

```
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
[ian@lyrebird ~]$ atrm 16 14 15
[ian@lyrebird ~]$ atq
17      Sat Jul 14 02:00:00 2007 a ian
```

Configure user access to job scheduling

If the file `/etc/cron.allow` exists, any non-root user must be listed in it in order to use `crontab` and the cron facility. If `/etc/cron.allow` does not exist, but `/etc/cron.deny` does exist, a non-root user who is listed in it cannot use `crontab` or the cron facility. If neither of these files exists, only the super user will be allowed to use this command. An empty `/etc/cron.deny` file allows all users to use the cron facility and is the default.

The corresponding `/etc/at.allow` and `/etc/at.deny` files have similar effects for the `at` facility.

Section 6. Data backup

This section covers material for topic 1.111.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Plan a backup strategy

- Dump a raw device to a file or restore a raw device from a file
- Perform partial and manual backups
- Verify the integrity of backup files
- Restore filesystems partially or fully from backups

Plan a backup strategy

Having a good backup is a necessary part of system administration, but deciding what to back up and when and how can be complex. Databases, such as customer orders or inventory, are usually critical to a business and many include specialized backup and recovery tools that are beyond the scope of this tutorial. At the other extreme, some files are temporary in nature and no backup is needed at all. In this section, we focus on system files and user data and discuss some of the considerations, methods, and tools for backup of such data.

There are three general approaches to backup:

1. A *full* backup is a complete backup, usually of a whole filesystem, directory, or group of related files. This takes the longest time to create, so it is usually used with one of the other two approaches.
2. A *differential* or *cumulative* backup is a backup of all things that have changed since the last full backup. Recovery requires the last full backup plus the latest differential backup.
3. An *incremental* backup is a backup of only those changes since the last incremental backup. Recovery requires the last full backup plus all of the incremental backups (in order) since the last full backup.

What to back up

When deciding what to back up, you should consider how volatile the data is. This will help you determine how often it should be backed up. Similarly, critical data should be backed up more often than non-critical data. Your operating system will probably be relatively easy to rebuild, particularly if you use a common image for several systems, although the files that customize each system would be more important to back up.

For programming staff, it may be sufficient to keep backups of repositories such as CVS repositories, while individual programmers' sandboxes may be less important. Depending on how important mail is to your operation, it may suffice to have infrequent mail backups, or it may be necessary to be able to recover mail to the

most recent date possible. You may want to keep backups of system cron files, but may not be so concerned about scheduled jobs for individual users.

The Filesystem Hierarchy Standard provides a classification of data that may help you with your backup choices. For details, see the tutorial [LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard](#).

Once you have decided what to back up, you need to decide how often to do a full backup and whether to do differential or incremental backups in between those full backups. Having made those decisions, the following suggestions will help you choose appropriate tools.

Automating backups

In the previous section, you learned how to schedule jobs, and the cron facility is ideal for helping to automate the scheduling of your backups. However, backups are frequently made to removable media, particularly tape, so operator intervention is probably going to be needed. You should create and use backup scripts to ensure that the backup process is as automatic and repeatable as possible.

Dump and restore raw devices

One way to make a full backup of a filesystem is to make an image of the partition on which it resides. A *raw device*, such as `/dev/hda1` or `/dev/sda2`, can be opened and read as a sequential file. Similarly, it can be written from a backup as a sequential file. This requires no knowledge on the part of the backup tool as to the filesystem layout, but does require that the restore be done to space that is at least as large as the original. Some tools that handle raw devices are *filesystem aware*, meaning that they understand one or more of the Linux filesystems. These utilities can dump from a raw device but do not dump unused parts of the partition. They may or may not require restoration to the same or larger sized partition. The `dd` command is an example of the first type, while the `dump` command is an example of the second type that is specific to the `ext2` and `ext3` filesystems.

The `dd` command

In its simplest form, the `dd` command copies an input file to an output file, where either file may be a raw device. For backing up a raw device, such as `/dev/hda1` or `/dev/sda2`, the input file would be a raw device. Ideally, the filesystem on the device should be unmounted, or at least mounted read only, to ensure that data does not change during the backup. Listing 39 shows an example.

Listing 39. Backup a partition using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 of=backup-1
```

```
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 49.3103 s, 21.2 MB/s
```

The `if` and `of` parameters specify the input and output files respectively. In this example, the input file is a raw device, `dev/sda3`, and the output file is a file, `backup-1`, in the root user's home directory. To dump the file to tape or floppy disk, you would specify something like `of=/dev/fd0` or `of=/dev/st0`.

Note that 1,044,610,560 bytes of data was copied and the output file is indeed that large, even though only about 3% of this particular partition is actually used. Unless you are copying to a tape with hardware compression, you will probably want to compress the data. Listing 40 shows one way to accomplish this, along with the output of `ls` and `df` commands, which show you the file sizes and the usage percentage of the filesystem on `/dev/sda3`.

Listing 40. Backup with compression using `dd`

```
[root@lyrebird ~]# dd if=/dev/sda3 | gzip > backup-2
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 117.723 s, 8.9 MB/s
[root@lyrebird ~]# ls -l backup-[12]
-rw-r--r-- 1 root root 1044610560 2007-07-08 15:17 backup-1
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
[root@lyrebird ~]# df -h /dev/sda3
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       972M   28M  944M   3% /grubfile
```

The `gzip` compression reduced the file size to about 20% of the uncompressed size. However, unused blocks may contain arbitrary data, so even the compressed backup may be much larger than the total data on the partition.

If you divide the size by the number of records processed by `dd`, you will see that `dd` is writing 512-byte blocks of data. When copying to a raw output device such as tape, this can result in a very inefficient operation, so `dd` can read or write data in much larger blocks. Specify the `obs` option to change the output size or the `ibs` option to specify the input block size. You can also specify just `bs` to set both input and output block sizes to a common value.

If you need multiple tapes or other removable storage to store your backup, you will need to break it into smaller pieces using a utility such as `split`.

If you need to skip blocks such as disk or tape labels, you can do so with `dd`. See the man page for examples.

Besides just copying data, the `dd` command can do several conversions, such as between ASCII and EBCDIC, between big-endian and little-endian, or between variable-length data records and fixed-length data records. Obviously these

conversions are likely to be useful when copying real files rather than raw devices. Again, see the man page for details.

The dump command

The `dump` command can be used for full, differential, or incremental backups on ext2 or ext3 filesystems. Listing 41 shows an example.

Listing 41. Backup with compression using dump

```
[root@lyrebird ~]# dump -0 -f backup-4 -j -u /dev/sda3
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-4
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzip)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 12285 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:47:48 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-4
DUMP: Volume 1 completed at: Sun Jul  8 16:47:57 2007
DUMP: Volume 1 took 0:00:09
DUMP: Volume 1 transfer rate: 819 kB/s
DUMP: Volume 1 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: 12260 blocks (11.97MB) on 1 volume(s)
DUMP: finished in 9 seconds, throughput 1362 kBytes/sec
DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Date this dump completed: Sun Jul  8 16:47:57 2007
DUMP: Average transfer rate: 819 kB/s
DUMP: Wrote 12260kB uncompressed, 7377kB compressed, 1.662:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[2-4]
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
-rw-r--r-- 1 root root 266932272 2007-07-08 15:44 backup-3
-rw-r--r-- 1 root root  7554939 2007-07-08 16:47 backup-4
```

In this example, `-0` specifies the *dump level* which is an integer, historically from 0 to 9, where 0 specifies a full dump. The `-f` option specifies the output file, which may be a raw device. Specify `-` to direct the output to stdout. The `-j` option specifies compression, with a default level of 2, using bzip compression. You can use the `-z` option to specify zlib compression if you prefer. The `-u` option causes the record of dump information, normally `/etc/dumpdates`, to be updated. Any parameters after the options represent a file or list of files, where the file may also be a raw device, as in this example. Notice how much smaller the backup is when the backup program is aware of the filesystem structure and can avoid the saving of unused blocks on the device.

If output is to a device such as tape, the `dump` command will prompt for another volume as each volume is filled. You can also provide multiple file names separated by commas. For example, if you wanted an unattended dump that required two tapes, you could load the tapes on `/dev/st0` and `/dev/st1`, schedule the `dump` command specifying both tapes as output, and go home to sleep.

When you specify a dump level greater than 0, an incremental dump is performed of all files that are new or have changed since the last dump at a lower level was taken. So a dump at level 1 will be a differential dump, even if a dump at level 2 or higher has been taken in the meantime. Listing 42 shows the result of updating the time stamp of an existing file on /dev/sda3 and creating a new file, then taking a dump at level 2. After that, another new file is created and a dump at level 1 is taken. The information from /etc/dumpdates is also shown. For brevity, part of the second dump output has been omitted.

Listing 42. Backup with compression using dump

```
[root@lyrebird ~]# dump -2 -f backup-5 -j -u /dev/sda3
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-5
DUMP: Label: GRUB
DUMP: Writing 10 Kilobyte records
DUMP: Compressing output at compression level 2 (bzlib)
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 91 blocks.
DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:55:47 2007
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing backup-5
DUMP: Volume 1 completed at: Sun Jul  8 16:55:47 2007
DUMP: 90 blocks (0.09MB) on 1 volume(s)
DUMP: finished in less than a second
DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
DUMP: Date this dump completed: Sun Jul  8 16:55:47 2007
DUMP: Average transfer rate: 0 kB/s
DUMP: Wrote 90kB uncompressed, 15kB compressed, 6.000:1
DUMP: DUMP IS DONE
[root@lyrebird ~]# echo "This data is even newer" >/grubfile/newerfile
[root@lyrebird ~]# dump -1 -f backup-6 -j -u -A backup-6-toc /dev/sda3
DUMP: Date of this level 1 dump: Sun Jul  8 17:08:18 2007
DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
DUMP: Dumping /dev/sda3 (/grubfile) to backup-6
...
DUMP: Wrote 100kB uncompressed, 16kB compressed, 6.250:1
DUMP: Archiving dump to backup-6-toc
DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[4-6]
-rw-r--r-- 1 root root 7554939 2007-07-08 16:47 backup-4
-rw-r--r-- 1 root root  16198 2007-07-08 16:55 backup-5
-rw-r--r-- 1 root root  16560 2007-07-08 17:08 backup-6
[root@lyrebird ~]# cat /etc/dumpdates
/dev/sda3 0 Sun Jul  8 16:47:47 2007 -0400
/dev/sda3 2 Sun Jul  8 16:55:46 2007 -0400
/dev/sda3 1 Sun Jul  8 17:08:18 2007 -0400
```

Notice that backup-6 is, indeed, larger than backup 5. The level 1 dump illustrates the use of the `-A` option to create a table of contents that can be used to determine if a file is on an archive without actually mounting the archive. This is particularly useful with tape or other removable archive volumes. You will see these examples again when we discuss restoring data later in this section.

The `dump` command can dump files or subdirectories, but you cannot update

`/etc/dumpdates` and only level 0, of full dump, is supported.

Listing 43 illustrates the `dump` command dumping a directory, `/usr/include/bits`, and its contents to floppy disk. In this case, the dump will not fit on a single floppy, so a new volume is required. The prompt and response are shown in bold.

Listing 43. Backup a directory to multiple volumes using `dump`

```
[root@lyrebird ~]# dump -0 -f /dev/fd0 /usr/include/bits
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Dumping /dev/sdb9 (/ (dir usr/include/bits)) to /dev/fd0
DUMP: Label: /
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 2790 blocks.
DUMP: Volume 1 started with block 1 at: Mon Jul  9 16:03:30 2007
DUMP: dumping (Pass III) [directories]
DUMP: End of tape detected
DUMP: Closing /dev/fd0
DUMP: Volume 1 completed at: Mon Jul  9 16:04:49 2007
DUMP: Volume 1 1470 blocks (1.44MB)
DUMP: Volume 1 took 0:01:19
DUMP: Volume 1 transfer rate: 18 kB/s
DUMP: Change Volumes: Mount volume #2
DUMP: Is the new volume mounted and ready to go?: ("yes" or "no") y
DUMP: Volume 2 started with block 1441 at: Mon Jul  9 16:05:10 2007
DUMP: Volume 2 begins with blocks from inode 2
DUMP: dumping (Pass IV) [regular files]
DUMP: Closing /dev/fd0
DUMP: Volume 2 completed at: Mon Jul  9 16:06:28 2007
DUMP: Volume 2 1410 blocks (1.38MB)
DUMP: Volume 2 took 0:01:18
DUMP: Volume 2 transfer rate: 18 kB/s
DUMP: 2850 blocks (2.78MB) on 2 volume(s)
DUMP: finished in 109 seconds, throughput 26 kBytes/sec
DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
DUMP: Date this dump completed: Mon Jul  9 16:06:28 2007
DUMP: Average transfer rate: 18 kB/s
DUMP: DUMP IS DONE
```

If you back up to tape, remember that the tape will usually be rewound after each job. Devices with a name like `/dev/st0` or `/dev/st1` automatically rewind. The corresponding non-rewind equivalent devices are `/dev/nst0` and `/dev/nst1`. In any event, you can always use the `mt` command to perform magnetic tape operations such as forward spacing over files and records, back spacing, rewinding, and writing EOF marks. See the man pages for `mt` and `st` for additional information.

If you select the `dump` levels judiciously, you can minimize the number of archives you need to restore to any particular level. See the man pages for `dump` for a suggestion based on the Towers of Hanoi puzzle.

As with the `dd` command, there are many options that are not covered in this brief introduction. See the man pages for more details.

Partial and manual backups

So far, you have learned about tools that work well for backing up whole filesystems. Sometimes your backup needs to target selected files or subdirectories without backing up the whole filesystem. For example, you might need a weekly backup of most of your system, but daily backups of your mail files. Two other programs, `cpio` and `tar`, are more commonly used for this purpose. Both can write archives to files or to devices such as tape or floppy disk, and both can restore from such archives. Of the two, `tar` is more commonly used today, possibly because it handles complete directories better, and GNU `tar` supports both `gzip` and `bzip` compression.

Using `cpio`

The `cpio` command operates in *copy-out* mode to create an archive, *copy-in* mode to restore an archive, or *copy-pass* mode to copy a set of files from one location to another. You use the `-o` or `--create` option for copy-out mode, the `-i` or `--extract` option for copy-in mode, and the `-p` or `--pass-through` option for copy-pass mode. Input is a list of files provided on `stdin`. Output is either to `stdout` or to a device or file specified with the `-f` or `--file` option.

Listing 44 shows how to generate a list of files using the `find` command. Note the use of the `-print0` option on `find` to generate null-terminate strings for file names, and the corresponding `--null` option on `cpio` to read this format. This will correctly handle file names that have embedded blank or newline characters.

Listing 44. Back up a home directory using `cpio`

```
[root@lyrebird ~]# find ~ian -depth -print0 | cpio --null -o
>backup-cpio-1
18855 blocks
```

If you'd like to see the files listed as they are archived, add the `-v` option to `cpio`.

As with other commands that can archive to tape, the block size may be specified. For details on this and other options, see the man page.

Using `tar`

The `tar` (originally from *Tape ARchive*) creates an archive file, or *tarfile* or *tarball*, from a set of input files or directories; it also restores files from such an archive. If a directory is given as input to `tar`, all files and subdirectories are automatically included, which makes `tar` very convenient for archiving subtrees of your directory structure.

As with the other archiving commands we have discussed, output can be to a file, a

device such as tape or diskette, or stdout. The output location is specified with the `-f` option. Other common options are `-c` to create an archive, `-x` to extract an archive, `-v` for verbose output, which lists the files being processed, `-z` to use gzip compression, and `-j` to use bzip2 compression. Most `tar` options have a short form using a single hyphen and a long form using a pair of hyphens. The short forms are illustrated here. See the man pages for the long form and for additional options.

Listing 45 shows how to create a backup of the system cron jobs using `tar`.

Listing 45. Backup of system cron jobs using tar

```
[root@lyrebird ~]# tar -czvf backup-tar-1 /etc/*crontab /etc/cron.d
tar: Removing leading `/' from member names
/etc/anacrontab
/etc/crontab
/etc/cron.d/
/etc/cron.d/sa-update
/etc/cron.d/smolt
```

In the first line of output, you are told that `tar` will remove the leading slash (/) from member names. This allows files to be restored to some other location for verification before replacing system files. It is a good idea to avoid mixing absolute path names with relative path names when creating an archive, since all will be relative when restoring from the archive.

The `tar` command can append additional files to an archive using the `-r` or `--append` option. This may cause multiple copies of a file in the archive. In such a case, the *last* one will be restored during a restore operation. You can use the `--occurrence` option to select a specific file among multiples. If the archive is on a regular filesystem instead of tape, you may use the `-u` or `--update` option to update an archive. This works like appending to an archive, except that the time stamps of the files in the archive are compared with those on the filesystem, and only files that have been modified since the archived version are appended. As mentioned, this does not work for tape archives.

As with the other commands you have studied here, there are many options that are not covered in this brief introduction. See the man or info pages for more details.

Backup file integrity

Backup file integrity is extremely important. There is no point in having a backup if it is bad. A good backup strategy also involves checking your backups.

The first step to ensuring backup integrity is to ensure that you have properly captured the data you are backing up. If the filesystem is unmounted or mounted read only, this is usually straightforward as the data you are backing up cannot

change during your backup. If you must back up filesystems, directories, or files that are subject to modification while you are taking the backup, you should verify that no changes have been made during your backup. If changes were made, you will need to have a strategy for capturing them, either by repeating the backup, or perhaps by replacing or superseding the affected files in your backup. Needless to say, this will also affect your restore procedures.

Assuming you took good backups, you will periodically need to verify your backups. One way is to restore the backup to a spare volume and verify that it matches what you backed up. This is easiest to do right before you allow updates on the filesystem you are backing up. If you back up to media such as CD or DVD, you may be able to use the `diff` command as part of your backup procedure to ensure that your backup is good. Remember that even good backups can deteriorate in storage, so you should check periodically, even if you do verify at the time of backup. Keeping digests using programs such as `md5sum` or `sha1sum` is also a good check on the integrity of a backup file.

Restore filesystems from backups

A counterpart to backing up files is the ability to restore them when needed. Occasionally you will want to restore an entire filesystem, but it is far more common to need to restore only specific files or perhaps a set of directories. Almost always you will restore to some temporary space and verify that what you have restored is indeed what you want and is consistent with the current state of your system before actually making the restored files live.

A related issue is the need to verify that the items you want happen to be on a particular backup, as often happens when a user needs access to a version of a file that was modified or perhaps deleted "sometime in the last week or two." With these thoughts in mind, let's look at some of the restoration options.

Restoring a dd archive

Recall that the `dd` command was not filesystem aware, so you will need to restore a dump of a partition to find out what is on it. Listing 46 shows how to restore the partition that was dumped back in Listing 39 to a partition, `/dev/sdc7`, that was specially created on a removable USB drive just for this purpose.

Listing 46. Restoring a partition using dd

```
[root@lyrebird ~]# dd if=backup-1 of=/dev/sdc7
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 44.0084 s, 23.7 MB/s
```

Recall that we added some files to the filesystem on `/dev/sda3` after this backup was taken. If you mount the newly restore partition and compare it with the original, you will see that this is indeed the case, as shown in Listing 47. Note that the file whose timestamp was updated using `touch` is not shown here, as you would expect.

Listing 47. Comparing the restored partition with current state

```
[root@lyrebird ~]# mount /dev/sdc7 /mnt/temp-dd/
[root@lyrebird ~]# diff -rq /grubfile/ /mnt/temp-dd/
Only in /grubfile/: newerfile
Only in /grubfile/: newfile
```

Restoring a dump archive using restore

Recall that our final use of `dump` was a differential backup and that we created a table of contents. Listing 48 shows how to use `restore` to check the files in the archive created by `dump`, using the archive itself (`backup-5`) or the table of contents (`backup-6-toc`).

Listing 48. Checking the contents of archives

```
[root@lyrebird ~]# restore -t -f backup-5
Dump tape is compressed.
Dump   date: Sun Jul  8 16:55:46 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 2 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
[root@lyrebird ~]# restore -t -A backup-6-toc
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
Starting inode numbers by volume:
  Volume 1: 2
      2      .
100481     ./ibshome
100482     ./ibshome/index.html
      16     ./newfile
      17     ./newerfile
```

The `restore` command can also compare the contents of an archive with the contents of the filesystem using the `-C` option. In Listing 49 we updated `newerfile` and then compared the backup with the filesystem.

Listing 49. Comparing an archive with a filesystem using restore

```
[root@lyrebird ~]# echo "something different" >/grubfile/newerfile
[root@lyrebird ~]# restore -C -f backup-6
Dump tape is compressed.
```

```
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
fileSYS = /grubfile
./newerfile: size has changed.
Some files were modified!  1 compare errors
```

The `restore` command can restore interactively or automatically. Listing 50 shows how to restore `newerfile` to root's home directory (so you could examine it before replacing the updated file if needed), then replace the updated file with the backup copy. This example illustrates interactive restoration.

Listing 50. Restoring a file using restore

```
[root@lyrebird ~]# restore -i -f backup-6
Dump tape is compressed.
restore > ?
Available commands are:
  ls [arg] - list directory
  cd arg - change directory
  pwd - print current directory
  add [arg] - add `arg' to list of files to be extracted
  delete [arg] - delete `arg' from list of files to be extracted
  extract - extract requested files
  setmodes - set modes of requested directories
  quit - immediately exit program
  what - list dump header information
  verbose - toggle verbose flag (useful with ``ls'')
  prompt - toggle the prompt display
  help or `?' - print this list
If no `arg' is supplied, the current directory is used
restore > ls new*
newerfile
newfile
restore > add newerfile
restore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume # (none if no more volumes): 1
set owner/mode for '.'? [yn] y
restore > q
[root@lyrebird ~]# mv -f newerfile /grubfile
```

Restoring a cpio archive

The `cpio` command in copy-in mode (option `-i` or `--extract`) can list the contents of an archive or restore selected files. When you list the files, specifying the `--absolute-filenames` option reduces the number of extraneous messages that `cpio` will otherwise issue as it strips any leading `/` characters from each path that has one. Partial output from listing our previous archive is shown in Listing 51.

Listing 51. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio -id --list --absolute-filenames <backup-cpio-1
```

```
/home/ian/.gstreamer-0.10/registry.i686.xml
/home/ian/.gstreamer-0.10
/home/ian/.Trash/gnome-terminal.desktop
/home/ian/.Trash
/home/ian/.bash_profile
```

Listing 52 shows how to restore all the files with "samp" in their path name or file name. The output has been piped through `uniq` to reduce the number of "Removing leading '/' ..." messages. You must specify the `-d` option to create directories; otherwise, all files are created in the current directory. Furthermore, `cpio` will not replace any newer files on the filesystem with archive copies unless you specify the `-u` or `--unconditional` option.

Listing 52. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio -ivd "*samp*" < backup-cpio-1 2>&1 |uniq
cpio: Removing leading `/' from member names
home/ian/crontab.samp
cpio: Removing leading `/' from member names
home/ian/sample.file
cpio: Removing leading `/' from member names
18855 blocks
```

Restoring a tar archive

The `tar` command can also compare archives with the current filesystem as well as restore files from archives. Use the `-d`, `--compare`, or `--diff` option to perform comparisons. The output will show files whose contents differ as well as files whose time stamps differ. Listing 53 shows verbose output (using option `-v`), from a comparison of the file created earlier and the files in `/etc` after `/etc/crontab` has been touched to alter its time stamp. The option `directory /` instructs `tar` to perform the comparison starting from the root directory rather than the current directory.

Listing 53. Comparing archives and files using tar

```
[root@lyrebird ~]# touch /etc/crontab
[root@lyrebird ~]# tar --diff -vf backup-tar-1 --directory /
etc/anacrontab
etc/crontab
etc/crontab: Mod time differs
etc/cron.d/
etc/cron.d/sa-update
etc/cron.d/smolt
```

Listing 54 shows how to extract just `/etc/crontab` and `/etc/anacrontab` into the current directory.

Listing 54. Extracting archive files using tar

```
[root@lyrebird ~]# tar -xzvf backup-tar-1 "*tab"
```

```
etc/anacrontab  
etc/crontab
```

Note that `tar`, in contrast to `cpio` creates the directory hierarchy for you automatically.

The next section of this tutorial shows you how to maintain system time.

Section 7. System time

This section covers material for topic 1.111.6 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

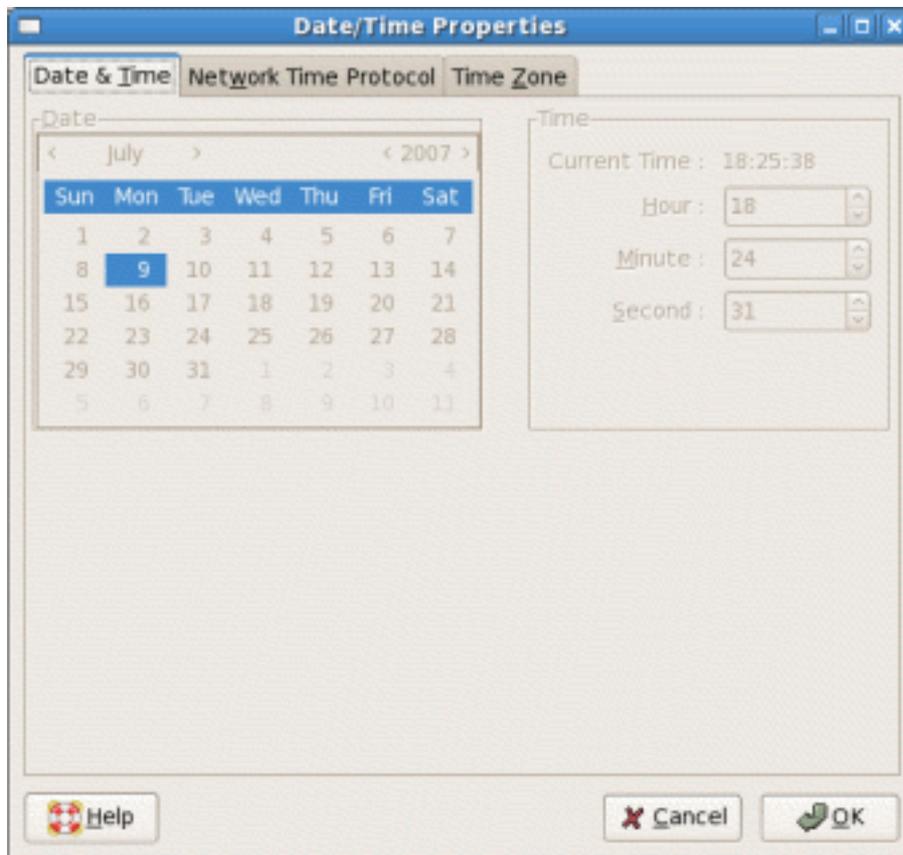
- Set the system date and time
- Set the BIOS clock to the correct UTC time
- Configure your time zone
- Configure the Network Time Protocol (NTP) service, including correcting for clock drift

Set the system date and time

System time on a Linux system is very important. You saw earlier how the cron and anacron facilities do things based on time, so they need an accurate time to base decisions on. Most of the backup and restore tools discussed in the previous section, along with development tools such as `make`, also depend on reliable time measurements. Most computers built since around 1980 include some kind of clock mechanism, and most built since 1984 or so have a persistent clock mechanism that keeps time even if the computer is turned off.

If you installed a Linux system graphically, you probably set the clock and chose a time zone suitable for your needs. You may have elected to use the Network Time Protocol (NTP) to set your clock, and you may or may not have elected to keep the system clock using Coordinated Universal Time or UTC. If you subsequently went to set the clock using graphical tools on a Fedora or Red Hat or similar system, you may have seen a dialog box like that in Figure 3.

Figure 3. Updating the date and time



Surprise! You can't actually set the clock yourself using this dialog. In this section you learn more about the difference between local clocks and NTP and how to set your system time.

No matter whether you live in New York, Budapest, Nakhodka, Ulan Bator, Bangkok, or Canberra, most of your Linux time computations are related to Coordinated Universal Time or UTC. If you run a dedicated Linux system, it is customary to keep the hardware clock set to UTC, but if you also boot another operating system such as Windows, you may need to set the hardware clock to local time. It really doesn't matter as far as Linux is concerned, except that there happen to be two different methods of keeping track of time zones internally in Linux, and if they don't agree, you can wind up with some odd time stamps on FAT filesystems, among other things. Listing 55 shows you how to use the `date` command to display the current date and time. The display is always in local time, even if your hardware clock keeps UTC time.

Listing 55. Displaying the current date and time

```
[root@lyrebird ~]# date;date -u
Mon Jul  9 22:40:01 EDT 2007
```

The `date` command supports a wide variety of possible output formats, some of which you already saw back in [Listing 28](#). See the man page for `date` if you'd like to learn more about the various date formats.

If you need to set the date, you can do this by providing a date and time as an argument. The required format is historical and is somewhat odd even to Americans and truly odd to the rest of the world. You must specify at least month, day, hour, and minute in MMDDhhmm format, and you may also append a two- or four-digit year (CCYY or YY) and optionally a period (.) followed by a two-digit number of seconds. Listing 56 shows an example that alters the system date by a little over a minute.

Listing 56. Setting the system date and time

```
[root@lyrebird ~]# date; date 0709221407;date
Mon Jul  9 23:12:37 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
```

Set the BIOS clock to UTC time

Your Linux system, along with most other current operating systems, actually has two clocks. The first is the hardware clock, sometimes called the Real Time Clock, RTC, or BIOS clock, which is usually tied to an oscillating quartz crystal that is accurate to within a few seconds per day. It is subject to variations such as ambient temperature. The second is the internal software clock, which is driven by counting system interrupts. It is subject to variations caused by high system load and interrupt latency. Nevertheless, your system typically reads the hardware clock at startup and from then on uses the software clock. The `date` command that you just learned about sets the software clock, not the hardware clock.

If you use the Network Time Protocol (NTP), you may possibly set the hardware clock when you first install the system and never worry about it again. If not, this part of the tutorial will show you how to display and set the hardware clock time.

You can use the `hwclock` command to display the current value of the hardware clock. Listing 57 shows the current value of both the system and hardware clocks.

Listing 57. System and hardware clock values

```
[root@lyrebird ~]# date;hwclock
Mon Jul  9 22:16:11 EDT 2007
Mon 09 Jul 2007 11:14:49 PM EDT -0.071616 seconds
```

Notice that the two values are different. You can synchronize the hardware clock

from the system clock using the `-w` or `--systohc` option of `hwclock`, and you can synchronize the system clock from the hardware clock using the `-s` or `--hctosys` option, as shown in Listing 58.

Listing 58. Setting the system clock from the hardware clock

```
[root@lyrebird ~]# date;hwclock;hwclock -s;date
Mon Jul  9 22:20:23 EDT 2007
Mon 09 Jul 2007 11:19:01 PM EDT  -0.414881 seconds
Mon Jul  9 23:19:02 EDT 2007
```

You may specify either the `--utc` or the `--localtime` option to have the system clock kept in UTC or local time. If no value is specified, the value is taken from the third line of `/etc/adjtime`.

The Linux kernel has a mode that copies the system time to the hardware clock every 11 minutes. This is off by default, but is turned on by NTP. Running anything that set the time the old fashioned way, such as `hwclock --hctosys`, turns it off, so it's a good idea to just let NTP do its work if you are using NTP. See the man page for `adjtimex` to find out how to check whether the clock is being updated every 11 minutes or not. You may need to install the `adjtimex` package as it is not always installed by default.

The `hwclock` command keeps track of changes made to the hardware clock in order to compensate for inaccuracies in the clock frequency. The necessary data points are kept in `/etc/adjtime`, which is an ASCII file. If you are not using the Network Time Protocol, you can use the `adjtimex` command to compensate for clock drift. Otherwise, the hardware clock will be adjusted approximately every 11 minutes by NTP. Besides showing whether your hardware clock is in local or UTC time, the first value in `/etc/adjtime` shows the amount of hardware clock drift per day (in seconds). Listing 59 shows two examples.

Listing 59. /etc/adjtime showing clock drift and local or UTC time.

```
[root@lyrebird ~]# cat /etc/adjtime
0.000990 1184019960 0.000000
1184019960
LOCAL
root@pinguino:~# cat /etc/adjtime
-0.003247 1182889954 0.000000
1182889954
LOCAL
```

Note that both these systems keep the hardware clock in local time, but the clock drifts are different — 0.000990 on lyrebird and -0.003247 on pinguino.

Configure your time zone

Your time zone is a measure of how far your local time differs from UTC. Information on available time zones that can be configured is kept in `/usr/share/zoneinfo`. Traditionally, `/etc/localtime` was a link to one of the time zone files in this directory tree, for example, `/usr/share/zoneinfo/Eire` or `/usr/share/zoneinfo/Australia/Hobart`. On modern systems it is much more likely to be a copy of the appropriate time zone data file since the `/usr/share` filesystem may not be mounted when the local time zone information is needed early in the boot process.

Similarly, another file, `/etc/timezone` was traditionally a link to `/etc/default/init` and was used to set the time zone environment variable `TZ`, and several locale-related environment variables. The file may or may not exist on your system. If it does, it may simply contain the name of the current time zone. You may also find time zone information in `/etc/sysconfig/clock`. Listing 60 shows these files from a Ubuntu 7.04 and a Fedora 7 system.

Listing 60. Time zone information in `/etc`

```
root@pinguino:~# cat /etc/timezone
America/New_York

[root@lyrebird ~]# cat /etc/sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
# The timezone of the system is defined by the contents of
/etc/localtime.
ZONE="America/New York"
UTC=false
ARC=false
```

Some systems such as Debian and Ubuntu have a `tzconfig` command to set the time zone. Others such as Fedora use `system-config-date` to set the time zone and to indicate whether the clock uses UTC or not. Listing 61 illustrates the use of the `tzconfig` command to display the current time zone.

Listing 61. Setting time zone with `tzconfig`

```
root@pinguino:~# tzconfig
Your current time zone is set to America/New_York
Do you want to change that? [n]:
Your time zone will not be changed
```

Configure the Network Time Protocol

The *Network Time Protocol (NTP)* is a protocol to synchronize computer clocks over a network. Synchronization is usually to UTC.

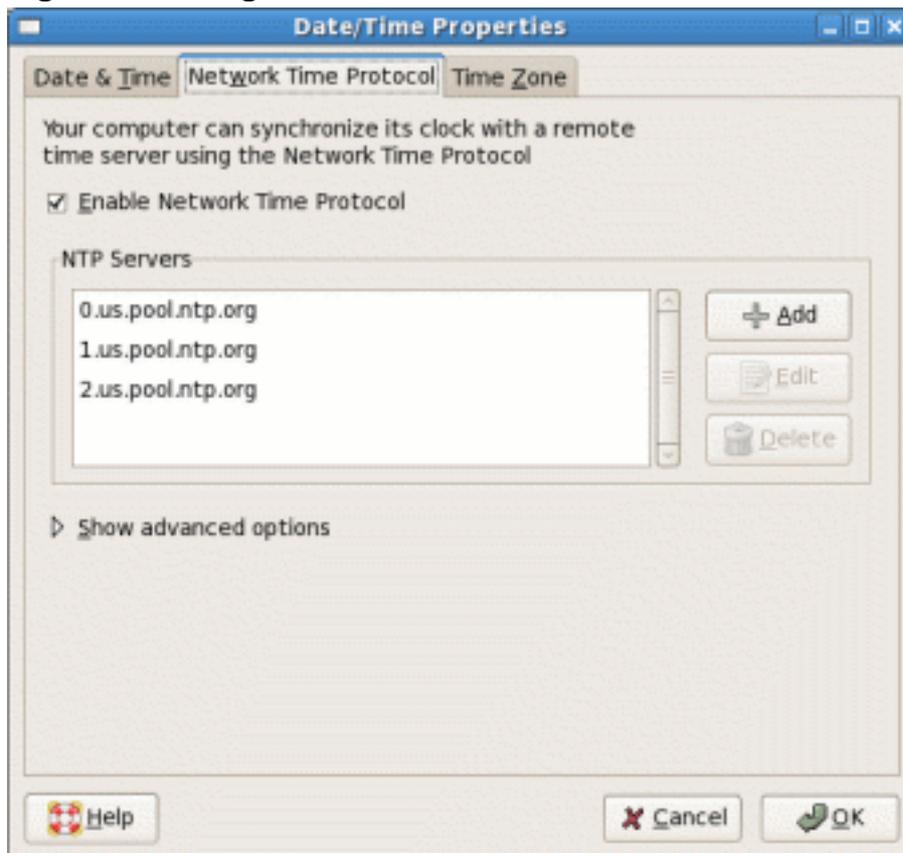
NTP version 3 is an Internet draft standard, formalized in RFC 1305. The current development version, NTP version 4 is a significant revision, which has not been formalized in an RFC. RFC 4330 describes Simple NTP (SNTP) version 4.

Time synchronization is accomplished by sending messages to *time servers*. The time returned is adjusted by an offset of half the round-trip delay. The accuracy of the time is therefore dependent on the network latency and the extent to which the latency is the same in both directions. The shorter the path to a time server, the more accurate the time is likely to be. See [Resources](#) for more detailed information than this simplistic description can provide.

There is a huge number of computers on the Internet, so time servers are organized into *strata*. A relatively small number of stratum 1 servers maintain very accurate time from a source such as an atomic clock. A larger number of stratum 2 servers get their time from stratum 1 servers and make it available to an even larger number of stratum 3 servers, and so on. To ease the load on time servers, a large number of volunteers donate time services through pool.ntp.org (see [Resources](#) for a link). Round robin DNS servers accomplish NTP load balancing by distributing NTP server requests among a pool of available servers.

If you use a graphical interface, you might be able to set your NTP time servers using a dialog similar to that in Figure 4. The fact that this system has enabled automatic time updates using NTP is why the dialog in Figure 3 did not allow the date and time to be changed.

Figure 4. Setting NTP servers



NTP configuration information is kept in `/etc/ntp.conf`, so you can also edit that file and then restart the `ntpd` daemon after you save it. Listing 62 shows an example `/etc/ntp.conf` file using the time servers from Figure 4.

Listing 62. Setting time zone with `tzconfig`

```
[root@lyrebird ~]# cat /etc/ntp.conf
# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#broadcast 192.168.1.255 key 42          # broadcast server
#broadcastclient          # broadcast client
#broadcast 224.0.1.1 key 42            # multicast server
#multicastclient 224.0.1.1            # multicast client
#manycastserver 239.255.254.254        # manycast server
#manycastclient 239.255.254.254 key 42 # manycast client

# Undisciplined Local Clock. This is a fake driver intended for backup
# and when no outside source of synchronized time is available.
#server 127.127.1.0 # local clock
#fudge 127.127.1.0 stratum 10

# Drift file. Put this in a directory which the daemon can write to.
# No symbolic links allowed, either, since the daemon updates the file
# by creating a temporary in the same directory and then rename()'ing
# it to the file.
driftfile /var/lib/ntp/drift

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8
server 0.us.pool.ntp.org
restrict 0.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 1.us.pool.ntp.org
restrict 1.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 2.us.pool.ntp.org
restrict 2.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
```

If you are using the `pool.ntp.org` time servers, these may be anywhere in the world. You will usually get better time by restricting your servers as in this example where `us.pool.ntp.org` is used, resulting in only U.S. servers being chosen. See [Resources](#)

for more information on the ntp.pool.org project.

NTP commands

You can use the `ntpdate` command to set your system time from an NTP time server as shown in Listing 63.

Listing 63. Setting system time from an NTP server using `ntpdate`

```
[root@lyrebird ~]# ntpdate 0.us.pool.ntp.org
10 Jul 10:27:39 ntpdate[15308]: adjust time server 66.199.242.154 offset
-0.007271 sec
```

Because the servers operate in round robin mode, the next time you run this command you will probably see a different server. Listing 64 shows the first few DNS responses for `0.us.ntp.pool.org` a few moments after the above `ntpdate` command was run.

Listing 64. Round robin NTP server pool

```
[root@lyrebird ~]# dig 0.pool.ntp.org +noall +answer | head -n 5
0.pool.ntp.org. 1062 IN A 217.116.227.3
0.pool.ntp.org. 1062 IN A 24.215.0.24
0.pool.ntp.org. 1062 IN A 62.66.254.154
0.pool.ntp.org. 1062 IN A 76.168.30.201
0.pool.ntp.org. 1062 IN A 81.169.139.140
```

The `ntpdate` command is now deprecated as the same function can be done using `ntpq` with the `-q` option, as shown in Listing 65.

Listing 65. Setting system time using `ntpd -q`

```
[root@lyrebird ~]# ntpd -q
ntpd: time slew -0.014406s
```

Note that the `ntpd` command uses the time server information from `/etc/ntp.conf`, or a configuration file provided on the command line. See the man page for more information and for information about other options for `ntpd`. Be aware also that if the `ntpd` daemon is running, `ntpd -q` will quietly exit, leaving a failure message in `/var/log/messages`.

Another related command is the `ntpq` command, which allows you to query the NTP daemon. See the man page for more details.

This brings us to the end of this tutorial. We have covered a lot of material on system administration. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- See the [Partimage homepage](#) for information on Partimage, a filesystem-aware partition dump and restore tool.
- "[/etc: Host-specific system configuration](#)" describes the Linux Standard Base (LSB) requirements for /etc.
- The [Network Time Protocol Project](#) produces a reference implementation of the NTP protocol, and implementation documentation.
- The [Network Time Synchronization Project](#) maintains an extensive array of documentation and background information, including briefing slides, on network time protocols.
- The [pool.ntp.org project](#) is a big virtual cluster of timeservers striving to provide reliable easy to use NTP service for millions of clients without putting a strain on the big popular timeservers.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content.](#)
- Get involved in the [developerWorks community](#) through our developer blogs, forums, podcasts, and community topics in our new [developerWorks spaces](#).

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 201 prep: Linux kernel

Intermediate Level Administration (LPIC-2) topic 201

Skill Level: Intermediate

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer
Gnosis Software

29 Aug 2005

Updated 20 Sep 2005

In this tutorial, David Mertz begins preparing you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 201. In this first of eight tutorials, you learn to understand, compile, and customize a Linux™ kernel.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams.

Each exam covers several topics, and each topic has a weight. The weights indicate the relative importance of each topic. Very roughly, expect more questions on the exam for topics with higher weight. The topics and their weights for LPI exam 201 are:

Topic 201

Linux kernel (weight 5). The focus of this tutorial.

Topic 202

System startup (weight 5).

Topic 203

Filesystem (weight 10).

Topic 204

Hardware (weight 8).

Topic 209

File and service sharing (weight 8).

Topic 211

System maintenance (weight 4).

Topic 213

System customization and automation (weight 3).

Topic 214

Troubleshooting (weight 6).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Linux kernel," the first of eight tutorials designed to prepare you for LPI exam 201. In this tutorial, you will learn how to compile and customize a Linux kernel.

The tutorial is organized according to the LPI objectives for this topic, as follows:

2.201.1 Kernel components (weight 1)

You will learn how to use kernel components that are necessary to specific hardware, hardware drivers, system resources, and requirements. You will learn about implementing different types of kernel images, identifying stable and development kernels and patches, as well as using kernel modules.

2.201.2 Compiling a kernel (weight 1)

You will learn how to properly compile a kernel to include or disable specific features of the Linux kernel as necessary. You will learn about compiling and recompiling the Linux kernel as needed, implementing updates and noting changes in a new kernel, creating a system `initrd` image, and installing new

kernels.

2.201.3 Patching a kernel (weight 2)

You will learn how to properly patch a kernel for various purposes including how to implement kernel updates, implement bug fixes, and add support for new hardware. You will also learn how to properly remove kernel patches from existing production kernels.

2.201.4 Customizing a kernel (weight 1)

You will learn how to customize a kernel for specific system requirements by patching, compiling, and editing configuration files as required. You will learn how to assess requirements for a kernel compile versus a kernel patch as well as build and configure kernel modules.

This tutorial is one of the few in this series that is about Linux itself, strictly speaking. That is, a variety of tools for networking, system maintenance, manipulating files and data, and so on, are important for a working Linux installation and are part of almost every Linux distribution. But the base kernel -- the bit of software that mediates between contending programs and access to hardware -- is the software managed by Linus Torvalds, and that is properly called "Linux itself."

One of the best things about the Linux kernel is that it is Free Software. Not only have many brilliant people contributed to making the Linux kernel better, but you, as system administrator, have access to the kernel source code. This gives you the power to configure and customize the kernel to fit your exact requirements.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. Kernel components

This section covers material for topic 2.201.1 for the Intermediate Level Administration (LPIC-2) exam 201. The topic has a weight of 1.

What makes up a kernel?

A Linux kernel is made up of the base kernel itself plus any number of kernel

modules. In many or most cases, the base kernel and a large collection of kernel modules are compiled at the same time and installed or distributed together, based on the code created by Linus Torvalds or customized by Linux distributors. A base kernel is always loaded during system boot and stays loaded during all uptime; kernel modules may or may not be loaded initially (though generally some are), and kernel modules may be loaded or unloaded during runtime.

The kernel module system allows the inclusion of extra modules that are compiled after, or separately from, the base kernel. Extra modules may be created either when you add hardware devices to a running Linux system or are sometimes distributed by third parties. Third parties sometime distribute kernel modules in binary form, though doing so takes away your capability as a system administrator to customize a kernel module. In any case, once a kernel module is loaded, it becomes part of the running kernel for as long as it remains loaded. Contrary to some conceptions, a kernel module is not simply an API for talking with a base kernel, but becomes patched in as part of the running kernel itself.

Kernel naming conventions

Linux kernels follow a naming/numbering convention that quickly tells you significant information about the kernel you are running. The convention used indicates a major number, minor number, revision, and, in some cases, vendor/customization string. This same convention applies to several types of files, including the kernel source archive, patches, and perhaps multiple base kernels (if you run several).

As well as the basic dot-separated sequence, Linux kernels follow a convention to distinguish stable from experimental branches. Stable branches use an even minor number, whereas experimental branches use an odd minor number. Revisions are simply sequential numbers that represent bug fixes and backward-compatible improvements. Customization strings often describe a vendor or specific feature. For example:

- `linux-2.4.37-foo.tar.gz`: Indicates a stable 2.4 kernel source archive from the vendor "Foo Industries"
- `/boot/bzImage-2.7.5-smp`: Indicates a compiled experimental 2.7 base kernel with SMP support enabled
- `patch-2.6.21.bz2`: Indicates a patch to update an earlier 2.6 stable kernel to revision 21

Kernel files

The Linux base kernel comes in two versions: *zImage*, which is limited to about 508

KB, and *bzImage* for larger kernels (up to about 2.5 MB). Generally, modern Linux distributions use the *bzImage* kernel format to allow inclusion of more features. You might expect that since the "z" in *zImage* indicates gzip compression, the "bz" in *bzImage* might mean bzip2 compression is used there. However, the "b" simply stands for "big" -- gzip compression is still used. In either case, as installed in the `/boot/` directory, the base kernel is often renamed as *vmlinuz*. Generally the file `/vmlinuz` is a link to a version names file such as `/boot/vmlinuz-2.6.10-5-386`.

There are a few other files in the `/boot/` directory associated with a base kernel that you should be aware of (sometimes you will find these at the file system root instead). `System.map` is a table showing the addresses for kernel symbols. `initrd.img` is sometimes used by the base kernel to create a simple file system in a ramdisk prior to mounting the full file system.

Kernel modules

Kernel modules contain extra kernel code that may be loaded after the base kernel. Modules typically provide one of the following functions:

- **Device drivers:** Support a specific type of hardware
- **File system drivers:** Provide the optional capability to read and/or write a particular file system
- **System calls:** Most are supported in the base kernel, but kernel modules can add or modify system services
- **Network drivers:** Implement a particular network protocol
- **Executable loaders:** Parse and load additional executable formats

Section 3. Compiling a kernel

This section covers material for topic 2.201.2 for the Intermediate Level Administration (LPIC-2) exam 201. The topic has a weight of 1.

Obtaining kernel sources

The first thing you need to do to compile a new Linux kernel is obtain the source code for one. The main place to find kernel sources is from the Linux Kernel

Archives (kernel.org; see [Resources](#) for a link). The provider of your distribution might also provide its own updated kernel sources that reflect vendor-specific enhancements. For example, you might fetch and unpack a recent kernel version with commands similar to these:

Listing 1. Fetching and unpacking kernel

```
% cd /tmp/src/  
% wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.12.tar.bz2  
% cd /usr/src/  
% tar jxvf /tmp/src/linux-2.6.12.tar.bz2
```

You may need root permissions to unpack the sources under `/usr/src/`. However, you are able to unpack or compile a kernel in a user directory. Check out kernel.org for other archive formats and download protocols.

Checking your kernel sources

If you have successfully obtained and unpacked a kernel source archive, your system should contain a directory such as `/usr/src/linux-2.6.12` (or a similar leaf directory if you unpacked the archive elsewhere). Of particular importance, that directory should contain a README file you might want to read for current information. Underneath this directory are numerous subdirectories containing source files, chiefly either `.c` or `.h` files. The main work of assembling these source files into a working kernel is coded into the file `Makefile`, which is utilized by the `make` utility.

Configuring the compilation

Once you have obtained and unpacked your kernel sources, you will want to configure your target kernel. There are three flags to the `make` command that you can use to configure kernel options. Technically, you can also manually edit the file `.config`, but in practice doing so is rarely desirable (you forgo extra informational context and can easily create an invalid configuration). The three flags are `config`, `menuconfig`, and `xconfig`.

Of these options, `make config` is almost as crude as manually editing the `.config` file; it requires you configure every option (out of hundreds) in a fixed order, with no backtracking. For text terminals, `make menuconfig` gives you an attractive curses screen that you can navigate to set just the options you wish to modify. The command `make xconfig` is similar for X11 interfaces but adds a bit extra graphical eye candy (especially pretty with Linux 2.6+).

For many kernel options you have three choices: (1) include the capability in the

base kernel; (2) include it as a kernel module; (3) omit the capability entirely. Generally, there is no harm (except a little extra compilation time) in creating numerous kernel modules, since they are not loaded unless needed. For space-constrained media, you might omit capabilities entirely.

Running the compilation

To actually build a kernel based on the options you have selected, you perform several steps:

- `make dep`: Only necessary on 2.4, no longer on 2.6.
- `make clean`: Cleans up prior object files, a good idea especially if this is not your first compilation of a given kernel tree.
- `make bzImage`: Builds the base kernel. In special circumstances you might use `make zImage` for a small kernel image. You might also use `make zlilo` to install the kernel directly within the lilo boot loader, or `make zdisk` to create a bootable floppy. Generally, it is a better idea to create the kernel image in a directory like `/usr/src/linux/arch/i386/boot/vmlinuz` using `make bzImage`, and manually copy from there.
- `make modules`: Builds all the loadable kernel modules you have configured for the build.
- `sudo make modules_install`: Installs all the built modules to a directory such as `/lib/modules/2.6.12/`, where the directory leaf is named after the kernel version.

Creating an initial ramdisk

If you built important boot drivers as modules, an initial ramdisk is a way of bootstrapping the need for their capabilities during the initial boot process. The especially applies to file system drivers that are compiled as kernel modules. Basically, an initial ramdisk is a magic root pseudo-partition that lives only in memory and is later `chrooted` to the real disk partition (for example, if your root partition is on RAID). Later tutorials in this series will cover this in more detail.

Creating an initial ramdisk image is performed with the command `mkinitrd`. Consult the manpage on your specific Linux distribution for the particular options given to the `mkinitrd` command. In the simplest case, you might run something like this:

Listing 2. Creating a ramdisk

```
% mkinitrd /boot/initrd-2.6.12 2.6.12
```

Installing the compiled Linux kernel

Once you have successfully compiled the base kernel and its associated modules (this might take a while -- maybe hours on a slow machine), you should copy the kernel image (`vmlinuz` or `bzImage`) and the `System.map` file to your `/boot/` directory.

Once you have copied the necessary kernel files to `/boot/`, and installed the kernel modules using `make modules_install`, you need to configure your boot loader -- typically `lilo` or `grub` to access the appropriate kernel(s). The next tutorial in this series provides information on configuring `lilo` and `grub`.

Further information

The `kernel.org` site contains a number of useful links to more information about kernel features and requirements for compilation. A particularly useful and detailed document is Kwan Lowe's *Kernel Rebuild Guide*. You'll find links to both in the [Resources](#) section.

Section 4. Patching a kernel

This section covers material for topic 2.201.3 for the Intermediate Level Administration (LPIC-2) exam 201. The topic has a weight of 2.

Obtaining a patch

Linux kernel sources are distributed as main source trees combined with much smaller patches. Generally, doing it this way allows you to obtain a "bleeding edge" kernel with much quicker downloads. This arrangement lets you apply special-purpose patches from sources other than `kernel.org`.

If you wish to patch several levels of changes, you will need to obtain each incremental patch. For example, suppose that by the time you read this, a Linux 2.6.14 kernel is available, and you had downloaded the 2.6.12 kernel in the prior section. You might run:

Listing 3. Getting incremental patches

```
% wget http://www.kernel.org/pub/linux/kernel/v2.6/patch-2.6.13.bz2
% wget http://www.kernel.org/pub/linux/kernel/v2.6/patch-2.6.14.bz2
```

Unpacking and applying patches

To apply patches, you must first unpack them using `bzip2` or `gzip`, depending on the compression archive format you downloaded, then apply each patch. For example:

Listing 4. Unzipping and applying patches

```
% bzip2 -d patch2.6.13.bz2
% bzip2 -d patch2.6.14.bz2
% cd /usr/src/linux-2.6.12
% patch -p1 < /path/to/patch2.6.13
% patch -p1 < /path/to/patch2.6.14
```

Once patches are applied, proceed with compilation as described in the prior section. `make clean` will remove extra object files that may not reflect the new changes.

Section 5. Customizing a kernel

This section covers material for topic 2.201.4 for the Intermediate Level Administration (LPIC-2) exam 201. The topic has a weight of 1.

About customization

Much of what you would think of as customizing a kernel was discussed in the section of this tutorial on compiling a kernel (specifically, the `make [x|menu]config` options). When compiling a base kernel and kernel modules, you may include or omit many kernel capabilities in order to achieve specific capabilities, run profiles, and memory usage.

This section looks at ways you can modify kernel behavior at runtime.

Finding information about a running kernel

Linux (and other UNIX-like operating systems) uses a special, generally consistent, and elegant technique to store information about a running kernel (or other running processes). The special directory `/proc/` contains pseudo-files and subdirectories with a wealth of information about the running system.

Each process that is created during the uptime of a Linux system creates its own numeric subdirectory with several status files. Much of this information is summarized by userlevel commands and system tools, but the underlying information resides in the `/proc/` file system.

Of particular note for understanding the status of the kernel itself are the contents of `/proc/sys/kernel`.

More about current processes

While the status of processes, especially userland processes, does not pertain to the kernel *per se*, it is important to understand these if you intend to tweak an underlying kernel. The easiest way to obtain a summary of processes is with the `ps` command (graphical and higher level tools also exist). With a process ID in mind, you can explore the running process. For example:

Listing 5. Exploring the running process

```
% ps
  PID TTY          TIME CMD
 16961 pts/2    00:00:00 bash
 17239 pts/2    00:00:00 ps
% ls /proc/16961
binfmt  cwd@    exe@   maps  mounts  stat   status
cmdline environ fd/    mem   root@  statm
```

This tutorial cannot address all the information contained in those process pseudo-files, but just as an example, let's look at part of `status`:

Listing 6. A look at the status pseudo-file

```
$ head -12 /proc/17268/status
Name:  bash
State: S (sleeping)
Tgid:  17268
Pid:   17268
PPid:  17266
TracerPid:
      0
Uid:   0           0           0           0
Gid:   0           0           0           0
FDSize: 256
Groups: 0
VmSize: 2640 kB
VmLck: 0 kB
```

The kernel process

As with user processes, the `/proc/` file system contains useful information about a running kernel. Of particular significance is the directory `/proc/sys/kernel/`:

Listing 7. `/proc/sys/kernel/` directory

```
% ls /proc/sys/kernel/
acct          domainname  msgmni      printk      shmall      threads-max
cad_pid       hostname    osrelease   random/     shmmax      version
cap-bound     hotplug     ostype      real-root-dev  shmmni
core_pattern  modprobe    overflowgid rtsig-max   swsusp
core_uses_pid msgmax      overflowuid rtsig-nr    sysrq
ctrl-alt-del  msgmnb      panic       sem         tainted
```

The contents of these pseudo-files show information on the running kernel. For example:

Listing 8. A look at the `ostype` pseudo-file

```
% cat /proc/sys/kernel/ostype
Linux
% cat /proc/sys/kernel/threads-max
4095
```

Already loaded kernel modules

As with other aspects of a running Linux system, information on loaded kernel modules lives in the `/proc/` file system, specifically in `/proc/modules`. Generally, however, you will access this information using the `lsmod` utility (which simply puts a header on the display of the raw contents of `/proc/modules`); `cat /proc/modules` displays the same information. Let's look at an example:

Listing 9. Contents of `/proc/modules`

```
% lsmod
Module              Size  Used by    Not tainted
lp                  8096   0
parport_pc         25096   1
parport            34176   1 [lp parport_pc]
sg                 34636   0 (autoclean) (unused)
st                 29488   0 (autoclean) (unused)
sr_mod             16920   0 (autoclean) (unused)
sd_mod             13100   0 (autoclean) (unused)
scsi_mod           103284   4 (autoclean) [sg st sr_mod sd_mod]
ide-cd              33856   0 (autoclean)
cdrom               31648   0 (autoclean) [sr_mod ide-cd]
nfsd                74256   8 (autoclean)
af_packet           14952   1 (autoclean)
ip_vs               83192   0 (autoclean)
```

```

floppy          55132  0
8139too        17160  1  (autoclean)
mii            3832  0  (autoclean) [8139too]
supermount     15296  2  (autoclean)
usb-uhci       24652  0  (unused)
usbcore        72992  1  [usb-uhci]
rtc            8060  0  (autoclean)
ext3           59916  2
jbd            38972  2  [ext3]

```

Loading additional kernel modules

There are two tools for loading kernel modules. The command `modprobe` is slightly higher level, and handles loading dependencies -- that is, other kernel modules a loaded kernel module may need. At heart, however, `modprobe` is just a wrapper for calling `insmod`.

For example, suppose you want to load support for the Reiser file system into the kernel (assuming it is not already compiled into the kernel). You can use the `modprobe -nv` option to just see what the command would do, but not actually load anything:

Listing 10. Checking dependencies with modprobe

```

% modprobe -nv reiserfs
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/fs/reiserfs/reiserfs.o.gz

```

In this case, there are no dependencies. In other cases, dependencies might exist (which would be handled by `modprobe` if run without `-n`). For example:

Listing 11. More modprobe

```

% modprobe -nv snd-emux-synth
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/drivers/sound/
soundcore.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/core/
snd.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/synth/
snd-util-mem.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/core/seq/
snd-seq-device.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/core/
snd-timer.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/core/seq/
snd-seq.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/core/seq/
snd-seq-midi-event.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/core/
snd-rawmidi.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/core/seq/
snd-seq-virmidi.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/core/seq/
snd-seq-midi-emul.o.gz
/sbin/insmod /lib/modules/2.4.21-0.13mdk/kernel/sound/synth/emux/

```

```
snd-emux-synth.o.gz
```

Suppose you want to load a kernel module now. You can use `modprobe` to load all dependencies along the way, but to be explicit you should use `insmod`.

From the information given above, you might think to run, for example, `insmod snd-emux-synth`. But if you do that without first loading the dependencies, you will receive complaints about "unresolved symbols." So let's try Reiser file system instead, which stands alone:

Listing 12. Loading a kernel module

```
% insmod reiserfs
Using /lib/modules/2.4.21-0.13mdk/kernel/fs/reiserfs/reiserfs.o.gz
```

Happily enough, your kernel will now support a new file system. You can mount a partition, read/write to it, and so on. For other system capabilities, the concept would be the same.

Removing loaded kernel modules

As with loading modules, unloading them can either be done at a higher level with `modprobe` or at a lower level with `rmmmod`. The higher level tool unloads everything in reverse dependency order. `rmmmod` just removes a single kernel module, but will fail if modules are in use (usually because of dependencies). For example:

Listing 13. Trying to unload modules with dependencies in use

```
% modprobe snd-emux-synth
% rmmmod soundcore
soundcore: Device or resource busy
% modprobe -rv snd-emux-synth
# delete snd-emux-synth
# delete snd-seq-midi-emul
# delete snd-seq-virmidi
# delete snd-rawmidi
# delete snd-seq-midi-event
# delete snd-seq
# delete snd-timer
# delete snd-seq-device
# delete snd-util-mem
# delete snd
# delete soundcore
```

However, if a kernel module is eligible for removal, `rmmmod` will unload it from memory, for example:

Listing 14. Unloading modules with no dependencies

```
% rmdir -v reiserfs
Checking reiserfs for persistent data
```

Automatically loading kernel modules

You can cause kernel modules to be loaded automatically, if you wish, using either the kernel module loader in recent Linux versions, or the `kernelld` daemon in older version. If you use these techniques, the kernel will detect the fact it does not support a particular system call, then attempt to load the appropriate kernel module.

However, unless you run in very memory-constrained systems, there is usually no reason not to simply load needed kernel modules during system startup (see the next tutorial in this series for more information). Some distributions may ship with the kernel module loader enabled.

Autocleaning kernel modules

As with automatic loading, autocleaning kernel modules is mostly only an issue for memory-constrained systems, such as embedded Linux systems. However, you should be aware that kernel modules may be loaded with the `insmod --autoclean` flag, which marks them as unloadable if they are not currently used.

The older `kernelld` daemon would make a call to `rmdir --all` periodically to remove unused kernel modules. In special circumstances (if you are not using `kernelld`, which you will not on recent Linux systems), you might add the command `rmdir --all` to your `crontab`, perhaps running once a minute or so. But mostly, this whole issue is superfluous, since kernel modules generally use much less memory than typical user processes do.

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Read Kwan Lowe's [Kernel Rebuild Guide](#) for more details on building a kernel.
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- Get the Linux kernel source at [kernel.org](#), the Linux Kernel Archives.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- [KernelNewbies.org](#) has lots of resources for people who are new to the kernel: an FAQ, an IRC channel, a mailing list, and a wiki.
- [KernelTrap](#) is a Web community devoted to sharing the latest in kernel development news.
- At [Kernel Traffic](#) you can find a newsletter that covers some of the discussion on the Linux kernel mailing list.
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

David Mertz, Ph.D.

David Mertz is Turing complete, but probably would not pass the Turing Test. For more on his life, see his [personal Web page](#). He's been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#).

LPI exam 201 prep: System startup

Intermediate Level Administration (LPIC-2) topic 202

Skill Level: Intermediate

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer
Gnosis Software

31 Aug 2005

In this tutorial, David Mertz begins continues you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 201. In this second of eight tutorials, you learn the steps a Linux™ system goes through during system initialization, and how to modify and customize those behaviors for your specific needs.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams.

Each exam covers several topics, and each topic has a weight. The weights indicate the relative importance of each topic. Very roughly, expect more questions on the exam for topics with higher weight. The topics and their weights for LPI exam 201 are:

Topic 201

Linux kernel (weight 5).

Topic 202

System startup (weight 5). The focus of this tutorial.

Topic 203

File systems (weight 10).

Topic 204

Hardware (weight 8).

Topic 209

File and service sharing (weight 8).

Topic 211

System maintenance (weight 4).

Topic 213

System customization and automation (weight 3).

Topic 214

Troubleshooting (weight 6).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "System startup," the second of eight tutorials designed to prepare you for LPI exam 201. In this tutorial, you will learn the steps a Linux system goes through during system initialization and how to modify and customize those behaviors for your specific needs.

The tutorial is organized according to the LPI objectives for this topic, as follows:

2.201.1 Customizing system startup and boot processes (weight 2)

You will learn to edit appropriate system startup scripts to customize standard system run levels and boot processes. This objective includes interacting with run levels and creating custom `initrd` images as needed.

2.201.2 System recovery (weight 3)

You will be able to properly manipulate a Linux system during the boot process and during recovery mode. This objective includes using both the `init` utility and `init= kernel` options.

This tutorial is at the border of Linux, strictly speaking. The [previous tutorial \(on topic](#)

201) addressed the kernel, which is the core of Linux. This tutorial moves on to the ancillary tools and scripts that are necessary to get the kernel running and to initialize a system to the point where it does something meaningful. Note that the scripts and tools associated with initialization are maintained by the creators of Linux distributions or individualized by system administrators rather than developed as part of the Linux kernel per se. Still, every Linux system -- even an embedded one -- requires some basic initialization steps. We'll review those steps here.

In later tutorials, we'll look at a variety of tools for networking, system maintenance, manipulating files and data, and so on, which are important for a working Linux installation and part of almost every Linux distribution, but are even less part of Linux per se than are initialization scripts.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. System startup and boot processes

What happens when you turn a Linux computer on?

Let's break the Linux boot process into nine steps that occur in almost every Linux configuration:

1. Hardware/firmware: The BIOS or firmware system reads the master boot record on the harddisk or other boot device (for example, CD, floppy, netboot, etc.).
2. A boot loader runs. Linux systems on x86 systems typically use LILO or GRUB. Some older systems might use loadlin to boot via an intermediate DOS partition. On Power PC® systems, this might be BootX or yaboot. In general, a *boot loader* is a simple program that knows where to look for the Linux kernel, perhaps choosing among several versions or even selecting other operating systems on the same machine.
3. The kernel loads.

4. The root filesystem is mounted. In some cases, a temporary ramdisk image is loaded before the true root filesystem to enable special drivers or modules that might be necessary for the true root filesystem. Once we have a root filesystem in place, we are ready for initialization proper.
5. Start the process `init`, the parent of all other Linux processes.
6. Read the contents of `/etc/inittab` to configure the remaining boot steps. Of special importance is the line in `/etc/inittab` that controls the runlevel the system will boot to (and therefore which further steps will be taken during initialization).
Actually, everything after this point is completely controlled by the content of the file `/etc/inittab`. Specifically, the scripts and tools that run generally follow some conventions, but in theory you could completely change `/etc/inittab` to run different scripts.

One specific setting in `/etc/inittab` is particularly crucial. A line similar to:

```
id:5:initdefault:
```

generally occurs near the top of the file, and sets the runlevel. This runlevel controls what actions are taken in the remainder on the `/etc/inittab` script.

Just what happens as an `/etc/inittab` script is processed? And specifically, what conventional files and directories are involved in the process?

7. Runlevel-neutral system initialization. Generally there are some initialization actions that are performed regardless of runlevel. These steps are indicated in `/etc/inittab` with a line like:

```
# System initialization.  
si::sysinit:/etc/rc.d/rc.sysinit
```

On some Linux systems (mostly Debian-based systems), you will see something more like:

```
si::sysinit:/etc/init.d/rcS
```

If the latter case, `/etc/init.d/rcS` is a script that simply runs each of the scripts matching `/etc/rcS.d/[Ss]??*`. On the other hand, if your system uses `/etc/rc.d/rc.sysinit`, that file contains a single long script to perform *all* the initialization.

8. Runlevel-specific system initialization. You may actually define as many actions as you like related to runlevel, and each action may pertain to one or more runlevels. As a rule, `/etc/inittab` will contain some lines like:

```
10:0:wait:/etc/rc.d/rc 0
# ...
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
```

In turn, the script `/etc/rc.d/rc` will run all the files matched by the pattern `/etc/rc$1.d/[KkSs]??*`. For example, on the sample system described that starts at runlevel 5, we would run (in order):

```
/etc/rc5.d/K15postgresql
/etc/rc5.d/S01switchprofile
/etc/rc5.d/S05harddrake
...
/etc/rc5.d/S55sshd
...
/etc/rc5.d/S99linuxconf
/etc/rc5.d/S99local
```

The files(s) starting with "K" or "k" are *kill scripts* that end processes or clean up their actions. Those starting with "S" or "s" are *startup scripts* that generally launch new processes or otherwise prepare the system to run at that runlevel. Most of these files will be shell scripts, and most will be links (often to files in `/etc/init.d`).

Most of the time, once a Linux system is running at a runlevel, you want to log into the system as a user. To let that happen, a program called `getty` runs to handle the login process. A number of variations on the basic `getty` are used by distribution creators, such as `agetty`, `mgetty`, and `mingetty`. All do basically the same thing.

9. Log in at the prompt. Our good friend `/etc/inittab` usually launches `getty` programs on one or more virtual terminals and does so for several different runlevels. Those are configured with lines like:

```
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

The first number reminds us of the virtual terminal where the `getty` runs; the next set of numbers are the several runlevels where this will happen (for example, launching `mingetty` in all of the runlevels 2, 3, 4, and 5).

Next steps might involve launching additional services, logging into a graphical environment, restoring UI settings, or other more personalized details that are outside the scope of this tutorial.

Understanding runlevels

The concept of runlevel is somewhat arbitrary, or at least it is not hardcoded into a Linux kernel. Valid runlevel numbers to set with the `initdefault` option (or override otherwise) are 0 - 6. By convention, the following meanings are given to each number:

Listing 1. Runlevels, defined

```
# Default runlevel. The runlevels used by Mandrake Linux are:
# 0 - Halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you don't have networking)
# 3 - Full multiuser mode
# 4 - Unused
# 5 - X11
# 6 - Reboot (Do NOT set initdefault to this)
```

This convention, as you can see, is as used in the Mandrake Linux distribution, but most distributions obey the same convention. Text-only or embedded distributions may not use some of the levels, but will still reserve those numbers.

Configuration lines in `/etc/inittab`

You have seen a number of `/etc/inittab` lines in examples, but it is worth understanding explicitly what these lines do. Each one has the format:

```
id:runlevels:action:process
```

The `id` field is a short abbreviation naming the configuration line (1 - 4 characters in recent versions of `init`; 1 - 2 in ancient ones). The `runlevels` is explained already. Next is the action taken by the line. Some actions are "special," such as:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

This action traps the Ctrl-Alt-Delete key sequence (regardless of runlevel). But most actions simply relate to spawning. A partial list of actions includes:

- `respawn`: The process will be restarted whenever it terminates (as with `getty`).
 - `wait`: The process will be started once when the specified runlevel is entered, and `init` will wait for its termination.
 - `once`: The process will be executed once when the specified runlevel is entered.
 - `boot`: The process will be executed during system boot (but after `sysinit`). The `runlevels` field is ignored.
-

Section 3. Customizing system startup and boot processes

What is a boot loader?

A few years ago, a program called LILO was pretty much universally used to boot Linux on x86 systems. The name LILO is short for "Linux LOader." Nowadays, another program called GRUB (GRand Unified Bootloader) is more popular. On non-x86 Linux systems, other boot loaders are used, but they are generally configured in the same manner as LILO or GRUB.

While there are differences in their configuration syntaxes, LILO and GRUB perform largely the same task. Basically, each presents a choice of operating systems (including, perhaps, multiple Linux kernels) and loads the selected OS kernel into memory. Both programs let you pass arguments on to a Linux kernel along the way, and both can be configured to boot non-Linux operating systems on the same machine.

Either LILO or GRUB (or other boot loaders) generally lives in the MBR (Master Boot Record) of the primary hard disk, which is automatically loaded by system BIOS. LILO was restricted to loading a specific raw sector from a harddisk. GRUB is more sophisticated in that it understands a number of filesystem types such as ext2/3, ReiserFS, VFAT, and UFS. This means that GRUB doesn't need to rewrite the MBR every time a configuration file is changed (the way LILO does).

Configuring the LILO boot loader

The LILO boot loader is configured with the contents of the file `/etc/lilo.conf`. For full

details on configuration options, read the manpage on `lilo.conf`. Several initial options control general behavior. For example, you will often see `boot=/dev/hda` or similar; this installs LILO to the MBR of the first IDE hard disk. You might also install LILO within a particular partition, usually because you use a different main boot loader. For example, `boot=/dev/sda3` installs LILO to the third partition of the first SCSI disk. Other options control the appearance and wait time of LILO.

Remember that after you have edited a `/etc/lilo.conf` configuration, you need to run LILO to actually install a new boot sector used during initialization. It is easy to forget to install new settings, but the boot loader itself cannot read the configuration except as encoded as raw sector offsets (which LILO calculates when run).

When using LILO you are mainly interested in the one or more `image=` lines and perhaps in some `other=` lines if you multiboot to other operating systems. A sample `/etc/lilo.conf` might contain:

Listing 2. Sample LILO configuration

```
image=/boot/bzImage-2.7.4
label="experimental"
image=/boot/vmlinuz
label="linux"
initrd=/boot/initrd.img
append="devfs=mount acpi=off quiet"
vga=788
read-only
other=/dev/hda3
label=dos
```

This configuration would allow you to choose at runtime either a 2.7.4 development kernel or a stable kernel (the latter happens to utilize an initial ramdrive during boot). You can also select a DOS installation installed to partition 3 on the first IDE drive.

Configuring the GRUB boot loader

A nice thing about GRUB is that it does not need to be reinstalled each time you change boot configuration. Of course, you *do* need to install it once in the first place, usually using a command like `grub-install /dev/hda`. Generally, distributions will do this for you during installation, so you may never explicitly run this.

However, since GRUB knows how to read many filesystems, normally you can simply change the contents of `/boot/grub/menu.lst` to change the options for the next bootup. Let's look at a sample configuration:

Listing 3. Sample GRUB configuration

```
timeout 5
```

```
color black/yellow yellow/black
default 0
password secretword

title linux
kernel (hd0,1)/boot/vmlinuz root=/dev/hda2 quiet
vga=788 acpi=off
initrd (hd0,1)/boot/initrd.img

title experimental
kernel (hd0,1)/boot/bzImage-2.7.4 root=/dev/hda2 quiet

title dos
root (hd0,4)
makeactive
chainloader +1
```

Changing options within the boot loader (LILO)

Both LILO and GRUB allow you to pass special parameters to the kernel you select. If you use LILO, you may pass boot prompt arguments by appending them to your kernel selection. For example, for a custom boot setting, you might type:

```
LILO: linux ether=9,0x300,0xd0000 root=/dev/ha2 vga=791
acpi=on
```

This line passes special parameters to the Ethernet module, specifies the root partition, chooses video mode, etc. Of course, it is not all that friendly, since you need to know the exact options available *and* type them correctly.

Of particular importance is the option to change the runlevel from the boot loader. For example, for recovery purposes, you may want to run in single-user mode, which you can do with the following:

```
LILO: experimental single
```

or:

```
LILO: linux 1
```

Another special option is the `init=` argument, which lets you use a program other than `init` as the first process. An option for a fallback situation might be `init=/bin/sh`, which at least gets you a Linux shell if `init` fails catastrophically.

Changing options within the boot loader (GRUB)

With GRUB, you have even more flexibility. In fact, GRUB is a whole basic shell that lets you change boot loader configurations and even read filesystems. For custom boot options, press "e" in the GRUB shell, then add options (such as a numeric

runlevel or the keyword "single" as with LILO). All the other boot prompt arguments you might type under LILO can be edited in a GRUB boot command, using simple readlines-style editing.

For some real sense of the power, you can open a GRUB command line. For example, suppose you think your `/etc/inittab` might be misconfigured, and you want to examine it before booting. You might type:

```
grub> cat (hd0,2)/etc/inittab
```

This would let you manually view your initialization without even launching any operating system. If there were a problem there, you might want to boot into single-user mode and fix it.

Customizing what comes after the boot loader

Once you understand the steps in a Linux post-kernel boot process (in other words, the `init` process and everything it calls), you also understand how to customize it. Basically, customization is just a matter of editing `/etc/inittab` and the various scripts in `/etc/rc?.d/` directories.

For example, I recently needed to customize the video bios on a Debian-based Linux laptop using a third-party tool. If this didn't run before X11 ran, my XOrg driver would not detect the correct video modes. Once I figured out what the issue was, the solution was as simple as creating the script `/etc/rcS.d/S56-resolution.sh`. In other words, I began running an extra script during every system startup.

Notably, I made sure this ran before `/etc/rcS.d/S70xorg-common` by the simple convention that scripts run in alphabetical order (if I wanted it to run later, I might have named it `S98-resolution.sh` instead). Arguably, I might have put this script only in the `/etc/rc5.d/` directory to run when X11 does -- but my way lets me manually run `startx` from other runlevels.

Everything in the initialization process is out in the open, right in the filesystem; almost all of it is in editable text scripts.

Section 4. System recovery

About recovery

The nicest thing about Linux from a maintenance perspective is that everything is a file. Of course, it can be perplexing at times to know *which* file something lives in. But as a rule, Linux recovery amounts to using basic filesystem utilities like `cp`, `mv`, and `rm`, and a text editor like `vi`. Of course, to automate these activities, tools like `grep`, `awk`, and `bash` are helpful; or at a higher level, `perl` or `python`. This tutorial does not address basic file manipulation.

Assuming you know how to manipulate and edit files, the only "gotcha" perhaps remaining for a broken system is not being able to use the filesystems at all.

Fixing a filesystem with fsck

Your best friend in repairing a broken filesystem is `fsck`. The [next tutorial \(on topic 203\)](#) has more information, so we will just introduce the tool here.

The tool called `fsck` is actually just a front-end for a number of more narrow `fsck.*` tools -- `fsck.ext2`, `fsck.ext3`, or `fsck.reiser`. You may specify the type explicitly using the `-t` option, but `fsck` will make an effort to figure it out on its own. Read the manpage for `fsck` or `fsck.*` for more details. The main thing you want to know is that the `-a` option will try to fix everything it can automatically.

You can check an unmounted filesystem by mentioning its raw device. For example, use `fsck /dev/hda8` to check a partition not in use. You can also check a rooted filesystem such as `fsck /home`, but generally do that only if the filesystem is already mounted as read-only, not as read-write.

Mounting and unmounting with mount and umount

A flexible feature of Linux systems is the fine-tuned control you have over mounting and unmounting filesystems. Unlike under Windows and some other operating systems, partitions are not automatically assigned locations by the Linux kernel, but are instead attached to the single `/` root hierarchy by the `mount` command. Moreover, different filesystem types (on different drives, even) may be mounted within the same hierarchy. You can unmount a particular partition with the `umount` command, specifying either the mount point (such as `/home`) or the raw device (such as `/dev/hda7`).

For recovery purposes, the ability to control mount points lets you do forensic analysis on partitions -- using `fsck` or other tools -- without risk of further damage to a damaged filesystem. You may also custom mount a filesystem using various options; the most important of these is mounting read-only using either of the synonyms `-r` or `-o ro`.

As a quick example, you might want to substitute one user directory location for

another, either because of damage to one or simply to expand disk space or move to a faster disk. You might perform this switch using something like:

```
# umount /home # old /dev/hda7 home dir
# mount -t xfs /dev/sda1 /home # new SCSI disk using XFS
# mount -t ext3 /dev/sda2 /tmp # also put the /tmp on SCSI
```

Mounting at bootup with /etc/fstab

For recovery, system upgrades, and special purposes, it is useful to be able to mount and unmount filesystems at will. But for day-to-day operation, you generally want a pretty fixed set of mounts to happen at every system boot. You control the mounts that happen at bootup by putting configuration lines in the file /etc/fstab. A typical configuration might look something like this:

Listing 4. Sample configuration in /etc/fstab

```
/dev/hda7 / ext3 defaults 1 1
none /dev/pts devpts mode=0620 0 0
/dev/hda9 /home ext3 defaults 1 2
none /mnt/cdrom supermount
dev=/dev/hdc,fs=auto,ro,--,iocharset=iso8859-1,codepage=850,umask=0 0 0
none /mnt/floppy supermount
dev=/dev/fd0,fs=auto,--,iocharset=iso8859-1,sync,codepage=850,umask=0 0 0
none /proc proc defaults 0 0
/dev/hda8 swap swap defaults 0 0
```

Find more details in the [next tutorial \(on topic 203\)](#).

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Boot loader showdown: Getting to know LILO and GRUB](#)" (developerWorks, August 2005) discusses how boot loaders work and can help you decide which of the two most popular -- LILO and GRUB -- might work best for you.
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

David Mertz, Ph.D.

David Mertz is Turing complete, but probably would not pass the Turing Test. For more on his life, see his [personal Web page](#). He's been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#).

LPI exam 201 prep: Filesystem

Intermediate Level Administration (LPIC-2) topic 203

Skill Level: Intermediate

David Mertz, Ph.D. (mertz@gnosis.cx)

Developer
Gnosis Software

31 Aug 2005

In this tutorial, David Mertz continues preparing you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 201. In this third of eight tutorials, you will learn how to control the mounting and un-mounting of filesystems, examine existing filesystems, create filesystems, and perform remedial actions on damaged filesystems.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams.

Each exam covers several topics, and each topic has a weight. The weights indicate the relative importance of each topic. Very roughly, expect more questions on the exam for topics with higher weight. The topics and their weights for LPI exam 201 are:

Topic 201

Linux kernel (weight 5).

Topic 202

System startup (weight 5).

Topic 203

Filesystem (weight 10). The focus of this tutorial.

Topic 204

Hardware (weight 8).

Topic 209

File and service sharing (weight 8).

Topic 211

System maintenance (weight 4).

Topic 213

System customization and automation (weight 3).

Topic 214

Troubleshooting (weight 6).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Filesystem," the third of eight tutorials designed to prepare you for LPI exam 201. In this tutorial, you will learn how to control the mounting and un-mounting of filesystems, examine existing filesystems, create filesystems, and perform remedial actions on damaged filesystems.

The tutorial is organized according to the LPI objectives for this topic, as follows:

2.203.1 Operating the Linux filesystem (weight 3)

You will be able to properly configure and navigate the standard Linux filesystem. This objective includes configuring and mounting various filesystem types. Also included is manipulating filesystems to adjust for disk space requirements or device additions.

2.203.2 Maintaining a Linux filesystem (weight 4)

You will be able to properly maintain a Linux filesystem using system utilities. This objective includes manipulating a standard ext2 filesystem.

2.203.3 Creating and configuring filesystem options (weight 3)

You will be able to configure automount filesystems. This objective includes configuring automount for network and device filesystems. Also included is creating non-ext2 filesystems for devices such as CD-ROMs.

This tutorial addresses elements of Linux as well as external tools that are useful for working with Linux systems. Support for filesystems, devices, and partitions is either compiled into the base kernel or included in kernel modules.

However, various tools that you are likely to use in managing these filesystems recognized by Linux are userland utilities and therefore only commonly included with Linux distributions rather than part of Linux itself. Nonetheless, filesystem tools are essential for working with pretty much every Linux system regardless of its intended use (even non-networked or embedded systems).

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. Creating and configuring filesystem options

Let's go out of order and start with creating and configuring filesystems and options.

Creating partitions

Before you can work with Linux filesystems, you need to create them. But before you can create a filesystem, you need to create a partition to put it on. As a brief primer, on x86 machines, hard disks may be divided into four primary partitions, but the last of those primary partitions may contain a number of extended partitions inside it.

In the past there were a number of restrictions about the highest cylinders where bootable partitions can occur, maximum disk sizes, locations of primary partitions on large disks, and so on. However, for the last five years or more, pretty much all system BIOSes flexibly handle disks of essentially unlimited size, and modern bootloaders (at least for Linux) have no important restrictions about partition sizes or locations.

The only rule that remains to worry about nowadays concerns operating systems other than Linux. Sometimes those still insist on living in primary partitions near the

front of a hard disk. Linux partitions are more than happy to reside on extended partitions and anywhere on any accessible disk drive.

There are several widely used tools in the Linux world for creating and manipulating partitions on hard disks. The oldest such tool is *fdisk*. Somewhat later, the *curses*-based *cdisk* became popular. GNU *parted* is also used in many distributions. Further, the installation systems for most Linux distributions and/or their graphical environments come with partitioning front-ends that provide friendlier interfaces to viewing and modifying partitions.

Of these tools, *fdisk* remains the most flexible and most forgiving tool. But "forgiving" is a slightly odd term to use here. Writing unintended partition-table information is a recipe for disaster regardless of what tool you use. But if your partitions have been created in somewhat non-standard ways, often by non-Linux operating systems and tools, *fdisk* will generally forge ahead where other tools might refuse to try at all. If it works, however, *cdisk* is generally friendlier and more interactive. And *parted* provides more powerful options about resizing and moving existing partitions non-destructively than *fdisk* or *cdisk*.

Whatever tool you use to create partitions, the concepts are similar. First, you need to perform these operations as root, ideally in single-user mode. And it's hard to make this point too strongly: *Be careful* when you modify partitions: ideally, have all important data backed up, and pay careful attention to what changes you make.

Before you start modifying a partition table, it is a good idea to be clear about what partitions currently exist. The command `fdisk -l /dev/hda` (or similar for other disks, for example `/dev/hdb` or `/dev/sda`) gives you information on existing partitions. `mount` is also helpful in understanding how these existing partitions are actually being used. If you wish to create new partitions, keep in mind any extra sectors within the fourth primary partition that might be available for additional extended partitions.

Let's see an example of a partition table on a Linux system of mine:

Listing 1. Sample partition table

```
% fdisk -l /dev/sda
Disk /dev/sda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders

Device Boot      Start         End      Blocks   Id  System
/dev/sda1    *           1         1216     9767488+   7   HPFS/NTFS
/dev/sda3             1217        4255     24410767+  83   Linux
/dev/sda4             4256        9729     43969905   5   Extended
/dev/sda5             4256        4380      1004031   82   Linux swap /
Solaris
/dev/sda6             4381        5597      9775521   83   Linux
```

This tells us several things. First of all, we can see that partition one is probably

used by a foreign operating system. And running `mount` will let us know:

```
% mount | head -1
/dev/sda3 on / type reiserfs (rw,noatime,notail,commit=600)
```

That is, the existing system is rooted on `/dev/sda3`. Perhaps most interestingly, the `/dev/sda4` partition extends to cylinder 9729, but the extended partitions within it use only part of that space.

After discovering some free space available on the drive, let's create a partition within it using `fdisk`:

```
% fdisk /dev/sda
```

The number of cylinders for this disk is set to 9729. There is nothing wrong with that, but this is larger than 1024 and could, in certain setups, cause problems with:

1. Software that runs at boot time (such as old versions of LILO).
2. Booting ant partitioning software from other operating systems (such as DOS FDISK, OS/2 FDISK).

Listing 2. Creating a partition

```
Command (m for help): n
Command action
l   logical (5 or over)
p   primary partition (1-4)
l
First cylinder (5598-9729), default 5598):
Using default value 5598
Last cylinder or +size or +sizeM or +sizeK (5598-9729):
+10000M

Command (m for help): w
The partition table has been altered!
```

Everything that follows a colon is typed in by the user (you). At this point, we have created a new 10 GB Linux partition:

```
/dev/sda7 5598 6814 9775521 83 Linux
```

Keep reading to find out how to *use* this partition. Note that you may need to reboot a system to make new partitions accessible.

Making a filesystem in a partition

Just having a partition is not quite enough; you need to make the filesystem. Above

we created a new Linux partition at `/dev/sda7`, but we need to decide which of the many filesystems Linux supports to use within that partition. Do we want the historical default `ext2`? Or the newer journaling-enhanced extension `ext3` format? Maybe we want one of the enhanced filesystems contributed to Linux by other parties: ReiserFS, XFS, JFS. Or maybe we need a filesystem that interoperates with another operating system, such as Minix, MSDOS, or VFAT (some others can be read if created already, but not always created with Linux tools).

All of the tools for making new filesystems follow the naming convention `mkfs.*`. That is, your system might have `mkfs.ext2`, `mkfs.minix`, `mkfs.xfs`, and so on, usually installed in `/sbin/`. Also, you may access each of these using the basic `mkfs -t <fstype>` switch. Several, but not all, of the filesystems also have compact forms like `mke3fs`. The filesystems that are available depend on your specific Linux distribution and version, and any extra tools you might have installed. `mkfs.ext2` is available on nearly every distribution.

The basics of making a filesystem are simple. Just run your desired `mkfs.*` tool against the partition you want the filesystem to exist on. For example:

```
% mkfs.xfs /dev/sda7
```

The displayed messages will vary according to the filesystem type you used. Generally, the messages give you information on the number of inodes, blocks, journaling type (if any), extents, and fragments relevant to that particular filesystem's usage strategy. Many of the filesystem creation tools warn you if you try to create a new filesystem on a partition with an existing filesystem, but not all of them will, so proceed with great caution (creating a new filesystem over an old one will probably result in data loss).

Making an ISO filesystem with mkisofs

A special case of making a filesystem is the creation of an *ISO filesystem*, which is a system image that may be written to a writeable CD or DVD device. An ISO filesystem is special in the sense that it is really just a (large) file with data laid out in a certain way rather than in an arrangement of a raw device like `/dev/cdrom` or `/dev/hdb3`.

The basic idea of creating an ISO filesystem -- which really means either an ISO9660 or HFS hybrid volume -- is simply to take the files in one or more existing hierarchies and arrange them into an ISO image. ISO9660 itself is limited to simple DOS-style 8.3 names, but the Rock Ridge and Joliet extensions allow storage of longer names and/or additional file attributes. For example, to create an image of a project, you might use a command like:

```
% mkisofs -o ProjectCD.iso -r ~/project-files ~/project-extras
```

In this case, we create an ISO image that uses Rock Ridge attributes (but unlike `-R` sets more useful values, such as all files readable) and contains a merge of all the files in two directories. Other options would let us add bootable headers to the image, create an HFS image, attach directories in specified locations other than root, and fine-tune file options.

Making an ISO filesystem with `cdrecord`

Transferring an ISO image to a recordable CD or DVD is often accomplished nowadays using a front-end tool, often a GUI interface. For example, both Gnome and KDE make CD burning part of their file-manager interface. Some commercial tools exist also. But for a system administrator, the older command-line tool `cdrecord` is a trusted utility that is present on most modern distributions and is much closer to "standard" than are other front-ends. Generally, the basic usage just requires specifying the device you want to write to and the ISO file you want to write.

As usual with Linux utilities, you may also specify a number of options to the record process, such as `-overburn` for CDs larger than 650 MB or a specific burn speed for your writer. See the manpage for `cdrecord` for current details.

You can find the device with the `-scanbus` option. The device you want is named as a numeric triple indicating the bus, not a regular block device in the filesystem. For example, you might see something like (abridged):

Listing 3. Finding a recordable device

```
% cdrecord -scanbus
[... ]
scsibus0:
0,0,0    0) 'ATA      ' 'WDC WD800UE-00HC' '09.0' Disk
0,1,0    1) *
[... ]
scsibus1:
1,0,0   100) 'Slimtype' 'DVDRW SOSW-852S ' 'PSB2' Removable
CD-ROM
[... ]
```

With bus information in hand, you can burn an image:

```
% sudo cdrecord -overburn -v speed=16 dev=1,0,0
/media/KNOPPIX_V3.6-2004-08-16-EN.iso
```

In this case, the image is oversized and I know my burner supports 16x. The action command output is rather verbose because of the `-v` option, but that helps in understanding the whole process.

Making an ISO filesystem with dd

Of final note, sometimes you want to create a brand new ISO image not out of some directories in your main filesystem, but rather from an already existing CD or DVD. To make an ISO image from a CD, just use the command `dd`, but refer to the raw block device for the CD rather than to the mounted location:

```
% dd if=/dev/cdrom of=project-cd.iso
```

You might wonder why not just use `cp` if the goal is to copy bytes. Actually, if you ignore a reported I/O error when the raw device runs out of bytes to copy, the `cp` command is *likely* to work. However, `dd` is better style (and doesn't complain, but instead reports a summary of activity).

Section 3. Operating the Linux filesystem

Mounting and unmounting with mount and umount

A flexible feature of Linux systems is the fine-tuned control you have over mounting and unmounting filesystems. Unlike under Windows and some other operating systems, partitions are not automatically assigned locations by the Linux kernel, but are instead attached to the single `/` root hierarchy by the `mount` command. Moreover, different filesystem types (on different drives, even) may be mounted within the same hierarchy. You can unmount a particular partition with the `umount` command, specifying either the mount point (such as `/home`) or the raw device (such as `/dev/hda7`).

For recovery purposes, the ability to control mount points lets you do forensic analysis on partitions -- using `fsck` or other tools -- without risk of further damage to a damaged filesystem. You may also custom mount a filesystem using various options; the most important of these is mounting read-only using either of the synonyms `-r` or `-o ro`.

As a quick example, you might want to substitute one user directory location for another, either because of damage to one or simply to expand disk space or move to a faster disk. You might perform this switch using something like:

```
# umount /home # old /dev/hda7 home dir
# mount -t xfs /dev/sda1 /home # new SCSI disk using XFS
# mount -t ext3 /dev/sda2 /tmp # also put the /tmp on SCSI
```

Default mounting

For day-to-day operation, you generally want a pretty fixed set of mounts to happen at every system boot. You control the mounts that happen at bootup by putting configuration lines in the file `/etc/fstab`. A typical configuration might look something like this:

Listing 4. Sample configuration for mounting at bootup

```
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
/dev/sda3 / reiserfs notail 0 1
/dev/sda5 none swap sw 0 0
/dev/sda6 /home ext3 rw 0 2
/dev/scd0 /media/cdrom0 udf,iso9660 ro,user,noauto 0 0
/media/Ubuntu-5.04-install-i386.iso /media/Ubuntu_5.04 iso9660
rw,loop 0 0
```

In the above listing, the first field (`<file system>`) is normally the block device to mount. The second field (`<mount point>`) is the mounted location. In some special cases, something other than a block device is given first. For `supermount` devices, you will see `none`. `/proc` is another odd case. You might also mount loopback devices, which are usually regular files.

The third field (`<type>`) and fourth field (`<options>`) are fairly straightforward; options depend on filesystem type and usage. The fifth field (`<dump>`) is usually zero. The sixth field (`<pass>`) should be 1 for the root filesystem and 2 for other filesystems that should be `fscked` during system boot.

Automounting with AMD and automount

Linux has quite a few ways of automatically mounting media that is removable (floppies, CDs, USB drives) or otherwise not of fixed availability (such as NFS filesystems). The goal of all these tools is similar, but each works slightly differently.

The tool AMD (automount daemon) is somewhat older and operates in userland. Basically, AMD runs periodically to see if any new mountable filesystems have become available generally for NFS filesystems. For the most part, AMD has been replaced in Linux distributions by `Autofs`, which runs as a kernel process.

You set up `Autofs` by compiling it into the kernel you use. After that, the behavior of the `Autofs` daemon (usually `/etc/init.d/autofs`) is controlled by the file `/etc/auto.master`, which in turn references a map file. For example:

```
# Sample auto.master file
# Format of this file: mountpoint map options
```

```
/mnt /etc/auto.mnt --timeout=10
```

The referenced `/etc/auto.mnt` specifies one or more subdirectories of `/mnt` that will be mounted (if access is requested). Unmounting will occur automatically, in this case 10 seconds after last access.

```
# Sample /etc/auto.mnt
floppy -fstype=auto,rw,sync,umask=002 :/dev/fd0
cdrom -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom
remote -fstype=nfs example.com:/some/dir
```

Automounting with supermount and submount

The tools `supermount` and `submount` are kernel-level tools (either compiled into the base kernel or kernel modules) to automatically mount removable media when accessed. `submount` is somewhat newer, but `supermount` is still probably used in more distributions. Neither tool is useful for NFS remote mounts, but either is more seamless than `Autofs` for local media.

In either case, devices requiring automounting are generally listed in the `/etc/fstab` configuration. The tools use slightly different syntaxes in `/etc/fstab`, but both are straightforward. A `supermount`-enabled `/etc/fstab` might contain the following:

```
# Example of supermount in /etc/fstab
none /mnt/cdrom supermount fs=auto,dev=/dev/cdrom 0 0
none /mnt/floppy supermount fs=auto,dev=/dev/fd0,--,user,rw 0
0
```

`submount` specifies the block device in the regular location rather than as a mount option. For example:

```
/dev/cdrom /mnt/cdrom subfs fs=cdfss,ro,users 0 0
/dev/fd0 /mnt/floppy subfs fs=floppyfss,rw,users 0 0
```

What is currently mounted?

A Linux user has several ways to see a list of current mounts. The `mount` command with no options (or with the `-l` option) lists currently mounted paths. If you like, you can filter the results with the `-t fstype` option.

The underlying dynamic information on mounted filesystems lives in `/etc/mtab`. The `mount` and `umount` commands and other systems processes will update this file to reflect current status; you should treat this file as read-only. A subset of the `mount` status information is additionally contained in `/proc/mounts`.

Special tools

The tool `sync` forces changed unwritten blocks to disk. You should not need to use this in normal situations, but you can sometimes check for disk problems by checking for a non-zero exit status. Modern filesystems, particularly journaling filesystems like `ext3`, `Reiser`, and `JFS`, effectively do syncing on every write.

If you like, you can manually disable or enable the use of a swapping or enable/disable swapping for particular devices. Normally, every device marked as type `swap` in `/etc/fstab` is used for swapping.

Section 4. Maintaining a Linux filesystem

Fixing a filesystem with `fsck`

Your best friend in repairing a broken filesystem is `fsck`.

The tool called `fsck` is actually just a front-end for a number of more narrow `fsck.*` tools -- `fsck.ext2`, `fsck.ext3`, or `fsck.reiser`. You may specify the type explicitly using the `-t` option, but `fsck` will make an effort to figure it out on its own. Read the manpage for `fsck` or `fsck.*` for more details. The main thing you want to know is that the `-a` option will try to fix everything it can automatically.

You can check an unmounted filesystem by mentioning its raw device. For example, use `fsck /dev/hda8` to check a partition not in use. You can also check a rooted filesystem such as `fsck /home`, but generally do that only if the filesystem is already mounted as read-only, not as read-write.

Checking blocks with `badblocks`

The `badblocks` utility does a lower-level test of the quality of a block device (or partition) than `fsck` does. `badblocks` may -- destructively or non-destructively -- examine the reliability of blocks on a device by writing and reading test patterns. The default option is `-n` for a slower mode that preserves existing data. For a brand-new partition with no existing files, you can (and probably should) use the `-w`. This tool simply informs you of bad blocks; it does not repair or mark them.

However, in practice, you are usually better off using the `badblock-checking` wrapper

in the `fsck.*` tool for your filesystem. For example, `e2fsck` (also called `fsck.ext2`) has the option `-c` to find and mark badblocks that the `badblocks` tool can detect. ReiserFS has similar `--check` and `--badblocks` options (but is not quite as automatic). Read the documentation for your particular filesystem for details on wrappers to `badblocks`.

Finding other maintenance utilities

Several tools are available for examining and fine-tuning Linux filesystems. In normal usage, the default settings for filesystems are well designed, but occasionally you will want to use filesystem tools for forensic analysis on crashed systems or to tune performance on systems with well-defined usage patterns.

Each filesystem type has its own set of tools; check the documentation for the filesystem you use for more details. Most have a similar array of tools. Some examples include:

- `dumpe2fs`: Output information about an ext2/3 filesystem.
- `tune2fs`: Adjust filesystem parameters on an ext2/3 filesystem.
- `debugfs`: Interactively fine-tune and examine an ext2/3 filesystem.
- `debugreiserfs`: Output information about a Reiser filesystem.
- `reiserfstune`: Adjust filesystem parameters on a Reiser filesystem.
- `xfs_admin`: Adjust filesystem parameters of an XFS filesystem.

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Common threads: Advanced filesystem implementor's guide, Parts 1 - 13](#)" (developerWorks, starting June 2001) is an excellent series on Linux filesystems.
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

David Mertz, Ph.D.

David Mertz is Turing complete, but probably would not pass the Turing Test. For more on his life, see his [personal Web page](#). He's been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#).

LPI exam 201 prep: Hardware

Intermediate Level Administration (LPIC-2) topic 204

Skill Level: Intermediate

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer
Gnosis Software

[Brad Huntting \(huntting@glarp.com\)](mailto:huntting@glarp.com)

Mathematician
University of Colorado

02 Sep 2005

In this tutorial, David Mertz and Brad Huntting continue preparing you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 201. In this fourth of eight tutorials, you learn how to add and configure hardware to a Linux™ system, including RAID arrays, PCMCIA cards, other storage devices, displays, video controllers, and other components.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams.

Each exam covers several topics, and each topic has a weight. The weights indicate the relative importance of each topic. Very roughly, expect more questions on the

exam for topics with higher weight. The topics and their weights for LPI exam 201 are:

Topic 201

Linux kernel (weight 5).

Topic 202

System startup (weight 5).

Topic 203

Filesystem (weight 10).

Topic 204

Hardware (weight 8). The focus of this tutorial.

Topic 209

File and service sharing (weight 8).

Topic 211

System maintenance (weight 4).

Topic 213

System customization and automation (weight 3).

Topic 214

Troubleshooting (weight 6).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Hardware," the fourth of eight tutorials designed to prepare you for LPI exam 201. In this tutorial, you learn how to add and configure hardware to a Linux system, including RAID arrays, PCMCIA cards, other storage devices, displays, video controllers, and other components.

The tutorial is organized according to the LPI objectives for this topic, as follows:

2.204.1 Configuring RAID (weight 2)

You will be able to configure and implement software RAID. This objective includes using mkraid tools and configuring RAID 0, 1, and 5.

2.204.2 Adding new hardware (weight 3)

You will be able to configure internal and external devices for a system including new hard disks, dumb terminal devices, serial UPS devices, multi-port

serial cards, and LCD panels.

2.204.3 Software and kernel configuration (weight 2)

You will be able to configure kernel options to support various hardware devices including UDMA66 drives and IDE CD burners. This objective includes using LVM (Logical Volume Manager) to manage hard disk drives and partitions as well as software tools to interact with hard disk settings.

2.204.4 Configuring PCMCIA devices (weight 1)

You will be able to configure a Linux installation to include PCMCIA support. This objective includes configuring PCMCIA devices, such as Ethernet adapters, to autodetect when inserted.

While you will often use userland tools to work with hardware devices, for the most part, basic support for those devices is provided by a Linux base kernel, kernel modules, or both. One notable exception to the close connection between the Linux kernel and hardware devices is in graphics cards and computer displays. A simple console text display is handled well enough by the Linux kernel (and even some graphics with framebuffer support), but generally advanced capabilities of graphics cards are controlled by XFree86 or more recently X.Org, X11 drivers. Almost all distributions include X11 and associated window managers and desktop environments; but for non-desktop servers, using X11 may be superfluous.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

In addition, some information on adding hardware is covered in two other tutorials: "[LPI exam 201 prep \(topic 201\): Linux kernel](#)" and "[LPI exam 201 prep \(topic 203\): Filesystems](#)." The LPI exam on hardware expects familiarity with kernel and filesystem tuning, so please refer to those other tutorials during exam preparation.

Section 2. Configuring RAID

What is RAID?

RAID (Redundant Array of Inexpensive Disks) provides mechanisms to combine multiple partitions on different hard drives into larger or more resilient virtual hard

drives. Numerous RAID levels were initially defined, but only three remain in common use: RAID-0 (disk striping), RAID-1 (mirroring), and RAID-5 (striping with parity information). RAID-4 is also occasionally used; it is similar to RAID-5 but puts parity information on exactly one drive rather than distributing it.

This tutorial discusses the *"new-style" RAID* under Linux (the default for 2.4 and 2.6 kernels with backports to earlier kernels available). The *"old-style" RAID* initially present in 2.0 and 2.2 kernels was buggy and should not be used. Specifically, *"new-style"* means the 0.90 RAID layer made by Ingo Molnar and others.

Using a RAID array

There are two basic parts to using a RAID array. The simple part is mounting the RAID device. Once a RAID (virtual) device is configured, it looks to the `mount` command the same as a regular block device partition. A RAID device is named as `/dev/mdN` once it is created, so you might mount it as:

```
% mount /dev/md0 /home
```

You can also include a line in `/etc/fstab` to mount the RAID virtual partition (this is usually the preferred method). The device driver reads superblocks of raw partitions to assemble a RAID partition once it has been created.

The more complicated part (more detailed, anyway) involves creating the RAID device out of relevant raw partitions. You can create a RAID partition with the tool `mkraid` combined with an `/etc/raidtab` configuration file.

You can also use the newer tool `mdadm`, which can usually operate without the need for a separate configuration file. In most distributions, `mdadm` is supplanting `raidtools` (which includes `mkraid`), but this tutorial discusses `mkraid` to follow the LPI exam objectives. The concepts are similar either way, but you should read the `mdadm` manpage to learn about its switches.

The layout of `/etc/raidtab`

The following definitions are used in the `/etc/raidtab` file to describe the components of a RAID. This list is not exhaustive.

- `raiddev`: The virtual disk partition provided by RAID (`/dev/md?`). This is the device that `mkfs` and `fsck` work with, and that is mounted in the same way as an actual hard disk partition.
- `raid-disk`: The underlying partitions used to build a RAID. They should be marked (with `fdisk` or similar tools) as partition type `0xFD`.

- `spare-disk`: These disks (typically there's only one) normally lie unused. When one of the raid disks fails the spare disk is brought online as a replacement.

Configuring RAID-0

RAID-0 or "disk striping" provides more disk I/O bandwidth at the cost of less reliability (a failure in any one of the raid-disks and you lose the entire RAID device). For example the following `/etc/raidtab` entry sets up a RAID-0 device:

```
raiddev /dev/md0
raid-level 0
nr-raid-disks 2
nr-spare-disks 0
chunk-size 32
persistent-superblock 1
device /dev/sda2
raid-disk 0
device /dev/sdb2
raid-disk 1
```

This defines a RAID-0 virtual device called `/dev/md0`. The first 32 KB chunk of `/dev/md0` is allocated to `/dev/sda2`, the next 32 KB go on `/dev/sdb2`, the third chunk is on `/dev/sda2`, etc.

To actually create the device, simply run:

```
% sudo mkraid /dev/md0
```

If you use `mdadm`, switches will configure these options rather than the `/etc/raidtab` file.

Configuring RAID-1

RAID-1 or "disk mirroring" simply keeps duplicate copies of the data on both block devices. RAID-1 gracefully handles a drive failure with no noticeable drop in performance. RAID-1 is generally considered expensive since half of your disk space is redundant. For example:

```
raiddev /dev/md0
raid-level 1
nr-raid-disks 2
nr-spare-disks 1
persistent-superblock 1
device /dev/sdb6
raid-disk 0
```

```
device      /dev/sdc5
raid-disk   1
device      /dev/sdd5
spare-disk  0
```

Data written to `/dev/md0` will be saved on both `/dev/sdb6` and `/dev/sdc5`. `/dev/sdd5` is configured as a *hot spare*. In the event `/dev/sdb6` or `/dev/sdc5` malfunctions, `/dev/sdd5` will be populated with data and brought online as a replacement.

Configuring RAID-5

RAID-5 requires at least three drives and uses error correction to get most of the benefits of disk striping but with the ability to survive a single drive failure. On the positive side, it requires only one extra redundant drive. On the negative side, RAID-5 is more complex; when a drive does fail, it drops into *degraded mode* which can dramatically impact I/O performance until a spare-disk can be brought online and repopulated.

```
raiddev /dev/md0
raid-level      5
nr-raid-disks  7
nr-spare-disks 0
persistent-superblock 1
parity-algorithm left-symmetric
chunk-size     32
device         /dev/sda3
raid-disk      0
device         /dev/sdb1
raid-disk      1
device         /dev/sdc1
raid-disk      2
device         /dev/sdd1
raid-disk      3
device         /dev/sde1
raid-disk      4
device         /dev/sdf1
raid-disk      5
device         /dev/sdg1
raid-disk      6
```

Using mke2fs or mke3fs

If you format a RAID-5 virtual device using `e2fs` or `e3fs`, you should pay attention to the *stride* option. The `-R stride=nn` option will allow `mke2fs` to better place different ext2-specific data structures in an intelligent way on the RAID device.

If the chunk size is 32 KB, it means that 32 KB of consecutive data will reside on one disk. If an ext2 filesystem has 4 KB block size then there will be eight filesystem blocks in one array chunk. We can indicate this information to the filesystem by running:

```
% mke2fs -b 4096 -R stride=8 /dev/md0
```

RAID-5 performance is greatly enhanced by providing the filesystem with correct stride information.

Kernel support and failures

Enabling the *persistent-superblock* feature allows the kernel to start the RAID automatically at boot time. New-style RAID uses the persistent superblock and is supported in 2.4 and 2.6 kernels. Patches are available to retrofit 2.0 and 2.2 kernels.

Here's what happens when a drive fails:

- **RAID-0:** All data is lost>
- **RAID-1/RAID-5:** The failed drive is taken offline and the spare disk (if it exists) is brought online and populated with data.

The document "The Software-RAID HOWTO" in the Linux HOWTO project discusses swapping in drives for failed or updated drives, including when such drives are hot-swappable and when a reboot will be required. Generally, SCSI (or Firewire) are hot-swappable while IDE drives are not.

Section 3. Adding new hardware

About hardware

Linux, especially in recent versions, has an amazingly robust and broad capability to utilize a variety of hardware devices. In general, there are two levels of support that you might need to worry about in supporting hardware. At a first level, there is a question of supporting a device at a basic system level; doing this is almost always by means of loading appropriate kernel modules for your hardware device.

At a second level, some devices need more-or-less separate support from the X11R6 subsystem: generally either XFree86 or X.Org (very rarely, a commercial X11 subsystem is used, but this tutorial does not discuss that).

Support for the main hot-swappable device categories -- such as those using

PCMCIA or USB interfaces -- are covered in their own topic sections to follow.

About X11

A quick note: X.Org is essentially the successor project to XFree86 (technically a fork). While XFree86 has not officially folded, almost all distribution support has shifted to X.Org because of licensing issues. Fortunately, except for some minor renaming of configuration files, the initial code base for both forks is largely the same; some newer special features are more likely to be supported in X.Org only.

X11R6 is a system for (networked) presentation of graphical applications on a user workstation. Perhaps counter-intuitively, an "X server" is the physical machine that a user concretely interacts with using its keyboard, pointing device(s), video card, display monitor, etc. An "X client" is the physical machine that devotes CPU time, disk space, and other non-presentation resources to running an application. In many or most Linux systems, the X server and X client are the self-same physical machine and a very efficient local communication channel is used for an application to communicate with user I/O devices.

An X server -- such as X.Org -- needs to provide device support for the I/O devices with which a user will interact with an application. Overwhelmingly, where there is any difficulty, it has to do with configuring video cards and display monitors. Fortunately, this difficulty has decreased in recent X.Org/XFree86 versions with much more automatic detection performed successfully. Technically, an X server also needs to support input devices -- keyboards and mice -- but that is usually fairly painless since these are well standardized interfaces. Everything else -- disk access, memory, network interfaces, printers, special devices like scanners, and so on -- are all handled by the X client application, generally by the underlying Linux kernel.

Kernel device support

Almost everything you need to know about device support in the Linux kernel boils down to finding, compiling, and loading the right kernel modules. That topic is covered extensively in the Topic 201 tutorial -- readers should consult that tutorial for most issues.

To review kernel modules, the main tools a system administrator needs to think about are `lsmod`, `insmod`, and `modprobe`. Also `rmmod` to a lesser extent. The tools `lsmod`, `insmod` and `rmmod` are low-level tools to, respectively, list, insert, and remove kernel modules for a running Linux kernel. `modprobe` performs these same functions at a higher level by examining dependencies, then making appropriate calls to `insmod` and `rmmod` behind the scenes.

Examining hardware devices

Several utilities are useful for scoping out what hardware you actually have available. The tool `lspci` provides detailed information on findable PCI devices (including those over PCMCIA or USB buses, in many cases). Correspondingly `setpci` can configure devices on PCI buses. `lspnp` lists plug-and-play BIOS device nodes and resources. `lsusb` similarly examines USB devices (and has a `setnpn` to modify configurations).

Setting up an X11 server (part one)

Basically, X.Org (or XFree86) come with a whole lot of video drivers and other peripheral drivers; you need to choose the right ones to use. Ultimately, the configuration for an X server lives in the rather detailed, and somewhat cryptic, file `/etc/X11/xorg.conf` (or `xfree86.conf`). A couple standard utilities can be used for somewhat friendlier modification of this file, but ultimately a text editor works. Some frontends included with X.Org itself include `xorgcfg` for graphical configuration (assuming you have it working well enough to use that) and `xorgconfig` for a text-based setup tool. Many Linux distributions package friendlier frontends.

The tool `SuperProbe` is often useful in detecting the model of your video card. You may also consult the database `/usr/X11R6/lib/X11/Cards` for detailed information on supported video cards.

Setting up an X11 server (part two)

Within the configuration file `/etc/X11/xorg.conf`, you should create a series of "Section" ... "EndSection" blocks, each of which defines a number of details and options about a particular device. These section names consist of:

```
* Files:           File pathnames
* ServerFlags:    Server flags
* Module:         Dynamic module loading
* InputDevice:    Input device description
* Device:         Graphics device description
* VideoAdaptor:   Xv video adaptor description
* Monitor:        Monitor description
* Modes:          Video modes descriptions
* Screen:         Screen configuration
* ServerLayout:   Overall layout
* DRI:            DRI-specific configuration
* Vendor:         Vendor-specific configuration
```

Setting up an X11 server (part three)

Among the sections, `Screen` acts as a master configuration for the display system. For example, you might define several `Monitor` sections, but select the one actually used with:

```
Section "Screen"
    Identifier      "Default Screen"
    Device          "My Video Card"
    Monitor         "Current Monitor"
    DefaultDepth    24
    SubSection "Display:
        Depth        24
        Modes         "1280x1024" "1024x768" "800x600"
    EndSubSection
    # more subsections and directives
Endsection
```

The section named `ServerLayout` is the real "master" configuration -- it refers to both the `Screen` to use and to various `InputDevice` sections. But if you have a problem, it is almost always with selecting the right `Device` or `Monitor`. Fortunately, DPMS monitors nowadays usually obviate the need to set painful `Modeline` options (in the bad old days, you needed to locate very specific timings on your monitors scan rates; usually DPMS handles this for you now).

Section 4. Configuring PCMCIA devices

About PCMCIA

PCMCIA devices are also sometimes called PC-Card devices. These (thick) credit-card sized peripherals are generally designed to be easily hot-swappable and transportable and are used most widely in notebook computers. However, some desktop or server configurations also have PCMCIA readers, sometimes in an external chassis connected via one of several possible buses (a special PCI or ISA card, a USB translator, etc). A variety of peripherals have been created in PCMCIA form factor, including wireless and Ethernet adaptors, microdrives, flash drives, modems, SCSI adapters, and other special-purpose devices.

Technically, a PCMCIA interface is a layer used to connect to an underlying ISA or PCI bus. For the most part, the translation is transparent -- the very same kernel modules or other tools that communicate with an ISA or PCI device will be used to manage the same protocol provided via PCMCIA. The only real special issue with PCMCIA devices is recognition of the insertion event and of the card type whose drivers should load.

Nowadays, the PCMCIA peripheral packaging is being eclipsed by USB and/or Firewire devices. While PCMCIA is a bit more convenient as a physical form-factor (usually hiding cards in the machine chassis), USB is closer to universal on a range of machines. As a result, many devices that have been packaged as PCMCIA in the past might now be packaged as USB "dongle" style devices; these are more readily available at retail outlets.

Recognizing a PCMCIA device (part one)

In modern kernels -- 2.4 and above -- PCMCIA support is available as a kernel module. Modern distribution will include this support, but if you compile a custom kernel, include the options `CONFIG_HOTPLUG`, `CONFIG_PCMCIA`, and `CONFIG_CARDBUS`. The same support was previously available in separately in the `pcmcia-cs` package.

The modules `pcmcia_core` and `pcmcia` support loading PCMCIA devices. `yenta_socket` is also generally loaded to support the CardBus interface (PCI-over-PCMCIA):

```
% lsmod | egrep '(yenta)|(pcmcia)'  
pcmcia                21380  3  atmel_cs  
yenta_socket         19584  1  
pcmcia_core          53568  3  atmel_cs,pcmcia,yenta_socket
```

Once a card is inserted into a PCMCIA slot, the daemon `cardmgr` looks up a card in the database `/etc/pcmcia/config` then loads appropriate supporting drivers as needed.

Recognizing a PCMCIA device (part two)

Let us take a look at the PCMCIA recognition in action. I inserted a card into a Linux laptop with a PCMCIA slot and with the previously listed kernel module support. I can use the tool `cardctl` to see what information this peripheral provided:

```
% cardctl ident  
Socket 0:  
  product info: "Belkin", "11Mbps-Wireless-Notebook-Network-Adapter"  
  manfid: 0x01bf, 0x3302  
  function: 6 (network)
```

This information is provided by the `pcmcia_core` kernel module by querying the physical card. Once the identification is available, `cardmgr` scans the database to figure out what driver(s) to load. Something like:

```
% grep -C 1 '0x01bf,0x3302' /etc/pcmcia/config
card "Belkin FSD6020 v2"
  manfid 0x01bf,0x3302
  bind "atmel_cs"
```

In this case, we want the kernel module `atmel_cs` to support the wireless interface this card provides. You can see what actually got loaded by looking at either `/var/lib/pcmcia/stab` or `/var/run/stab`, depending on your system:

```
% cat /var/run/stab
Socket 0: Belkin FSD6020 v2
0      network atmel_cs      0      eth2
```

Debugging a PCMCIA device

In the above example, everything worked seamlessly. The card was recognized, drivers loaded, and the capabilities ready to go. That is the best case. If things are not as smooth, you might not find a driver to load.

If you are confident that your PCMCIA device can use an existing driver (for example, it is compatible with another chipset), you can manually run `insmod` to load the appropriate kernel module. Or if you use this card repeatedly, you can edit `/etc/pcmcia/config` to support your card, indicating the needed driver. However, guessing a needed module is unlikely to succeed -- you need to make sure your card really is compatible with some other known PCMCIA card.

Loading PCMCIA devices can be customized with the setup scripts in `/etc/pcmcia/`, each named for a category of function. For example, when an 802.11b card like the previous example loads, the script `/etc/pcmcia/wireless` runs. You can customize these scripts if your device has special setup requirements.

Using "schemes" for different configurations

If you need to use a PCMCIA device in multiple configurations, you may use the `cardctl scheme` command to set (or query) a configuration. For example:

```
% sudo cardctl scheme foo
checking: eth2
/sbin/ifdown: interface eth2 not configured
Changing scheme to 'foo'...
Ignoring unknown interface eth2=eth2.
% cardctl scheme
Current scheme: 'foo'.
```

In this case, I have not actually defined the `foo` scheme, but in general, if you

change a scheme, device reconfiguration is attempted. Schemes may be used in setup scripts by examining the `$ADDRESS` variable:

```
# in /etc/pcmcia/network.opts (called by /etc/pcmcia/network)
case "$ADDRESS" in
work,*,*,*)
    # definitions for network in work scheme ...
    ;;
default,*,*,*)
    # definitions for network in default scheme ...
    ;;
esac
```

You may of course, set schemes in initialization scripts or via other triggering events (in a `cron` job, via a GUI interface, etc.).

Section 5. Configuring Universal Serial Bus devices

About USB

As the section on PCMCIA mentioned, USB is somewhat newer technology that has largely eclipsed PCMCIA in importance. USB allows chaining of up to 127 devices on the same bus using a flexible radial topology of hubs and devices. USB comes in several versions with increasing speeds, the latest being 2.0. The latest USB version theoretically supports up to 480 MBsec. USB 1.1 supported the lower speed of 12 MBsec. In practice, there are a lot of reasons that particular devices might, in fact, operate much slower than these theoretical numbers -- but it is still a reasonably fast bus interface.

Recognizing a USB device (part one)

At an administrative level, USB works very similarly to PCMCIA. The kernel module is `usbcore`. Support is better in 2.4+ kernels than in earlier 2.2 kernels. Above the `usbcore` level, one of several kernel modules support: `uhci`, `uhci_hcp`, `ohci`, `ohci_hcp`, `ehci`, `ehci_hcp`. Exactly which kernel modules you need depends on the chipset your machine uses; and in the case of `ehci` whether it supports USB 2.0 high speed. Generally, if your machine support `ehci` (or `ehci_hcp`), you will also want a backward-compatible `ehci` module loaded. Brad Hards' "The Linux USB sub-system" contains details on exactly which chipsets support which kernel modules. For multiuse kernels, you should just compile in all the USB modules.

Assuming you get a kernel with the correct support, the hotplug subsystem should handle loading any drivers needed for a specific inserted USB device. The file `/proc/bus/usb/devices` contains detailed information on the currently available USB devices (both hubs and peripherals).

Recognizing a USB device (part two)

Normally the USB bus is mounted as a dynamically generated filesystem similar to the `/proc/` filesystem. The filesystem type is known as either `usbdevfs` or `usbfs`. Depending on your distribution, `/proc/bus/usb/` might get mounted either in initialization scripts such as `/etc/rcS.d/S02mountvirtfs` or via an `/etc/fstab` configuration. In the latter case, you might have a line like:

```
# /etc/fstab
none /proc/bus/usb usbdevfs defaults 0 0
```

Otherwise, an initialization script might run something like:

```
mount -t usbdevfs none /proc/bus/usb
```

The recognition of devices and control over the USB subsystem is contained in the `/etc/hotplug/`, especially within `/etc/hotplug/usb.rc` and `/etc/hotplug/usb.agent`. Inserting a USB device will `modprobe` a driver. You may customize the initialization of a device further by creating a `/etc/hotplug/usb/$DRIVER` script for your particular peripheral.

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Common threads: Advanced filesystem implementor's guide, Parts 1 - 13](#)" (developerWorks, starting June 2001) is an excellent series on Linux filesystems.
- "[Understanding Linux configuration files](#)" (developerWorks, December 2001) explains configuration files on a Linux system that control user permissions, system applications, daemons, services, and other administrative tasks in a multi-user, multi-tasking environment.
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the authors

David Mertz, Ph.D.

David Mertz is Turing complete, but probably would not pass the Turing Test. For more on his life, see his [personal Web page](#). He's been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#).

Brad Huntting

Brad has been doing UNIX® systems administration and network engineering for

about 14 years at several companies. He is currently working on a Ph.D. in Applied Mathematics at the University of Colorado in Boulder, and pays the bills by doing UNIX support for the Computer Science department.

LPI exam prep: Networking configuration

Intermediate Level Administration (LPIC-2) topic 205

Skill Level: Intermediate

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer
Gnosis Software

08 Nov 2005

This is the first of seven tutorials covering intermediate network administration on Linux®. In this tutorial, David Mertz shows you how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11), through the routing of network addresses. Higher level servers that may operate on these configured networks are covered in later tutorials.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 202: Tutorials and topics

LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic 205): Networking configuration	(This tutorial) Learn how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11), through the routing of network addresses. See detailed objectives below.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	Coming soon
Topic 207	LPI exam 202 prep (topic 207): DNS	Coming soon
Topic 208	LPI exam 202 prep (topic 208): Web services	Coming soon
Topic 210	LPI exam 202 prep (topic 210): Network client management	Coming soon
Topic 212	LPI exam 202 prep (topic 212): System security	Coming soon
Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	Coming soon

To start preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#). To prepare for the other exam in certification level 2, see the [developerWorks tutorials for LPI exam 201](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Networking configuration", the first of seven tutorials covering intermediate network administration on Linux. In this tutorial, you learn how to configure a basic TCP/IP network from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses. Higher level servers that may operate on these configured networks are covered in later tutorials.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Networking configuration: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
2.205.1 Basic networking configuration	Weight 5	Configure a network device to be able to connect to a local network and a wide-area network. This objective includes being able to communicate between various subnets within a single network, configure dialup access using mgetty, configure dialup access using a modem or ISDN, configure authentication protocols such as PAP and CHAP, and configure TCP/IP logging.
2.205.2 Advanced network configuration and troubleshooting	Weight 3	Configure a network device to implement various network authentication schemes. This objective includes configuring a multi-homed network device, configuring a virtual private network, and resolving networking and communication problems.

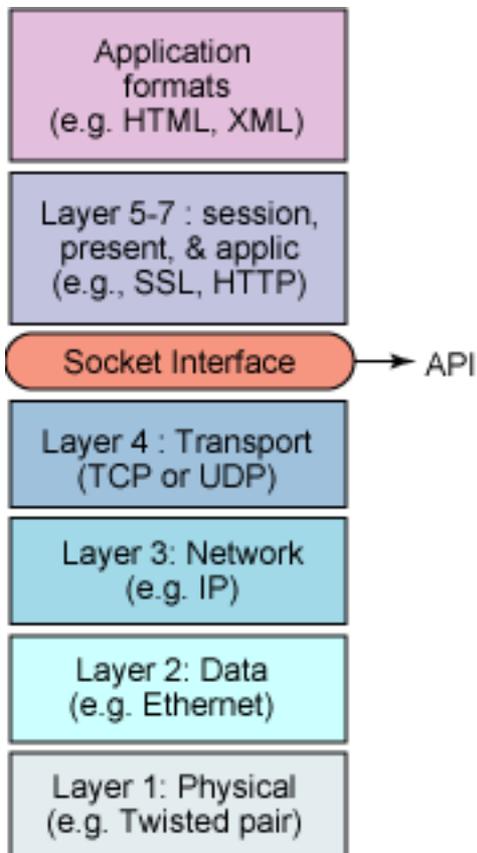
Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. About network configuration

When thinking about Linux networking and network configuration, it helps to keep in mind the *OSI seven-layer reference model*:

Figure 1. The OSI seven-layer reference model



What we call "network configuration" essentially lives on the second and third layers -- the *Data Link Layer* and the *Network Layer* -- and in the interfaces between them. In practice, this amounts to either Ethernet or serial interfaces like modems for the Data Link Layer, and the Internet Protocol (IP) for the Network Layer. Later tutorials in this series deal with higher-level layers, though most server applications discussed do not cleanly separate all seven layers (or even the top four where they operate).

The first network layer is the *Physical Layer*, the actual wires (or wireless channels) and circuits. A practical network administrator needs to be ready to inspect cabling and install new network peripherals from time to time (but those issues are outside the scope of these tutorials). Clearly, however, a loose wire, fried Ethernet card, or broken plug is just as capable of creating network problems as is misconfigured software.

Layer four is the *Transport Layer*, concretely, this means either TCP or UDP in IP networks. TCP and UDP are used at higher levels via the Berkeley Sockets Interface, a well-tested library found on all modern computer systems. For background on how applications (such as those discussed in later tutorials of this series) use TCP or UDP read the tutorials "[Programming Linux sockets, Part 1](#)" and "[Programming Linux sockets, Part 2](#)."

Other resources

As with most Linux tools, the manpages contain valuable information. For more in-depth information, the Linux Documentation Project has a variety of useful documents, especially its HOWTOs. Many books on Linux networking, such as O'Reilly's *TCP/IP Network Administration* by Craig Hunt, can be quite helpful. You'll find links to these and more resources in this tutorial's [Resources](#) section.

Section 3. Basic networking configuration

Address resolution protocol

The first thing to understand about Ethernet devices -- either 802.11 a/b/g wireless or the more traditional CAT5/CAT6 wired networks -- is that every Ethernet device has a unique six-byte ID in it. Those IDs are assigned in blocks to manufacturers; you can look up the Ether number assignments at IANA. Ethernet generally "just works" at the hardware level, but a system needs to map an Ethernet ID to the IP address it will use to enable IP networking.

The Address Resolution Protocol (ARP) lets machines discover each others' IP address within a local Ethernet network. As a protocol, ARP is generally implemented within network device drivers (kernel modules); the arp tool lets you examine the status of the ARP system and tweak it in some ways. At this point, we assume that each machine has been configured to know its own IP address, either by static assignment or dynamically with DHCP.

When a Linux system (or any device with Ethernet) wishes to address an IP address, the ARP request message "who is X.X.X.X tell Y.Y.Y.Y" is sent using the Ethernet broadcast address. The target system forms an ARP response "X.X.X.X is hh:hh:hh:hh:hh:hh" and sends it to the requesting host. An ARP response is cached for a short time in `/proc/net/arp` to avoid the need to continually reestablish the mapping between hardware Ethernet addresses and IP addresses.

Gorry Fairhurst provides a nice description of ARP (see [Resources](#)).

The arp utility

The Linux utility arp lets you examine and modify the status of ARP mappings. A

simple status report might look like Listing 1:

Listing 1. ARP status report

```
$ arp -n
Address          HWtype  HWaddress          Flags Mask  Iface
192.168.2.1     ether   00:03:2F:09:61:C7  C          eth0
```

This tells you the specific hardware device assigned to IP address 192.168.2.1 on this network (the number used is suggestive of a router/gateway, which it is, in this case). The fact that only this single mapping is listed does not necessarily mean that no other devices exist on the local network, but simply that the ARP records for other devices have expired. ARP expires records after a short time -- on the order of minutes rather than seconds or hours -- to allow networks to reconfigure themselves if hosts are added or removed or if settings are changed on machines. By caching an ARP record for a short time, a new request should not be necessary during most client/server application sessions.

Any sort of IP request on a host that may be on the local network causes the kernel to send out an ARP request; if an ARP reply is received, add the host to the ARP cache (as in Listing 2):

Listing 2. Interacting with additional IP addresses

```
$ ping -c 1 192.168.2.101 > /dev/null
$ ping -c 1 192.168.2.101 > /dev/null
$ ping -c 1 192.168.2.102 > /dev/null
$ ping -c 1 192.168.32.32 > /dev/null
$ ping -c 1 192.168.32.32 > /dev/null
$ arp -n
Address          HWtype  HWaddress          Flags Mask  Iface
192.168.2.1     ether   00:03:2F:09:61:C7  C          eth0
192.168.2.101   ether   00:30:65:2C:01:11  C          eth0
192.168.2.100   ether   00:11:24:9D:1E:4B  C          eth0
192.168.2.102   ether   00:48:54:83:82:AD  C          eth0
```

In this case, the first four addresses really exist on the local Ethernet network, but 192.168.32.32 does not, so no ARP reply is received. Notice also that addresses you may succeed in connecting to via non-local routing will also not cause anything to be added to the ARP cache (see Listing 3):

Listing 3. Nothing added to the ARP cache

```
$ ping -c 1 google.com
PING google.com (216.239.57.99) 56(84) bytes of data.
64 bytes from 216.239.57.99: icmp_seq=1 ttl=235 time=109 ms
--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 109.123/109.123/109.123/0.000 ms
$ arp -n
Address          HWtype  HWaddress          Flags Mask  Iface
```

```
192.168.2.1 ether 00:03:2F:09:61:C7 C eth0
```

Google is *reachable* (because routing is already configured), but 216.239.57.99 is non-local and is not added to ARP. The seventh tutorial in this series, on topic 214, deals with network troubleshooting and demonstrates manually setting ARP.

PPP, PAP, and CHAP

Point-to-Point Protocol (PPP) is used to establish Internet links over dial-up modems, direct serial connections, DSL, and other types of point-to-point links (sometimes including PPPoE as a "pseudo-layer over Ethernet -- more of a handshake protocol). The `pppd` daemon works together with the kernel PPP driver to establish and maintain a PPP link with another system (called the peer) and to negotiate Internet Protocol (IP) addresses for each end of the link.

PPP, specifically `pppd`, authenticates its peer and/or supplies authentication information to the peer. Such authentication is performed using either the simple password system Password Authentication Protocol (PAP) or the per-session Challenge Handshake Authentication Protocol (CHAP). Of the two, CHAP is more secure if both ends support it.

Options for PPP in general are stored in `/etc/ppp/options`. Configuration of PAP is via the PAP secrets file `/etc/ppp/pap-secrets`; for CHAP it is in the CHAP secrets file `/etc/ppp/chap-secrets`.

The PAP/CHAP secrets file

The `/etc/ppp/pap-secrets` file contains whitespace-separated fields for *client*, *server*, *secret*, and *acceptable local IP address*. The last may be blank (and generally is for dynamic IP allocation). The PAP secrets file should be configured correspondingly for each peer. Even though PPP is a peer protocol, for connection purposes we call the requesting machine the client and the waiting machine the server. For example, the machine `bacchus` on my network might have a configuration like:

Listing 4. Configuring pap-secrets on the bacchus

```
# Every regular user can use PPP and uses passwords from /etc/passwd
# INBOUND connections
# client  server  secret          acceptable local IP addresses
*         bacchus  ""             *
chaos    bacchus  chaos-password
# OUTBOUND connections
bacchus  *        bacchus-password
```

The machine `bacchus` will accept connections claiming to be any regular user or

also claiming to be the machine chaos (and demanding the password `chaos-password` in the latter case). To other machines, bacchus will simply use its own name and offer the password `bacchus-password` to every peer.

Correspondingly, the machine chaos on my network might have:

Listing 5. Machine chaos makes more conservative connection choices

```
# client  server  secret  acceptable local IP addresses
chaos    bacchus chaos-password
bacchus  chaos   bacchus-password
```

Machine chaos is more conservative regarding to whom it will connect. It is willing to exchange credentials only with bacchus. You may configure each `/etc/ppp/options` file to decide if credentials are demanded, though.

Using CHAP secrets requires that you allow for both peer machines to authenticate each other. As long as bi-direction authentication is configured in PAP secrets, a CHAP secrets file may look just the same as the above examples.

Connecting with mgetty

The PAP secrets file can be used with the `AUTO_PPP` function of mgetty. mgetty 0.99+ is preconfigured to start up pppd with the `login` option. This tells pppd to consult `/etc/passwd` (and `/etc/shadow` in turn) after a user has passed this file.

In general, a getty program may be configured to allow connections from serial devices, including modems and direct serial ports. For example, for a hard-wired line or a console tty, you might run:

```
/sbin/getty 9600 ttyS1
```

in your inittab. For a old style dial-in line with a 9600/2400/1200 baud modem:

```
/sbin/getty -mt60 ttyS1 9600,2400,1200.
```

Configuring routing

In the discussion of Address Resolution Protocol, we saw how IP addresses are assigned within a local network. However, to communicate with machines outside of a local network, it is necessary to have a *gateway/router*. Basically, a gateway is simply a machine that connects to more than one network and can therefore take packets transmitted within one network and re-transmit them on other networks it is connected to. This is where the name "Internet" comes from: It is a "network of

networks" in which every gateway can eventually reach every other network that is said to be "on the Internet."

The fifth tutorial in this series, on topic 210, deals with network client management, and discusses DHCP. DHCP will assign both client IP addresses and gateway addresses. However, with a fixed IP address on a client or in debugging situations, the Linux command `route` allows you to view and modify routing tables. The newer command `ip` also lets you modify routing tables using a somewhat more powerful syntax.

A routing table simply lets you determine which gateway or host to send a packet to, given a specific pattern in the address. An address pattern is specified by combining an address with a *subnet mask*. A subnet mask is a bit pattern, usually represented in dotted quad notation, that tells the kernel which bits of a destination to treat as the *network address* and which remaining bits to treat as a *subnet*. The command `ip` can accept the simpler `/NN` format for bitmasks. In general, in a mask and address, zero bits are "wildcards."

For example, a simple network with a single external gateway is likely to have a routing table similar to Listing 6:

Listing 6. Typical simple routing table

```
$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
0.0.0.0 192.168.2.1 0.0.0.0 UG 0 0 0 eth0
```

What this means is simply that any IP address that matches "192.168.2.*" is on the local network and will be delivered directly to the proper host (resolved with ARP). Any other address will be sent to the gateway "192.168.2.1", which will be required to forward a packet appropriately. The machine 192.168.2.1 must be connected to one or more external networks.

However, for a more complex case, you may route specific patterns differently. In an invented example, let's say that you want to route specific `/16` addresses through other gateways. You could do what is shown in Listing 7:

Listing 7. Changing routing on `/16` networks

```
$ route add -net 216.109.0.0 netmask 255.255.0.0 gw 192.168.2.2
$ route add -net 216.239.0.0 netmask 255.255.0.0 gw 192.168.2.3
$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
216.109.0.0 192.168.2.2 255.255.0.0 UG 0 0 0 eth0
216.239.0.0 192.168.2.3 255.255.0.0 UG 0 0 0 eth0
```

```
0.0.0.0      192.168.2.1  0.0.0.0      UG  0      0      0 eth0
```

Addresses of the form "216.109.*" and "216.239.*" will now be routed through the gateways 192.168.2.2 and 192.168.2.3, respectively (both on the local network themselves). Addresses that are local, or outside the pattern spaces given, will be routed as before. You can use the command `route delete` correspondingly to remove routes.

Section 4. Advanced network configuration and troubleshooting

About network utilities

Linux comes with a number of standard utilities you can use to customize and troubleshoot a network configuration. Although much of Linux networking code lives in the kernel itself, almost everything about the behavior of networks is configurable using command-line utilities. Many distributions also come with higher level and/or graphical high-level configuration tools, but those do not contain anything that can not be performed and scripted using the command-line tools.

The ping utility

The most basic way to check whether a Linux host has access to an IP address (or to a named host, once DNS and/or `/etc/hosts` is configured) is with the utility **ping**. `ping` operates at the basic IP layer and it does not rely on the Data Link Layer like TCP or IP. `ping` instead uses the Internet Control Message Protocol (ICMP). If you cannot reach a host with `ping`, you can assume you will not reach it with any other tool, so `ping` is always the first step in establishing whether a connection to a host is available (man `ping` has details on options).

By default, `ping` sends a message every one second until cancelled, but you may change timings, limit message count, and output details. When `ping` is run, it returns some details on round-trip time and dropped packages, but for the most part either you can ping a host or you cannot. Listing 8 gives some examples:

Listing 8. Local and non-local ping examples

```
$ ping -c 2 -i 2 google.com
```

```

PING google.com (216.239.37.99): 56 data bytes
64 bytes from 216.239.37.99: icmp_seq=0 ttl=237 time=43.861 ms
64 bytes from 216.239.37.99: icmp_seq=1 ttl=237 time=36.956 ms

--- google.com ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 36.956/40.408/43.861 ms

$ ping 192.168.2.102
PING 192.168.2.102 (192.168.2.102): 56 data bytes
64 bytes from 192.168.2.102: icmp_seq=0 ttl=255 time=4.64 ms
64 bytes from 192.168.2.102: icmp_seq=1 ttl=255 time=2.176 ms
^C
--- 192.168.2.102 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 2.176/3.408/4.64 ms

```

The ifconfig utility

Network interfaces are configured with the tool **ifconfig**. Usually, this is run as part of the initialization process, but in some cases interfaces may be modified and tuned later (especially for debugging). If you run `ifconfig` with no switches, it displays the current status. You may use the forms `ifconfig <interface> up` and `ifconfig <interface> down` to start and stop network interfaces. Some other switches change the display or limit the display to specific interfaces. See `man ifconfig` for more details.

An informational display might look something like Listing 9:

Listing 9. Using ifconfig to examine network interfaces

```

$ ifconfig
eth0  Link encap:Ethernet  HWaddr 00:12:F0:21:4C:F8
      inet addr:192.168.2.103  Bcast:192.168.2.255  Mask:255.255.255.0
      inet6 addr: fe80::212:f0ff:fe21:4cf8/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:540 errors:0 dropped:0 overruns:0 frame:0
      TX packets:233 errors:0 dropped:0 overruns:0 carrier:1
      collisions:0 txqueuelen:1000
      RX bytes:49600 (48.4 KiB)  TX bytes:42067 (41.0 KiB)
      Interrupt:21 Base address:0xc000 Memory:ffcf000-ffcf0fff

ppp0  Link encap:Point-Point Protocol
      inet addr:10.144.153.104  P-t-P:10.144.153.51  Mask:255.255.255.0
      UP POINTOPOINT RUNNING  MTU:552  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0
      TX packets:0 errors:0 dropped:0 overruns:0

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:4043 errors:0 dropped:0 overruns:0 frame:0
      TX packets:4043 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:368044 (359.4 KiB)  TX bytes:368044 (359.4 KiB)

```

In this display, two networks are configured, one on Ethernet and one on PPP (plus the local loopback). In other cases, you might have multiple Ethernet interfaces configured or other interface types. If so, the system is called *multi-homed*.

The netstat utility

Linux utilities tend to overlap in functionality. The tool **netstat** displays information that may also be provided by several utilities, such as `ifconfig` and `route`. You may also find extensive general statistics on network activity. For example:

Listing 10. Network statistics report

```
$ netstat -s
Ip:
  12317 total packets received
  0 forwarded
  0 incoming packets discarded
  12255 incoming packets delivered
  11978 requests sent out
Icmp:
  1 ICMP messages received
  0 input ICMP message failed.
  ICMP input histogram:
    echo replies: 1
  0 ICMP messages sent
  0 ICMP messages failed
  ICMP output histogram:
Tcp:
  7 active connections openings
  5 passive connection openings
  0 failed connection attempts
  0 connection resets received
  3 connections established
  11987 segments received
  11885 segments send out
  0 segments retransmitted
  0 bad segments received.
  3 resets sent
Udp:
  101 packets received
  0 packets to unknown port received.
  0 packet receive errors
  92 packets sent
TcpExt:
  1 TCP sockets finished time wait in fast timer
  1490 delayed acks sent
  Quick ack mode was activated 5 times
  3632 packets directly queued to recvmsg prequeue.
  126114 of bytes directly received from backlog
  161977 of bytes directly received from prequeue
  1751 packet headers predicted
  3469 packets header predicted and directly queued to user
  17 acknowledgments not containing data received
  4696 predicted acknowledgments
  0 TCP data loss events
```

Other utilities

There are several other utilities you should be aware of for network configuration. As usual, their respective manpages contain full usage information. Detailed discussion of these is in the seventh tutorial in this series, on topic 214, which deals with network troubleshooting.

tcpdump lets you monitor all the packets that pass through network interfaces, optionally limited to particular interfaces or filtered on various criteria. Often saving this packet summary information, then filtering or summarizing it with text-processing tools, is useful for debugging network problem. For example, you can examine packets communicated with a particular remote host.

lsof lists open files on a running Linux system. But in particular, you may use the `lsof -i` option to examine only the pseudo files for a particular IP connection or for network connections in general. For example:

Listing 11. Using `lsof` to examine pseudo files for connections

```
$ lsof -i
COMMAND      PID USER   FD   TYPE DEVICE SIZE NODE
      NAME
vino-serv 7812 dgm    33u  IPv4  12824
      TCP *:5900 (LISTEN)
gnome-cup 7832 dgm    18u  IPv4  12865
      TCP localhost.localdomain:32771->localhost.localdomain:ipp (ESTABLISHED)
telnet     8909 dgm     3u  IPv4  15771
      TCP 192.168.2.103:32777->192.168.2.102:telnet (ESTABLISHED)
```

nc and **netcat** are aliases. **netcat** is a simple UNIX utility that reads and writes data across network connections using TCP or UDP protocol. It is a "back-end" tool that can be used directly or driven by other programs and scripts. In many respects, **netcat** is similar to **telnet**, but is more versatile in allowing UDP interaction and sending unfiltered binary data.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Read Kwan Lowe's [Kernel Rebuild Guide](#) for more details on building a kernel.
- Learn how applications use TCP or UDP in these tutorials:
 - "[Programming Linux sockets, Part 1](#)" (developerWorks, October 2003)
 - "[Programming Linux sockets, Part 2](#)" (developerWorks, January 2004)
- *TCP/IP Network Administration, Third Edition* by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- Find [Ether number assignments](#) at IANA.
- Gorry Fairhurst provides a [nice description of ARP](#).
- Many Linux User Groups have local and distance study groups for LPI exams -- here's a list or more than [700 LUGs around the world](#).
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).

Get products and technologies

- Get the Linux kernel source at the [Linux Kernel Archives](#).
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

David Mertz, Ph.D.

David Mertz has been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#) . For more on David, see his [personal Web page](#).

LPI exam prep: Mail and news

Intermediate Level Administration (LPIC-2) topic 206

Skill Level: Intermediate

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer
Gnosis Software

22 Nov 2005

This is the second of seven tutorials covering intermediate network administration on Linux®. In this tutorial, David Mertz discusses how to use Linux as a mail server and as a news server. Overall, e-mail is probably the main use of the Internet, and Linux is perhaps the best platform for e-mail services. This tutorial covers mail transport, local mail filtering, and mailing list maintenance software. It also briefly discusses server software for the NNTP protocol.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 202: Tutorials and topics

LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic 205): Networking configuration	Learn how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	(This tutorial) Learn how to use Linux as a mail server and as a news server. Learn about mail transport, local mail filtering, mailing list maintenance software, and server software for the NNTP protocol. See detailed objectives below.
Topic 207	LPI exam 202 prep (topic 207): DNS	Coming soon
Topic 208	LPI exam 202 prep (topic 208): Web services	Coming soon
Topic 210	LPI exam 202 prep (topic 210): Network client management	Coming soon
Topic 212	LPI exam 202 prep (topic 212): System security	Coming soon
Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	Coming soon

To start preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#). To prepare for the other exam in certification level 2, see the [developerWorks tutorials for LPI exam 201](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Mail and news," the second of seven tutorials covering intermediate network administration on Linux. In this tutorial, you learn how to use Linux as a mail

server and as a news server. This tutorial covers mail transport, local mail filtering, and mailing list maintenance software. It also briefly discusses server software for the NNTP protocol.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Mail and news: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
2.206.1 Configuring mailing lists	Weight 1	Install and maintain mailing lists using Majordomo. Monitor Majordomo problems by viewing Majordomo logs.
2.206.2 Using Sendmail	Weight 4	Manage a Sendmail configuration including e-mail aliases, mail quotas, and virtual mail domains. This objective includes configuring internal mail relays and monitoring SMTP servers.
2.206.3 Managing mail traffic	Weight 3	Implement client mail management software to filter, sort, and monitor incoming user mail. This objective includes using software such as Procmail on both the server and client side.
2.206.4 Serving news	Weight 1	Install and configure news servers using INN. This objective includes customizing and monitoring served newsgroups.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. About mail and news

The broad use of Linux for mail and news servers has led to the development of improved tools over time. When the LPI certification exams were developed, the most popular tools were *Sendmail* for mail transport, *Procmail* for local mail handling, *Majordomo* for mailing lists, and *innd* (InterNetNews daemon) for NNTP. The last of these is still probably the default choice for news; however, despite its technical strengths, the NNTP protocol has been somewhat eclipsed by e-mail mailing lists and Web-based discussion forums.

Of the other tools, Sendmail and Procmail are still widely used, although not as ubiquitously as they once were. The most popular upgrade or replacement for Sendmail is postfix, which contains facilities for backwards compatibility with Sendmail. The local mail handling field is well populated with choices, but Procmail is still popular. On the other hand, Majordomo is a slight anachronism nowadays. Just as Majordomo largely replaced the earlier listserv software, mailman has more recently eclipsed Majordomo. However, to match the current LPI topic areas, this tutorial discusses Majordomo.

Other resources

As with most Linux tools, it is always useful to examine the man pages for any utilities discussed. Versions and switches might change between utility or kernel version or with different Linux distributions. For more in-depth information, the Linux Documentation Project has a variety of useful documents, especially its HOWTOs. See the [Resources](#) section for a link. A variety of books on Linux networking have been published; I have found O'Reilly's *TCP/IP Network Administration*, by Craig Hunt, to be quite helpful (find whatever edition is most current when you read this; see [Resources](#) for a link).

Section 3. Configuring mailing lists

What does Majordomo do?

A mailing list manager program is basically a local extension for a mail transport program (MTA) such as Sendmail. Basically, the MTA running on a system passes off a set of addresses to the control of the mailing list manager, and the mailing list manager modifies, processes, and perhaps re-mails the messages it receives. Some messages received by a mailing list manager are messages meant for distribution to the mailing list itself (perhaps needing to be verified for permission to distribute to the list(s)). Other messages are control messages that change the status of the

mailing list, such as the subscription options of a particular subscriber. A mailing list manager does not perform mail delivery itself, but passes that function to its supporting MTA.

As the introduction to this tutorial stated, Majordomo is not currently the state-of-the-art choice for mailing lists. Rather, the best choice for a new installation of a mailing list is probably Mailman. Majordomo, however, is still perfectly functional and is installed on many older systems which continue to operate without problem (sometimes supporting lists that have been operational for many years).

There is a wrinkle with Majordomo versions, however. Some years ago, a rewrite of the Majordomo 1.x series was started, called Majordomo2. Unfortunately, that rewrite fizzled out without ever reaching release status. While Majordomo2 (in a beta version) may be used in a very small number of systems, Majordomo 1.9.5 is the most recent stable version and is the version discussed in this tutorial.

Installing Majordomo

You can obtain an archive of the Majordomo software at the Majordomo site (see [Resources](#) for a link).

After unpacking a file that will be named something like `majordomo-1.94.5.tgz`, be sure to read the `INSTALL` file carefully. You need to follow all the steps it describes for getting a working Majordomo system. Building the system uses the usual `make`; `make install` steps of most source installs, as well as `make install-wrapper`. The install can and should verify itself with a command like `cd /usr/local/majordomo-1.94.5; ./wrapper config-test` (the `make install` provides details in a message).

Before building, modify the Makefile and create and/or modify `majordomo.cf`. As a starting point, you can copy the latter file from `sample.cf` in the source distribution. In the Makefile, a number of environment variables are set, but the most critical and subtle of these is probably `W_GROUP`. This is the *numeric* gid of the group Majordomo will run under, almost always the group "daemon." The gid for daemon is 1 on most systems, but be sure to check using the following:

```
$ id daemon
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

Other variables in Makefile include `PERL` for the path to the interpreter, and `W_HOME` for the location where Majordomo will be installed.

Your new `majordomo.cf` file also needs to be edited before the `make install`. The Perl variables that need to be modified appear mainly near the top of the file. Definitely adjust `$whereami` and `$homedir`, and examine the others to make sure

they are sensible.

Telling Sendmail to use Majordomo

The final step in installation is convincing Sendmail to talk with Majordomo. Within the `/etc/sendmail.cf` file, this involves a line like this:

```
OA/path/to/majordomo/majordomo.aliases
```

If you use the M4 processor to generate Sendmail configuration files, you can use a line like this:

```
define(`ALIAS_FILE',`/etc/aliases,/path/to/majordomo/majordomo.aliases')
```

The sample `majordomo.aliases` contains some sample values:

Listing 1. Sample `majordomo.aliases`

```
majordomo:  "|/usr/test/majordomo-1.94.5/wrapper majordomo"
majordomo-owner:  you
owner-majordomo:  you
test:           "|/usr/test/majordomo-1.94.5/wrapper resend -l test test-list"
test-list:      :include:/usr/test/majordomo-1.94.5/lists/test
owner-test:     you
test-owner:     you
test-request:   you
```

These, of course, need to be customized for your particular setup. In particular, "you" means the name of the list administrator (who is not necessarily the overall system administrator).

Creating a new Majordomo list

The sample setup given above created a list called "test," with addresses for "test-owner," "test-request," etc. for administering the list. In real use, you will probably want lists with other names. To accomplish that, do the following:

1. Switch to the directory `$listdir`, as defined in `majordomo.cf`.
2. Create files called `my-list-name` and `my-list-name.info` (adjust appropriately); `chmod` them to 664. The latter file contains an introduction to the list.
3. Create several aliases in your `majordomo.aliases` file, following the pattern of the "test" examples -- for example, "foo-owner," "foo,"

"foo-request," and so on.

4. Send requests to `subscribe`, `unsubscribe`, `signoff`, and so on, for members of the list.
5. Create an archive directory in the location specified by the `$filedir` and `$filedir_suffix` variables.
6. Create a digest subdirectory under `$digest_work_dir`. Use the same name as the digest list (for example: `test-digest`).
7. Make sure everything is owned by user `majordomo`, group `majordomo`, and is writable by both owner and group (in other words, mode `664` for files and mode `775` for directories).
8. Issue a `config <listname> <listname>admin` command to Majordomo. This will cause it to create a default configuration file for the list, and send it back to you.

Section 4. Using Sendmail

What does Sendmail do?

Sendmail is a Mail Transport Agent (MTA). It routes, modifies, and delivers mail message across heterogeneous mail systems. With a history somewhat parallel to that of mailing list software, Sendmail has a "permanent beta" version called Sendmail X that is intended as an upgrade/replacement for the stable Sendmail 8.x series; however, much as Mailman has largely supplanted Majordomo, several MTAs have partially eclipsed Sendmail. The chief such new MTA is Postfix, but Qmail and Exim are also widely used. Nonetheless, Sendmail still remains, at least by a narrow margin, the most widely used MTA on Linux systems. As of September 16, 2005, the latest stable release of Sendmail was 8.13.5.

Not just one book, but many books, have been written on Sendmail. See [Resources](#) for a list of available books. The most comprehensive of these is *Sendmail, Third Edition* (O'Reilly, 2002) by Bryan Costales with Eric Allman. At 1,232 pages, this book covers quite a lot more than this tutorial can touch on.

While Sendmail in principle supports a number of mail transport protocols such as UUCP, by far the most widely used is Simple Mail Transport Protocol (SMTP), which here includes Extended SMTP (ESMTP) for enhanced MIME encoded message

bodies. At heart, mail that is not forwarded to other SMTP hosts is delivered to the local system by putting messages in local files. Local Mail User Agents (MUAs) read messages that Sendmail (or another MTA) puts in local files (and often also fetch mail using POP3 or IMAP), but generally call on Sendmail to deliver outgoing messages. Some MUAs, however, directly communicate with SMTP servers (such as Sendmail instances, local or remote) rather than placing messages in the Sendmail queue for later processing. Usually the Sendmail queue is in `/var/spool/mqueue/`.

Installing Sendmail

The first thing to do is obtain a copy of the current Sendmail software from sendmail.org (see [Resources](#) for a link), for example, `sendmail.8.13.5.tar.gz`. Unpack it as usual. Unlike many applications that use the `make; make install` pattern, building Sendmail is performed with `sh Build`. After the initial build, `cd` to the `cf/cf/` subdirectory; copy a suitable `*.mc` file as `sendmail.mc`; customize `sendmail.mc`; and run the following to generate a `sendmail.cf` file:

```
$ m4 ../m4/cf.m4 sendmail.mc > sendmail.cf
```

You may also use the shortcut `sh Build sendmail.cf`. This may seem mysterious, but both these commands generate an actual Sendmail configuration from a more readable format using the M4 macro processor. Actual `sendmail.cf` files, though editable ASCII, are quite cryptic and should only be modified minimally by hand.

Finally, copy the `sendmail` binary from a location that will be something like `obj.Linux.2.6.10-5-386.i686/sendmail/sendmail` to its final location (back up an old one if it exists), typically `/usr/sbin/`, and copy your `sendmail.cf` file to `/etc/mail/sendmail.cf`. You can also do the latter in the `cf/cf/` subdirectory with `sh Build install-cf`. You will probably need to `su` or `sudo` to obtain file permissions for the relevant directories.

A number of utilities come with Sendmail: `makemap`, `mailstats`, etc. Each corresponding directory has a `README` and can be installed by running `sh Build install` from the subdirectory.

The `sendmail.cf` file

The main complexity, and the main function, of Sendmail is in its `sendmail.cf` file. This configuration file contains some settings for the Sendmail environment, but principally it contains patterns for addresses to rewrite and/or deliver by certain mechanisms.

Two rewrite mechanism that may be configured are the `genericstable` and `virtusertable`, which let you map local users to and from external addresses. For either mapping, you first create an aliases file as plain text. For example:

Listing 2. Outbound mappings

```
david          david.mertz@gmail.com
root          root@gnosis.cx
dqm@gnosis.lan david.mertz@gmail.com
```

Or, for incoming mail mapped to local accounts:

Listing 3. Inbound mappings

```
david@mail.gnosis.cx      david
david@smtp.gnosis.cx     david
david@otherdomain.net    david
@mail.gnosis.cx          %1@external-host.com
owner@list.gnosis.cx     owner%3
jax@bar.com              error:5.7.0:550 Address invalid
```

To compile these aliases, use the `makemap` utility:

```
$ makemap dbm /etc/mail/virtusertable < inbound
$ makemap hash /etc/mail/genericstable < outbound
```

Enabling use of these maps can be configured using M4 macros in `sendmail.cf` (or in whatever configuration file you use).

Listing 4. Enabling mappings in `sendmail.cf`

```
DOMAIN(gnosis.cx)dnl
FEATURE(`virtusertable', `dbm /etc/mail/virtusertable')dnl
FEATURE(`genericstable', `hash /etc/mail/genericstable')dnl
GENERIC_DOMAIN_FILE(`/etc/mail/generics-domains')dnl
```

A number of things are going on here. The `DOMAIN` macro indicates that a file like `cf/domain/gnosis.cx.m4` is used for additional macros. The `FEATURE` macros enable use of the `virtusertable` and `genericstable`. The `GENERIC_DOMAIN_FILE` macro defines the domains that qualify for remapping for names in `genericstable`.

Rewriting will follow all the rules indicated. In test mode (`sendmail -bt`), you can examine the rewriting that is performed for specific addresses. For example, using `genericstable`, mail to the local user "david" will be delivered to `david.mertz@gmail.com` externally. Assuming `localhost` is defined in `/etc/mail/generics-domains`, mail to `david@localhost` will go to the same place.

In the other direction, mail coming in for david@mail.gnosis.cx will be rewritten and delivered to local user "david." Multiple domains can be manipulated by Sendmail at the same time, so david@otherdomain.net will also be delivered locally.

The full power comes in some of the wildcard symbols. Any mail sent to mail.gnosis.cx that is not specifically directed to a local user will be forwarded to the same username at external-host.com. But that's a simple pattern. More interestingly, the %3 can be used to expand multiple extra name information, so owner-foo@list.gnosis.cx and owner-bar@list.gnosis.cx will be delivered to local users "owner-foo" and "owner-bar," respectively (unless they undergo further rewriting). These local users might be mailing list processing systems or other automated message handlers. As a special case, you can raise an error for a given address rather than rewrite it further.

What we have looked at so far really just scratches the surface of the rewriting rules you can add to Sendmail, but they give you an initial feel. Buy one of the large books on the topic to learn more details.

Running Sendmail

Sendmail can run in a number of modes. The most common mode is as a daemon that stays in the background and periodically process its queue. For example, running:

```
$ /usr/sbin/sendmail -bd -q10m
```

tells Sendmail to run as a daemon and check its queue every ten minutes. You can also run Sendmail a single time to process the queue at once, but not daemonize:

```
$ /usr/sbin/sendmail -q
```

As mentioned above, Sendmail has a test mode to examine address rewriting rules. For example (taken from the *Linux Network Administrators Guide*; see [Resources](#) for a link):

Listing 5. Sendmail test mode

```
$ /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vstout.vbrew.com
rewrite: ruleset 3 input: isaac @ vstout . vbrew . com
rewrite: ruleset 96 input: isaac < @ vstout . vbrew . com >
rewrite: ruleset 96 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 3 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 0 input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199 input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 98 input: isaac < @ vstout . vbrew . com . >
```

```
rewrite: ruleset 98 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198 input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset 0 returns: $# local $: isaac
```

Section 5. Managing e-mail traffic

What does Procmail do?

Procmail is a mail processor. Basically, once Sendmail or another MTA has delivered mail to a local mailbox, you might use an MUA to process the mail in your inbox. You save some messages to various folders; you delete others; you forward other messages to various interested parties; you reply to others; and so on. Doing these tasks in an MUA is a manual and interactive process, and is potentially time consuming.

Procmail is a program that can do these tasks for you automatically whenever the required processing can be stated in a rule-driven way. Obviously, when you write back to your mother about her personal e-mail, some personal attention is required; but for a large class of other messages, it is useful to specify in advance exactly what you would like to happen when a given message is received. The rules that can drive automated message handling might involve specific pattern-based header fields, certain contents in a message body, or even calls out to more specific and specialized external programs such as statistical spam filters.

Enabling Procmail

Procmail probably came installed with your Linux distribution. If not, you can obtain the source archive at procmail.org (see [Resources](#)). As of this writing, the latest version is 3.22. You may also be able to install Procmail as a binary using the install system of your Linux distribution (for example, on Debian: `apt-get install procmail`). Building from source is a straightforward `make install`. All Procmail needs to operate is the procmail binary and a `~/.procmailrc` configuration file (or possibly a global `/etc/procmailrc` file).

Beyond installing Procmail in the first place, you need to get your local mail system to utilize Procmail. An older mechanism to process mail through Procmail is to use a `.forward` file; this will still often work on a per-user basis. Usually, a user will create a file, `~/.forward`, that contains something like this:

```
|/usr/local/bin/procmail
```

This will pipe each incoming message to Procmail. However, a better and more common way to utilize Procmail is to tell your MTA to talk directly to Procmail in the first place. In Sendmail, this involves enabling the `local_procmail` feature by putting the following in your `sendmail.mc` file:

```
FEATURE(`local_procmail', `/usr/bin/procmail', `procmail -Y -a  
$h -d $u')
```

Once Procmail is enabled, it needs a file, `~/.procmailrc`, which contains the set of rules it processes in handling a given message. Procmail is not a daemon, but rather a text processing tool that accepts exactly one e-mail message at a time via STDIN.

Rules in `~/.procmailrc`

At heart, Procmail is just a set of regular expression recipes. You may also define environment variables in the same fashion as in a shell script. Recipes are executed in order, but flags may be used to execute a particular condition only if the prior condition is satisfied (**A**) or is not satisfied (**E**). Some Procmail recipes are delivery recipes, and others are non-delivery recipes; the former terminates processing of a given message, unless the `c` flag is given to explicitly continue processing. Probably the most common action of a recipe is to store a message in a named mailbox, but you may also pipe a message to another program or forward the message to a list of addresses.

A recipe usually starts with a lock (optionally with a specific lock file; otherwise, it is done automatically) and some flags, followed by some rules, and then by exactly one action. For example:

```
:0 [flags] [ : [locallockfile] ]  
<zero or more conditions (one per line)>  
<exactly one action line>
```

Of particular note are the implied flag **H** to match the header and **B** to match the body. Patterns normally are case-insensitive, but the **D** flag can force case-sensitive matching.

If a condition begins with `*`, everything after that character is an `egrep` expression. Otherwise, if a line starts with `<` or `>`, it checks the size of a message as being smaller or larger than a given number of bytes. The `$` prefix allows shell substitutions.

An action that is simply a file name saves a message to that mailbox. Use the special pseudo-file `/dev/null` to delete a message. A pipe character (`|`) passes

the message to another program, such as the digest-splitting utility for mail that is distributed with Procmail. The exclamation prefix (!) forwards a message as an action (but negates a condition in a rule). Some examples:

Listing 6. Sample ~/.procmailrc file

```
:0:
* ^Subject:.*Digest                # split digests and save parts
* ^From:.*foo-digest
|formail +1 -ds cat >>mailing_lists_mailbox

:0:
* !(To|Cc).*mertz@gnosis.cx        # my main account here
* !(To|Cc).*david.mertz@gmail.com  # I get mail from here
* !From.*gnosis\.cx                # I trust gnosis not to spam
* !From.*list.*@                   # don't trash mailing lists
* !From.*good-buddy                 # sometimes Bcc's me mail
spam

:0:
* ^Subject.*[MY-LIST]               # redistribute MY-LIST messages
! member@example.com, member2@example.net, member3@example.edu

:0:
* ^Cc.*joe@somewhere.org           # save to both inbox and JOE mbox
{
  :0 c
  $DEFAULT

  :0
  JOE
}
```

Section 6. Serving NNTP news

What does InterNetNews do?

NNTP is a nice protocol for "pull" distribution of messages to any users who are interested in a given topic. The Usenet is a large collection of "newsgroups" on thousands of different topics that distribute messages via NNTP. Being a pull protocol, an NNTP server gathers the current messages available from a decentralized network of servers, selecting only those newsgroups that the site administrator chooses to include. When a new message is posted to a given newsgroup, it propagates non-hierarchically from the server to all the other servers on the Internet interested in subscribing to that particular newsgroup.

From an end-user perspective, a mailing list can appear very similar to a newsgroup. In either case, the user composes and posts messages, and reads messages

written by other people. In the ancient days of Usenet and the Internet, mailing lists were not as capable of presenting discussion topics in a "threaded" fashion as newsgroups now do automatically. But for a number of years, mail clients have done a good job of inferring the discussion threads within mailing lists.

The main difference between newsgroups and mailing lists is in their underlying network protocol. A mailing list still relies on one centralized mail server that accepts all the messages destined for a particular list, and distributes that message via e-mail to all users who have indicated an interest (and have been approved, either by automatic or human-moderated subscription mechanisms). In contrast, NNTP connects every node to every other one without relying on a central server; each NNTP server simply talks to the other servers "nearby," and messages rather rapidly reach the whole world.

InterNetNews (INN) is an NNTP server that was first written in 1992, and has been actively maintained since then. As of this writing, INN is at version 2.4.1. The home page for INN includes releases and documentation (see [Resources](#) for a link).

Setting up INN

After obtaining and unpacking the current source release, building INN is a straightforward `./configure; make; make install` sequence. To build INN, you need to have Perl and yacc (or bison) installed. This creates a number of files, mostly in the `/usr/local/news/` directory (which you probably do not have if INN has not been installed previously).

Before running the `innd` daemon (as user "news"), you should modify a number of configuration files. Full details are outside our scope, but a longer tutorial entitled *Installing and Running a Usenet News Server with INN and FreeBSD* on the full set of files needing attention is available online (see [Resources](#) for a link). Many of the permissions and quota issues are handled by the make system, but you might want to double check these configurations.

A file to pay particular attention to is the quota setup in `/usr/local/news/etc/storage.conf`. This controls which newsgroups are subscribed to and how much history from each newsgroup to maintain. Once the quota is reached, older messages are purged from a given newsgroup (on the local server, not from Usenet as a whole). For example, `storage.conf` might contain something like this:

Listing 7. Sample `storage.conf` configuration

```
method cnfs {
  newsgroups: alt.binaries.*
  class: 1
  size: 0,1000000
  options: BINARIES
```

```
}  
method cnfs {  
  newsgroups: *  
  class: 2  
  size: 0,100000  
  options: NOTBINRY  
}
```

The `class` value specifies the order in which different rules are evaluated.

Once all the various configuration files are tweaked, just running `innd` as a daemon (probably launched for an initialization script) monitors the upstream servers configured by `/usr/local/news/etc/innfeed.conf`, `/usr/local/news/etc/incoming.conf`, and `/usr/local/news/etc/newsfeeds`.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- View this [700 Linux User Groups around the world](#) -- many LUGs have local and distance study groups for LPI exams.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- *TCP/IP Network Administration, Third Edition* by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- The *Linux Network Administrators Guide* is an extensive online book covering many facets of Linux networking. Of specific interest for testing Sendmail is section 18.9, [Testing Your Configuration](#).
- The [Sendmail site](#) offers a [list of Sendmail books](#) for reference, including *Sendmail, Third Edition* (O'Reilly, December 2002).
- See *Installing and Running a Usenet News Server with INN and FreeBSD* for a more complete tutorial on installing and running an INN server.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).

Get products and technologies

- At the [Majordomo site](#), find downloads and information for Majordomo.
- At [procmal.org](#), download the latest version of Procmal.
- At [sendmail.org](#), download the latest version of Sendmail.
- Go to the [INN home page](#) for InterNetNews information and downloads.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Browse the [developerWorks blogs](#) and get involved in the developerWorks

community.

About the author

David Mertz, Ph.D.

David Mertz has been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#) . For more on David, see his [personal Web page](#).

LPI exam prep: Domain Name System (DNS)

Intermediate Level Administration (LPIC-2) topic 207

Skill Level: Intermediate

[David Mertz \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer
Gnosis Software

01 Dec 2005

This is the third of [seven tutorials](#) covering intermediate network administration on Linux®. In this tutorial, David Mertz gives an introduction to DNS and discusses how to use Linux as a DNS server, chiefly using BIND 9. He shows how to set up and configure the service, how to create forward and reverse lookup zones, and how to ensure that the server is secure from attacks.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 202: Tutorials and topics		
LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic 205): Networking configuration	Learn how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	Learn how to use Linux as a mail server and as a news server. Learn about mail transport, local mail filtering, mailing list maintenance software, and server software for the NNTP protocol.
Topic 207	LPI exam 202 prep (topic 207): DNS	(This tutorial) Learn how to use Linux as a DNS server, chiefly using BIND. Learn how to perform a basic BIND configuration, manage DNS zones, and secure a DNS server. See detailed objectives below.
Topic 208	LPI exam 202 prep (topic 208): Web services	Coming soon
Topic 210	LPI exam 202 prep (topic 210): Network client management	Coming soon
Topic 212	LPI exam 202 prep (topic 212): System security	Coming soon
Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	Coming soon

To start preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#). To prepare for the other exam in certification level 2, see the [developerWorks tutorials for LPI exam 201](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Domain Name System," the third of seven tutorials covering intermediate network administration on Linux. In this tutorial, you get a solid overview of DNS fundamentals and learn how to use Linux as a DNS server. You learn about setting up and configuring a BIND server, including working with `named.conf` and other configuration files; you also learn about forward and reverse DNS zones, as well as the basics of DNS security, including running BIND in a `chroot` jail and the DNS Security Extensions.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Domain Name System: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
2.207.1 Basic BIND 8 configuration	Weight 2	Configure BIND to function as a caching-only DNS server. This objective includes the ability to convert a BIND 4.9 <code>named.boot</code> file to the BIND 8.x <code>named.conf</code> format, and reload the DNS by using <code>kill</code> or <code>ndc</code> . This objective also includes configuring logging and options such as <code>directory</code> location for zone files.
2.207.2 Create and maintain DNS zones	Weight 3	Create a zone file for a forward or reverse zone or root-level server. This objective includes setting appropriate values for the SOA resource record, NS records, and MX records. Also included is adding hosts with A resource records and CNAME records as appropriate, adding hosts to reverse zones with PTR records, and adding the zone to the <code>/etc/named.conf</code> file using the <code>zone</code> statement with appropriate type, file, and masters values. You should also be able to delegate a zone to another DNS server.
2.207.3 Securing a DNS server	Weight 3	Configure BIND to run as a non-root user, and configure BIND to run in a <code>chroot</code> jail. This objective includes configuring DNSSEC statements such as <code>key</code> and <code>trusted-keys</code> to prevent

domain spoofing. Also included is the ability to configure a split DNS configuration using the `forwarders` statement, and specifying a non-standard version number string in response to queries.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. About DNS

The Domain Name System very usefully allows users of TCP/IP applications to refer to servers by symbolic name rather than by numeric IP address. The Berkeley Internet Name Domain (BIND) software provides a server daemon called *named* that answers requests for information about the IP address assigned to a symbolic name (or a reverse lookup or other information). On the client side of the DNS system, a *resolver* is a set of libraries that applications may use to communicate with DNS servers. The BIND package comes with several client utilities to assist in configuring, querying, and debugging a BIND 9 server: `dig`, `nslookup`, `host`, and `rndc` (formerly `ndc`). In essence, these utilities call the same libraries as other client applications do, but give direct feedback on answers provided by DNS servers.

About BIND

At the time of writing, the current version of BIND is 9.3.1. The first stable version of the BIND 9 series was released in October 2000. You may find BIND 8, which is still maintained for security patches (currently at 8.4.6), on some older installations, but as a rule, upgrade to BIND 9 where possible. Truly ancient systems might have BIND 4 installed on them, but you should upgrade those as soon as possible since BIND 4 is deprecated.

All versions of BIND may be obtained from the Internet Systems Consortium (ISC; see [Resources](#) for a link). Documentation and other resources on BIND are also at that site.

Because the LPI objectives call specifically for knowledge of BIND 8 configuration, and we cover BIND 9 here, we recommend that you review the BIND 8 information on the ISC site before taking the LPI 202 exam.

Other resources

As with most Linux tools, it is always useful to examine the manpages for any utilities discussed. Versions and switches might change between utility or kernel version or with different Linux distributions. For more in-depth information, the Linux Documentation Project (see [Resources](#) for a link) has a number of useful HOWTOs. A variety of good books on Linux networking have been published, in particular O'Reilly's *TCP/IP Network Administration* is quite helpful (find whatever edition is most current when you read this; see [Resources](#) for a link).

For DNS and BIND information specifically, O'Reilly's *DNS and BIND, Fourth Edition* is a good resource (see [Resources](#) for a link); at 622 pages, it covers more detail than this tutorial can. Several other books on BIND are available from various publishers.

Section 3. Understanding domain name system queries

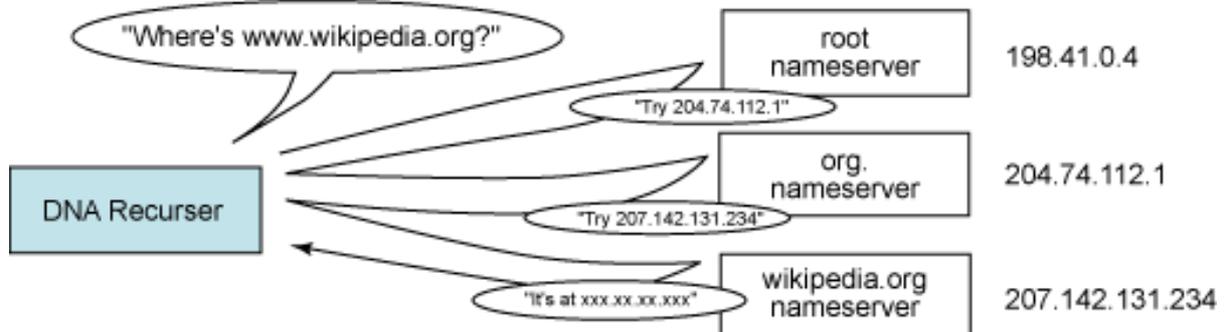
The topology of DNS

DNS is a hierarchical system of domain *zones*. Each zone provides a limited set of answers about domain name mappings, the ones within its own subdomain. A given server will query a more general server when it does not know a mapping and, if necessary, follow redirect suggestions until it finds the correct answer (or determines that no answer can be found, producing an error). When a local *named* server finds the answer to a DNS query, it caches that answer for a configurable amount of time (typically on the order of hours rather than seconds or days). By caching DNS queries, the overall network demand is lowered considerably, especially on top-level-domain (TLD) servers. The article on DNS at Wikipedia (see [Resources](#) for a link) is an excellent starting point for understanding the overall architecture. This tutorial borrows a public domain diagram from that site (see Figure 1 below).

A diagram of a hypothetical DNS query makes it easy to understand the overall lookup process. Suppose your local machine wishes to contact the symbolic domain name `www.wikipedia.org`. To find the corresponding IP address, your machine would first consult the local nameserver you have configured for a client machine. This

local nameserver may run on the same machine as the client application; it may run on a DNS server on your local LAN; or it may be provided by your ISP. In almost all cases, it is an instance of BIND's `named`. This local nameserver will first check its cache, but assuming no cached information is available, will perform steps as in the following diagram:

Figure 1. Example of DNS recursion



Understand that in this diagram, the "DNS Recursor" is the actual DNS server (named), not the client application that talks to it.

DNS uses TCP and UDP on port 53 to serve requests. Almost all DNS queries consist of a single UDP request from the client followed by a single UDP reply from the server.

How an application knows where to find a DNS server

Configuring client application access to its DNS server(s) is quite straightforward. The entire configuration is contained in the file `/etc/resolve.conf`, whose job is principally to specify the IP addresses for one or more "local" DNS servers. You may manually configure `/etc/resolve.conf` with known DNS servers; however, if you use DHCP to configure a client, the DHCP handshaking process will add this information to `/etc/resolve.conf` automatically (you may still read it or even modify it after DHCP sets it up, but it will be reset on reboot). The library code modified by `/etc/resolv.conf` is called the "DNS resolver."

If more than one DNS server is configured in an `/etc/resolv.conf` file, secondary and tertiary DNS servers will be consulted if the primary server fails to provide an answer within the specified timeout period. A maximum of three DNS servers may be configured.

The basic entry within an `/etc/resolv.conf` file contains the `nameserver` <IP-addr> entries. Some other entries let you modify returned answers. For example, the directives `domain` and `search` let you expand names without dots in them (like machines on the local LAN). The `options` directive lets you change timeouts per DNS server, turn on debugging, decide when to append full domain

names, and change other aspects of DNS resolver behavior. For example, one of my machines is configured with:

Listing 1. Modifying options to configure DNS servers

```
# cat /etc/resolv.conf
search gnosis.lan
nameserver 0.0.0.0
nameserver 192.168.2.1
nameserver 151.203.0.84
options timeout:3
```

The first directive lets this machine know that machines on the local LAN use the private domain *gnosis.lan*, so the simple name *bacchus* may be resolved as *bacchus.gnosis.lan*. More than one space-separated domain may be listed in the `search` directive.

Next, I list several DNS servers to try. The first is the local machine itself, which can be referred to either as *0.0.0.0* or by its official IP address, but not with a loopback address. The next `nameserver` directive lists my home-office router that connects my LAN to the Internet (and provides DHCP and DNS services). The tertiary nameserver is one provided by my ISP. I also set an option to use a three-second timeout on each nameserver rather than the default five seconds.

DNS client utilities

BIND 9 comes with four main client utilities. Three of those -- `dig`, `nslookup`, and `host` -- perform similar functions, more or less in descending order of detail. All three utilities are simply command-line utilities to exercise the DNS resolver. Essentially they do what many client applications do internally: these utilities simply provide output on the results of lookups on STDOUT. The most powerful of the three utilities, `dig`, also has the most options to limit or configure its output.

These utilities are most often used to look up an IP address from a symbolic domain name, but you may also perform reverse lookups or other record types other than default "A" records. For example, the command `host -t MX gnosis.cx` will show you mail servers associated with *gnosis.cx*. Some examples might help:

Listing 2. host query of google.com

```
$ host google.com
google.com has address 72.14.207.99
google.com has address 64.233.187.99
```

Listing 3. host MX query of gnosis.cx

```
$ host -t MX gnosis.cx
gnosis.cx mail is handled by 10 mail.gnosis.cx.
```

For the `nslookup` utility:

Listing 4. nslookup using default (machine-local) server

```
$ nslookup gnosis.cx
Server:          0.0.0.0
Address:         0.0.0.0#53

Non-authoritative answer:
Name:   gnosis.cx
Address: 64.41.64.172
```

Or a reverse lookup using the `dig` utility and specifying a non-default nameserver:

Listing 5. dig reverse lookup specifying a non-default nameserver

```
$ dig @192.168.2.2 -x 64.233.187.99

; <<>> DiG 9.2.4 <<>> @192.168.2.2 -x 64.233.187.99
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 3950
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;99.187.233.64.in-addr.arpa.      IN  PTR

;; AUTHORITY SECTION:
187.233.64.in-addr.arpa. 2613  IN  SOA  ns1.google.com. dns-admin.google.com.
2004041601 21600 3600 1038800 86400

;; Query time: 1 msec
;; SERVER: 192.168.2.2#53(192.168.2.2)
;; WHEN: Thu Nov 10 02:00:27 2005
;; MSG SIZE rcvd: 104
```

The remaining BIND 9 utility to keep in mind is `rndc`. The `rndc` utility controls the operation of a name server. It supersedes the `ndc` utility that was provided in older BIND releases. If `rndc` is invoked with no command-line options or arguments, it prints a short summary of the supported commands. See the manpage for `rndc` for full information on its use.

Section 4. Basic BIND configuration and running a name server

BIND configurations

When you run the `named` daemon to provide a DNS server, you may choose from three modes of operation: *master*, *slave*, and *caching-only*. The `named` daemon itself looks in its configuration files, chiefly `/etc/bind/named.conf`, to determine its operating mode.

In master mode, the `named` server acts as the authoritative source for all information about its zone. Domain information provided by the server is loaded from a local disk file that is manually modified or updated. Each DNS zone should have exactly one master server.

In slave mode, the `named` server transfers its zone information from the master server for its zone. Technically, a multi-zone server can be master of one zone and slave for another, but more commonly a LAN installation has a single hierarchy between master and slave or caching-only servers. A slave server transfers complete zone information to local files from its master server, so the answers provided by a slave server are still considered authoritative.

In caching-only mode, the `named` server keeps no zone files. Every query relies on some other name server for an initial answer, but to minimize bandwidth usage, previous queries are cached by the caching-only server. However, any novel query must be answered by a query sent over the network. Caching-only servers are most common on local machines where client applications can often perform a lookup without any network traffic.

In the `/etc/resolv.conf` configuration I offered as an example earlier in [Listing 1](#), `0.0.0.0` is a caching-only server, `192.168.2.1` is a slave server, and `151.203.0.84` is a master server. You cannot tell this for certain just based on the order or IP addresses used, but the use of the local machine pseudo-IP address `0.0.0.0` suggests that it is running a caching-only server.

Configuring `named.conf`

There are some standard features that pretty much every `/etc/bind/named.conf` file has. An initial `options` directive configures some basic information. After that, several `zone` directives provide information on how to handle various zone requests. Domains given in `zone` directives as IP addresses represent initial portions of IP address ranges, but are indicated "backwards." Symbolic names may define zones, too, allowing further specified subdomains.

`named.conf` files (and other BIND configuration files) follow C-like formatting conventions, in large part. Both C-style block comments (`/* comment */`) and

C++-style line comments (`// comment`) are allowed, as are shell-style line comments (`# comment`). Directives are followed by semicolon-divided statements surrounded by curly brackets.

To start, here are some common options. My local `/etc/bind/named.conf` begins with:

Listing 6. My local `named.conf` starts like this

```
include "/etc/bind/named.conf.options";
```

But you may also use the `options` directive directly:

Listing 7. Configuring `named.conf` options

```
options {
    directory "/var/bind";
    forwarders { 192.168.2.1; 192.168.3.1};
    // forward only;
}
```

This setup lets files specified without a full path be located in a relative directory; it also tells the local `named` to look first in 192.168.2.1 and 192.168.3.1 for non-cached results. The `forward only` directive (commented out here) says to look only in those nameservers on the local LAN rather than ask the root servers on the Internet.

A special `zone` directive is present in nearly all `named.conf` files:

Listing 8. Hint zone for root servers

```
zone "." {
    type hint;
    file "/etc/bind/db.root";
};
```

The contents of `db.root` (sometimes called `named.ca` for "certifying authority") is special. It points to canonical root servers, the domain registrars themselves. Their values change rarely, but you may obtain an official latest version from `ftp.rs.internic.net`. This is not a file a regular administrator will modify.

Beyond the root zone hint, a `named.conf` file will contain some master and/or slave zones. For example, for the local loopback:

Listing 9. Loopback zone configuration

```
zone "127.in-addr.arpa" {
    type master;
    file "/etc/bind/db.127";
```

```
};
```

More interestingly, a named server might act as master for a domain (and provide reverse lookup):

Listing 10. External zone configuration

```
zone "example.com" {
    type master;
    file "example.com.hosts"; // file relative to /var/bind
};
// Reverse lookup for 64.41.* IP addresses (backward IP address)
zone "41.64.in-addr.arpa" {
    type master;
    file "41.64.rev";
};
```

For a slave configuration, you might instead use:

Listing 11. External zone configuration (slave)

```
zone "example.com" {
    type slave;
    file "example.com.hosts"; // file relative to /var/bind
    masters { 192.168.2.1; };
};
// Reverse lookup for 64.41.* IP addresses (backward IP address)
zone "41.64.in-addr.arpa" {
    type slave;
    file "41.64.rev";
    masters { 192.168.2.1; };
};
```

Other configuration files

The `named.conf` file references a number of other configuration files with the `file` directive. These names are dependent on your specific setup, but in general will contain various resource records that are defined in RFC 1033 (*Domain Administrators Operations Guide*; see [Resources](#) for a link). The standard resource records are:

SOA

Start of authority. Parameters affecting an entire zone.

NS

Nameserver. A domain's name server.

A

Address. Hostname to IP address.

PTR

Pointer. IP address to hostname.

MX

Mail exchange. Where to deliver mail for a domain.

CNAME

Canonical name. Hostname alias.

TXT

Text. Stores arbitrary values.

The format of a record is: <name> <time-to-live> IN <type> <data>.

The name and time-to-live are optional and default to the last values used. The literal string IN means Internet and is always used in practice. The resource record files may also contain directives, which begin with a dollar sign. The most common of these is probably \$TTL, which sets a default time-to-live. For example, a somewhat trivial record file for the 127.*localhost looks like this:

Listing 12. A trivial record file

```
# cat /etc/bind/db.127
; BIND reverse data file for local loopback interface
;
$TTL      604800
@         IN      SOA      localhost. root.localhost. (
                        1          ; Serial
                        604800     ; Refresh
                        86400      ; Retry
                        2419200    ; Expire
                        604800 )   ; Negative Cache TTL
;
@         IN      NS       localhost.
1.0.0     IN      PTR      localhost.
```

Other directives are \$ORIGIN, which sets the domain name used to complete any relative domain name; \$INCLUDE, which reads an external file; and \$GENERATE, which creates a series of resource records covering a range of IP addresses.

Section 5. Create and maintain DNS zones

Reverse zone files

Reverse zone files (often indicated with a `.rev` extension) contain mappings from zone-specific IP addresses to symbolic names. For example, you might have a file `/var/bind/41.64.rev` that contains:

Listing 13. Reverse zone file for 64.41.*

```
$TTL 86400
; IP address to hostname
@      IN      SOA      example.com.  mail.example.com. (
                                2001061401 ; Serial
                                21600      ; Refresh
                                1800       ; Retry
                                604800     ; Expire
                                900        ; Negative cach TTL

                                IN      NS      ns1.example.com.
                                IN      NS      ns2.example.com.
; Define names for 64.41.2.1, 64.41.2.2, etc.
1.2    IN      PTR      foo.example.com.
2.2    IN      PTR      bar.example.com.
3.2    IN      PTR      baz.example.com.
```

Forward zone files

Forward zone files (often named as *domain.hosts*) contain the crucial "A" records for mapping symbolic names to IP addresses. For example, you might have a file `/var/bind/example.com.hosts` that contains the following:

Listing 14. Forward zone file for example.com

```
$TTL 86400
; Hostname to IP address
@      IN      SOA      example.com.  mail.example.com. (
                                2001061401 ; Serial
                                21600      ; Refresh
                                1800       ; Retry
                                604800     ; Expire
                                900        ; Negative cach TTL

                                IN      NS      ns1.example.com.
                                IN      NS      ns2.example.com.
localhost  IN      A      127.0.0.1
foo        IN      A      64.41.2.1
www        IN      CNAME  foo.example.com
bar        IN      A      64.41.2.2
bar        IN      A      64.41.2.3
```

Section 6. Securing a DNS server

Securing a DNS server

As with many services, it's a good idea to run BIND in a so-called *chroot jail*. This limits BIND's access to other files or system resources, should a vulnerability or bug exist in BIND. Find a more detailed tutorial on running BIND with chroot at the "Chroot-BIND HOWTO" (see [Resources](#) for a link).

The general point of this procedure is that it is unwise to run BIND as the root user, or even as a common special user like "nobody." Often the user "named" is created to run BIND. The files used by this special user are placed in a local directory like /chroot/named/ and appropriate relative subdirectories.

BIND 9 provides much cleaner support for chroot restrictions than did BIND 8; a clean compile should suffice without special switches or Makefile tweaks.

DNSSEC

Beyond securing the local machine that runs BIND, it is also desirable to provide security guarantees to the DNS protocol itself. The DNS Security Extensions (DNSSEC) are a set of extensions to DNS that provide authenticity and integrity.

DNS is based on UDP rather than on TCP, and therefore does not have a mechanism for verifying a packet source. This limitation makes spoofing and interception attacks possible. In other words, DNS requesters might be fed malicious information, for example to redirect communication to an intruder's host. By adding cryptographic Transactional Signatures (TSIG) to DNS requests, DNSSEC can prevent spoofing of DNS responses. Each BIND 9 server that wishes to communicate securely must enable DNSSEC, but the enhanced protocol is otherwise backwards compatible. The first thing DNS servers must do -- those that intend to communicate securely, anyway -- is generate *key pairs*. This works in a manner very similar to SSH keys for host and server. For example:

Listing 15. Generating DNSSEC keys

```
dnssec-keygen -r /dev/urandom -a HMAC-MD5 -b 128 -n HOST \
primary-secondary.my.dom
# ls Kprimary-secondary.my.dom.*
Kprimary-secondary.my.dom.+157+46713.key
Kprimary-secondary.my.dom.+157+46713.private
```

As the filenames suggest, this generates public and private keys for the configured host, and the public key will be distributed to other servers. Get a good introduction to DNSSEC in "The Basics of DNSSEC" on the O'Reilly Network (see [Resources](#) for a link).

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- The [article on DNS at Wikipedia](#) is a good starting point for understanding the overall architecture.
- *[DNS and BIND, Fourth Edition](#)* by Paul Albitz and Cricket Liu (O'Reilly, 2001) thoroughly covers both DNS and BIND.
- *[TCP/IP Network Administration, Third Edition](#)* by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- See RFC 1033, the *[Domain Administrators Operations Guide](#)*, for the definitions of standard resource records.
- The [Chroot-BIND HOWTO](#) shows how to configure BIND to run in a chroot jail.
- The [Basics of DNSSEC](#) provides a good introduction to DNSSEC.
- View this [700 Linux User Groups around the world](#) -- many LUGs have local and distance study groups for LPI exams.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).

Get products and technologies

- [Download BIND](#) from the [Internet Systems Consortium](#).
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

David Mertz

David Mertz has been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#) . For more on David, see his [personal Web page](#).

LPI exam prep: Web services

Intermediate Level Administration (LPIC-2) topic 208

Skill Level: Intermediate

[David Mertz \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer

Gnosis Software, Inc.

25 Apr 2006

In this tutorial, the fourth in a [series of seven tutorials](#) covering intermediate network administration on Linux, David Mertz continues preparing you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 208. Here, David Mertz discusses how to configure and run the Apache HTTP server and the Squid proxy server.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 202: Tutorials and topics

LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic 205): Networking configuration	Learn how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	Learn how to use Linux as a mail server and as a news server. Learn about mail transport, local mail filtering, mailing list maintenance software, and server software for the NNTP protocol.
Topic 207	LPI exam 202 prep (topic 207): DNS	Learn how to use Linux as a DNS server, chiefly using BIND. Learn how to perform a basic BIND configuration, manage DNS zones, and secure a DNS server.
Topic 208	LPI exam 202 prep (topic 208): Web services	(This tutorial) Learn how to install and configure the Apache Web server, and learn how to implement the Squid proxy server. See detailed objectives below.
Topic 210	LPI exam 202 prep (topic 210): Network client management	Coming soon
Topic 212	LPI exam 202 prep (topic 212): System security	Coming soon
Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	Coming soon

To start preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#). To prepare for the other exam in certification level 2, see the [developerWorks tutorials for LPI exam 201](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Web services," the fourth of seven tutorials covering intermediate network administration on Linux. In this tutorial, you learn how to configure and run the Apache HTTP server and the Squid Web Proxy Cache.

As with the other tutorials in the developerWorks 201 and 202 series, this tutorial is intended to serve as a study guide and entry point for exam preparation, rather than complete documentation on the subject. Readers are encouraged to consult LPI's [detailed objectives list](#) and to supplement the information provided here with other material as needed.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Web services: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
2.208.1 Implementing a Web server	Weight 2	Install and configure a Web server. This objective includes monitoring the server's load and performance, restricting client user access, configuring support for scripting languages as modules, and setting up client user authentication. Also included is the ability to configure server options to restrict usage of resources.
2.208.2 Maintaining a Web server	Weight 2	Configure a Web server to use virtual hosts, Secure Sockets Layer (SSL), and customize file access.
2.208.3 Implementing a proxy server	Weight 2	Install and configure a proxy server, including access policies, authentication, and resource usage.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. About Apache and Squid

Apache Web server

Apache is the predominant Web server on the Internet as a whole; it is even more predominant among Linux servers. A few more special-purpose Web servers are available (some offering higher performance for specific tasks), but Apache is always the default choice.

Apache comes pre-installed on most Linux distributions and is often already running after being launched during initialization, even if you have not specifically configured a Web server. If Apache is not installed, use the normal installation system of your distribution to install it, or [download the latest HTTP server](#) from the Apache HTTP Server Project. Many extra capabilities are provided by modules, many are distributed with Apache itself, and others are available from third parties.

Even though the latest Apache has been at the 2.x level since 2001, Apache 1.3.x is still in widespread use, and the 1.3.x series continues to be maintained for bug fixes and security updates. Some minor configuration differences exist between 1.3 and 2.x; a few modules are available for 1.3 that are not available for 2.x. The latest releases as of this tutorial are 1.3.34 (stable), 2.0.55 (stable), and 2.1.9 (beta).

As a rule, a new server should use the latest stable version in the 2.x series. Unless you have a specific need for an unusual older module, 2.x provides good stability, more capabilities, and overall better performance (in some tasks, such as in PHP support, 1.3 still performs better). Moving forward, new features will certainly be better supported in 2.x than in 1.3.x.

Squid proxy server

Squid is a proxy-caching server for Web clients that supports the HTTP, FTP, TLS, SSL, and HTTPS protocols. By running a cache on a local network, or at least closer to your network than the resources queried, speed can be improved and network bandwidth reduced. When the same resource is requested multiple times by machines served by the same Squid server, the resource is delivered from a server-local copy rather than requiring the request to go out over multiple network routers and to potentially slow or overload destination servers.

You can configure Squid as an explicit proxy that must be configured in each Web client (browser), or you can configure it to intercept all Web requests out of a LAN

and cache all such traffic. You can configure Squid with various options regarding how long and under what conditions to keep pages cached.

Other resources

As with most Linux tools, it is always useful to examine the manpages for any utilities discussed. Versions and switches might change between utility or kernel version or with different Linux distributions. For more in-depth information, the Linux Documentation Project has a variety of useful documents, especially its HOWTOs. A variety of books on Linux networking have been published; I have found O'Reilly's *TCP/IP Network Administration*, by Craig Hunt, to be quite helpful. (See [Resources](#) later in this tutorial for links.)

Many good books have been written on working with Apache. Some are concerned with general administration, while others cover particular modules or special configurations of Apache. Check your favorite bookseller for a range of available titles.

Section 3. Implementing a Web server

A swarm of daemons

Launching Apache is similar to launching any other daemon. Usually you want to put its launch in your system initialization scripts, but in principle you may launch Apache at any time. On most systems, the Apache server is called *httpd*, though it may be called *apache2* instead. The server is probably installed in `/usr/sbin/`, but other locations are possible depending on the distribution and how you installed the server.

Most of the time you will launch Apache with no options, but the `-d serverroot` and `-f config` options are worth keeping in mind. The first lets you specify a location on the local disks from where content will be served; the second lets you specify a non-default configuration file. A configuration file may override the `-f` option using the `ServerRoot` directive (most do). By default, configuration files are either `apache2.conf` or `httpd.conf`, depending on compilation options. These files might live at `/etc/apache2/`, `/etc/apache/`, `/etc/httpd/conf/`, `/etc/httpd/apache/conf`, or a few other locations depending on version, Linux distribution, and how you installed or compiled Apache. Checking `man apache2` or `man httpd` should give you system-specific details.

The Apache daemon is unusual when compared with other servers in that it usually creates several running copies of itself. The primary copy simply spawns the others, while these secondary copies service the actual incoming requests. The goal of having multiple running copies is to act as a "pool" for requests that may arrive in bundles; additional copies of the daemon are launched as needed, according to several configuration parameters. The primary copy usually runs as root, but the other copies run as a more restricted user for security reasons. For example:

Listing 1. The many faces of multiple running copies of Apache

```
# ps axu | grep apache2
root      6620    Ss   Nov12  0:00 /usr/sbin/apache2 -k start -DSSL
www-data  6621    S    Nov12  0:00 /usr/sbin/apache2 -k start -DSSL
www-data  6622    S1   Nov12  0:00 /usr/sbin/apache2 -k start -DSSL
www-data  6624    S1   Nov12  0:00 /usr/sbin/apache2 -k start -DSSL
dqm       313     S+   03:44  0:00 man apache2
root      637     S+   03:59  0:00 grep apache2
```

On many systems, the restricted user will be *nobody*. In Listing 1, it is `www-data`.

Including configuration files

As mentioned, the behavior of Apache is affected by directives in its configuration file. For Apache2 systems, the main configuration file is likely to reside at `/etc/apache2/apache2.conf`, but often this file will contain multiple `Include` statements to add configuration information from other files, possibly by wildcard patterns. Overall, an Apache configuration is likely to contain hundreds of directives and options (most not specifically documented in this tutorial).

A few files are particularly likely to be included. You might see `httpd.conf` for "user" settings, to utilize prior Apache 1.3 configuration files that use that name. Virtual hosts are typically specified in separate configuration files, matched on a wildcard, like in the following:

Listing 2. Specifying virtual hosts

```
# Include the virtual host configurations:
Include /etc/apache2/sites-enabled/[^.#]*
```

With Apache 2.x, modules are typically specified in separate configuration files as well (more often in the same file in 1.3.x). For example, a system of mine includes:

Listing 3. From `/etc/apache2/apache2.conf`

```
# Include module configuration:
Include /etc/apache2/mods-enabled/*.load
```

```
Include /etc/apache2/mods-enabled/*.conf
```

Actually using a module in a running Apache server requires two steps in the configuration file, both loading it and enabling it:

Listing 4. Loading an optional Apache module

```
# cat /etc/apache2/mods-enabled/userdir.load
LoadModule userdir_module /usr/lib/apache2/modules/mod_userdir.so
# cat /etc/apache2/mods-enabled/userdir.conf
<IfModule mod_userdir.c>
    UserDir public_html
    UserDir disabled root

    <Directory /home/*/public_html>
        AllowOverride FileInfo AuthConfig Limit
        Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    </Directory>
</IfModule>
```

The wildcards in the `Include` lines will insert all the `.load` and `.conf` files in the `/etc/apache2/mods-enabled/` directory.

Notice the general pattern: Basic directives are one-line commands with some options; more complex directives nest commands using an XML-like open/close tag. You just have to know for each directive whether it is one-line or open/close style -- you cannot choose among styles at will.

Log files

An important class of configuration directives concern logging of Apache operations. You can maintain different types of information and degrees of detail for Apache operations. Keeping an error log is always a good idea; you can specify it with the single directive:

Listing 5. Specifying an error log

```
# Global error log.
ErrorLog /var/log/apache2/error.log
```

You can customize other logs of server access, of referrers, and of other information to fit your individual setup. A logging operation is configured with two directives. First, a `LogFormat` directive uses a set of special variables to specify what goes in the log file; second, a `CustomLog` directive tells Apache to actually record events in the specified format. You can specify an unlimited number of formats regardless of whether each one is actually used. This allows you to switch logging details on and off, based on evolving needs.

Variables in a `LogFormat` are similar to shell variables, but with a leading `%`. Some variables have single letters, while others have long names surrounded by brackets, as shown in Listing 6.

Listing 6. LogFormat variables

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
CustomLog /var/log/apache2/referer_log combined
```

Consult a book or full Apache documentation for the list of variables. Commonly used ones include `%h` for IP address of requesting client, `%t` for datetime of the request, `%>s` for HTTP status code, and the misspelled `%{Referer}` for the referring site that led to the served page.

The name used in the `LogFormat` and `CustomLog` directives is arbitrary. In Listing 6, the name `combined` was used, but it could just as well be `myfoobarlog`. However, a few names are commonly used and come with sample configuration files, such as `combined`, `common`, `referer`, and `agent`. These specific formats are typically supported directly by log-analyzer tools.

Section 4. Maintaining a Web server

Virtual hosts, multi-homing, and per-directory options

Individual directories served by an Apache server may have their own configuration options. However, the main configuration may limit which options can be configured locally. If per-directory configuration is desired, use the `AccessFileName` directive and typically specify the local configuration filename of `.htaccess`. The limitations of local configuration are specified within a `<Directory>` directive. For example:

Listing 7. Example of directory directive

```
#Let's have some Icons, shall we?
Alias /icons/ "/usr/share/apache2/icons/"
<Directory "/usr/share/apache2/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Often working in conjunction with per-directory options, Apache can service *virtual*

hosts. Multiple domain names may be served from the same Apache process, each accessing an appropriate directory. You can define virtual hosts with the `<VirtualHost>` directive; place configuration files in an included directory, such as `/etc/apache2/sites-enabled/`, or in a main configuration file. For example, you might specify:

Listing 8. Configuring virtual hosts

```
<VirtualHost "foo.example.com">
  ServerAdmin webmaster@foo.example.com
  DocumentRoot /var/www/foo
  ServerName foo.example.com
  <Directory /var/www/foo>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>
  ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
  <Directory "/usr/lib/cgi-bin">
    AllowOverride None
    Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
    Order allow,deny
    Allow from all
  </Directory>
  CustomLog /var/log/apache2/foo_access.log combined
</VirtualHost>
<VirtualHost "bar.example.org">
  DocumentRoot /var/www/bar
  ServerName bar.example.org
</VirtualHost>
<VirtualHost *>
  DocumentRoot /var/www
</VirtualHost>
```

The final `*` option picks up any HTTP requests that are not directed to one of the explicitly specified names (like those addressed by IP address or addressed as an unspecified symbolic domain that also resolves to the server machine). For virtual domains to work, DNS must define each alias with a CNAME record.

Multi-homed servers sound similar to virtual hosting, but the concept is different. Using *multi-homing*, you may specify the IP addresses to which a machine is connected in order to allow Web requests. For example, you might provide HTTP access only to the local LAN, but not to the outside world. If you specify an address to listen on, you may also indicate a non-default port. The default value for `BindAddress` is `*`, which means to accept requests on every IP address under which the server may be reached. A mixed example might look like:

Listing 9. Configuring multi-homing

```
BindAddress 192.168.2.2
Listen 192.168.2.2:8000
Listen 64.41.64.172:8080
```

In this case, we will accept all client requests from the local LAN (that use the 192.168.2.2 address) on the default port 80 and on the special port 8000. This Apache installation will also monitor client HTTP requests from the WAN address, but only on port 8080.

Limiting Web access

You may enable *per-directory* server access with the `Order`, `Allow from`, and `Deny from` commands within a `<Directory>` directive. Denied or allowed addresses may be specified by full or partial hostnames or IP addresses. `Order` lets you give precedence between the accept list and the deny list.

In many cases, you need more fine-tuned control than you can gain by simply allowing particular hosts to access your Web server. To enable user login requirements, use the `Auth*` family of commands, again within the `<Directory>` directive. For example, to set up Basic Authentication, you might use a directive as shown in Listing 10.

Listing 10. Configuring Basic Authentication

```
<Directory "/var/www/baz">
  AuthName "Baz"
  AuthType Basic
  AuthUserFile /etc/apache2/http.passwords
  AuthGroupFile /etc/apache2/http.groups
  Require john jill sally bob
</Directory>
```

You may also specify Basic Authentication within a per-directory `.htaccess` file. Digest Authentication is more secure than Basic, but Digest Authentication is less widely implemented in browsers. However, the weakness of Basic (that it transmits passwords in cleartext) is better solved with an SSL layer, anyway.

Support for SSL encryption of Web traffic is provided by the module `mod_ssl`. When SSL is used, data transmitted between server and client is encrypted with a dynamically negotiated password that is resistant to interception. All major browsers support SSL. For more information on configuring Apache 2.x with `mod_ssl`, see the description on the Apache Web site (see [Resources](#) for a link).

Section 5. Implementing a proxy server

Installing and running Squid

In most distributions, you can install Squid using the normal installation procedures. Get the source version of Squid from the Squid Web Proxy Cache Web site (see [Resources](#) for a link). Building from source uses the basic `./configure; make; make install` sequence.

Once installed, you may simply run it as root, `/usr/sbin/squid` (or whatever location your distribution uses, perhaps `/usr/local/sbin/`). Of course, to do much useful, you will want to configure the Squid configuration file at `/etc/squid/squid.conf`, `/usr/local/squid/etc/squid.conf`, or wherever precisely your system locates `squid.conf`. As with almost all daemons, you may use a different configuration file, in this case with the `-f` option.

Ports, IP addresses, http_access, and ACLs

The most important configuration options for Squid are the `http_port` options you select. You may monitor whichever ports you wish, optionally attaching each one to a particular IP address or hostname. The default port for Squid is 3128, allowing any IP address that connects to the Squid server. To cache only for a LAN, specify the local IP address instead as shown:

Listing 11. Caching Squid only for a LAN

```
# default (disabled)
# http_port 3128
# LAN only
http_port 192.168.2.2:3128
```

You may also enable caching via other Squid servers using the `icp_port` and `htcp_port`. The IPC and HTCP protocols are used for caches to communicate between themselves rather than by Web servers and clients themselves. To cache multicasts, use `mcast_groups`.

To let clients connect to your Squid server, you need to give them permission to do so. Unlike a Web server, Squid is not entirely generous with its resources. In the simple case, we can just use a couple of subnet/netmask or CIDR (Classless Internet Domain Routing) patterns to control permissions:

Listing 12. Simple Squid access permissions

```
http_access deny 10.0.1.0/255.255.255.0
http_access allow 10.0.0.0/8
icp_access allow 10.0.0.0/8
```

You can use the `acl` directive to name access control lists (ACLs). You can name `src` ACLs that simply specify address ranges as in Listing 12, but you can also create other types of ACLs. For example:

Listing 13. Fine-tuned access permissions

```
acl mynetwork      src          192.168/16
acl asp            urlpath_regex  \.asp$
acl bad_ports     port          70 873
acl javascript    rep_mime_type -i ^application/x-javascript$
# what HTTP access to allow classes
http_access deny asp          # don't cache active server pages
http_access deny bad_ports   # don't cache gopher or rsync
http_access deny javascript  # don't cache javascript content
http_access allow mynetwork  # allow LAN everything not denied
```

Listing 13 shows only a small subset of the available ACL types. See a sample `squid.conf` for examples of many others. Or take a look at the Access control documentation (Chapter 6) in the Squid User's Guide (see [Resources](#) for a link).

In Listing 13, we decide not to cache URLs that end with `.asp` (probably dynamic content), not to cache ports 70 and 873, and not to cache returned JavaScript objects. Other than what is denied, machines on the LAN (the `/16` range given) will have all their requests cached. Notice that each ACL defined has a unique, but arbitrary, name (use names that make sense; Squid does not reserve the names).

Caching modes

The simplest way to run Squid is in proxy mode. If you do this, clients must be explicitly configured to use the cache. Web browser clients have configuration screens that allow them to specify a proxy address and port rather than a direct HTTP connection. This setup makes configuring Squid very simple, but it makes clients do some setup work if they want to benefit from the Squid cache.

You can also configure Squid to run as a transparent cache. To do this, you need to either configure policy-based routing (outside of Squid itself, using `ipchains` or `ipfilter`) or use your Squid server as a gateway. Assuming you can direct external requests via the Squid server, Squid needs to be configured as follows. You may need to recompile Squid with the `--enable-ipf-transparent` option; however, in most Linux installations, this should already be fine. To configure the server for transparent caching (once it gets the redirected packets), add something like Listing 14 to your `squid.conf`:

Listing 14. Configuring Squid for transparent caching

```
httpd_accel_host virtual
httpd_accel_port 80
```

```
httpd_accel_with_proxy on  
httpd_accel_uses_host_header on
```

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Get more information on [configuring Apache 2.x with mod_ssl](#).
- Read the tutorial "[Customizing Apache for maximum performance](#)" (developerWorks, June 2002) to learn how to tweak Apache for particular environments and needs.
- The chapter on [Access Control and Access Control Operators](#) in the Squid User's Guide describes the available ACL types.
- *[TCP/IP Network Administration, Third Edition](#)* by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- The [Linux Users Groups WorldWide](#) home page lists 700 Linux user groups around the world. Many LUGs have local and distance study groups for LPI exams.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- In the [developerWorks Linux zone](#), find more resources for Linux developers.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- Download the latest [Apache Web server](#).
- Download [Squid](#) and additional Squid documentation.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content](#).
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the author

David Mertz

David Mertz thinks that artificial languages are perfectly natural, but natural languages seem a bit artificial. You can reach David at mertz@gnosis.cx; you can investigate all aspects of his life at his [personal Web page](#). Check out his book, [Text Processing in Python](#). Suggestions and recommendations on past or future columns are welcome.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

LPI exam 201 prep: File and service sharing

Intermediate Level Administration (LPIC-2) topic 209

Skill Level: Intermediate

[Brad Huntting \(hunting@glarp.com\)](mailto:hunting@glarp.com)

Mathematician
University of Colorado

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer
Gnosis Software

02 Sep 2005

In this tutorial, Brad Huntting and David Mertz continue preparing you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 201. In this fifth of eight tutorials, you learn how to use a Linux™ system as a networked file server using any of several protocols supported by Linux.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams.

Each exam covers several topics, and each topic has a weight. The weights indicate the relative importance of each topic. Very roughly, expect more questions on the exam for topics with higher weight. The topics and their weights for LPI exam 201

are:

Topic 201

Linux kernel (weight 5).

Topic 202

System startup (weight 5).

Topic 203

Filesystems (weight 10).

Topic 204

Hardware (weight 8).

Topic 209

File and service sharing (weight 8). The focus of this tutorial.

Topic 211

System maintenance (weight 4).

Topic 213

System customization and automation (weight 3).

Topic 214

Troubleshooting (weight 6).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "File and service sharing," the fifth of eight tutorials designed to prepare you for LPI exam 201. In this tutorial, you learn how to use a Linux system as a networked file server using any of several protocols supported by Linux.

The tutorial is organized according to the LPI objectives for this topic, as follows:

2.209.1 Configuring a Samba server (weight 5)

You will be able to set up a Samba server for various clients. This objective includes setting up a login script for Samba clients and setting up an nmbd WINS server. Also included is changing the workgroup in which a server participates, defining a shared directory in smb.conf, defining a shared printer in smb.conf, using nmblookup to test WINS server functionality, and using the smbmount command to mount an SMB share on a Linux client.

2.209.2 Configuring an NFS server (weight 3)

You will be able to create an exports file and specify filesystems to be exported. This objective includes editing exports file entries to restrict access to certain hosts, subnets, or netgroups. Also included is specifying mount options in the exports file, configuring user ID mapping, mounting an NFS filesystem on a client, and using mount options to specify soft or hard and background retries, signal handling, locking, and block size. You should also be able to configure tcpwrappers to further secure NFS.

The current LPI exam objectives for topic 209 exam cover NFS and Samba. But if you are a system administrator designing a server configuration, you should also consider whether FTP, SCP/SSH, HTTP, or other protocols might, in fact, meet your requirements.

One of the most significant uses for Linux, particularly in a server context, is to provide shared files to client systems. In fact, in a general way, serving files is probably most of what all networking is used for. This tutorial -- and in fact, this series of tutorials -- will not address peer-to-peer file-sharing servers such as BitTorrent. Rather, this tutorial looks only at older client-server arrangements: A central server that provides disk stores for multiple clients. Even when clients upload files, those are always stored and served by the server, rather than in a decentralized fashion.

Protocols widely used for file serving include HTTP (the WWW), TFTP (Trivial File Transfer Protocol), FTP (File Transfer Protocol), SCP (Secure Copy Protocol, a specialized use of SSH), RCP (Remote Copy Protocol, generally deprecated), NFS (Network File System), and Samba (server message block). HTTP and SSH will be discussed in upcoming tutorials for LPI exam 202, as will security issues around FTP. TFTP and RCP are special purpose or deprecated and will not be addressed in these tutorials.

This tutorial looks at NFS and Samba in some detail and briefly describes FTP. NFS and Samba are network file-sharing protocols that allow mostly transparent access to remote filesystems. FTP might require a custom FTP client program, although many desktop environments or tools (on Linux or otherwise) hide the details of this negotiation and effectively present the same user interface as an NFS- or Samba-mounted drive.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. Configuring an NFS server

Using NFS on a client

If the server is properly configured and the client has appropriate permissions, mounting a remote filesystem with NFS requires only the `mount` command:

```
mount -t nfs my.nfs.server.com:/path/on/server /path/on/client
```

or a suitable entry in `/etc/fstab`:

```
my.nfs.server.com:/path/on/server /path/on/client nfs rw,soft  
0 0
```

The `soft` option tells the kernel to send an I/O error (EIO) to user processes in the event of network difficulties. The default `hard` option will cause processes to hang while the NFS server is unreachable.

In addition, the helper programs `rpc.lockd`, `rpc.statd`, and `rpc.quotad` may be run on client and/or server.

Configuring an NFS server (part one)

An NFS server requires three distinct programs, as well as three optional programs.

When an NFS client mounts an NFS filesystem, it contacts the following server daemons, most of which must run standalone (as opposed to being started from `inetd`):

- `portmap`: Sometimes named `portmapper` or `rpc.bind`.
- `rpc.mountd`: Sometimes `mounted`.
- `rpc.nfsd`: Sometimes `nfsd`.

In addition, there are three optional helper programs: `rpc.lockd`, `rpc.statd`, and `rpc.quotad` which, respectively, provide global locking, accelerate the `lstat` family of syscalls (used by `ls -l`, etc.), and provide support for quotas.

Configuring an NFS server (part two)

All three NFS-related servers use TCPwrappers (`tcpd`) for access control and therefore may require entries in `/etc/host.allow`.

Neither `nfsd` nor `portmap` normally require any configuration beyond `/etc/hosts.allow`.

The configuration file for `mountd` is (indirectly) `/etc/exports`. It says which filesystems can be mounted by which clients. Under the Linux implementation of NFS, `/etc/exports` is not directly parsed by `mountd`. Instead, the `exportfs -a` command parses `/etc/exports` and writes the result to `/var/lib/nfs/xtab` where `mountd` can read it. There are other flags to `exportfs` which allow these two files to be desynchronized. That is, you may temporarily add or remove exported directories without modifying the semi-permanent records in `/etc/exports`.

Administrators of other Unix-like servers should note that the syntax of the Linux `/etc/exports` file differs significantly from that of SunOS or BSD.

Configuring `/etc/hosts.allow` and `/etc/hosts.deny`

The configuration file `/etc/hosts.allow` describes hosts that are allowed to connect to a Linux system. This configuration is not specific to NFS, but a system needs to be permitted to connect in the first place to use an NFS server. Similarly, `/etc/hosts.deny` is a list of hosts prohibited from connecting.

Slightly unintuitively, first allowed hosts are searched, then denied hosts, but anything left unmatched is granted access. This does not mean that the login mechanisms of individual servers are not still operative, but a cautious administrator might deny anything not explicitly permitted (a little paranoia is good) by using:

```
# /etc/hosts.deny
ALL:ALL EXCEPT localhost:DENY
```

With an `/etc/hosts.deny` set to deny everything (except connections from `LOCALHOST`), only those connections explicitly permitted will be allowed. For example:

```
#/etc/hosts.allow
# Allow localhost and intra-net domain to use all servers
ALL : 127.0.0.1, 192.168.
# Let everyone ssh here except 216.73.92.* and .microsoft.com
sshd: ALL EXCEPT 216.73.92. .microsoft.com : ALLOW
# Let users in the *.example.net domain ftp in
ftpd: .example.net
```

Configuring /etc/exports

Here's a sample /etc/export file:

```
# sample /etc/exports file / master(rw)
trusty(rw,no_root_squash) /projects proj*.local.domain(rw)
/usr *.local.domain(ro) @trusted(rw) /home/joe
pc001(rw,all_squash,anonuid=150,anongid=100) /pub
(ro,insecure,all_squash)
```

Normally, `root` (uid 0) on the client is treated as `nobody` (uid 65534) on the server; this is called *root squashing* since it protects files owned by root (and not group/other writable) from being altered by NFS clients. The `no_root_squash` tag disables this behavior and allows the root user on `trusty` full access to the / partition. This can be useful for installing and configuring software.

The `/usr` partition will be read only for all hosts except those in the "trusted" netgroup.

When `/home/joe` is mounted by `pc001`, all remote users (regardless of uid/gid) will be treated as if they have uid=150, gid=100. This is useful if the remote NFS client is a single-user workstation or does not support different users (like with DOS).

Normally, Linux (and other Unix-like operating systems) reserves the use of TCP and UDP ports 1-1023 (so called *secure ports*) for use by processes running as root. To ensure that the root user has initiated a remote NFS mount, the NFS server normally requires remote clients to use secure ports when mounting NFS filesystems. This convention, however, is not honored by some operating systems (notably Windows). In such cases, the *insecure* option allows the NFS client to use any TCP/UDP port. This is usually required when serving Windows clients.

NFS utilities

`nfsstat` displays a time series of NFS-related statistics (client and/or server) regarding the local machine similar to `iostat` and `vmstat`.

The `showmount` command queries `mountd` and shows which clients are currently mounting filesystems. As NFS is a stateless protocol and the `mountd` daemon is queried infrequently, the output of `showmount` can become inaccurate. Unfortunately, there is not really any way to force `showmount` to become accurate. However, where it is inaccurate, `showmount` almost always errs in showing stale mounts rather than omitting active mounts (relatively harmlessly).

In this context, "stateless" means that the `nfsd` daemons that serve up the actual file data have no memory of which files are open, nor even which clients have which

partitions mounted. Each request (readblock, writeblock, etc.) contains all the information needed to complete it (partition id provided by `mountd`, inode number, block number, read/write/etc., data). The HTTP protocol is similar in this respect. An upside of statelessness if the server reboots, the clients will notice only a brief period of interrupted access.

Section 3. Configuring a samba server

Samba server configuration

The Samba server `smbd` provides file and print services (largely for Windows clients). While it can be started from `inetd`, it is typically run as a stand alone daemon `smbd -D`. `nmbd` is the NetBios nameserver (or WINS server). It too can be run from `inetd`, but is more typically run as a stand alone daemon `nmbd -D`. Samba can function as a server in a Windows WORKGROUP, as well as Primary Domain Controller.

The configuration file for both `smbd` and `nmbd` is `/etc/samba/smb.conf`. Copious configuration parameters are described in the `smb.conf` manpage. The `lmhosts` file is used to map NetBios names to IP addresses. Its format is similar to (but not identical to) the `/etc/hosts` file.

There are several excellent HOWTOs on the subject of Samba configuration as well as several books. This section touches on the basic ideas with pointers to more complete documentation.

Setting up a home-directory file share

The following `smb.conf` snippet allows users to access their (local) home directories from remote Samba clients:

```
[homes]
comment = Home Directories
browseable = no
```

This is usually included in the default `smb.conf` file.

Setting up a print share with CUPS

Of the numerous Unix printing systems, CUPS is the least antiquated and probably the currently most popular. Depending on your distribution, CUPS may be enabled in the default `smb.conf`. Here is a simple example of a CUPS print share:

```
[global]
load printers = yes
printing = cups
printcap name = cups

[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
public = yes
guest ok = yes
writable = no
printable = yes
printer admin = root

[print$]
comment = Printer Drivers
path = /etc/samba/drivers
browseable = yes
guest ok = no
read only = yes
write list = root
```

CUPS can provide `ppd` (Postscript printer description) files and Windows drivers for clients which, when setup properly, allows remote users to take advantage of the full range of a printer's features (color versus black-and-white, resolution, paper-tray select, double- vs. single-sided printing, etc.). Traditional Unix printing systems are quite cumbersome by comparison. Consult the `cupsaddsmb` manpage for more information.

Authentication

Samba (unlike NFS) requires individual users to authenticate with the server. As with any network-authentication service, care should be taken to insure that passwords are never passed over the network unencrypted. See the section on encrypting passwords in the `smb.conf` manpage for details.

There are a variety of mechanisms Samba can use to authenticate remote users (clients). By their nature most of these are incompatible with the standard Unix password hash. The notable exception is when passwords are passed over the wire in the clear, unencrypted, which is almost always a bad idea.

Assuming you encrypt passwords on the wire, `smbpasswd` will usually be used to set up users with an initial Samba password. The "Unix password sync" option allows `smbpasswd` to change Unix passwords whenever users change their Samba password.

Alternatively, the `pam_smb` module, when configured, can authenticate Linux users using the Samba database directly. As if that's not enough choices, LDAP can be used to authenticate Samba and/or Linux users.

Debugging Samba

When configuring a Samba server, the `testparm` (also called `smbtestparm`) command can be quite useful. It will parse the `smb.conf` file and report any problems.

The `nmblookup` command does for Samba what `nslookup` does for DNS; it queries the NetBios directory. See the `nmblookup` manpage for more details.

Samba client configuration

The `smbclient` command provides FTP-like access to a Samba file share. Transparent access to SMB file shares is trickier; see the `smbmount` manpage or the `sharif` package for more info.

Section 4. Configuring File Transfer Protocol servers

About FTP

FTP is an old and widely used network protocol. FTP is normally run over two separate ports, 20 and 21. Port 21 is used as a control stream (transmitting login information and commands) while port 20 is used as the data stream over which actual file content is transmitted.

Generally, FTP is not considered a very secure protocol in the sense that in its default mode of operation, control information -- login passwords -- are transmitted in the clear. For that matter, data streams are also unencrypted, but FTP shares that feature with NFS and Samba (for secure data channels, SSH/SCP is a better choice). It *is* possible to layer FTP's control port over SSH, hence protecting control information.

Traditional FTP clients provide their own shell environment over which to transmit control commands and configure connections. Sometimes GUI frontends are used to provide friendlier interfaces to FTP transfers. However these days, many

non-dedicated tools incorporate FTP -- everything from file managers to text editors are often happy to work with files served by an FTP server.

Anonymous FTP

For what FTP is most often used for, security is not usually an issue. Probably most often, FTP servers are used for "anonymous FTP" -- that is, data that is available to the world at large and therefore doesn't require much security. By convention, a username of *anonymous* is configured to allow access and an identifying password (often an email address) is requested but not verified. Sometimes a username/password is required, but such a combination is provided without any deep user authentication (for instance, with people who want to volunteer for a project).

Most Web browsers and many file managers and tools support FTP servers transparently. Often these tools will use an FTP URL to request a file (or also to upload a file to a server). For example, the command-line tool `wget` will retrieve files from FTP servers using the following:

```
$ wget ftp://example.net/pub/somefile
$ wget ftp://user:passwd@example.net/pub/somefile
```

File managers will often mount an FTP server in a manner that is essentially identical to a local filesystem or NFS or Samba drive (this does not, however, use the `mount` and `/etc/fstab` system; such pseudo-partitions are usually named by their URL).

Choices of FTP servers

Given the age and ubiquity of FTP, a bewildering number of implementations are available and installed with various Linux distributions. Configuring the FTP server you decide to use will require a visit to the documentation accompanying the particular server.

Some popular Linux FTP servers include

- `wu-ftp`.
- `vsftpd`.
- `ProFTPD`.
- `BSD ftpd`.

- TUX FTP.

There are many less used ones as well. In most every case, the configuration of a server will live in a file like `/etc/FOOftpd.conf` (for an appropriate value of "FOO"). I am fond of `vsftpd` which is both fast and avoids known security glitches (the "vs" stands for "very secure").

A sample FTPd configuration file

Given the wealth of servers, configuration syntaxes will differ. But a few concepts taken from `/etc/vsftpd.conf` illustrate the types of options other servers provide. For `vsftpd`, each option takes the form `option=value` with the usual hash marks for comment lines. Most other FTPd configuration files are similar.

- `anonymous_enable`: Controls whether anonymous logins are permitted.
- `anon_world_readable_only`: When enabled, anonymous users will only be allowed to download world-readable files.
- `chroot_local_user`: If enabled, local users will be placed in a `chroot()` jail in their home directory after login.
- `pasv_enable`: Should the server use the "passive FTP" style in which clients initiate ports (helps with firewalls at clients).
- `ssl_enable`: If enabled, `vsftpd` will support SSL secure connections.
- `tcp_wrappers`: If enabled incoming connections will be fed through access control (like `/etc/hosts.allow` and `/etc/hosts.deny`).

Launching an FTP server

In the simplest case, you may start an FTP server the same way you might launch any daemon:

```
% sudo vsftpd
```

At this point the server will listen for incoming connections according the rules configured in its configuration file. You may also launch an FTP server from an "network super-server" such as `inetd` or `xinetd`. The LPI 202 tutorials will discuss these super-servers.

Launching a daemon individually, even if in appropriate startup scripts -- either for a particular runlevel or in `/etc/rcS.d/` -- gives you finer control over the behavior of an

FTP server.

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Introduction to Samba](#)" (developerWorks, June 2000) is a two-part series that shows how to set up and configure a Samba server.
- [Samba Installation, Configuration, and Sizing Guide](#) (IBM Redbook, May 2003) gives you the basics of installing and configuring Samba.
- "[Using Network File System](#)" (developerWorks, February 2005) is a primer on using NFS.
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the authors

Brad Huntting

Brad has been doing UNIX® systems administration and network engineering for about 14 years at several companies. He is currently working on a Ph.D. in Applied Mathematics at the University of Colorado in Boulder, and pays the bills by doing UNIX support for the Computer Science department.

David Mertz, Ph.D.

David Mertz is Turing complete, but probably would not pass the Turing Test. For

more on his life, see his [personal Web page](#). He's been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#) .

LPI exam prep: Network client management

Intermediate Level Administration (LPIC-2) topic 210

Skill Level: Intermediate

[David Mertz \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer

Gnosis Software, Inc.

24 May 2006

In this tutorial, the fifth in a [series of seven tutorials](#) covering intermediate network administration on Linux, David Mertz continues preparing you to take the Linux Professional Institute Intermediate Level Administration (LPIC-2) Exam 202. By following this tutorial, you will examine several protocols' centralized configuration of network settings on clients within a network. DHCP is widely used to establish basic handshaking to clients machines such as assigning IP addresses. At a higher level, NIS and (more often) LDAP are used for arbitrary shared information among machines on a network. This tutorial also discusses PAM, which is a flexible, networked, user authentication system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each

exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 202: Tutorials and topics		
LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic 205): Networking configuration	Learn how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	Learn how to use Linux as a mail server and as a news server. Learn about mail transport, local mail filtering, mailing list maintenance software, and server software for the NNTP protocol.
Topic 207	LPI exam 202 prep (topic 207): DNS	Learn how to use Linux as a DNS server, chiefly using BIND. Learn how to perform a basic BIND configuration, manage DNS zones, and secure a DNS server.
Topic 208	LPI exam 202 prep (topic 208): Web services	Learn how to install and configure the Apache Web server, and learn how to implement the Squid proxy server.
Topic 210	LPI exam 202 prep (topic 210): Network client management	(This tutorial) Learn how to configure a DHCP server, an NIS client and server, an LDAP server, and PAM authentication support. See detailed objectives below.
Topic 212	LPI exam 202 prep (topic 212): System security	Coming soon
Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	Coming soon

To start preparing for certification level 1, see the [developerWorks tutorials for LPI exam 101](#). To prepare for the other exam in certification level 2, see the [developerWorks tutorials for LPI exam 201](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Network client management," the fifth of seven tutorials covering intermediate network administration on Linux. In this tutorial, you learn about several protocols' centralized configuration of network settings on clients within a network, look at how DHCP is widely used to establish basic handshaking to clients machines (such as assigning IP addresses), and see how, at a higher level, NIS and (more often) LDAP are used for arbitrary shared information among machines on a network. This tutorial also discusses PAM (Pluggable Authentication Module), a flexible, networked, user-authentication system.

As with the other tutorials in the developerWorks 201 and 202 series, this tutorial is intended to serve as a study guide and entry point for exam preparation, rather than complete documentation on the subject. Readers are encouraged to consult LPI's [detailed objectives list](#) and to supplement the information provided here with other material as needed.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Web services: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
2.210.1 DHCP configuration	Weight 2	Configure a DHCP server. This objective includes setting default and per client options, adding static hosts and BOOTP hosts. Also included is configuring a DHCP relay agent and maintaining the DHCP server.
2.210.2 NIS configuration	Weight 1	Configure an NIS server. This objective includes configuring a system as an NIS client.
2.210.3 LDAP configuration	Weight 1	Configure an LDAP server. This objective includes working with directory hierarchy, groups, hosts, services, and adding other data to the hierarchy. Also included is importing and adding items, as well as adding and managing users.
2.210.4	Weight 2	Configure PAM to support

PAM authentication

authentication using various available methods.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. Introduction

About DHCP

Dynamic Host Configuration Protocol (DHCP) is a successor to the older BOOTP protocol. The principle role of a DHCP server is to assign IP addresses to client machines that may connect or disconnect from a network. Most IP networks, even those with stable topologies and client lists, use DHCP to prevent conflicts in IP address allocation.

Additionally, a DHCP server provides clients with routing and subnet information, DNS addresses, and in some cases other information. DHCP assignments may have varying durations, ranging from short to infinite, depending on server configuration and client request details. In fact, DHCP is consistent with assigning fixed IP addresses to specific machines (via their MAC hardware addresses), but in any case prevents conflicts among machines.

The formal specification of DHCP is RFC 2131 (see [Resources](#) later in this tutorial for a link).

About NIS

The Network Information Service (NIS) is Sun Microsystems' "Yellow Pages" (YP) client-server directory service protocol for distributing system configuration data such as user and host names on a computer network.

NIS/YP is used for keeping a central directory of users, hostnames, and most other useful things in a computer network. For example, in a common UNIX environment, the list of users (for authentication) is placed in `/etc/passwd`. Using NIS adds another

"global" user list which is used for authenticating users on any host.

For the most part, NIS has been superseded by the more general and more secure LDAP for general use.

A good starting point for further information on NIS is the "The Linux NIS(YP)/NYS/NIS+ HOWTO" (see [Resources](#) for a link).

About LDAP

The Lightweight Directory Access Protocol (LDAP) is a client-server protocol for accessing directory services, specifically X.500-based directory services.

An LDAP directory is similar to a database, but tends to contain more descriptive, attribute-based information. As such, LDAP provides enough flexibility for storing any type of network-shared information. The information in a directory is read much more often than it is written, so it is tuned to give quick-response to high-volume lookup or search operations.

LDAP has the ability to replicate information widely in order to increase availability and reliability while reducing response time. When directory information is replicated, any temporary inconsistencies between replicas will become synced over time.

The formal specification of LDAP is RFC 2251 (see [Resources](#) for a link).

About PAM

Linux-PAM (Pluggable Authentication Modules for Linux) is a suite of shared libraries that enable the local system administrator to choose how applications authenticate users.

A PAM-aware application can switch at runtime between authentication mechanism(s). Indeed, you may entirely upgrade the local authentication system without recompiling the applications themselves. This PAM library is configured locally with a system file, `/etc/pam.conf` (or a series of configuration files located in `/etc/pam.d/`) to authenticate a user request via the locally available authentication modules. The modules themselves will usually be located in the directory `/lib/security` and take the form of dynamically loadable object files.

The Linux-PAM System Administrators' Guide is a good starting point for further information (see [Resources](#) for a link).

Other resources

As with most Linux tools, it is always useful to examine the manpages for any utilities discussed. Versions and switches might change between utility or kernel version or with different Linux distributions. For more in-depth information, the Linux Documentation Project has a variety of useful documents, especially its HOWTOs. A variety of books on Linux networking have been published; I have found O'Reilly's *TCP/IP Network Administration*, by Craig Hunt to be quite helpful. (See [Resources](#) for links.)

Section 3. DHCP configuration

The overall protocol

As with many network protocols, the Dynamic Host Configuration Protocol (DHCP) is a client/server interface. A DHCP client is a much simpler program, both internally and to configure, than is a DHCP server. Essentially, the job of a DHCP client is to broadcast a DHCPDISCOVER message on its local physical subnet, then await a response.

The DHCPDISCOVER message MAY include options that suggest values for the network address and lease duration. Servers that receive a DHCPDISCOVER message should respond to the requesting MAC address with a DHCPOFFER message. The client, in turn, responds with a DHCPREQUEST message to one of the offering servers, usually to the first (and only) responding server.

The actual configuration parameters a client uses are received in a DHCPACK message. At that point, the client has received an allocated IP address and its communications will move, so to speak, from the Data Link Layer (Ethernet) to the Network Layer (IP).

The client process

A DHCP client typically only needs to be configured with the set of information it wishes to obtain. For example, Debian-based distributions typically use the DHCP client, `dhclient`, which is configured with the `/etc/dhcp3/dhclient.conf` file. The sample file that is distributed with the `dhcp3-client` package has all but one configuration option commented out. The one enabled option might look like:

Listing 1. Option for `dhclient.conf`

```
request subnet-mask, broadcast-address, time-offset, routers,  
domain-name, domain-name-servers, host-name,  
netbios-name-servers, netbios-scope;
```

In this example, the default configuration, the client essentially just says "ask for everything possible." In the negotiation messages, the DHCPACK message from the server will contain information for all these requested values which the client will use once they are received. Client IP address is implied in this list since that configuration is always negotiated.

As well as specifying timeout and lease time options (and a few others), a client *may*, but need not in most cases, put some restrictions on the IP addresses it wishes to use. To exclude a particular one, you can use `reject 192.33.137.209;`. To specify the explicit address the client wishes to use, then use `fixed-address 192.5.5.213;`.

A client may reject a lease offer with the DHCPDECLINE message, but servers will try to fulfill requests where possible. A DHCP server may also make a fixed assignment of a particular IP address to a requesting MAC address; configuring a per-machine IP address is more often done with server configuration than with client configuration.

In order to keep track of acquired leases, dhclient keeps a list of leases it has been assigned in the `/var/lib/dhcp3/dhclient.leases` file (the path may vary across distros); this way a non-expired DHCP lease is not lost if a system disconnects from the physical network and/or reboots.

The server process

A DHCP server needs to know a bit more about its options since it provides various information to clients in DHCP leases and also must assure that IP addresses are uniquely assigned per client. The DHCP server usually runs as the daemon, `dhcpd`, and takes its configuration information from `/etc/dhcpd.conf` (this path may vary across distros). A single `dhcpd` daemon may manage multiple subnets, generally if multiple physical networks connect to a server; most frequently however, one server manages one subnet. Listing 2 is a fairly complete example of a server configuration.

Listing 2. dhcpd.conf configuration options

```
# default/max lease in seconds: day/week  
default-lease-time 86400;  
max-lease-time 604800;  
option subnet-mask 255.255.255.0;  
option broadcast-address 192.168.2.255;  
option routers 192.168.2.1;  
# DNS locally, fallback to outside local domain
```

```
option domain-name-servers 192.168.2.1, 151.203.0.84;
option domain-name "example.com";
# Set the subnet in which IP address are pooled
subnet 192.168.2.0 netmask 255.255.255.0 {
    range 192.168.2.100 192.168.2.199;
}
# Assign a fixed IP to a couple machines by MAC address
group {
    use-host-decl-names true;
    host first.example.com {
        hardware ethernet 00:00:c1:a2:de:10;
        fixed-address 192.168.2.200;
    }
    host second.example.com {
        hardware ethernet 00:dd:cc:a1:78:34;
        fixed-address 192.168.2.201;
    }
}
```

When a client sends out a broadcast message to a server running with this configuration, it will either receive a lease on 192.168.2.200 or 192.168.2.201 if it has the specified MAC address, or it will receive a lease on an available address in the pool 192.168.2.100 through 192.168.2.199.

A client may also use the DHCPINFORM message to tell a server that it already has an assigned IP address (by manual configuration), but wishes to obtain other configuration information. Notice that informing a server that a client *is* using a particular IP address is not the same as a requesting a specific IP address; in the latter case, the server may or may not grant the request depending on existing leases. In the former case, the server has no voice in the decision and no lease is granted per se at all (however, servers will try to avoid assigning IP addresses known to be in use to new requesting clients).

When leases expire, clients and servers must negotiate new leases for configuration parameters to remain valid. Shorter leases may be used where configuration information on a server is likely to change (for example, with dynamic DNS via an external WAN). A client may gracefully terminate a lease by sending the DHCPRELEASE message, but this is not required for correct operation (clients sometimes crash, reboot, or become disconnected without the opportunity to send this message, after all).

Absent a release message, a lease is maintained by the server for whatever time terms it was granted on, so a rebooted machine will often continue using its prior lease (which will be stored in `dhclient.leases` on both server and client).

Section 4. NIS configuration

When to use NIS

Most of the utilities associated with NIS are still named with a *yp* prefix because it was formerly known as "Sun Yellow Pages"; trademark issues forced the name change to NIS. NIS is used to let a network of machines share information such as users and groups (the contents of `/etc/passwd` and `/etc/group`, respectively), allowing users rights on any machine within a NIS domain.

NIS operates in a manner similar to DNS in defining domains where information is distributed and also in allowing master and slave servers to hierarchically distribute information within a domain. In fact, in principle NIS could be used in place of DNS by distributing domain name information found in `/etc/hosts`, but this is rarely done in practice. NIS has a certain flexibility in that any type of information can, in principle, be put into a NIS database (which is in DBM format, though the tool `makedbm` in the NIS server package converts flat files to this format, generally "behind the scenes").

There is also a service called NIS+ which was intended to supersede NIS and includes data encryption and authentication; however, NIS+ is not backwards compatible with NIS and is not widely used.

Before you start

To run any of the NIS utilities you need to run the daemon `/sbin/portmap` which converts RPC program numbers into TCP/IP (or UDP/IP) protocol port numbers since NIS clients make RPC calls. Most Linux distributions launch `/sbin/portmap` in their startup scripts, but you should check that this daemon is running with `% ps -ax | grep portmap`.

If not already running, install `/sbin/portmap` and include it in your system startup scripts.

NIS client utilities (ypbind daemon)

A NIS client includes the tools **ypbind**, **ypwhich**, **ypcat**, **yppoll**, and **ypmatch**. The daemon `ypbind` must run as root, and is normally launched as part of system startup scripts (though it is not required to do so).

The other tools rely on the services of `ypbind` but run at a user level. Old version of `ypbind` broadcast a binding request on the local network, but this allows a malicious NIS server to answer the request and provide bad user and group information to clients. It is preferable to configure specific servers for `ypbind` to connect to using the `/etc/yp.conf` file. If multiple servers are configured (or if broadcast is used despite the danger), `ypbind` may switch bound servers each 15 minutes according to which is

NIS sources themselves are configured in `/etc/nsswitch.conf`. The name might suggest this is strictly for name server lookup, but a variety of information types are described. Basically, this configuration describes the order in which information sources are searched. The name `nis` in an order means information obtained from a NIS server; the name `files` means to use an appropriate local configuration file. The name `dns` is used for the `hosts` option.

As well, you may specify what to do if an initial source does not contain the information desired: `return` means to give up (and unless NIS was simply altogether unresponsive, `continue` means to try the next source for that datum). For example:

Listing 6. `/etc/nsswitch.conf`

```
passwd:      compat
group:       compat
shadow:      compat
hosts:       dns [!UNAVAIL=return] files
networks:    nis [NOTFOUND=return] files
ethers:      nis [NOTFOUND=continue] files
protocols:  nis [NOTFOUND=return] files
rpc:         nis [NOTFOUND=return] files
services:    nis [NOTFOUND=return] files
```

NIS client user utilities

The utilities **ypwhich**, **ypcat**, **ypoll**, and **ypmatch** are used at a user level to query NIS information:

- **ypwhich** prints the name of a NIS server.
- **ypcat** prints the values of all the keys in a NIS database.
- **ypoll** prints the version and master server of a NIS map.
- **ypmatch** prints the values of one or more keys from a NIS map.

See the corresponding manpages of each utility for more usage information.

NIS server utilities (**ypinit**, **ypserv**)

A NIS server uses the `ypserv` daemon to provide NIS databases to clients and is configured with the `/etc/ypserv.conf` file. As I mentioned, you may run both master and slave NIS servers within a domain. The set of NIS databases is initialized on a master server using (just the first time you run it; after that use `make -C /var/yp` like this: `% /usr/lib/yp/ypinit -m`).

A slave server is really just a NIS client that gets its databases from the master server (and runs ypserv). To copy master server information to the locally running slave server, use `% /usr/lib/yp/ypinit -s my.nist.domain`.

On the master server, the NIS databases are built from information in (some of) the following familiar configuration files:

- /etc/passwd,
- /etc/group,
- /etc/hosts,
- /etc/networks,
- /etc/services,
- /etc/protocols,
- /etc/netgroup,
- /etc/rpc.

Exactly what databases are exported is configured in /var/yp/Makefile which also propagates changes when it is rebuilt.

Slave servers will be notified of any change to the NIS maps when they are rebuilt (via the yppush program) and automatically retrieve the necessary changes in order to synchronize their databases. NIS clients do not need to do this since they continually talk to the NIS server to read the information stored in its DBM databases.

Section 5. LDAP configuration

When to use LDAP

In principle, the Lightweight Directory Access Protocol is similar in purpose to NIS. Both distribute some structured information about network configurations from client to server; however, LDAP goes further in hierarchically structuring which information is distributed to which clients, redirecting requests to other LDAP servers where necessary and in building in security mechanisms. Moreover, LDAP provides mechanisms and tools for clients to update information held in LDAP servers which in turn distribute that information to other clients requesting it (subject to

permissions, of course).

Installation

Before you run OpenLDAP (the Free Software implementation generally used on Linux, though some commercial implementations exist), you will need to install or verify the existence of several requisite libraries:

- The OpenSSL Transport Layer Security (TLS) services may be obtained from the [OpenSSL Project](#) (or via the install mechanisms of your Linux distribution).
- Kerberos Authentication Services are optionally supported, but this is very desirable. Either [MIT Kerberos](#) or [Heimdal Kerberos](#) will work.
- Simple Authentication and Security Layer may be installed as part of your base distro, but may be obtained as [Cyrus SASL](#) as well.
- [Sleepycat Software Berkeley DB](#) is recommended, though other DBM implementations are probably compatible.
- Posix threads and TCP wrappers are desirable, if not strictly required.

Given that you have satisfied these prerequisites, download the [OpenLDAP library](#) and perform the more-or-less usual dance:

Listing 7. The usual OpenLDAP install dance

```
% ./configure
% make depend
% make
% make test # make sure things went OK
% su root -c 'make install'
```

After basic installation, you need to configure the slapd configuration, usually at `/usr/local/etc/openldap/slapd.conf`. Setup should include your domain components:

Listing 8. Domain components to include with slapd.conf

```
database bdb
suffix "dc=eng,dc=uni,dc=edu,dc=eu"
rootdn "cn=Manager,dc=eng,dc=uni,dc=edu,dc=eu"
rootpw <secret>
directory /usr/local/var/openldap-data
```

In order to find a value for `<secret>`, use the utility `slappasswd` and use this encrypted base64 encoded string for your "`<secret>`":

Listing 9. Discovering your "secret"

```
% slappasswd
New password: *****
Re-enter new password: *****
{SSHA}YzPqL5Jio2+17NFiy/pAz8pqS5Ko13fH
```

To get more information on permission, replication, and other options you can configure in `slapd.conf`, take a careful look at the detailed manpages.

Launching the `slapd` daemon is pretty much like starting any daemon; you can test to see it worked with `ldapsearch`:

Listing 10. Testing to see if `slapd` worked

```
su root -c /usr/local/libexec/slapd
ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
```

If all went well, you should see something like this:

Listing 11. Working `slapd` response

```
dn:
namingContexts: dc=eng,dc=uni,dc=edu,dc=eu
```

Adding data with an LDIF file

The data format used in LDAP is a binary format, but a ASCII serialization called LDAP Data Interchange Format (LDIF) is used for exporting and importing data to an LDAP database. Binary data in LDIF is represented as base64 encoding. OpenLDAP includes tools for exporting data from LDAP servers to LDIF (**ldapsearch**), importing data from LDIF to LDAP servers (**ldapadd**), and applying a set of changes described in LDIF to LDAP servers (**ldapmodify** and **ldapdelete**).

Moreover, LDIF is one of the formats for importing and exporting address book data for the Mozilla Application Suite and other user application-level tools. Even Microsoft Windows 2000 Server and Windows Server 2003 include an LDIF tool, **LDIFDE**, to transfer data to and from Active Directory.

To manually add information to an LDAP server, first create an LDIF file:

Listing 12. Creating the sample LDIF file, `example.ldif`

```
dn: dc=example,dc=com
objectclass: dcObject
```

```
objectclass: organization
o: Example Company
dc: example

dn: cn=Manager,dc=example,dc=com
objectclass: organizationalRole
cn: Manager
```

Then add it using `% ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f example.ldif`.

Obviously, you replace the example.com domain with one appropriate to your site. As a rule, LDAP domain hierarchies and names match those used by familiar DNS names. You will be required to enter the `rootpw` value you specified in `slapd.conf`.

Querying LDAP databases

There is a tool, **slurpd** (Standalone LDAP Update Replication Daemon), that replicates a complete database of information; but for individual data values, you will either use a tool like `ldapsearch` or more likely LDAP support will be built into some application you run. The **slapcat** tool is also available for dumping an LDAP database into LDIF. For example, many Mail User Agents (MUAs) can use LDAP to locate address and contact information.

Within applications, including ones you might program yourself using application or scripting languages, LDAP resources may be accessed with LDAP URLs. These have the form of

```
ldap://host:port/dn?attributes?scope?filter?extensions.
```

Most of these fields are optional. The default hostname is `localhost`; the default port is `389`. The default root distinguished name is the empty string. If you need authentication information, specify it in the extensions portion of the URL.

In addition to LDAP URLs, many LDAP servers and clients also support the non-standard but widely used LDAPS URLs. LDAPS URLs use SSL connections instead of plaintext connections and use a default port of `636`:

```
ldaps://host:port/dn?attributes?scope?filter?extensions.
```

Section 6. PAM authentication

When to use PAM

The first thing to keep in mind about Pluggable Authentication Modules (PAM) is that it is not an application or protocol itself. Rather, this is a collection of libraries that applications may have been compiled to utilize. If an application is PAM-enabled, the security policy of that application can be configured by a system administrator without modifying or upgrading the application itself. Many Linux tools, especially daemons and servers, are PAM-enabled.

A quick way to check if a given tool is *probably* PAM-enabled is to use `ldd` to check which libraries it uses. For example, I might wonder whether my login utility is PAM-enabled:

Listing 13. Is my login PAM-enabled?

```
% ldd /bin/login | grep libpam
libpam.so.0 => /lib/libpam.so.0 (0xb7fa8000)
libpam_misc.so.0 => /lib/libpam_misc.so.0 (0xb7fa5000)
```

The use of `libpam.so` and `libpam_misc.so` by `login` does not fully guarantee that the PAM facilities are actually being used and used correctly by this tool, but it is a good suggestion. Likewise, perhaps I wonder similarly about my Apache and FTP servers:

Listing 14. How about Apache and FTP servers?

```
% ldd /usr/sbin/apache2 | grep libpam
% ldd /bin/login | grep libpam
libpam.so.0 => /lib/libpam.so.0 (0xb7fa8000)
libpam_misc.so.0 => /lib/libpam_misc.so.0 (0xb7fa5000)
```

So I know my particular Apache installation is not PAM-enabled (though versions are available that include this).

To check more thoroughly if PAM is fully working with a given tool, you can create a PAM configuration file for the program. For example, to test the login utility, you might create a file `/etc/pam.d/login` (but notice that it probably already exists on your system with a more meaningful setting, so do not delete the existing copy):

Listing 15. Checking login for PAM with `etc/pam.d/login`

```
auth      required      pam_permit.so
auth      required      pam_warn.so
```

Now running a properly PAM-enabled `login` will let anyone login, but it will log the action to the system log. If `syslog` shows an entry, PAM is enabled for this application. Readers will notice that this is about the worst configuration you could invent for `login` since it gives *anyone* shell access. Having noticed that, be warned

that PAM should be configured with a certain caution.

PAM configuration

PAM works with two different types of configuration files. Most `libpam.so` libraries are compiled in the "use the better one if available, but settle for the worse one" mode. However, you might also have a PAM library compiled as "use the better one, but also check the worse one." Let me explain that.

The currently preferred way to configure PAM is with files in the directory `/etc/pam.d/` that are named the same as the service whose security they describe. An older and less preferred style used a single file, `/etc/pam.conf`, to set security policy for all applications. From a maintainability point of view, the per-application configuration files are just easier to work with and may also be symlinked to "duplicate" a security policy from one application in another. Both configuration styles look basically the same. The single `/etc/pam.conf` file contains lines of the form:

Listing 16. Both configuration files contain

```
<service> <module-type> <control-flag> <module-path> <args>
```

In per-application configuration files, the first field is omitted since it is already the same as the filename. In the older style, the test-only login configuration we saw would look like:

Listing 17. `/etc/pam.conf`

```
login    auth    required    pam_permit.so
login    auth    required    pam_warn.so
```

The `<module-type>` field may have one of four values:

- `auth` (authentication),
- `account` (non-authentication permissions based on system of user status),
- `session` (perform actions before/after service used), and
- `password` (update user authentication tokens).

The `<control-flag>` field is used to "stack" modules which allows you rather subtle control of when a method is required, whether it's performed at all, and when some other fallback is acceptable. Its options are

- required,
- requisite,
- sufficient,
- optional, and
- include.

I will discuss these in the next panel.

The `<module-path>` we have seen in our examples; it names a shared library either in the expected module location if no path is given or at an explicit location if it starts with a `"/`. For example, in Listing 17, you might have specified `/lib/security/pam_warn.so`. The `<args>` might be anything, depending on what a particular module needs to configure its operation, though a few generic arguments should be supported by most PAM modules. Notice that PAM modules are extensible. If someone writes a new PAM module, you can simply drop it into `/lib/security` and all your applications can use it once their configuration file is updated to indicate that.

A PAM permission example

To see how the `<control-flag>` works, let's develop an example that is moderately complex. First thing we should do is create a special *OTHER* service. If this exists and no PAM policy is defined for a service, *OTHER*'s policy is used. A safe default might be like Listing 18:

Listing 18. `/etc/pam.d/other`

```
auth        required    pam_warn.so
auth        required    pam_deny.so
@include    safe-account
@include    safe-password
@include    safe-session
```

In this example, an attempt at authentication is first logged to syslog and is then denied. The `@include` statements just include contents from elsewhere, such as `/etc/pam.d/safe-account` and friends, where these "safe" definitions might contain similar warn-then-deny instructions for the other `<module-type>`'s.

Now let's configure access for our hypothetical classified-db application. Being rather concerned about access, for a user to use this application, he or she needs to provide either a matched retinal print or a fingerprint and also enter a password. The password, however, might be stored in either the local `/etc/passwd` and `/etc/shadow` configurations or available via one of two outside database servers.

None of the security modules I use in this example actually exist (to my knowledge), except `pam_unix.so` which is old-fashioned UNIX-style password access.

Listing 19. `/etc/pam.d/classified-db`

```

auth      optional    pam_unix.so
auth      optional    pam_database1.so    try_first_pass
auth      optional    pam_database2.so    use_first_pass
auth      requisite   pam_sompasswd.so
auth      sufficient  pam_fingerprint.so  master=file1 7points
auth      required   pam_retinaprint.so

```

The flow through this configuration is modestly complex. First we try password authentication by three `optional` module types. Since these are `optional`, failing one does not stop the authentication process nor satisfy it. The standard UNIX authentication password is tried first (the user is prompted to enter a password). After that, we check passwords in `database1`, but we first use the generic module argument `try_first_pass` to see if the UNIX password is the same one in the database; only if it is not do we prompt for an additional password. For `database2` however, we only try to authenticate using the UNIX password that the user entered (generic argument `use_first_pass`).

Having checked against some `optional` password systems, we use a hypothetical `pam_sompasswd.so` that needs to determine whether any of the earlier password checks succeeded (perhaps using a semaphore; but how that is done is left open for hypothetical modules). The point is that since this check is `requisite`, if it fails no further authentication is done and the failure is reported.

The final two authentication checks (if they are reached) are `sufficient`. That is, satisfying one of them returns an overall success status to the calling application. So first we try a fingerprint check using `pam_fingerprint.so` (notice some hypothetical arguments are passed to the module). Only if this fails -- maybe because of the absence of a fingerprint scanner as well as for a bad fingerprint -- is the retinal scan attempted. Likewise, if the retinal scan succeeds, that is `sufficient`. However, just to demonstrate all the flags, we actually use `required` here which means that even if retinal scanning succeeds, we would continue checking other methods (but no more exist in this example, so `sufficient` would do the same thing).

There is also a way of specifying more fine-tuned compound flags for the `<control-flag>` using bracketed `[value1=action1 ...]` sets, but the basic keywords usually suffice.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Learn about [DHCP](#) (RFC 2131) by reading the formal specifications.
- Read "[The Linux NIS\(YP\)/NYS/NIS+ HOWTO](#)" to learn about NIS.
- The formal specification of [LDAP is RFC 2251](#).
- [The Linux-PAM System Administrators' Guide](#) is a good starting point for more on the Pluggable Authentication Modules.
- [TCP/IP Network Administration, Third Edition](#) by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- This list of more than [700 Linux User Groups around the world](#) can help you find local and distance study groups for LPI exams.
- For more in-depth information, the [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- In the [developerWorks Linux zone](#), find more resources for Linux developers.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the author

David Mertz

David Mertz thinks that artificial languages are perfectly natural, but natural

languages seem a bit artificial. You can reach David at mertz@gnosis.cx; you can investigate all aspects of his life at his [personal Web page](#). Check out his book, [Text Processing in Python](#). Suggestions and recommendations on past or future columns are welcome.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

LPI exam 201 prep: System maintenance

Intermediate Level Administration (LPIC-2) topic 211

Skill Level: Intermediate

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)
Developer
Gnosis Software

02 Sep 2005

In this tutorial, David Mertz begins preparing you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 201. In this sixth of eight tutorials, you learn basic concepts of system logging, software packaging, and backup strategies.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams.

Each exam covers several topics, and each topic has a weight. The weights indicate the relative importance of each topic. Very roughly, expect more questions on the exam for topics with higher weight. The topics and their weights for LPI exam 201 are:

Topic 201

Linux kernel (weight 5).

Topic 202

System startup (weight 5).

Topic 203

Filesystem (weight 10).

Topic 204

Hardware (weight 8).

Topic 209

File and service sharing (weight 8).

Topic 211

System maintenance (weight 4). The focus of this tutorial.

Topic 213

System customization and automation (weight 3).

Topic 214

Troubleshooting (weight 6).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "System maintenance," the sixth of eight tutorials designed to prepare you for LPI exam 201. In this tutorial, you learn basic concepts of system logging, software packaging, and backup strategies.

The tutorial is organized according to the LPI objectives for this topic, as follows:

2.211.1 System logging (weight 1)

You will be able to configure syslogd to act as a central network log server.

This objective also includes configuring syslogd to send log output to a central log server, logging remote connections, and using grep and other text utilities to automate log analysis.

2.211.2 Packaging software (weight 1)

You will be able to build a package. This objective includes building (or rebuilding) both RPM and DEB packaged software.

2.211.3 Backup operations (weight 2)

You will be able to create an offsite backup storage plan.

This tutorial, like the corresponding LPI exam, is a grab-bag of several topics that do

not cleanly fall into other categories. System logging and analyzing log files are important tasks for a system administrator to be familiar with. Likewise, a maintained system should carry out a sensible backup strategy using standard Linux tools.

Not every system administrator will need to create custom software packages, but administrators of multiple (similar) installations will need to install site- or company-specific software packages as part of their duties. This tutorial looks at the Debian and RPM package formats, and touches on basic "tarballs."

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. System logging

About logging

Many processes and servers under a Linux system append changing status information to "log files." These log files often live in the `/var/log/` directory and often begin with a time stamp indicating when the described event occurred. But for better or worse, there is no real consistency in the precise format of log files. The one feature you can pretty much count on is that Linux log files are plain ASCII files, and contain one "event" per line of the file. Often (but not always) log files contain a (relatively) consistent set of space- or tab-delimited data fields.

Some processes, especially Internet services, handle log file writes within their own process. At heart, writing to a log file is just an append to an open file handle. But many programs (especially daemons and `cron`'d jobs) use the standard syslog API to let the `syslogd` or `klogd` daemons handle the specific logging process.

Parsing log files

Exactly how you go about parsing a log file depends greatly on the specific format it takes. For log files with regular table format, tools like `cut`, `split`, `head`, and `tail` are likely to be useful. For all log files, `grep` is a great tool for finding and filtering contents of interest. For more complex processing tasks, you are likely to think of

sed, *awk*, *perl*, or *python* as tools of choice.

For a good introduction to many of the text processing tools you are most likely to use in processing and analyzing log files, see David's IBM developerWorks tutorial on the GNU text processing utilities. A number of good higher-level tools also exist for working with log files, but these tools are usually distribution-specific and/or non-standard (but often Free Software) utilities.

Logging with syslogd and klogd

The daemon klogd intercepts and logs Linux kernel messages. As a rule, klogd will utilize the more general syslogd capabilities, but in special cases it may log messages directly to a file.

The general daemon syslogd provides logging for many programs. Every logged message contains at least a time and a hostname field and usually a program name field. The specific behavior of syslogd is controlled by the configuration file `/etc/syslog.conf`. Application (including kernel) messages may be logged to files, usually living under `/var/log/` or remotely over a network socket.

Configuring `/etc/syslog.conf`

The file `/etc/syslog.conf` contains a set of rules, one per line. Empty lines and lines starting with a `#` are ignored. Each rule consists of two whitespace-separated fields, a selector field, and an action field. The selector, in turn, contains one of more dot-separated facility/priority pairs. A facility is a subsystem that wishes to log messages and can have the (case-insensitive) values: `auth`, `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `security`, `syslog`, `user`, `uucp`, and `local0` through `local7`.

Priorities have a specific order and matching a given priority means "this one or higher" unless an initial `=` (or `!=`) is used. Priorities are, in ascending order: `debug`, `info`, `notice`, `warning` or `warn`, `err` or `error`, `crit`, `alert`, `emerg` or `panic` (several names have synonyms). `none` means that no priority is indicated.

Both facilities and priorities may accept the `*` wildcard. Multiple facilities may be comma-separated and multiple selectors may be semi-colon separated. For example:

```
# from /etc/syslog.conf
# all kernel messages
kern.*                -/var/log/kern.log
# `catch-all' logfile
*.=info;*.=notice;*.=warn;\
  auth,authpriv.none;\
```

```

cron,daemon.none;\
mail,news.none      -/var/log/messages
# Emergencies are sent to everybody logged in
*.emerg              *

```

Configuring remote logging

To enable remote logging of syslogd messages (really application messages, but handled by syslogd), you must first enable the "syslog" service on both the listening and the sending machines. To do this, you need to add a line to each `/etc/services` configuration file containing something like:

```
syslog      514/UDP
```

To configure the local (sending) syslogd to send messages to a remote host, you specify a regular facility and priority but give an action beginning with an "@" symbol for the destination host. A host may be configured in the usual fashion, either `/etc/hosts` or via DNS (it need not be resolved already when syslogd first launches). For example:

```

# from /etc/syslog.conf
# log all critical messages to master.example.com
*.crit                @master.example.com
# log all mail messages except info level to mail.example.com
mail.*;mail.!=info   @mail.example.com

```

Rotating log files

Often you will not want to let particular log files grow unboundedly. The utility *logrotate* may be used to archive older logged information. Usually *logrotate* is run as a `cron` job, generally daily. *logrotate* allows automatic rotation, compression, removal, and mailing of log files. Each log file may be handled daily, weekly, monthly, or only when it grows too large.

The behavior of *logrotate* is controlled by the configuration file `/etc/logrotate.conf` (or some other file, if specified). The configuration file may contain both global options and file-specific options. Generally, archived logs are saved for a finite time period and are given sequential backup names. For example, one system of mine contains the following files due to its rotation schedule.

```

-rw-r----- 1 root adm 4135 2005-08-10 04:00 /var/log/syslog
-rw-r----- 1 root adm 6022 2005-08-09 07:36 /var/log/syslog.0
-rw-r----- 1 root adm  883 2005-08-08 07:35 /var/log/syslog.1.gz
-rw-r----- 1 root adm  931 2005-08-07 07:35 /var/log/syslog.2.gz
-rw-r----- 1 root adm  888 2005-08-06 07:35 /var/log/syslog.3.gz

```

```
-rw-r----- 1 root adm 9494 2005-08-05 07:35 /var/log/syslog.4.gz  
-rw-r----- 1 root adm 8931 2005-08-04 07:35 /var/log/syslog.5.gz
```

Section 3. Packaging software

In the beginning was the tarball

For custom software distribution on Linux, there is actually much less needed than you might think. Linux has a fairly clean standard about where files of various types should reside and installing custom software, at its heart, need not involve much more than putting the right files in the right places.

The Linux tool *tar* (for "tape archive," though it need not, and usually does not, utilize tapes) is perfectly adequate to create an archive of files with specified filesystem locations. For distribution, you generally want to compress a tar archive with *gzip* (or *bzip2*). See the final section of this tutorial on backup for more information on these utilities. A compressed tar archive is generally named with the extensions *.tar.gz* or *.tgz* (or *.tar.bz2*).

Early Linux distributions -- and some current ones like Slackware -- use simple tarballs as their distribution mechanism. For a custom distribution of in-house software to centrally maintained Linux systems, this continues to be the simplest approach.

Custom archive formats

Many programming languages and other tools come with custom distribution systems that are neutral between Linux distributions and usually between altogether different operating systems. Python has its *distutils* tools and archive format; Perl has CPAN archives; Java has *.jar* files; Ruby has *gems*. Many non-language applications have a standard system for distributing plug-ins or other enhancements to a base application as well.

While you can perfectly well use an package format like DEB or RPM to distribute a Python package for example, it often makes more sense to follow the packaging standard of the tool the package is created for. Of course, for system-level utilities and applications or for most compiled userland applications, that standard is the Linux distribution packaging standards. But for custom tools written in specific programming languages, something different might promote easier reuse of your

tools across distributions and platforms (whether in-house or external users are the intended target).

The "big two" package formats

There are two main package formats used by Linux distributions: Redhat Package Manager (RPM) and Debian (DEB). Both are similar in purpose but different in details. In general, either one is a format for an "enhanced" archive of files. The enhancements provided by these package formats include annotations for version numbers, dependencies of one application upon other applications or libraries, human-readable descriptions of packaged tools, and a general mechanism for managing the installation, upgrade, and de-installation of packaged tools.

Under DEB files, the nested configuration file *control* contains most of the package metadata. For RPM files, the file *spec* plays this role. The full details of creating good packages in either format is beyond this tutorial, but we will outline the basics here.

What is in a .deb file?

A DEB package is created with the archive tool and cousin of tar, *ar* (or by some higher-level tool that utilizes *ar*). Therefore we can use *ar* to see just what is really inside a .deb file. Normally we would use higher-level tools like *dpkg*, *dpkg-deb*, or *apt-get* to actually work with a DEB package. For example:

```
% ar tv unzip_5.51-2ubuntu1.1_i386.deb
rw-r--r-- 0/0      4 Aug  1 07:23 2005 debian-binary
rw-r--r-- 0/0    1007 Aug  1 07:23 2005 control.tar.gz
rw-r--r-- 0/0  133475 Aug  1 07:23 2005 data.tar.gz
```

The file *debian-binary* simply contains the DEB version (currently 2.0). The tarball *data.tar.gz* contains the actual application files -- executables, documentation, manual pages, configuration files, and so on.

The tarball *control.tar.gz* is the most interesting from a packaging perspective. Let us look at the example DEB package we chose:

```
% tar tvfz control.tar.gz
drwxr-xr-x root/root      0 2005-08-01 07:23:43 ./
-rw-r--r-- root/root    970 2005-08-01 07:23:43 ./md5sums
-rw-r--r-- root/root    593 2005-08-01 07:23:43 ./control
```

As you might expect, *md5sums* contains cryptographic hashes of all the distributed files for verification purposes. The file *control* is where the metadata lives. In some

cases you might also find or wish to include scripts called *postinst* and *prerm* in `control.tar.gz` to take special steps after installation or before removal, respectively.

Creating a DEB control file

The installation scripts can do anything a shell script might. (Look at some examples in existing packages to get an idea.) But those scripts are optional and often not needed or included. Required for a `.deb` package is its control file. The format of this file contains various metadata fields and is best illustrated by showing an example:

```
% cat control
Package: unzip
Version: 5.51-2ubuntu1.1
Section: utils
Priority: optional
Architecture: i386
Depends: libc6 (>= 2.3.2.ds1-4)
Suggests: zip
Conflicts: unzip-crypt (<< 5.41)
Replaces: unzip-crypt (<< 5.41)
Installed-Size: 308
Maintainer: Santiago Vila <sanvila@debian.org>
Description: De-archiver for .zip files
 InfoZIP's unzip program. With the exception of multi-volume archives
 (ie, .ZIP files that are split across several disks using PKZIP's /& option),
 this can handle any file produced either by PKZIP, or the corresponding
 InfoZIP zip program.
.
This version supports encryption.
```

Basically, except the custom data values, you should make your control file look just like this one. For non-CPU specific packages -- either scripts, pure documentation, or source code -- use `Architecture: all`.

Making a DEB package

Creating a DEB package is performed with the tool `dpkg-deb`. We cannot cover all the intricacies of good packaging, but the basic idea is to create a working directory, `./debian/`, and put the necessary contents into it before running `dpkg-deb`. You will also want to set permissions on your files to match their intended state when installed. For example:

```
% mkdir -p ./debian/usr/bin/
% cp foo-util ./debian/usr/bin          # copy executable/script
% mkdir -p ./debian/usr/share/man/man1
% cp foo-util.1 ./debian/usr/share/man/man1 # copy the manpage
% gzip --best ./debian/usr/share/man/man1/foo-util.1
% find ./debian -type d | xargs chmod 755 # set dir permissions
% mkdir -p ./debian/DEBIAN
% cp control ./debian/DEBIAN           # first create a matching 'control'
% dpkg-deb --build debian              # create the archive
```

```
% mv debian.deb foo-util_1.3-1all.deb # rename to final package name
```

More on DEB package creation

In the previous panel you can see that our local directory structure underneath `./debian/` is meant to match the intended installation structure. A few more points on creating a good package are worth observing.

- Generally you should create a file as part of your distribution called `./debian/usr/share/doc/foo-util/copyright` (adjust for package name).
- It is also good practice to create the files `./debian/usr/share/doc/foo-util/changelog.gz` and `./debian/usr/share/doc/foo-utils/changelog.Debian.gz`.
- The tool *lintian* will check for questionable features in a DEB package. Not everything *lintian* complains about is strictly necessary to fix; but if you intend wider distribution, it is a good idea to fix all issues it raises.
- The tool *fakeroot* is helpful for packaging with the right owner. Usually a destination wants tools installed as root, not as the individual user who happened to generate the package (*lintian* will warn about this). You can accomplish this with:

```
% fakeroot dpkg-deb --build debian
```

What is in an .rpm file?

RPM takes a slightly different strategy than DEB does in creating packages. Its configuration file is called *spec* rather than *control*, but the *spec* file also does more work than a *control* file does. All the details of steps needed for pre-installation, post-installation, pre-removal, and installation itself, are contained as embedded script files in a *spec* configuration. In fact, the *spec* format even comes with macros for common actions.

Once you create an RPM package, you do so with the `rpm -b` utility. For example:

```
% rpm -ba foo-util-1.3.spec # perform all build steps
```

This package build process does not rely on specific named directories as with DEB, but rather on directives in the more complex *spec* file.

Creating RPM metadata

The basic metadata in an RPM is much like that in a DEB. For example, `foo-util-1.3.spec` might contain something like:

```
# spec file for foo-util 1.3
Summary: A utility that fully foos
Name: foo-util
Version: 1.3
Release: 1
Copyright: GPL
Group: Application/Misc
Source: ftp://example.com/foo-util.tgz
URL: http://example.com/about-foo.html
Distribution: MyLinux
Vendor: Acme Systems
Packager: John Doe <jdoe@acme.example.com>

%description
The foo-util program is an advanced fooser that combines the
capabilities of OneTwo's foo-tool and those in the GPL bar-util.
```

Scripting in an RPM

Several sections of an RPM spec file may contain mini shell scripts. These include:

- `%prep`: Steps to perform to get the build ready such as clean out earlier (partial) builds. Often the following macro is helpful and sufficient:

```
%prep
%setup
```

- `%build`: Steps to actually build the tool. If you use the `make` facility, this might amount to:

```
%build
make
```

- `%install`: Steps to install the tool. Again, if you use `make` this might mean:

```
%install
make install
```

- `%files`: You *must* include a list of files that are part of the package. Even though your Makefile might use these files, the package manager

program (rpm) will not know about them unless you include them here:

```
%files
%doc README
/usr/bin/foo-util
/usr/share/man/man1/foo-util.1
```

Section 4. Backup operations

About backup

The first rule about making backups is: Do it! It is all too easy in server administration -- or even just with Linux on the desktop -- to neglect backing up on a schedule suited to your requirements.

The easiest way to carry out backups in a systematic and schedules way is to perform them on a `cron` job. See the Topic 213 tutorial for a discussion of configuring *crontab*. In part, the choice of backup schedule depends on the backup tool and media you choose to use.

Backup to tape is a traditional technique and tape drives continue to offer the largest capacity of relatively inexpensive media. But recently, writeable or rewriteable CDs and DVD have become ubiquitous and will often make reasonable removable media for backups.

What to back up

A nice thing about Linux is that it uses a predicable and hierarchical arrangement of files. As a consequence, you rarely need to backup an entire filesystem hierarchy; most of a Linux filesystem hierarchy can be reinstalled from distribution media easily enough. In large environments, a master server image might be used to create a basic Linux system which can be customized by restoring a few selected files that were backed up elsewhere.

Basically, what you want backed up is `/home/`, `/etc/`, `/usr/local/`, and maybe `/root/` and `/boot/`. Often you will also want to backup some parts of `/var/`, such as `/var/log/` and `/var/mail/`.

Backup with cp and scp

Perhaps the simplest way to perform a backup is with `cp` or `scp` and the `-r` (recurse) switch. The former copies to "local" media (but including NFS mounts), the latter can copy to remote servers in a securely encrypted fashion. For either, you need a mounted drive with sufficient space to accommodate the files you want to backup, of course. To gain any real hardware protection, you want the partition you copy to to be a different physical device than the drive(s) you are backing up from.

Copying with `cp` or `scp` can be part of an incremental backup schedule. The trick here is to use the utility `find` to figure out which files have been modified recently. Here is a simple example where we copy all the files in `/home/` that have been modified in the 1st day:

```
#!/bin/bash
# File: backup-daily.sh
# ++ Run this on a daily cron job ++
#-- first make sure the target directories exist
for d in `find /home -type d` ; do mkdir -p /mnt/backup$d ; done
#-- then copy all the recently modified files (one day)
for f in `find /home -mtime -1` ; do cp $f /mnt/backup$f ; done
```

The `cp -u` flag is somewhat similar, but it depends on the continuity of the target filesystem between backup events. The `find` approach works fine if you change the mount point of `/mnt/backup` to a different NFS location. And the `find` system works equally well with `scp` once you specify the necessary login information to the remote site.

Backup with tar

Although `cp` and `scp` are workable for backup, generally `tar` sees wider use, being designed specifically for creating tape archives. Despite the origin of the name, `tar` is equally capable of creating a simple `.tar` file as raw data on a tape drive. For example, you might backup to a tape drive using a command like:

```
% tar -cvf /dev/rmt0 /home # Archive /home to tape
```

To direct the output to a file is hardly any different:

```
% tar -cvf /mnt/backup/2005-08-12.tar /home
```

In fact, since `gzip` is streamable, you can easily compress an archive during creation:

```
% tar -cv /home | gzip - > /mnt/backup/2005-08-12.tgz
```

You can combine tar with find in the same ways shown for cp or scp. To list the files on a tape drive, you might use:

```
% tar -tvf /dev/rmt0
```

To retrieve a particular file:

```
% tar -xvf /dev/rmt0 file.name
```

Backup with cpio

The utility *cpio* is a superset of tar. It handles tar archives, but will also work with several other formats and has many more options built in. *cpio* comes with a huge wealth of switches to filter which files are backed up and even supports remote backup internally (rather than needing to pipe through scp or the like). The main advantage *cpio* has over tar is that you can both add files to archives and remove files back out.

Here are some quick examples of *cpio*:

- Create a file archive on a tape device: `% find /home -print | cpio -ocBv /dev/rmt0.`
- List the entries in a file archive on a tape device: `% cpio -itcvB < /dev/rmt0.`
- Retrieve a file from a tape drive: `% cpio -icvdBum file.name < /dev/rmt0.`

Backup with dump and restore

A set of tools named *dump* and *restore* (or with related names) are sometimes used to backup whole filesystems. Unfortunately, these tools are specific to filesystem types and are not uniformly available. For example, the original *dump* and *restore* are ext2/3-specific while the tools *xfsdump* and *xfsrestore* are used for XFS filesystems. Not every filesystem type has the equivalent tools and even if they do, switches are not necessarily uniform.

It is good to be aware of these utilities, but they are not very uniform across Linux systems. For some purposes -- like if you only use XFS partitions -- the performance of *dump* and *restore* can be a great boost over a simple tar or *cpio*.

Incremental backup with rsync

rsync is utility that provides fast incremental file transfer. For automated remote backups, *rsync* is often the best tool for the job. A nice feature of *rsync* over other tools is that it can optionally enforce two-way synchronization. That is, rather than simply backing up newer or changed files, it can also automatically remove locally deleted files from the remote backup.

To get a sense of the options, this moderately complex script (located at the *rsync* Web pages) is useful to look at:

```
#!/bin/sh
# This script does personal backups to a rsync backup server. You will
# end up with a 7 day rotating incremental backup. The incrementals will
# go into subdirs named after the day of the week, and the current
# full backup goes into a directory called "current"
# tridge@linuxcare.com
# directory to backup
BDIR=/home/$USER
# excludes file - this contains a wildcard pats of files to exclude
EXCLUDES=$HOME/cron/excludes
# the name of the backup machine
BSERVER=owl
# your password on the backup server
export RSYNC_PASSWORD=XXXXXX
BACKUPDIR=`date +%A`
OPTS="--force --ignore-errors --delete-excluded --exclude-from=$EXCLUDES
--delete --backup --backup-dir=/$BACKUPDIR -a"
export PATH=$PATH:/bin:/usr/bin:/usr/local/bin
# the following line clears the last weeks incremental directory
[ -d $HOME/emptydir ] || mkdir $HOME/emptydir
rsync --delete -a $HOME/emptydir/ $BSERVER::$USER/$BACKUPDIR/
rmdir $HOME/emptydir
# now the actual transfer
rsync $OPTS $BDIR $BSERVER::$USER/current
```

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Using the GNU text utilities](#)" (developerWorks, March 2004) shows you how to use the GNU text utilities collection to process log files, documentation, structured text databases, and other textual sources of data or content.
- "[Understanding Linux configuration files](#) (developerWorks, December 2001)" explains configuration files on a Linux system that control user permissions, system applications, daemons, services, and other administrative tasks in a multi-user, multi-tasking environment.
- "[Windows-to-Linux roadmap: Part 8. Backup and recovery](#)" (developerWorks, November 2003) is a quick guide to Linux backup and recovery.
- [Installation Guide and Reference: Software Product Packaging Concepts](#) discusses the concepts of software packaging.
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- Find [sample Samba scripts](#) at the rsync Web pages.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the author

David Mertz, Ph.D.

David Mertz is Turing complete, but probably would not pass the Turing Test. For more on his life, see his [personal Web page](#). He's been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#).

LPI exam prep: System security

Intermediate Level Administration (LPIC-2) topic 212

Skill Level: Intermediate

[David Mertz \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer

Gnosis Software, Inc.

13 Jun 2006

In this tutorial, the sixth in a [series of seven tutorials](#) covering intermediate network administration on Linux®, David Mertz continues preparing you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 202. By necessity, this tutorial touches briefly on a wide array of Linux-related topics from a security-conscious network server perspective, including general issues of routing, firewalls, and NAT translation and the relevant tools. It addresses setting security policies for FTP and SSH; reviews general access control with `tcpd`, `hosts.allow`, and friends; and presents some basic security monitoring tools and shows where to find security resources.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each

exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic 205): Networking configuration	Learn how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	Learn how to use Linux as a mail server and as a news server. Learn about mail transport, local mail filtering, mailing list maintenance software, and server software for the NNTP protocol.
Topic 207	LPI exam 202 prep (topic 207): DNS	Learn how to use Linux as a DNS server, chiefly using BIND. Learn how to perform a basic BIND configuration, manage DNS zones, and secure a DNS server.
Topic 208	LPI exam 202 prep (topic 208): Web services	Learn how to install and configure the Apache Web server, and learn how to implement the Squid proxy server.
Topic 210	LPI exam 202 prep (topic 210): Network client management	Learn how to configure a DHCP server, an NIS client and server, an LDAP server, and PAM authentication support. See detailed objectives below.
Topic 212	LPI exam 202 prep (topic 212): System security	(This tutorial) Learn how to configure a router, secure FTP servers, configure SSH, and perform various other security administration tasks. See detailed objectives below.
Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	Coming soon

To start preparing for certification level 1, see the [developerWorks tutorials for LPI](#)

[exam 101](#). To prepare for the other exam in certification level 2, see the [developerWorks tutorials for LPI exam 201](#). Read more about the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "System security," the sixth of seven tutorials covering intermediate network administration on Linux. In this tutorial, you learn about a wide array of topics related to using Linux as a security-conscious network server. Such issues as routing, firewalls, and NAT translation (and the tools to manage them) are covered, as well as setting security policies for FTP and SSH. You also learn about general access control with `tcpd`, `hosts.allow`, and `friends` (revisiting the discussion in [LPI exam 201 prep \(topic 209\): File and service sharing](#)). Finally, you learn about some basic security monitoring tools, as well as where to find security resources.

As with the other tutorials in the developerWorks 201 and 202 series, this tutorial is intended to serve as a study guide and entry point for exam preparation, rather than complete documentation on the subject. Readers are encouraged to consult LPI's [detailed objectives list](#) and to supplement the information provided here with other material as needed.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

LPI exam objective	Objective weight	Objective summary
2.212.2 Configuring a router	Weight 2	Configure a system to perform network address translation (NAT, IP masquerading), and state its significance in protecting a network. This objective includes configuring port redirection, managing filter rules, and averting attacks.
2.212.3 Securing FTP servers	Weight 2	Configure an FTP server for anonymous downloads and uploads. This objective includes precautions to be taken if anonymous uploads are permitted and configuring user access.
2.212.4	Weight 2	Configure an SSH daemon.

Secure shell (SSH)		This objective includes managing keys, configuring SSH for users, forwarding an application protocol over SSH, and managing the SSH login.
2.212.5 TCP_wrappers	Weight 1	Configure tcpwrappers to allow connections to specified servers only from certain hosts or subnets.
2.212.6 Security tasks	Weight 3	Install and configure a secure authentication system; perform basic security auditing of source code; receive security alerts from various sources; audit servers for open e-mail relays and anonymous FTP servers; install, configure, and run intrusion detection systems; and apply security patches and bug fixes.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Other resources

As with most Linux tools, it is always useful to examine the manpages for any utilities discussed. Versions and switches might change between utility or kernel version or with different Linux distributions. For more in-depth information, the Linux Documentation Project has a variety of useful documents, especially its HOWTOs. Also, a variety of books on Linux system security have been published; I have found O'Reilly's *TCP/IP Network Administration*, by Craig Hunt to be quite helpful. See the [Resources](#) section for links.

Section 2. Configuring a router

About packet filtering

The Linux kernel includes the "netfilter" infrastructure, which enables you to filter network packages. Usually this capability is compiled into the base kernel, but a kernel module may be needed. Either way, module loading should be seamless (for instance, running `iptables` will load `iptables_filter.o` if it needs it).

Packet filtering is controlled with the utility `iptables` in modern Linux systems; older systems used `ipchains`. Before that, it was `ipfwadm`. While you *can* still use `ipchains` in conjunction with recent kernels if backward compatibility is needed, you will almost always prefer the enhanced capabilities and improved syntax in `iptables`. That said, most of the concepts and switches in `iptables` are compatible enhancements to `ipchains`.

Depending on the exact scenario of filtering (firewall, NAT, etc.), filtering and address translation may occur either before or after routing itself. The same `ipchains` tool is used in either case, but different rule sets ("chains") are used for the cases -- at base, `INPUT` and `OUTPUT`. However, filtering can also affect the routing decision by filtering on the `FORWARD` chain; this may lead to dropping packets rather than routing them.

Routing

As well as filtering with `iptables` (or legacy `ipchains`), the Linux kernel performs *routing* of IP packets it receives. Routing is a simpler process than filtering, though the two are conceptually related.

During routing, a host simply looks at a destination IP address and decides whether it knows how deliver a packet directly to that address or whether a gateway is available that knows how to deliver to that address. If a host can neither deliver a packet itself nor knows what gateway to forward it to, the packet is dropped. However, typical configurations include a "default gateway" that handles every otherwise unspecified address.

Configuration and display of routing information is performed with the utility `route`. However, routing may either be *static* or *dynamic*.

With static routing, delivery is determined by a routing table that is explicitly configured by invocations of the `route` command and its `add` or `del` commands. However, configuring dynamic routing using the `routed` or `gated` daemons that broadcast routing information to adjacent routing daemons is often more useful.

The `routed` daemon supports the Routing Information Protocol (RIP); the `gated` daemon adds support for a number of other protocols -- and can use multiple

protocols at once -- such as:

- Routing Information Protocol Next Generation (RIPng)
- Exterior Gateway Protocol (EGP)
- Border Gateway Protocol (BGP) and BGP4+
- Defense Communications Network Local-Network Protocol (HELLO)
- Open Shortest Path First (OSPF)
- Intermediate System to Intermediate System (IS-IS)
- Internet Control Message Protocol (ICMP and ICMPv6)/Router Discovery

Let's look at a fairly typical static routing table:

Listing 1. Typical static routing table

```
% /sbin/route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
66.98.217.0 * 255.255.255.0 U 0 0 0 eth0
10.10.12.0 * 255.255.254.0 U 0 0 0 eth1
66.98.216.0 * 255.255.254.0 U 0 0 0 eth0
169.254.0.0 * 255.255.0.0 U 0 0 0 eth1
default evls-66-98-216- 0.0.0.0 UG 0 0 0 eth0
```

This means that addresses in the 66.98.217/24 and 66.98.216/23 ranges will be directly delivered over `eth0`. Address ranges 10.10.12/23 and 169.254/16 will be delivered on `eth1`. Anything left over will be sent to the gateway `evls-66-98-216-1.evlservers.net` (the name is cut off in the `route` display; you could also use `route -n` to see that name was IP address 66.98.216.1). If you wanted to add a different gateway for some other address ranges, you might run something like this:

Listing 2. Adding new gateway for other address ranges

```
% route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.2.1 dev eth0
```

For a machine that serves as a gateway itself, you will generally want to run dynamic routing, using the `routed` or `gated` daemons, which may supplement a smaller number of static routes. The `routed` daemon is configured by the contents of `/etc/gateways`. The `gated` daemon is more modern and has more capabilities (as indicated); it is configured by `/etc/gated.conf`. Generally, if you use either of these, you will want to launch them in your startup scripts. You must *not* run both `routed` and `gated` on the same machine because results will be unpredictable and almost certainly undesirable.

Filtering with iptables

The Linux kernel stores a table of filter rules for IP packets that form a sort of state-machine. Sets of rules that are processed in sequence are known as "(firewall) chains." When one chain meets a condition, one of the possible actions is to shift control to processing another chain as in a state-machine. Before you have added any rules or states, three chains are automatically present: `INPUT`, `OUTPUT`, and `FORWARD`. The `INPUT` chain is where a packet addressed to the host machine passes and potentially from there to a local application process. The `FORWARD` chain is where a packet addressed to a different machine passes, assuming forwarding is enabled and the routing system knows how to forward that packet. A packet generated on the local host is sent into the `OUTPUT` chain for filtering -- if it passes the filters in the `OUTPUT` chain (or any linked chains), it is routed out over its network interface.

One action that a rule can take is to `DROP` a packet; in that case, no further rule processing or state transition is taken for that packet. But if a packet is not dropped, the next rule in a chain is examined to see if it matches the packet. In some cases, satisfaction of a rule will branch process to a different chain and its set of rules. Creation, deletion, or modification of rules and of chains in which rules live is performed with the tool `iptables`. In older Linux systems, the same function was done using `ipchains` instead. The concepts behind both tools and even for the ancient `ipfwadm` are similar, but `iptables` syntax is discussed here.

A rule specifies a set of conditions that a packet might meet and what action to take if the packet does meet that condition. As mentioned, one common action is to `DROP` packets. For example, suppose you wanted (for some reason) to disable `ping` on the loopback interface (the `ICMP` interface). You could make this happen with:

Listing 3. Disabling on the loopback interface

```
% iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
```

Of course, that is a silly rule and we probably want to remove it after we test it, like this:

Listing 4. DROPPing the silly rule

```
% iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
```

Deleting a rule with the `-D` option requires either exactly the same options as specified when it was added or specification by rule number (which you *must* determine in the first place) like this:

Listing 5. Specifying rule number so deletion can work

```
% iptables -D INPUT 1
```

A more interesting rule might look at source and destination addresses in packets. For example, suppose that a problem remote network is trying to utilize services on a particular subnet of your network. You might block this on your gateway/firewall machine with:

Listing 6. Blocking the gateway/firewall machine

```
% iptables -A INPUT -s 66.98.216/24 -d 64.41.64/24 -j DROP
```

Doing this will stop anything from the `66.98.216.*` IP block from communicating with anything in the local `64.41.64.*` subnet. Of course, singling out a specific IP block for blacklisting is fairly limited as protection. A more likely scenario might be to *allow* only a specific IP block to access a local subnet:

Listing 7. Letting a specific IP block access a local subnet

```
% iptables -A INPUT -s ! 66.98.216/24 -d 64.41.64/24 -j DROP
```

In this case, *only* the `66.98.216.*` IP block can access the specified subnet. Moreover, you can use a symbolic name for an address and can specify a particular protocol to be filtered. You can also select a specific network interface (like `eth0`) to filter, but that is less commonly useful. For example, to let only a specific remote network access a local Web server, you might use:

Listing 8. Letting a specific remote network address access a local Web server

```
% iptables -A INPUT -s ! example.com -d 64.41.64.124 -p TCP -sport 80 -j DROP
```

You can specify a number of other options with `iptables`, including rate limits on the number of packets that will be allowed or filtering on TCP flags, for example. See the manpage for `iptables` for more details.

User-defined chains

You have seen the basics of adding rules to the automatic chains. But much of the configurability in `iptables` comes with adding user-defined chains and branching to them if patterns are matched. New chains are defined with the `-N` option; you have already seen branching using the special target `DROP`. `ACCEPT` is also a

special target with the obvious meaning. Also, special targets `RETURN` and `QUEUE` are available. The first means to stop processing a given chain and return to its parent/caller. The `QUEUE` handler lets you pass packets to a user-space process for further processing (which might be logging, modification of the packet, or more elaborate filtering than `iptables` supports). The simple example in Rusty Russell's "Linux 2.4 Packet Filtering HOWTO" is a good example of adding a user-defined chain:

Listing 9. Adding a user-defined chain

```
# Create chain to block new connections, except established locally
% iptables -N block
% iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
% iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
% iptables -A block -j DROP # DROP everything else not ACCEPT'd
# Jump to that chain from INPUT and FORWARD chains
% iptables -A INPUT -j block
% iptables -A FORWARD -j block
```

Notice that the `block` chain `ACCEPTs` in a limited class of cases, then the final rule `DROPS` everything not previously `ACCEPTED`.

Once you have established some chains, whether adding rules to the automatic chains or adding user-defined chains, you may use the `-L` option to view the current rules.

Network address translation vs. firewalls

The examples we have looked at are mostly in the class of firewall rules. But network address translation (NAT) is also configured by `iptables`.

Basically, NAT is a way of using connection tracking to masquerade packets coming from a local subnet address as the external WAN address before sending them out "over the wire" (on the `OUTPUT` chain). The gateway/router that performs NAT needs to remember which local host connected to which remote host and reverse the address translation if packets arrive back from the remote host.

From a filtering perspective though, you simply pretend that NAT does not exist. The rules you specify should simply use the "real" local addresses regardless of how NAT might masquerade them to the outside world. Enabling masquerading, such as in basic NAT, just uses the below `iptables` command. To use this you will need to make sure the kernel module `iptables_nat` is loaded and also turn on IP forwarding:

Listing 10. Enabling the masquerade

```
% modprobe iptables_nat # Load the kernel module
% iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
% echo 1 > /proc/sys/net/piv4/ip_forward # Turn on IP forwarding
```

This capability is called *source NAT* -- the address of the outgoing packet is modified. *destination NAT* (DNAT) also exists to enable port forwarding, load sharing, and transparent proxying. In those cases, incoming packets are modified to get to the relevant local host or subnet.

But most of the time when users or administrators talk about NAT, they mean source NAT. If you mean to configure destination NAT, you would specify `PREROUTING` rather than `POSTROUTING`. For DNAT, the packets are transformed before they are routed.

Section 3. Securing FTP servers

FTP servers

Many different FTP servers are available for Linux, and different distributions offer different servers. Naturally, configuration of different servers varies, though most tend to follow similar configuration directives.

A popular FTP server is vsftpd (the Very Secure FTP daemon). ProFTP is also in wide use, as are wu-ftp and ncftpd.

For many purposes, FTP is not really needed at all. For example, secure transfers for users who have accounts on a server machine can often be accomplished using `scp` (secure copy), which relies on the underlying SSH installation, but otherwise mostly mimics the familiar `cp` command.

The configuration file for vsftpd is `/etc/vsftpd.conf`. Other FTP servers use similar files.

FTP configuration options

Here are a few options to keep in mind in `/etc/vsftpd.conf` (and probably in your server if you use a different one):

- `anonymous_enabled`: Lets anonymous users log in using the usernames "anonymous" or "ftp".

- `anon_mkdir_write_enable`: Lets anonymous users create directories (within world-writable parent directories).
- `anon_upload_enable`: Lets anonymous users upload files.
- `anon_world_readable_only`: "YES" by default; rarely a good idea to change it. Only lets anonymous FTP access world-readable files.
- `chroot_list_enable`: Specifies a set of users (listed in `/etc/vsftpd.chroot-list`) in a "chroot jail" in their home directory upon login.
- `ssl_enable`: Supports SSL-encrypted connections.

Read the manpages for your FTP server for more complete options. Generally, running an FTP server is as simple as tweaking a configuration file and running the server within your initialization scripts.

Section 4. Secure shell (SSH)

Client and server

Most every Linux machine (as well as most other operating systems) should have a secure shell (SSH) client. Often, the OpenSSH version is used, but a variety of compatible SSH clients are sometimes used. While an SSH client is essential to connect to a host, the larger security issues arise in properly configuring an SSH server.

Since a client initiates a connection to a server, the client is actively choosing to trust the server. Just having an SSH client does not allow any kind of access *into* a machine; therefore, it does not expose vulnerabilities.

Configuring a server is also not particularly complex; the server daemon is designed to enable and enforce good security practices. But clearly it is a server that is sharing resources with clients based on requests from the clients the server decides to honor.

The SSH protocol has two versions, version 1 and version 2. In modern systems, using protocol version 2 is always preferred, but generally both clients and servers maintain backward compatibility with version 1 (unless this capability is disabled with configuration options). This lets you connect to increasingly uncommon version 1-only systems.

Somewhat different configuration files are used in version 1 and version 2 protocols. For protocol version 1, a client first creates an RSA key pair using `ssh-keygen` and stores the private key in `$HOME/.ssh/identity` and the public key in `$HOME/.ssh/identity.pub`. This same `identity.pub` should be appended to the remote `$HOME/.ssh/authorized_keys` files.

Obviously, there is a chicken-and-egg problem here: How can you copy a file to a remote system before you have access? Fortunately, SSH also supports a fallback authentication method of sending encrypted-on-the-wire passwords that are evaluated through the usual remote-system login tests (such as, the user account must exist, and the right password must be provided).

Protocol 2 supports both RSA and DSA keys, but RSA authentication is somewhat enhanced rather than identical to that in protocol 1. For protocol 2, private keys are stored in `$HOME/.ssh/id_rsa` and `$HOME/.ssh/id_dsa`. Protocol 2 also supports a number of extra confidentiality and integrity algorithms: AES, 3DES, Blowfish, CAST128, HMAC-MD5, HMAC-SHA1, and so on. The server can be configured as to preferred algorithms and order of fallbacks.

For general configuration options rather than key information, the client stores its keys in `/etc/ssh/ssh_config` (or if available, in `/$HOME/.ssh/config`). Client options can also be configured with the `-o` switch; a particularly common switch is the `-X` or `-x` to enable or disable X11 forwarding. If enabled, the X11 port is tunneled through SSH to enable encrypted X11 connections.

Tools like `scp` also use similar port forwarding over SSH. For example, on the local machine I am working on, I can launch onto the local display an X11 application that only exists remotely (on my local subnet in this case):

Listing 11. Launching a remote X11 application

```
$ which gedit # not on local system
$ ssh -X dqm@192.168.2.2
Password:
Linux averatec 2.6.10-5-386 #1 Mon Oct 10 11:15:41 UTC 2005 i686 GNU/Linux
No mail.
Last login: Thu Feb 23 03:51:15 2006 from 192.168.2.101
dqm@averatec:~$ gedit &
```

Configuring the server

The `sshd` daemon, specifically the OpenSSH version, enables secure encrypted communications between two untrusted hosts over an unsecured network. The base `sshd` server is normally started during initialization and listens for client connections forking a new daemon for each client connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

As with the client tool, the `sshd` server accepts a variety of options on the command line, but is normally configured by the file `/etc/ssh/sshd_config`. A number of other configuration files are also used. For example, the access controls `/etc/hosts.allow` and `/etc/hosts.deny` are honored. Keys are stored in a similar fashion to the client side, in `/etc/ssh/ssh_host_key` (protocol 1), `/etc/ssh/ssh_host_dsa_key`, `/etc/ssh/ssh_host_rsa_key`, and public keys in `/etc/ssh/ssh_host_dsa_key.pub` and friends. Also, as with the client, you will use `ssh-keygen` to generate keys in the first place. See the manpage for `sshd` and `ssh-keygen` for details on configuration files and copying generated keys to appropriate files.

Many configuration options are in `/etc/ssh/sshd_config`, and the default values are generally sensible (and sensibly secure). A few options are worth mentioning:

- `AllowTcpForwarding` enables or disables port forwarding (tunneling), and is set to "YES" by default.
- `Ciphers` controls the list and order of encryption algorithms to be utilized.
- `AllowUsers` and `AllowGroups` accept wildcard patterns and allow you to control which users may even attempt further authentication.
- `DenyGroups` and `DenyUsers` act symmetrically as you would expect.
- `PermitRootLogin` lets the root user SSH into a machine.
- `Protocol` lets you specify whether both protocol versions are accepted (and if not, which one is).
- `TCPKeepAlive` is good to look at if you are losing SSH connections. A "keepalive" message is sent to check connections if this is enabled, but this can cause disconnection if transient errors occur in the route.

SSH tunneling

OpenSSH lets you create a tunnel to encapsulate another protocol within an encrypted SSH channel. This capability is enabled on the `sshd` server by default, but could have been disabled with command-line or configuration-file options. Assuming the capability is enabled, clients can easily emulate whatever port/protocol they wish to use for a connection. For example, to create a tunnel for telnet:

Listing 12. Digging a tunnel for telnet

```
% ssh -2 -N -f -L 5023:localhost:23 user@foo.example.com
% telnet localhost 5023
```

Of course, this example is fairly pointless since an SSH command shell does the same thing as a telnet shell. But you could create a POP3, HTTP, SMTP, FTP, X11,

or other protocol connection in this analogous manner. The basic concept is that a particular local host port acts as if it were the remote service with actual communication packets traveling over the SSH connection in encrypted form.

The options we used in the example are:

- `-2` (use protocol 2),
- `-N` (no command/tunnel only),
- `-f` (SSH in background), and
- `-L`, (describe tunnel as "localport:remotehost:remoteport").

The server (with username) is also specified.

Section 5. TCP_wrappers

What is "TCP_wrappers"?

The first thing to know about TCP_wrappers is that you *should not* use it, and it is not actively maintained. However, you might find the `tcpd` daemon from TCP_wrappers still running on a legacy system. In its time, this was a good application, but its functionality has been superceded by iptables and other tools. The general purpose of TCP_wrappers is to monitor and filter incoming requests for SYSTAT, FINGER, FTP, TELNET, RLOGIN, RSH, EXEC, TFTP, TALK, and other network services.

TCP_wrappers can be configured in a couple of ways. One is to substitute `tcpd` for other servers, providing arguments to pass control on to the particular server once `tcpd` has done its logging and filtering. Another method leaves the network daemons alone and modifies the `inetd` configuration file. For example, an entry such as:

```
tftp dgram udp wait root /usr/etc/tcpd in.tftpd -s /tftpboot
```

causes an incoming tftp request to run through the wrapper program (`tcpd`) with a process name `in.tftpd`.

Section 6. Security tasks

This objective is a hodgepodge of tasks -- all important for maintaining a secure network -- that I can't hope to do justice to in the space of this tutorial. I recommend spending time familiarizing yourself with the resources and tools listed in this section.

Web sites worth monitoring for security issues and patches include:

- [Security focus news](#): The Security Focus Web site is one of the best sites for reporting and discussing security issues and specific vulnerabilities. The site includes several newsletters and alerts that you can subscribe to, as well as general columns and searchable bug reports.
- [The Bugtraq mailing list](#): A full-disclosure moderated mailing list for the *detailed* discussion and announcement of computer security vulnerabilities: what they are, how to exploit them, and how to fix them.
- [CERT Coordination Center](#): Hosted by Carnegie Mellon University, CERT has a range of advisories similar to the Security Focus site, with a bit more emphasis on tutorials and guidelines. Keeping track of multiple such sites is a good way to make sure you are current on all the security incidents affecting your OS, distribution, and specific tools or servers.
- [Computer Incident Advisory Capability](#): CIAC Information Bulletins are distributed to the Department of Energy community to notify sites of computer security vulnerabilities and recommended actions. Similarly, CIAC Advisory Notices serve to alert sites to severe, time-critical vulnerabilities and solutions to be applied as soon as possible. CIAC Technical Bulletins cover technical security issues and analyses of a less time-sensitive nature.
- Information on [securing open mail relays](#): A common vulnerability on systems with mail servers is failure to properly secure systems against malicious use by spammers and fraudulent mailers. The Open Relay Database provides tutorials on security-particular mail tools, open relay testing online tools, and a database of known problem servers that can be used to configure blacklists if site administrators so desire.

Tools to monitor security you might consider running include:

- [Open Source Tripwire](#): A security and data integrity tool for monitoring and alerting on specific file changes.

- [scanlogd](#) : A TCP port scan detection tool.
- [Snort](#): Network intrusion prevention and detection using a rule-driven language; it uses signature-, protocol-, and anomaly-based inspection methods.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- *[TCP/IP Network Administration, Third Edition](#)* by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- For more in-depth information, the [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- In the [developerWorks Linux zone](#), find more resources for Linux developers.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- This list of more than [700 Linux User Groups around the world](#) can help you find local and distance study groups for LPI exams.
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the author

David Mertz

David Mertz has been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book *[Text Processing in Python](#)*. For more on David, see his [personal Web page](#).

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

LPI exam 201 prep: System customization and automation

Intermediate Level Administration (LPIC-2) topic 213

Skill Level: Intermediate

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer
Gnosis Software

[Brad Hunting \(hunting@glarp.com\)](mailto:hunting@glarp.com)

Mathematician
University of Colorado

01 Sep 2005

In this tutorial, David Mertz and Brad Hunting begin preparing you to take the Linux™ Professional Institute Intermediate® Level Administration (LPIC-2) Exam 201. In this seventh of eight tutorials, you learn basic approaches to scripting and automating system events, including report and status generation, clean up, and general maintenance.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams.

Each exam covers several topics, and each topic has a weight. The weights indicate the relative importance of each topic. You can expect more questions on the exam for topics with higher weight. The topics and their weights for LPI exam 201 are:

Topic 201

Linux kernel (weight 5).

Topic 202

System startup (weight 5).

Topic 203

Filesystem (weight 10).

Topic 204

Hardware (weight 8).

Topic 209

File and service sharing (weight 8).

Topic 211

System maintenance (weight 4).

Topic 213

System customization and automation (weight 3). The focus of this tutorial.

Topic 214

Troubleshooting (weight 6).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques. For details, please contact info@lpi.org.

About this tutorial

Welcome to "System customization and automation," the seventh of eight tutorials designed to prepare you for LPI exam 201. In this tutorial, you learn several basic approaches to scripting and automating system events, such as report and status generation, clean up, and general maintenance.

The tutorial is organized according to the LPI objectives for this topic:

2.213.1 Automating tasks using scripts (weight 3)

You will be able to write simple Perl scripts that make use of modules where appropriate, use the Perl taint mode to secure data, and install Perl modules from Comprehensive Perl Archive Network (CPAN). This objective includes using sed and awk in scripts, as well as using scripts to check for process execution and generating alerts by e-mail or pager when a process dies. You

should be able to write and schedule automatic execution of scripts to parse logs for alerts and e-mail them to administrators, synchronize files across machines using `rsync`, monitor files for changes and generate e-mail alerts, and write a script that notifies administrators when specified users log in or out.

One of the task categories a system administrator must perform is to automate events that need to occur periodically and to efficiently handle other events that occur sporadically. For automatic scheduling, your primary tools are `cron` and `at`. Tasks, whether regularly scheduled or manually launched, can be scripted with various languages, including `bash`, `awk`, `Perl`, or `Python`. Tools in the GNU text utilities are often useful as part of many processing tasks; these are most often used within `bash` scripts since more sophisticated languages like `awk`, `Perl`, and `Python` build in most of the capabilities in the text utilities.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. Automating periodic tasks

Configuring cron

The daemon `cron` is used to run commands periodically. You can use `cron` for a wide variety of scheduled system housekeeping and administration tasks. If there's an event or task that needs to regularly occur, it should be controlled by `cron`. `Cron` wakes up every minute to check whether it needs to do anything, but it cannot perform tasks more than once per minute. (If you need to do that, you probably want a daemon, not a "cron job.") `Cron` logs its action to the `syslog` facility.

`Cron` searches several places for configuration files that indicate environment settings and commands to run. The first is in `/etc/crontab`, which contains system tasks. The `/etc/cron.d/` directory can contain multiple configuration files that are treated as supplements to `/etc/crontab`. Special packages can add files (matching the package name) to `/etc/cron.d/`, but system administrators should use `/etc/crontab`.

User-level `cron` configurations are stored in `/var/spool/cron/crontabs/$USER`.

However, these should always be configured using the `crontab` tool. Using `crontab`, users can schedule their own recurrent tasks.

Scheduling daily, weekly, and monthly jobs

Jobs that should run on a simple daily, weekly, or monthly schedule -- which are the most commonly used schedules -- follow a special convention. Directories called `/etc/cron.daily/`, `/etc/cron.weekly/`, and `/etc/cron.monthly/` are created to include collections of scripts to run on those respective schedules. Adding or removing scripts from these directories is a simple way to schedule system tasks. For example, a system I maintain rotates its logs daily with a script file using:

Listing 1. Sample daily script file

```
$ cat /etc/cron.daily/logrotate
#!/bin/sh
test -x /usr/sbin/logrotate || exit 0
/usr/sbin/logrotate /etc/logrotate.conf
```

Cron and anacron

You can use `anacron` to execute commands periodically with a frequency specified in days. Unlike `cron`, `anacron` checks whether each job has been executed in the last n days (where n is the period specified for that job, as opposed to whether the current time matches the scheduled execution). If not, `anacron` runs the job's command after waiting for the number of minutes specified as the delay parameter. Therefore, on machines that are not running continuously, periodic jobs are executed once the machine is actually running (obviously, the exact timing can vary, but the task will not be forgotten).

`Anacron` reads a list of jobs from the configuration file `/etc/anacrontab`. Each job entry specifies a period in days, a delay in minutes, a unique job identifier, and a shell command. For example, on one Linux system I maintain, `anacron` is used to run daily, weekly, and monthly jobs even if the machine is not running at the scheduled time of day:

Listing 2. Sample anacron configuration file

```
$ cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
# These replace cron's entries
1      5      cron.daily      nice run-parts --report /etc/cron.daily
7      10     cron.weekly     nice run-parts --report /etc/cron.weekly
@monthly 15     cron.monthly    nice run-parts --report /etc/cron.monthly
```

The contents of a crontab

The format of `/etc/crontab` (or the contents of `/etc/cron.d/` files) is slightly different from that of user crontab files. Basically, this just amounts to an extra field in `/etc/crontab` that indicates the user a command runs as. This is not needed for user crontab files since they are already stored in a file matching username (`/var/spool/cron/crontabs/$USER`).

Each line of `/etc/crontab` either sets an environment variable or configures a recurring job. Comment and blank lines are ignored. For cron jobs, the first five fields specify times to run (where each zero-based field may have a list and/or a range). The fields are minute, hour, day of month, month, day of week (space- or tab-separated). An asterisk (*) in any position indicates *any*. For example, to run a task at midnight on Tuesdays and Thursdays during August through October, you could use:

```
# line in /etc/crontab
0 0 * 7-9 2,5 root /usr/local/bin/the-task -opt1 -opt2
```

Using special scheduling values

Some common scheduling patterns have shortcut names you can use in place of the first five fields:

@reboot

Run once, at startup.

@yearly

Run once a year, "0 0 1 1 *".

@annually

Same as @yearly.

@monthly

Run once a month, "0 0 1 * *".

@weekly

Run once a week, "0 0 * * 0".

@daily

Run once a day, "0 0 * * *".

@midnight

Same as @daily.

@hourly

Run once an hour, "0 * * * *".

For example, you could have a configuration containing:

```
@hourly root /usr/local/bin/hourly-task
0,29 * * * * root /usr/local/bin/twice-hourly-task
```

Using crontab

To set up a user-level scheduled task, use the `crontab` command (as opposed to the `/etc/crontab` file). Specifically, `crontab -e` launches an editor to modify a file. You can list current jobs with `crontab -l` and remove the file with `crontab -r`. Or you can specify `crontab -u user` to schedule tasks for a given user, but the default is to do so for yourself (permission limits apply).

The `/etc/cron.allow` file, if present, must contain the names of all users allowed to schedule jobs. Alternately, if there is no `/etc/cron.allow`, then a user must not be in the `/etc/cron.deny` file if allowed to schedule tasks. If neither file exists, everyone can use `crontab`.

Section 3. Automating one-time tasks

Using the at command

If you need to schedule a task to run in the future, you can use the `at` command, which takes a command from STDIN or from a file (using the `-f` option), and accepts time descriptions in a flexible collection of formats.

A family of commands is used in association with the `at` command: `atq` lists pending tasks; `atrm` removes a task from the pending queue; and `batch` works much like `at`, except it defers running a job until the system load is low.

Permissions

Similar to `/etc/cron.allow` and `/etc/cron.deny`, the `at` command has `/etc/at.allow` and `/etc/at.deny` files to configure permissions. The `/etc/at.allow` file, if present, must

contain all users allowed to schedule jobs. Alternately, if there is no `/etc/at.allow`, then a user must not be in `/etc/at.deny` if allowed to schedule tasks. If neither file exists, everyone may use `at`.

Time specifications

See the manpage on your `at` version for full details. You can specify a particular time as `HH:MM`, which schedules an event to happen when that time next occurs. (If the time has already passed today, it means tomorrow.) If you use 12-hour time, you can also add `a.m.` or `p.m.` You can give a date as `MMDDYY`, `MM/DD/YY`, `DD.MM.YY`, or `month-name-day`. You can also increment from the current time with `now + N units`, in which `N` is a number and `units` are minutes, hours, days, or weeks. The words *today* and *tomorrow* keep their obvious meaning, as do *midnight* and *noon* (*teatime* is 4 p.m.). Some examples:

```
% at -f ./foo.sh 10am Jul 31 % echo 'bar -opt' | at 1:30
tomorrow
```

The exact definition of the time specification is in `/usr/share/doc/at/timespec`.

Section 4. Tips for scripts

Outside resources

Many excellent books are available on `awk`, Perl, `bash`, and Python. The coauthor of this tutorial (naturally) recommends his own title, [Text Processing in Python](#), as a good starting point for scripting in Python.

Most scripts you write for system administration focus on text manipulation such as extracting values from logs and configuration files and generating reports and summaries. It also means cleaning up system cruft and sending notifications of tasks performed.

The most common scripts in Linux system administration are written in `bash`. `bash` itself has relatively few built-in capabilities, but `bash` makes it particularly easy to utilize external tools (including basic file utilities such as `ls`, `find`, `rm`, and `cd`) and text tools (like those found in the GNU text utilities).

Bash tips

One particularly helpful setting to include in bash scripts that run on a schedule is the `set -x` switch, which echoes the commands run to `STDERR`. This is helpful in debugging scripts when they don't produce the desired effect. Another useful option during testing is `set -n`, which causes a script to look for syntax problems, but not actually to run. Obviously, you don't want a `-n` version scheduled in `cron` or `at`, but to get it up and running, it can help.

Listing 3. Sample cron job that runs a bash script

```
#!/bin/bash
exec 2>/tmp/my_stderr
set -x
# functional commands here
```

This redirects `STDERR` to a file and outputs the commands run to `STDERR`. Examining that file later can be useful.

The manpage for bash is good, though quite long. You may find all the options that the built-in `set` can accept particularly interesting.

A common task in a system administration script is to process a collection of files, often with the files of interest identified using the `find` command. However, a problem can arise when file names contain white space or newline characters. Much of the looping and processing of file names you are likely to do can be confused by these internal white space characters. For example, these two commands are different:

```
% rm foo bar baz bam
% rm 'foo bar' 'baz bam'
```

The first command unlinks four files (assuming they exist to start with); the second removes just two files, each with an internal space in the name. File names with spaces are particularly common in multimedia content.

Fortunately, the GNU version of the `find` command has a `-print0` option to `NULL` terminate each result; and the `xargs` command has a corresponding `-0` command to treat arguments as `NULL` separated. Putting these together, you can clean up stray files that might contain white space in their names using:

Listing 4. Cleaning up file names with spaces

```
#!/bin/bash
# Cleanup some old files
set -x
```

```
find /home/dqm \( -name '*.core' -o -name '#*' \) -print0 \
| xargs -0 rm -f
```

Perl taint mode

Perl has a handy switch `-T` to enable taint mode. In this mode, Perl takes a variety of extra security precautions, but primarily it limits execution of commands arising from external input. If you use `sudo` execution, taint mode might be enabled automatically, but the safest thing is to start your administration scripts with:

```
#!/usr/local/bin/perl -T
```

Once you do this, all command line arguments, environment variables, locale information (see `perllocale`), results of certain system calls (`readdir()`, `readlink()`, the variable of `shmread()`, the messages returned by `msgrcv()`, the password, `gcos` and shell fields returned by the `getpwxxx()` calls), and all file inputs are marked as "tainted." Tainted data cannot be used directly or indirectly in any command that invokes a sub-shell nor in any command that modifies files, directories, or processes, with a few exceptions.

It's possible to untaint particular external values by carefully checking them for expected patterns:

Listing 5. Untainting external values

```
if ($data =~ /^[[-\@w.]+$/) {
    $data = $1;           # $data now untainted
} else {
    die "Bad data in $data"; # log this somewhere
}
```

Perl CPAN packages

One of the handy things about Perl is that it comes with a convenient mechanism for installing extra support packages; it's called Comprehensive Perl Archive Network (CPAN). RubyGems is similar in function. Python, unfortunately, does not yet have an automated installation mechanism, but it comes with more in the default installation. Simpler languages like `bash` and `awk` do not really have many add-ons to install in an analogous sense.

The manpage on the `cpan` command is a good place to get started, especially if you have a task to perform for which you think someone might have done most of the work already. Look for candidate modules at [CPAN](#).

`cpan` has both an interactive shell and a command-line operation. Once configured

(run the interactive shell once to be prompted for configuration options), `cpan` handles dependencies and download locations in an automated manner. For example, suppose you discover you have a system administration task that involves processing configuration files in YAML (yaml Ain't Markup Language) format. Installing support for YAML is as simple as:

```
% cpan -i YAML # maybe with 'sudo' first
```

Once installed, your scripts can contain `use YAML;` at the top. This goes for any capabilities for which someone has created a package.

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Understanding Linux configuration files](#)" (developerWorks, December 2001) shows you how to set up configuration files on a Linux system that control user permissions, system applications, daemons, services, and other administrative tasks in a multiuser, multitasking environment.
- The developerWorks tutorial "[Using the GNU text utilities](#)" (developerWorks, March 2004) introduces you to scripting utilities.
- These [scripting articles on developerWorks](#) provide lots of resources on using scripts to automate tasks in Linux.
- *[Text Processing in Python](#)* by David Mertz, co-author of this tutorial, is an excellent source for Python scripts.
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- The [CPAN Module Archive](#) is your source for Perl modules.
- [Order the SEK for Linux](#): Get this two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the authors

David Mertz, Ph.D.

David Mertz is Turing complete, but probably would not pass the Turing Test. For more about his life, see his [personal Web page](#). He's been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book *[Text Processing in Python](#)*. You can contact David at mertz@gnosis.cx.

Brad Huntting

Brad has been doing UNIX® systems administration and network engineering for about 14 years at several companies. He is currently working on a Ph.D. in Applied Mathematics at the University of Colorado in Boulder, and pays the bills by doing UNIX support for the Computer Science department. Contact Brad at huntting@glarp.com.

LPI exam 202 prep: Network troubleshooting

Intermediate Level Administration (LPIC-2) topic 214

Skill Level: Intermediate

[David Mertz \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)
Developer
Gnosis Software, Inc.

28 Jun 2006

In this tutorial, the last of a [series of seven tutorials](#) covering intermediate network administration on Linux®, David Mertz finishes preparing you to take the Linux Professional Institute Intermediate Level Administration (LPIC-2) Exam 202. This tutorial revisits earlier tutorials in the LPI 202 series, focusing on how to use the basic tools you've already covered to fix networking problems. The tool review is divided into two categories: configuration tools and diagnostic tools.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks.

- To prepare for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#).
- To prepare for certification level 2, see the [developerWorks tutorials for LPI exams 201 and 202](#).

View the [entire set of developerWorks LPI tutorials](#).

For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 202: Tutorials and topics		
LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic 205): Networking configuration	Learn how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	Learn how to use Linux as a mail server and as a news server. Learn about mail transport, local mail filtering, mailing list maintenance software, and server software for the NNTP protocol.
Topic 207	LPI exam 202 prep (topic 207): DNS	Learn how to use Linux as a DNS server, chiefly using BIND. Learn how to perform a basic BIND configuration, manage DNS zones, and secure a DNS server.
Topic 208	LPI exam 202 prep (topic 208): Web services	Learn how to install and configure the Apache Web server, and learn how to implement the Squid proxy server.
Topic 210	LPI exam 202 prep (topic 210): Network client management	Learn how to configure a DHCP server, an NIS client and server, an LDAP server, and PAM authentication support. See detailed objectives below.
Topic 212	LPI exam 202 prep (topic 212): System security	Learn how to configure a router, secure FTP servers, configure SSH, and perform various other security administration tasks.

Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	(This tutorial) Review tools and commands that let you detect and solve networking problems. See detailed objectives below.
-----------	---	---

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Network troubleshooting," the last of seven tutorials covering intermediate network administration on Linux. This tutorial re-examines the material covered in the first six tutorials on the Linux Professional Institute's 202 exam topics to give some general context for the entire series. Highlighted here are some of the tools you've previously covered -- `ifconfig`, `route`, `hostname`, `dmesg`, `netstat`, `ping`, `traceroute`, etc. -- with the focus on using those tools to fix problems.

As with the other tutorials in the developerWorks 201 and 202 series, this tutorial is intended to serve as a study guide and entry point for exam preparation, rather than complete documentation on the subject. Readers are encouraged to consult LPI's [detailed objectives list](#) and to supplement the information provided here with other material as needed.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Network troubleshooting: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
2.214.7 Troubleshooting network issues	Weight 1	Identify and correct common network setup issues. This objective includes knowledge of locations for basic configuration files and commands.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial. This tutorial builds on material covered in the [previous six tutorials in the LPI exam 202 series](#).

Other resources

As with most Linux tools, it is always useful to examine the manpages for any utilities discussed. Versions and switches might change between utility or kernel version or with different Linux distributions. For more in depth information, the [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs. A variety of books on Linux networking have been published; I have found O'Reilly's *TCP/IP Network Administration*, by Craig Hunt to be quite helpful (find whatever edition is most current when you read this).

Section 2. Network configuration tools

About network troubleshooting

To troubleshoot a network configuration, you need to know how to use several of the tools discussed in this tutorial series; you also need to be familiar with the configuration files that affect network status and behavior. This tutorial summarizes the main tools and configuration files you should be familiar with for effective troubleshooting.

For simplicity, this tutorial groups the tools according to whether a given tool applies more to configuration of a network in the first place or to diagnosis of network problems. Of course, in practice those elements are rarely separate.

ifconfig

[LPI exam 202 prep \(topic 205\): Networking configuration](#) discusses `ifconfig` in greater detail. This utility both reports on the current status of network interfaces and lets you modify the configuration of those interfaces. In most cases, if *something* is wrong with a network -- like a particular machine does not appear to access the network at all -- running `ifconfig` with no options is usually the first step you should take. If this fails to report active interfaces, you can be pretty sure that the local machine itself has a configuration problem. "Active" in this case means that it shows an IP address assigned; in most cases, you should expect to see a number of packets in the RX and TX lines:

Listing 1. Using ifconfig

```
eth0    Link encap:Ethernet  HWaddr 00:C0:9F:21:2F:25
        inet addr:192.168.216.90  Bcast:66.98.217.255  Mask:255.255.254.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:6193735  errors:0  dropped:0  overruns:0  frame:0
        TX packets:6982479  errors:0  dropped:0  overruns:0  carrier:0
```

Attempting to activate an interface with something like `ifconfig eth0 up ...` is a good first step to try to see if an interface *can* be activated (in many cases, filling in additional options in the line).

route

[LPI exam 202 prep \(topic 205\): Networking configuration](#) discusses `route` in greater detail. This utility lets you both view and modify the routing tables currently in effect for a local machine and a local network. Using `route`, you may add and delete routes, set netmasks and gateways, and perform various other tweaking tasks.

For the most part, calls to `route` should be performed in initialization scripts, but in attempting to diagnose and fix problems, experimenting with routing options can help (you can copy successes to appropriate initialization scripts for later use).

hostname

This utility also has aliases to employ different aspects of the utility:

- `domainname`
- `nodename`
- `dnsdomainname`
- `nisdomainname`
- `ypdomainname`

You control these capabilities with switches to `hostname` itself.

`hostname` is used to either set or display the current host, domain, or node name of the system. These names are used by many of the networking programs to identify the machine. The domain name is also used by NIS/YP.

dmesg

The utility `dmesg` allows you to examine kernel log messages; it works in cooperation with `syslogd`. Any kernel process, including those related to

networking, are best accessed using the `dmesg` utility, often filtered using other tools such as `grep`, as well as switches to `dmesg`.

Manually setting ARP

You almost never need or want to mess with automatically discovered ARP records. However, you may want to manually configure the ARP cache in debugging situations. The utility `arp` lets you do this. The key flag options in the `arp` utility are `-d` for delete, `-s` for set, and `-f` for set-from-file (default file is `/etc/ethers`).

For example, suppose that communication with a specific IP address on the local network is erratic or unreliable. One possible cause of this situation is if multiple machines are incorrectly configured to use the same IP address. When an ARP request is broadcast over the Ethernet network, it is not pre-determined which machine will respond first with an ARP reply. The end result might be that the data packets are delivered to one machine one time and to a different machine another time.

Using `arp -n` to debug the actual IP assignment is a first step. If you can determine that the IP address at issue does not map to the correct Ethernet device, that is a strong clue about what is going on.

But beyond that somewhat random detection, you can force the right ARP mapping using the `arp -s` (or `-f`) option. Set an IP to map to the actual Ethernet device it should map to; manually configured mapping will not expire unless specifically set to do so using the `temp` flag. If a manual ARP mapping fixes the data loss problem, this is a strong sign the problem is over-assigned IP addresses.

Section 3. Network diagnostic tools

netstat

[LPI exam 202 prep \(topic 205\): Networking configuration](#) discusses `netstat` in greater detail. This utility displays a variety of information on network connections, routing tables, interface statistics, masquerade connections, and multicast memberships. Among other things, `netstat` provides fairly detailed statistics on packets that have been handled in various ways.

The manpage for `netstat` provides information on the wide range of switches and options available. This utility is a good general-purpose tool for digging into details of

the status of networking on the local machine.

ping

A good starting point in finding out if you can connect to a given host from the current machine (by either IP number or symbolic name) is the utility `ping`. As well as establishing that a route exists at all -- including the resolution of names via DNS or other means if a symbolic name is used -- `ping` gives you information on round-trip times that may be indicative of network congestion or routing delays. Sometimes `ping` will indicate a percentage of dropped packets, but in practical use you almost always see either 100 or 0 percent of packets lost by `ping` requests.

traceroute

The utility `traceroute` is a bit like a `ping` on steroids. Rather than simply report the fact that a route exists to a given host, `traceroute` reports complete details on all the hops taken along the way, including the timing of each router. Routes may change over time, either because of dynamic changes in the Internet or because of routing changes you have implemented locally. At a given moment though, `traceroute` shows you an actual followed path.

Listing 2. traceroute shows the actual followed path

```
$ traceroute google.com
traceroute: Warning: google.com has multiple addresses; using 64.233.187.99
traceroute to google.com (64.233.187.99), 30 hops max, 38 byte packets
 1  evls-66-98-216-1.evlservers.net (66.98.216.1)  0.466 ms  0.424 ms  0.323 ms
 2  ivhou-207-218-245-3.ev1.net (207.218.245.3)  0.650 ms  0.452 ms  0.491 ms
 3  ivhou-207-218-223-9.ev1.net (207.218.223.9)  0.497 ms  0.467 ms  0.490 ms
 4  gateway.mfn.com (216.200.251.25)  36.487 ms  1.277 ms  1.156 ms
 5  so-5-0-0.mpr1.atl6.us.above.net (64.125.29.65)  13.824 ms  14.073 ms  13.826 ms
 6  64.124.229.173.google.com (64.124.229.173)  13.786 ms  13.940 ms  14.019 ms
 7  72.14.236.175 (72.14.236.175)  14.783 ms  14.749 ms  14.476 ms
 8  216.239.49.226 (216.239.49.226)  16.651 ms  16.421 ms  17.648 ms
 9  64.233.187.99 (64.233.187.99)  14.816 ms  14.913 ms  14.775 ms
```

host, nslookup, and dig

All three utilities -- `host`, `nslookup`, and `dig` -- are used for querying DNS entries; they largely overlap in their capabilities. Generally, `nslookup` provided enhancement to `host`, and `dig` in turn enhanced `nslookup` (though none of the three are exactly backward- or forward-compatible with the others). All the tools rely on the same underlying kernel facilities, so reported results should be consistent in all cases (except where level of detail differs). For example, each of the three is used to query `google.com`:

Listing 3. Using host, nslookup, and dig to query Google

```
$ host google.com
google.com has address 64.233.187.99
google.com has address 64.233.167.99
google.com has address 72.14.207.99

$ nslookup google.com
Server:      207.218.192.39
Address:     207.218.192.39#53

Non-authoritative answer:
Name:   google.com
Address: 64.233.167.99
Name:   google.com
Address: 72.14.207.99
Name:   google.com
Address: 64.233.187.99

$ dig google.com
; <<>> DiG 9.2.4 <<>> google.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46137
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.                IN      A

;; ANSWER SECTION:
google.com.                295     IN      A      64.233.167.99
google.com.                295     IN      A      72.14.207.99
google.com.                295     IN      A      64.233.187.99

;; Query time: 16 msec
;; SERVER: 207.218.192.39#53(207.218.192.39)
;; WHEN: Mon Apr 17 01:08:42 2006
;; MSG SIZE rcvd: 76
```

Section 4. Network configuration files

`/etc/network/` and `/etc/sysconfig/network-scripts/`

The directory `/etc/network/` contains a variety of data about the current network on some Linux distributions, especially in the file `/etc/network/interfaces`. Various utilities, especially `ifup` and `ifdown` (or `iwup` and `iwdown` for wireless interfaces), are contained in `/etc/sysconfig/network-scripts/` on some distributions (but the same scripts may live elsewhere instead on your distribution).

`/var/log/syslog` and `/var/log/messages`

Messages logged by the kernel or the `syslogd` facility are stored in the log files `/var/log/syslog` and `/var/log/messages`. [LPI exam 201 prep \(topic 211\): System maintenance](#) discusses system logging in greater detail. The utility `dmesg` is generally used to examine logs.

`/etc/resolv.conf`

[LPI exam 202 prep \(topic 207\): Domain Name System](#) discusses `/etc/resolv.conf` in greater detail. Generally, this file simply contains the information needed to find domain name servers. It may be configured either manually or via dynamic means such as RIP, DHCP, or NIS.

`/etc/hosts`

The file `/etc/hosts` is usually the first place a Linux system looks to attempt to resolve a symbolic hostname. You can add entries to either bypass DNS lookup (or sometimes YP or NIS facilities) or to name hosts that are not available on DNS, often because they are strictly names on the local network. See the examples in Listing 4.

Listing 4. `/etc/hosts`, the place to resolve symbolic hostnames

```
$ cat /etc/hosts
# Set some local addresses
127.0.0.1    localhost
255.255.255.255 broadcasthost
192.168.2.1  artemis.gnosis.lan
192.168.2.2  bacchus.gnosis.lan
# Set undesirable site patterns to loopback
127.0.0.1   *.doubleclick.com
127.0.0.1   *.advertising.com
127.0.0.1   *.valueclick.com
```

`/etc/hostname` and `/etc/HOSTNAME`

The file `/etc/HOSTNAME` (on some systems without the capitalization) is sometimes used for the symbolic name of the localhost as known on the network. However, use of this file varies between distributions; generally `/etc/hosts` is used exclusively on modern distributions.

`/etc/hosts.allow` and `/etc/hosts.deny`

[LPI exam 201 prep \(topic 209\): File and service sharing](#) and [LPI exam 202 prep \(topic 212\): System security](#) discusses the files `/etc/hosts.allow` and `/etc/hosts.deny`

in greater detail. These configuration files are used for positive and negative access lists by a variety of network tools. Read the manpages on these configuration files for more information on the specification of wildcards, ranges, and specific permissions that may be granted or denied.

Beyond initial setup to enforce system security, you often want to examine the content of these when a connection fails that "just seems like" it should be working. Generally, examining access control issues comes after examining basic interface and routing information in a debugging effort. That is, if you cannot reach a particular host at all (or it cannot reach you), it does not matter whether the host has permissions to use the services you provide. But selective failures in connections and service utilization can often be because of access control issues.

Section 5. A final word

Take advantage of every resource

For the subjects addressed in this tutorial, possibly the best resource for further information is the rest of this tutorial series. Nearly all the topics addressed here are detailed further in previous tutorials.

Quite a few people have written step-by-step guides to fixing a broken Linux network. One that looks good is "[Simple Network Troubleshooting](#)." Debian's similar quick guide is "[How To Set Up A Linux Network](#)." Since tutorials come and go and are updated on different schedules as distributions and commands change, you can always search the Internet to find currently available sources.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- *[TCP/IP Network Administration, Third Edition](#)* by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- For more in-depth information, the [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- In the [developerWorks Linux zone](#), find more resources for Linux developers.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- This list of more than [700 Linux User Groups around the world](#) can help you find local and distance study groups for LPI exams.
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the author

David Mertz

David Mertz has been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book *[Text Processing in Python](#)*. For more on David, see his [personal Web page](#).

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

LPI exam 201 prep: Troubleshooting

Intermediate Level Administration (LPIC-2) topic 214

Skill Level: Intermediate

[Brad Huntting \(hunting@glarp.com\)](mailto:hunting@glarp.com)

Mathematician

University of Colorado

[David Mertz, Ph.D. \(mertz@gnosis.cx\)](mailto:mertz@gnosis.cx)

Developer

Gnosis Software

02 Sep 2005

In this tutorial, Brad Huntting and David Mertz continue preparing you to take the Linux Professional Institute® Intermediate Level Administration (LPIC-2) Exam 201. The last of eight tutorials, this tutorial focuses on what you can do when things go wrong. It builds on material already covered in more detail in earlier tutorials.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at junior and intermediate levels. To attain each level of certification, you must pass two LPI exams.

Each exam covers several topics, and each topic has a weight. The weights indicate the relative importance of each topic. Very roughly, expect more questions on the exam for topics with higher weight. The topics and their weights for LPI exam 201

are:

Topic 201

Linux kernel (weight 5).

Topic 202

System startup (weight 5).

Topic 203

File systems (weight 10).

Topic 204

Hardware (weight 8).

Topic 209

File and service sharing (weight 8).

Topic 211

System maintenance (weight 4).

Topic 213

System customization and automation (weight 3).

Topic 214

Troubleshooting (weight 6). The focus of this tutorial.

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Troubleshooting," the last of eight tutorials designed to prepare you for LPI exam 201. In this tutorial, you review material already covered in more detail in earlier tutorials but with a focus on learning what to do when things go wrong.

The tutorial is organized according to the LPI objectives for this topic, as follows:

2.214.2 Creating recovery disks (weight 1)

You will be able to create both a standard bootdisk for system entrance and a recovery disk for system repair.

2.214.3 Identifying boot stages (weight 1)

You will be able to determine, from bootup text, the four stages of boot sequence and distinguish between each.

2.214.4 Troubleshooting LILO (weight 1)

You will be able to determine specific stage failures and corrective techniques.

2.214.5 General troubleshooting (weight 1)

You will be able to recognize and identify boot loader- and kernel-specific stages and utilize kernel boot messages to diagnose kernel errors. This objective includes identifying and correcting common hardware issues, and determining if the problem is hardware or software.

2.214.6 Troubleshooting system resources (weight 1)

You will be able to identify, diagnose, and repair your local system environment.

2.214.8 Troubleshooting environment configurations (weight 1)

You will be able to identify common local system and user environment configuration issues and common repair techniques.

Troubleshooting a Linux system that is not working -- or more especially, is not *entirely* working -- is probably the most frustrating part of system administration. Linux is not special in this regard. With Linux, common tasks -- adding new hardware or software, tweaking the behavior of installed software, or scripting and scheduling routine operations -- are straightforward. But when a running system suddenly stops, you are presented with a dizzying collection of things that *might* be wrong.

This tutorial cannot solve all real-world problems, but it does point to tools that are helpful in the troubleshooting process (most touched briefly on in other tutorials) and reminds you of the best places to look for sources of trouble. In troubleshooting, a good understanding of Linux, combined with patience and elbow grease, will go a long way.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

Section 2. Creating recovery disks

Before you recover

When, in course of system events, a Linux installation gets so messed up that it just cannot boot normally, a good first course of action is to attempt to "boot single-user" to fix the problem. Single-user mode (runlevel 1) allows you to access and modify many problems without worrying about permission issues or files being locked by other users and processes. Moreover, single-user mode runs with a minimum of services, daemons, and tasks that might confound -- or even be the cause of -- your underlying problem.

In some cases, however, even files needed for single-user mode such as `/etc/passwd`, `/etc/fstab`, `/etc/inittab`, `/sbin/init`, `/dev/console`, etc., can become corrupted and the system will not boot, even into single-user mode. In times like this, a recovery disk (a bootable disk which contains the essentials needed to repair a broken partition, sometimes called a *repair disk*) can be used to get the system back on its feet.

Recovery CDs

Today, standalone CD (or DVD) distributions such as Knoppix can be used as the "Cadillac" of recovery disks, sporting a vast array of drivers, debugging tools, and even a Web browser. Even if you have special needs such as unusual drivers (like kernel modules) or recovery software, you are usually best off downloading a Linux LiveCD like Knoppix and customizing the contents of the ISO image before burning a recovery disk. See the tutorial for Topic 203 for information on loop mounting an ISO image.

Recovery floppies

For very old systems without bootable CD-ROMs, floppies containing only the bare essentials have to be used for recovery. Niceties like *emacs* and *vim* are typically too bulky to fit on a single recovery floppy, so an experienced system administrator should be prepared to use a stripped-down editor like *ed* to fix corrupted configuration files.

To create custom boot floppies, the user should read the Linux Documentation Project's "Linux-Bootdisk HOWTO." In many cases, however, Linux distributions will offer to create a recovery diskettes at installation time or later using included tools. Be sure to create and store recover disks *before* you really need them!

Section 3. Identifying boot stages

About booting

The tutorial on Topic 202 contains much more extensive information about Linux boot sequences. We will review the stages in quick summary for this tutorial.

The first stage of bootup has changed little since IBM®-compatible PCs with hard disks were first introduced. The BIOS reads the first sector of the boot disk into memory and runs it. This 512-byte Master Boot Record (MBR), which also stores the fdisk label, turns around and loads the "OS loader" (either GRUB or LILO) from the "active" partition.

Booting the kernel

Once the system BIOS on an x86 system (other architectures vary slightly) runs the MBR, several steps happen for the Linux kernel to load. If the boot does not succeed, determining where it fails is the first step in figuring out what needs to be fixed.

- Load Boot Loader (LILO/Grub).
- Boot loader starts and hands off control to kernel.
- Kernel: Load base kernel and other essential kernel modules.
- Hardware initialization and setup:
 - Kernel starts.
 - Initialize kernel infrastructure (VM, scheduler, etc.).
 - Device Drivers Probe and Attach.
 - Root filesystem mounted.

Booting the userland

Assuming a base kernel, kernel modules, and root filesystem start successfully (or at least successfully enough not to freeze completely), the system initialization process starts:

- Daemon initialization and setup
 - /sbin/init started as process 1
 - /sbin/init reads /etc/inittab

- `/etc/rc<n>.d/S*` scripts are run by `/etc/init.d/rc`
 - Disks fsck'ed
 - Network interfaces configured
 - Daemons started
 - `getty(s)` started on console and serial ports
-

Section 4. Troubleshooting LILO and GRUB

Looking for the boot loader

When GRUB starts up it flashes "GRUB loading, please wait..." on the screen and then jumps into its boot menu. LILO usually displays a `LILO:` prompt (but some versions jump directly to a menu). If you do not reach a LILO or GRUB screen, it is probably time for a separate recovery disk. Generally, you will know you have this problem with some kind of BIOS message about not finding a boot sector or maybe even not finding a hard disk.

If your system cannot locate a usable boot sector, it is possible that your LILO or GRUB installation became corrupted. Sometimes other operating systems step on boot sectors (or overzealous "virus protection programs" on Windows®) and occasionally other problems arise. A recovery disk will let you reinstall LILO or GRUB. Typically you will want to `chroot` to your old root filesystem to utilize your prior LILO or GRUB configuration files.

In the hard-disk-not-found case, hardware failure is likely (and you will hope you made a backup); often in the no-boot-sector case too. See the tutorial on Topic 202 for more information on LILO and GRUB.

Configuring LILO and GRUB

LILO is configured with the file `/etc/lilo.conf` or by another configuration file specified at the command line. To reinstall LILO, first make sure your `lilo.conf` file really matches your current system configuration (the right partition and disk number information especially; this can become mixed up if you swap hard disk and/or change partitions). Once you have verified your configuration, simply run `/sbin/lilo` as root (or in single-user mode).

Configuring and reinstalling GRUB is essentially the same process as for LILO. GRUB's configuration file is `/boot/grub/menu.lst`, but it is read each time the system boots. Likewise, the GRUB shell lives at `/boot/grub/`, but you can choose which hard disk the basic GRUB MBR lives at with the `grout=` option in `/boot/grub/menu.lst`. To install GRUB to an MBR, run a command like `grub-install /dev/hda`, which will check the currently mounted `/boot/` for its configuration.

Section 5. General troubleshooting

The Linux filesystem structure

Linux distributions vary a bit in where they put files. The Linux Filesystem Hierarchy Standard is making progress in furthering standardization of this. A few directories are already well standardized and are particularly important to look at for startup and runtime problems:

- `/proc/` is the virtual filesystem with information on processes and system status. All the guts of a running system live here, in a sense. See Topic 201 for more information.
- `/var/log/` is where the log files live. If something goes wrong, it is good that helpful information is in some log file here.
- `/` is generally the root of the filesystem under Linux which simply contains other nested directories. On some systems bootup files like `vmlinuz` and `initrd.img` might be here rather than in `/boot/`.
- `/boot/` stores files directly used in the kernel boot process.
- `/lib/modules/` is where kernel modules live, nested under this directory in subdirectories named for the current kernel version (if you boot multiple kernel versions, multiple directories should exist). For example:

```
% ls /lib/modules/2.6.10-5-386/kernel/  
arch crypto drivers fs lib net security sound
```

Finding bootup messages

During actual system boot, messages can scroll by quickly and you may not have time to identify problems or unexpected initialization activities. Some information of interest is likely to be logged with `syslog`, but the basic kernel and kernel module

messages can be examined with the tool *dmesg*.

Hardware/system identification tools

The tutorial for Topic 203 (Hardware) contains more information on hardware identification. Generally, you should keep in mind these system tools:

- *lspci*: Lists all PCI devices.
- *lsmod*: Lists loaded kernel modules.
- *lsusb*: Lists USB devices.
- *lspnp*: Lists Plug-and-Play devices.
- *lshw*: Lists hardware.

Slightly farther from hardware, but still useful:

- *lsuf*: Lists open files.
- *insmod*: Loads kernel modules.
- *rmmod*: Removes kernel modules.
- *modprobe*: Higher level *insmod*/*rmmod*/*lsmod* wrapper.
- *uname*: Prints system information (kernel version, etc.).
- *strace*: Traces system calls.

When you get desperate in trying to use binaries, whether libraries or applications, the tool *strings* can really rescue you (but expect to work at it). Crucial information like hard-coded paths are sometimes buried in binaries and with a good measure of trial-and-error, you can find them by looking for the strings within binaries.

Section 6. Troubleshooting system resources and environment configurations

The initialization

As Topic 202 addresses in more detail, the startup of Linux after the kernel loads is

controlled by the `init` process. The main controlling configuration for `init` is `/etc/inittab`. The `/etc/inittab` contains details on what steps to perform at each runlevel. But perhaps the most crucial thing it does is actually set the runlevel for later actions. If your system is having troubles booting, a setting a different runlevel may help the process. The key line is something like:

```
# The default runlevel. (in /etc/inittab)
id:2:initdefault:
```

Resource scripts

The `rc` scripts run at boot time, shutdown time, and whenever the system changes runlevels and they are responsible for starting and stopping most system daemons. In most (read *modern*) Linux distributions, the `rc` scripts live in the directory `/etc/init.d/` and are linked to the directories `/etc/rc<N>.d/` (for $N=runlevel$) where they are named "S*" for startup scripts and "K*" for shutdown scripts. The system never runs the scripts from `/etc/init.d` directly, but looks for them in `/etc/rc<N>.d/[SK]*`.

The system shell

Rarely, but conceivably, a system administrator may wish to change the system-wide shell startup script `/etc/profile`. Such a change affects every interactive shell (except for `/bin/tcsh` users or other non-`/bin/sh`-compatible shells). Corrupting this file can easily lead to a situation where no one can login and a recovery disk may be needed to rectify the situation. The normal method of changing shell behavior on an individual basis is by modifying `/home/$USER/.bash_profile` and `/home/$USER/.bashrc`.

Configure kernel parameters

The `sysctl` system (see the manpage for `sysctl`) was taken from BSD UNIX® and is be used to tune some system resources. Run `sysctl -a` to see what variables can be controlled by `sysctl` and what they are set to. The `sysctl` utility is most useful to tuning networking parameters as well as some kernel parameters. Use the file `/etc/sysctl.conf` set `sysctl` parameters at boot time.

Dynamic libraries

On most systems, dynamic libraries are constantly being added, updated, replaced, and removed. Since almost every program on the system needs to find and load dynamic libraries, the names, version numbers, and locations of most dynamic libraries are cached by the `ldconfig` program. Normally only the dynamic libraries in

the system directories `/lib/` and `/usr/lib/` are cached. To add more directories to the system-wide default search path, add the directory names (such as `/usr/X11R6/lib`) to the `/etc/ld.so.conf` and run `ldconfig` as root.

System logging

Topic 211 discusses `syslog` in detail. The key file to keep in mind if you have problems (and especially if you want to make sure you can analyze them later) is `/etc/syslog.conf`. By changing the contents of this configuration, you have granular control over what events are logged and where they are logged -- including potentially using mail or remote machines. If problems occur, make sure the subsystems where you think they arise log information in a manner you can readily examine forensically.

Periodic events

On almost every Linux system, the `cron` daemon is running. See Topic 213 for more details on `cron` and `crontab`. In general, a source of potential problems to keep in mind is scripts that are run intermittently. It is possible that what you believe to be a kernel or application issue results from unrelated scripts that are run behind the scenes by `cron`.

An extreme measure is to kill `cron` altogether. You can find its process number with `ps ax | grep cron` and `kill` can terminate it. A less drastic measure is to edit `/etc/crontab` to run a more conservative collection of tasks; also be sure to constrain the user tasks that are scheduled by editing `/etc/cron.allow` and `/etc/cron.deny`. Even though users usually should not have enough permissions to cause system-wide problems, a good first step is to temporarily disable user `crontab` configurations and see if that resolves problems.

Resources

Learn

- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- "[Linux hardware stability guide](#)" (developerWorks, March 2001) is a two-part guide on troubleshooting Linux hardware.
- "[Common threads: Advanced filesystem implementor's guide, Parts 1 - 13](#)" (developerWorks, starting June 2001) is an excellent series on Linux filesystems.
- "[Understanding Linux configuration files](#)" (developerWorks, December 2001) explains configuration files on a Linux system that control user permissions, system applications, daemons, services, and other administrative tasks in a multi-user, multi-tasking environment.
- "[System recovery with Knoppix](#)" (developerWorks, October 2003) shows how to access a non-booting Linux system with a Knoppix CD.
- "[Assess system security using a Linux LiveCD](#)" (developerWorks, July 2005) introduces four LiveCD offerings that specialize in nailing down vulnerabilities.
- Read the Linux Documentation Project's [Linux-Bootdisk HOWTO](#).
- Find more resources for Linux developers in the [developerWorks Linux zone](#).

Get products and technologies

- Read about and download [Knoppix](#).
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Build your next development project on Linux with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the authors

Brad Huntting

Brad has been doing UNIX systems administration and network engineering for about 14 years at several companies. He is currently working on a Ph.D. in Applied Mathematics at the University of Colorado in Boulder, and pays the bills by doing UNIX support for the Computer Science department.

David Mertz, Ph.D.

David Mertz is Turing complete, but probably would not pass the Turing Test. For more on his life, see his [personal Web page](#). He's been writing the developerWorks columns *Charming Python* and *XML Matters* since 2000. Check out his book [Text Processing in Python](#).

LPI exam 301 prep, Topic 301: Concepts, architecture, and design

Senior Level Linux Professional (LPIC-3)

Skill Level: Intermediate

[Sean A. Walberg](mailto:sean@ertw.com) (sean@ertw.com)
Senior Network Engineer

23 Oct 2007

In this tutorial, Sean Walberg helps you prepare to take the Linux Professional Institute® Senior Level Linux Professional (LPIC-3) exam. In this first in a series of six tutorials, Sean introduces you to Lightweight Directory Access Protocol (LDAP) concepts, architecture, and design. By the end of this tutorial, you will know about LDAP concepts and architecture, directory design, and schemas.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *advanced level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102. To attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active advanced-level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the five junior, advanced, and

senior certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. Table 1 lists the six topics and corresponding developerWorks tutorials for LPI exam 301.

Table 1. LPI exam 301: Tutorials and topics

LPI exam 301 topic	developerWorks tutorial	Tutorial summary
Topic 301	LPI exam 301 prep: Concepts, architecture, and design	(This tutorial.) Learn about LDAP concepts and architecture, learn how to design and implement an LDAP directory, and learn about schemas. See the detailed objectives below.
Topic 302	LPI exam 301 prep: Installation and development	Coming soon.
Topic 303	LPI exam 301 prep: Configuration	Coming soon.
Topic 304	LPI exam 301 prep: Usage	Coming soon.
Topic 305	LPI exam 301 prep: Integration and migration	Coming soon.
Topic 306	LPI exam 301 prep: Capacity planning	Coming soon.

To pass exam 301 (and attain certification level 3), you should:

- Have several years experience with installing and maintaining Linux® on a number of computers for various purposes.
- Have integration experience with diverse technologies and operating systems.
- Have professional experience as, or training for, an enterprise-level Linux professional (including having experience as a part of another role).
- Know advanced and enterprise levels of Linux administration including installation, management, security, troubleshooting, and maintenance.
- Be able to use open source tools to measure capacity planning and troubleshoot resource problems.
- Have professional experience using LDAP to integrate with UNIX® and Microsoft® Windows® services, including Samba, Pluggable Authentication Modules (PAM), e-mail, and Microsoft Active Directory directory service.
- Be able to plan, architect, design, build, and implement a full environment

using Samba and LDAP as well as measure the capacity planning and security of the services.

- Be able to create scripts in Bash or Perl or have knowledge of at least one system-programming language (such as C).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular.

About this tutorial

Welcome to "Concepts, architecture, and design," the first of six tutorials designed to prepare you for LPI exam 301. In this tutorial, you learn about LDAP concepts and architecture, how to design and implement an LDAP directory, and about schemas.

This tutorial is organized according to the [LPI objectives for this topic](#). Very roughly, expect more questions on the exam for objectives with higher weights.

Objectives

Table 2 provides the detailed objectives for this tutorial.

Table 2. Concepts, architecture, and design: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
301.1 Concepts and architecture	3	Be familiar with LDAP and X.500 concepts.
301.2 Directory design	2	Design and implement an LDAP directory while planning an appropriate Directory Information Tree to avoid redundancy. You should have an understanding of the types of data that are appropriate for storage in an LDAP directory.
301.3 Schemas	3	Be familiar with schema concepts and the base schema files included with an OpenLDAP installation.

Prerequisites

To get the most from this tutorial, you should have an advanced knowledge of Linux

and a working Linux system on which to practice the commands covered.

If your fundamental Linux skills are a bit rusty, you may want to first review the [tutorials for the LPIC-1 and LPIC-2 exams](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

System requirements

To follow along with the examples in these tutorials, you need a Linux workstation with the OpenLDAP package and support for PAM. Most modern distributions meet these requirements.

Section 2. Concepts and architecture

This section covers material for topic 301.1 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 3.

In this section, learn about:

- LDAP and X.500 technical specification
- Attribute definitions
- Directory namespaces
- Distinguished names
- LDAP Data Interchange Format
- Meta-directories
- Changetype operations

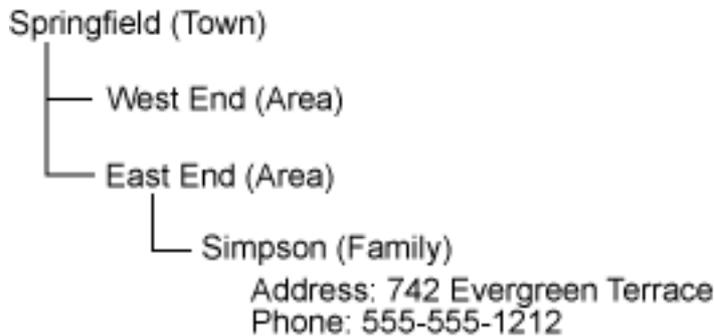
Most of the LPIC-3 exam focuses on the use of the Lightweight Directory Access Protocol (LDAP). Accordingly, the first objective involves understanding what LDAP is, what it does, and some of the basic terminology behind the concept. When you understand this, you will be able to move on to designing your directory and integrating your applications with the directory.

LDAP, what is it?

Before talking about LDAP, let's review the concept of directories. The classic example of a directory is the phone book, where people are listed in alphabetical order along with their phone numbers and addresses. Each person (or family) represents an object, and the phone number and address are attributes of that object. Though not always obvious at a glance, some objects are businesses instead of people, and these may include fax numbers or hours of operation.

Unlike its printed counterpart, a computer directory is hierarchical in nature, allowing for objects to be placed under other objects to indicate a parent-child relationship. For instance, the phone directory could be extended to have objects representing areas of the city, each with the people and business objects falling into their respective area objects. These area objects would then fall under a city object, which might further fall under a state or province object, and so forth, much like Figure 1. This would make a printed copy much harder to use because you would need to know the name and geographical location, but computers are made to sort information and search various parts of the directory, so this is not a problem.

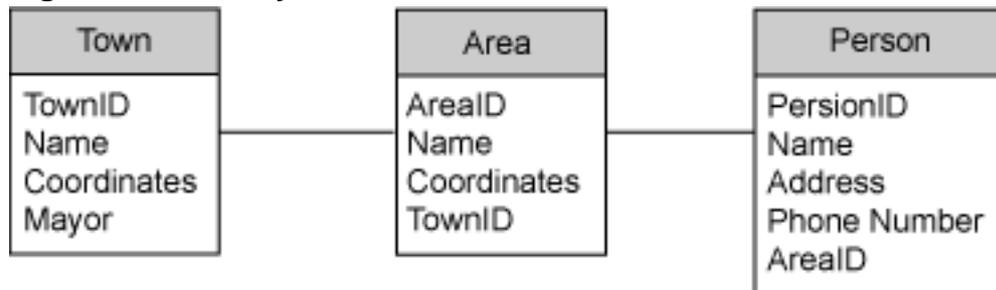
Figure 1. A sample directory



Looking at Figure 1, knowing where the Simpson's record is tells you more than just the address and phone number. You also know they are in the East end in the town of Springfield. This structure is called a tree. Here, the root of the tree is the Springfield object, and the various objects represent further levels of branching.

This directory-based approach to storing data is quite different than the relational databases that you may be familiar with. To compare the two models, Figure 2 shows what the telephone data might look like if modeled as a relational database.

Figure 2. Directory data modeled in relational form



In the relational model, each type of data is a separate table that allows different types of information to be held. Each table also has a link to its parent table so that the relationships between the objects can be held. Note that the tables would have to be altered to add more information fields.

Remember that nothing about the directory model places any restrictions on how the data may be stored on disk. In fact, OpenLDAP supports many back ends including flat files and Structured Query Language (SQL) databases. The mechanics of laying out the tables on disk are largely hidden from you. For instance, Active Directory provides an LDAP interface to its proprietary back end.

The history of LDAP

LDAP was conceived in Request for Comments (RFC) 1487 as a lightweight way to access an X.500 directory instead of the more complex Directory Access Protocol. (See the [Resources](#) section for links to this and related RFCs.) X.500 is a standard (and a family of standards) from the International Telecommunication Union (ITU, formerly the CCITT) that specifies how directories are to be implemented. You may be familiar with the X.509 standard that forms the core of most Public Key Infrastructure (PKI) and Secure Sockets Layer (SSL) certificates. LDAP has since evolved to version 3 and is defined in RFC 4511.

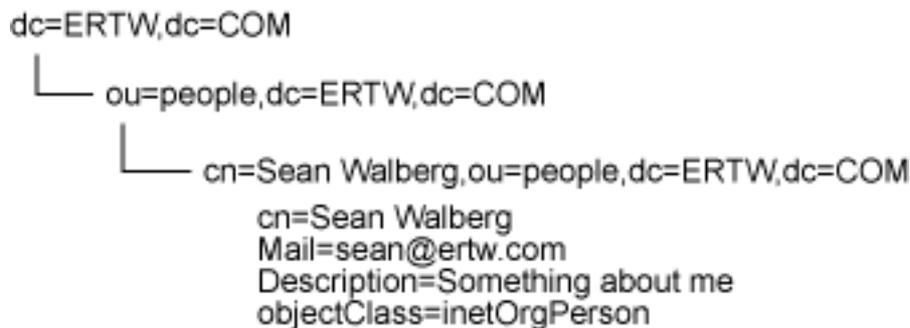
Connecting to an X.500 database initially required the use of the Open Systems Interconnection (OSI) suite of protocols and, in true ITU fashion, required understanding of thick stacks of protocol documentation. LDAP allowed Internet Protocol (IP)-based networks to connect to the same directory with far fewer development cycles than using OSI protocols. Eventually the popularity of IP networks led to the creation of LDAP servers that support only as many X.500 concepts as necessary.

Despite the triumph of LDAP and IP over X.500 and OSI, the underlying organization of the directory data is still X.500-ish. Concepts that you will learn over the course of this tutorial, such as Distinguished Names and Object Identifiers, are brought up from X.500.

X.500 was intended as a way to create a global directory system, mostly to assist with the X.400 series of standards for e-mail. LDAP can be used as a global directory with some effort, but it is mostly used within an enterprise.

A closer look at naming and attributes

In the LDAP world, names are important. Names let you access and search records, and often the name gives an indication of where the record is within the LDAP tree. Figure 3 shows a typical LDAP tree.

Figure 3. A typical LDAP tree showing a user

At the top, or *root*, of the tree is an entity called `dc=ertw,dc=com`. The `dc` is short for *domain component*. Because `ertw` is under the `.com` top-level domain, the two are separated into two different units. Components of a name are concatenated with a comma when using the X.500 nomenclature, with the new components being added to the left. Nothing technically prevents you from referring to the root as `dc=ertw.com`, though in the interest of future interoperability it is best to have the domain components separate (in fact, RFC 2247 recommends the separate domain components).

`dc=ertw,dc=com` is a way to uniquely identify that entity in the tree. In X.500 parlance, this is called the *distinguished name*, or the *DN*. The DN is much like a primary key in the relational-database world because there can be only one entity with a given DN within the tree. The DN of the topmost entry is called the *Root DN*.

Under the root DN is an object with the DN of `ou=people,dc=ertw,dc=com`. `ou` means *organizational unit*, and you can be sure it falls under the root DN because the `ou` appears immediately to the left of the root DN. You can also call `ou=people` the *relative distinguished name*, or *RDN*, because it is unique within its level. Put in recursive terms, the DN of an entity is the entity's RDN plus the DN of the parent. Most LDAP browsers show only the RDN because it eliminates redundancy.

Moving down the tree to `cn=Sean Walberg,ou=people,dc=ertw,dc=com`, you find the record for a person. `cn` means *common name*. For the first time, though, a record has some additional information in the form of *attributes*. Attributes provide additional information about the entity. In fact, you'll see the leftmost component of the DN is duplicated; in this case, it's the `cn` attribute. Put another way, the RDN of an entity is composed of one (or more) attributes of the entity.

While `mail` and `description` are easy enough to understand, `objectClass` is not as obvious. An object class is a group of attributes that correspond to a particular entity type. One object class may contain attributes for people and another for UNIX accounts. By applying the two object classes to an entity, both sets of attributes are available to be stored.

Each object class is assigned an object identifier (OID) that uniquely identifies it. The object class also specifies the attributes, and which ones are mandatory and which

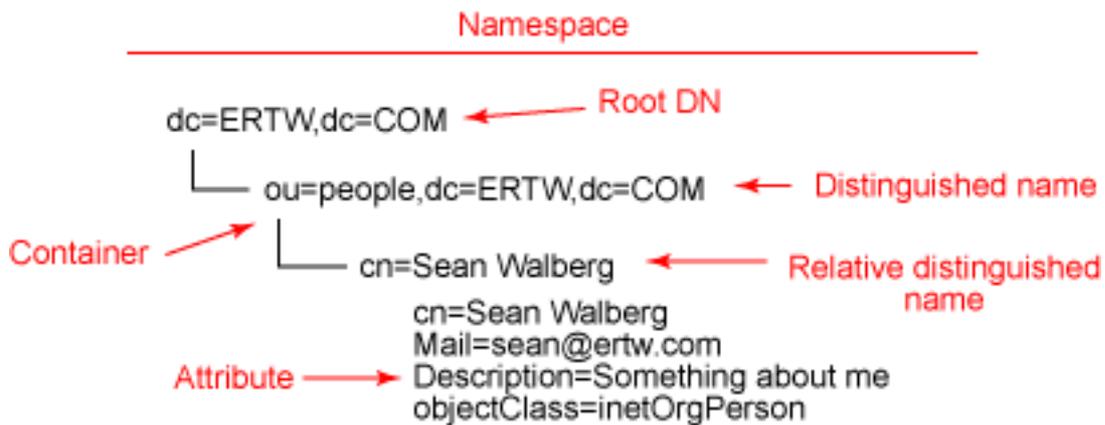
are optional. Mandatory attributes must have some data for the entity to be saved. The object class also identifies the type of data held and whether multiple attributes of the same name are allowed. For instance, a person might have only one employee number but multiple first names (for example, Bob, Robert, and Rob).

The bottom-level objects are not the only ones to have object classes associated with them. These objects, called *containers*, also have object classes and attributes. The `people` `ou` is of type `organizationalUnit` and has a `description` attribute along with `ou=people` to create the RDN. The root of the tree is of type `dcObject` and `organization`. Knowing which object classes to assign an object depends on what is being held in the object and under it. Refer to the [Schemas](#) section for more details.

The root DN also defines the *namespace* of the tree or, to be more technical, the *Directory Information Tree* (DIT). Something ending in `dc=ibm,dc=com` would fall outside of the namespace from [Figure 3](#), whereas the record for Sean Walberg falls within the namespace. With that in mind, though, it is possible that one LDAP server contains multiple namespaces. A somewhat abstract item called the *Root DSE* contains the information about all the namespaces available on the server. DSE means the DSA-Specific Entry, and DSA means Directory System Agent (that is, the LDAP server).

[Figure 4](#) summarizes the terminology associated with the LDAP tree.

Figure 4. Summary of LDAP terminology



Finally, an LDAP tree can be synchronized with other trees or data sources. For instance, one branch of the tree could come from a security system, another from a customer database, and the rest could be stored in the LDAP server. This is called a *meta-directory* and is intended to be a single source of data for applications such as single sign-on.

The LDIF file

Data can get into an LDAP server in one of two ways. Either it can be loaded in over the network, using the LDAP protocol, or it can be imported from the server through a file in the *LDAP Data Interchange Format* (LDIF). LDIF can be used at any time, such as to create the initial tree, and to perform a bulk add or modify of the data some time later. The output of a search can also be in LDIF for easy parsing or import to another server. The full specification for LDIF is in RFC 2849 (see [Resources](#) for a link).

Adding records

The LDIF that generated the tree from Figure 3 is shown in Listing 1.

Listing 1. A simple LDIF file to populate a tree

```
# This is a comment
dn: dc=ertw,dc=com
dc: ertw
description: This is my company
  the description continues on the next line
  indented by one space
objectClass: dcObject
objectClass: organization
o: ERTW.COM

dn: ou=people,dc=ertw,dc=com
ou: people
description: Container for users
objectclass: organizationalunit

dn: cn=Sean Walberg,ou=people,dc=ertw,dc=com
objectclass: inetOrgPerson
cn: Sean Walberg
cn: Sean A. Walberg
sn: Walberg
homephone: 555-111-2222
mail: sean@ertw.com
description: Watch out for this guy
ou: Engineering
```

Before delving into the details of the LDIF file, note that the attribute names are case insensitive. That is, `objectclass` is the same as both `objectClass` and `OBJECTCLASS`. Many people choose to capitalize the first letter of each word except the first, such as `objectClass`, `homePhone`, and `thisIsAreallyLongAttribute`.

The first line of the LDIF shows a UNIX-style comment, which is prefixed by a hash sign (#), otherwise known as a pound sign or an octothorpe. LDIF is a standard ASCII file and can be edited by humans, so comments can be helpful. Comments are ignored by the LDAP server, though.

Records in the LDIF file are separated by a blank line and contain a list of attributes and values separated by a colon (:). Records begin with the `dn` attribute, which identifies the distinguished name of the record. [Figure 1](#), therefore, shows three

records: the `dc=ertw, ou=people, and cn=Sean Walberg` RDNs, respectively.

Choosing attributes

The attribute names may be confusing at this point. How do you choose which object class to assign a record? How do you find out which attributes are available? How do you know that `o` stands for organization?

To put it very simply, the answers to all of these questions lie in the schema, which is covered in [Schemas](#). The schema provides a description of which attributes mean what. The schema also maps attributes into object classes. Adding an object class to a record allows you to use the attributes that fall within it.

The final piece of the puzzle is to understand how the LDAP tree is to be used. If you're going to be authenticating UNIX accounts against the tree, your users had better have an object class that gives them the same `userid` attribute that your system is looking for.

Looking back at [Figure 1](#), you can see the first record defined is the root of the tree. The distinguished name comes first. Next comes a list of all the attributes and values, separated by a colon. Colons within the value do not need any special treatment. The LDAP tools understand that the first colon separates the attribute from the value. If you need to define two values for an attribute, then simply list them as two separate lines. For example, the root object defines two object classes.

Each record must define at least one object class. The object class, in turn, may require that certain attributes be present. In the case of the root object, the `dcObject` object class requires that a *domain component*, or `dc`, be defined, and the `organization` object class requires that an *organization* attribute, or `o`, be defined. Because an object must have an attribute and value corresponding to the RDN, the `dcObject` object class is required to import the `dc` attribute. Defining an `o` attribute is not required to create a valid record.

A `description` is also used on the root object to describe the company. The purpose here is to demonstrate the comment format. If your value needs to span multiple lines, start each new line with a leading space instead of a value. Remember that specifying multiple `attribute: value` pairs defines multiple instances of the attribute.

The second record in [Figure 1](#) defines an `organizationalUnit`, which is a container for people objects in this case. The third defines a user of type `inetOrgPerson`, which provides common attributes for defining people within an organization. Note that two `cn` attributes are defined; one is also used in the DN of the record. The second, with the middle initial, will help for searching, but it is the first that is required to satisfy the condition that the RDN be defined.

In the user record there is also an `ou` that does not correspond to the `organizationalUnit` the user is in. The container the user object belongs to can always be found by parsing the DN. This `ou` attribute refers to something defined by the user, in this case a department. No referential integrity is imposed by the server, though the application may be looking for a valid DN such as `ou=Engineering,ou=Groups,dc=ertw,dc=com`.

The only other restriction placed on LDIF files that add records is that the tree must be built in order, from the root. [Figure 1](#) shows the root object being built, then an `ou`, then a user within that `ou`. Now that the structure is built, users can be added directly to the `people` container, but if a new container is to be used, it must be created first.

The LDIF behind adding objects is quite easy. The format gets more complex when objects must be changed or deleted. LDIF defines a `changetype`, which can be one of the following:

- `add` adds an item (default).
- `delete` deletes the item specified by the DN.
- `modrdn` renames the specified object within the current container, or moves the object to another part of the tree.
- `moddn` is synonymous with `modrdn`.
- `modify` makes changes to attributes within the current DN.

Deleting users

Deleting an item is the simplest case, only requiring the `dn` and `changetype`. Listing 2 shows a user being deleted.

Listing 2. Deleting a user with LDIF

```
dn: cn=Fred Smith,ou=people,dc=ertw,dc=com
changetype: delete
```

Manipulating the DN

Manipulating the DN of the object is slightly more complex. Despite the fact that there are two commands, `moddn` and `modrdn`, they do the same thing! The operation consists of three separate parts:

1. Specify the new RDN (leftmost component of the DN).
2. Determine if the old RDN should be replaced by the new RDN within the

record, or if it should be left.

3. Optionally, move the record to a new part of the tree by specifying a new parent DN.

Consider Jane Smith, who changes her name to Jane Doe. The first thing to do is change her `cn` attribute to reflect the name change. Because the new name is the primary way she wishes to be referred to, and the common name forms part of the DN, the `moddn` operation is appropriate. (If the common name weren't part of the DN, this would be an attribute change, which is covered in the next section.) The second choice is to determine if the `cn: Jane Smith` should stay in addition to `cn: Jane Doe`, which allows people to search for her under either name. Listing 3 shows the LDIF that performs the change.

Listing 3. LDIF to change a user's RDN

```
# Specify the record to operate on
dn: cn=Jane Smith,ou=people,dc=ertw,dc=com
changetype: moddn
# Specify the new RDN, including the attribute
newrdn: cn=Jane Doe
# Should the old RDN (cn=Jane Smith) be deleted? 1/0, Default = 1 (yes)
deleteoldrdn: 0
```

Listing 3 begins by identifying Jane's record, then the `moddn` operator. The new RDN is specified, continuing to use a common name type but with the new name. Finally, `deleteoldrdn` directs the server to keep the old name. Note that while `newrdn` is the only necessary option to the `moddn` `changetype`, if you omit `deleteoldrdn`, the action is to delete the old RDN. According to RFC 2849, `deleteoldrdn` is a required element.

Should the new Mrs. Jane Doe be sent to a new part of the tree, such as a move to `ou=managers,dc=ertw,dc=com`, the LDIF must specify the new part of the tree somehow, such as in Listing 4.

Listing 4. Moving a record to a new part of the tree

```
dn: cn=Jane Doe,ou=people,dc=ertw,dc=com
changetype: modrdn
newrdn: cn=Jane Doe
deleteoldrdn: 0
newsuperior: ou=managers,dc=ertw,dc=com
```

Curiously, a new RDN must be specified even though it is identical to the old one, and the OpenLDAP parser now requires that `deleteoldrdn` is present despite it being meaningless when the RDN stays the same. `newsuperior` follows, which is the DN of the new parent in the tree.

One final note on the `modrdn` operation is that the order matters, unlike most other LDIF formats. After the `changetype` comes the `newrdn`, followed by `deleteoldrdn`, and, optionally, `newsuperior`.

Modifying attributes

The final `changetype` is `modify`, which is used to modify attributes of a record. Based on the earlier discussion of `moddn`, it should be clear that `modify` does not apply to the DN or the RDN of a record.

Listing 5 shows several modifications made to a single record.

Listing 5. Modifying a record through LDIF

```
dn: cn=Sean Walberg,dc=ertw,dc=com
changetype: modify
replace: homePhone
homePhone: 555-222-3333
-
changetype: modify
add: title
title: network guy
-
changetype: modify
delete: mail
-
```

The LDIF for the `modify` operation looks similar to the others. It begins with the DN of the record, then the `changetype`. After that comes either `replace:`, `add:`, or `delete:`, followed by the attribute. For `delete`, this is enough information. The others require the attribute:value pair. Each change is followed by a dash (-) on a blank line, including the final change.

LDIF has an easy-to-read format, both for humans and computers. For bulk import and export of data, LDIF is a useful tool.

Section 3. Directory design

This section covers material for topic 301.2 for the Senior Level Linux Professional (LPIC-3) exam 301. The topic has a weight of 2.

In this section, learn about:

- Defining LDAP directory content

- Directory organization
- How to plan appropriate Directory Information Trees

Determining if LDAP is appropriate

Like any other tool, LDAP is not appropriate for every solution. Before choosing LDAP, you must ask yourself some questions:

- How often will changes be made, and what kind of changes are they?
- What will be using the data?
- What kind of queries will be made against the data?
- Is the information hierarchical in nature?

LDAP databases are geared toward read-intensive operations. People may only change personal information a few times a year, but we can expect to look up attributes far more than that, such as resolving the `UserID` owning a file to a printable name when looking through a directory. LDAP data tends to be heavily indexed, which means every change to the underlying data requires multiple changes to the indexes that help the server find the data later. LDAP also doesn't offer a means to make mass updates to the tree, other than performing a search and then modifying each individual DN.

The LDAP specifications do not define transactions, which are common in relational databases. Transactions ensure that all the operations within the transaction succeed, or else the data is rolled back to the pre-transaction state (individual servers may implement this, but it is not a requirement). These limitations on updating data and the lack of transactions make LDAP a poor choice for bank transactions.

Determining the user, or the consumer, of the data is also important. If none of the consumers speak LDAP, then LDAP may not be a good fit. To LDAP's credit, it is an extremely simple protocol to implement and is available for most languages on most platforms. With a mere handful of available operations defined, it can be integrated into existing applications with ease.

LDAP provides search functionality, but with nowhere near the level of a relational database. The LDAP server may store the underlying data using SQL, but you as the user are abstracted from this and cannot make use of it. Thus, you are limited to the search filters that are supported by your LDAP server. These filters will be investigated in more detail in later articles in this series, but for now understand that the filters are just that -- filters. You can perform some powerful searches, such as "show me all the employees who live in Washington and those that live in Texas who

are over 40." Statements equivalent to the SQL `GROUP BY` are not available, though. LDAP is best suited for look-up style operations, such as "show me the username of the person with UID 4131," and "give me the names of everyone working for Jim Smith."

Finally, the information you are trying to store should lend itself to hierarchical storage. You could store a flat list of data in an LDAP server, but it would probably be a waste of resources.

When you can answer these questions and have determined that LDAP is the correct solution, it is time to design the directory tree.

Organizing your tree

The foremost idea to keep in your mind is that reorganization of the tree is undesirable. The goal is to break down your objects such that each branch of the tree holds objects of a similar type, but the chances of an object having to be moved is low. The reasons for this are twofold. One is that it's just a hassle to do. The second is that a move changes the DN of the object, and then you have to update all the objects that reference the moved object.

The root DN

The root of your tree should be something that represents your company. RFC 2247 calls for the use of the `dc` attribute and the familiar `dc=example,dc=com` format to map the company's primary domain into a DN. Your company probably has many Internet domains but has one that is preferred. It is also common to choose something under the `local` top-level domain (TLD), which doesn't currently exist on the Internet. Microsoft has long suggested using this TLD for their Active Directory implementations despite it not being a reserved TLD.

If you don't want to use a domain name in your root DN, you can simply have an object with `objectclass: organization`, which gives a root DN of `o=My Corporation`.

Filling in the structure

Deciding what goes at the next level of the DIT is difficult. It is often tempting to describe the company's organizational structure as a series of nested `organizationalUnit` objects, but companies are constantly reorganizing, which breaks the first rule. Instead, consider using attributes to store this information instead.

Depending on your use of the LDAP server and how you choose to name objects, you may want to keep all users in one tree or separate them according to role. For

instance, you can create one container for employees and one for customers, or you can group them all into a single container. The choice depends on your applications and how you plan on managing the tree. If the sales department takes care of managing the customer list, and the systems administrators take care of the staff, it might be better to create two containers. Figure 5 shows a DIT that has been broken down into customers and users.

Figure 5. An LDAP tree with separate branches for users and customers

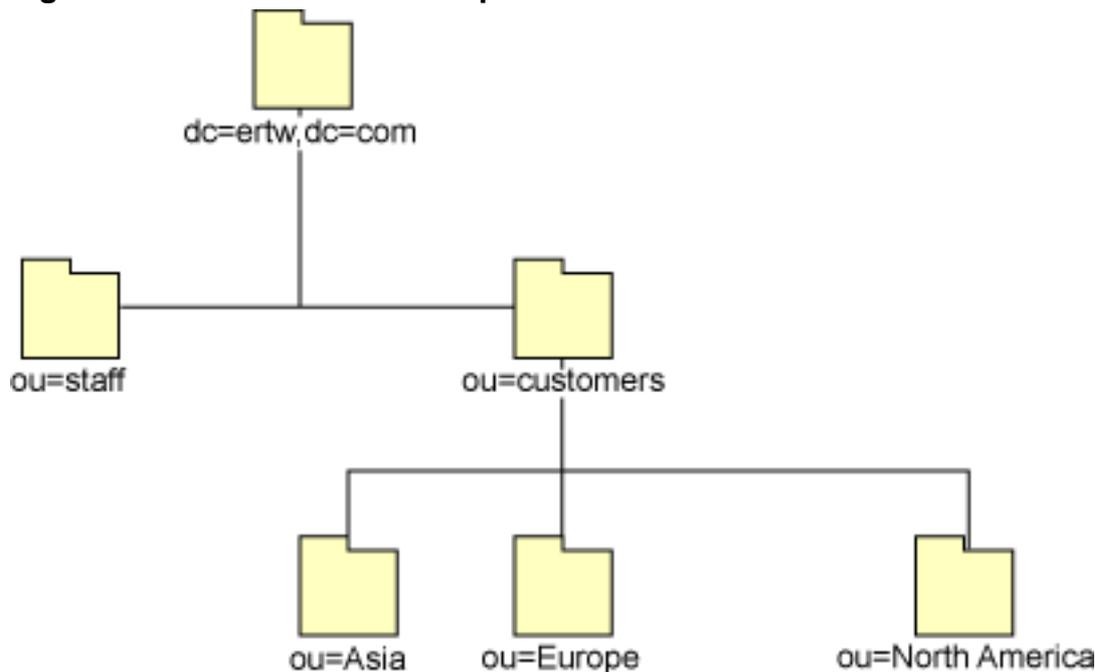


Figure 5 shows one grouping for staff and one for customers. All staff reside in the same organizational unit (OU), but customers are broken up by region. In this situation, this allows different people to manage their own region's customers.

If you plan on using LDAP for authentication of UNIX resources, you must then decide where to store that information. User accounts have already been taken care of earlier, but you will need to store groups, and possibly other maps such as hosts, services, networks, and aliases. Which of these you store in LDAP, and which you leave as local files depends on you and if you need to be able to update them centrally.

The simplest case is to store each map in a separate `organizationalUnit`. You will find that when configuring your UNIX system to read this information, you need to specify the DN of the container and any filters. If you store groups in the user OU, you will have to write some filters. You also may have to adjust the filters if you ever make any structural changes to the staff OU.

Determining the object classes

So far the discussion has centered on the layout of the LDAP tree with the basic goal being to avoid having to rename objects in the future. After you decide upon the tree, you must then decide which object classes to use. It is certainly possible to assign different object classes to objects under the same branch of the tree, but this will almost certainly lead to maintenance problems in the future.

To add more stress to the situation, it's not always possible to add more object classes to an object if you make a mistake. LDAP schemas define two types of object classes: *structural* and *auxiliary*. Structural object classes usually inherit properties from other object classes in a chain that ends up at an object class called `top`. Structural object classes can be said to define the object's identity, while auxiliary object classes are there to add attributes. `organizationalUnit` is a structural object class, as is `inetOrgPerson`. Going back to [Listing 1](#), the top-level entry had two object classes: `dcObject` and `organization`. `organization` is the structural object class. `dcObject` plays the auxiliary role by defining the `dc` attribute

The part that can cause problems is that an entry can only have one structural object class. Sometimes you see `inetOrgPerson`, `organizationalPerson`, `person`, and `top` in the same record, but they are all part of the same inheritance tree. `inetOrgPerson` and `account` are both structural, are not in the same inheritance tree, and therefore can't be used together. Some LDAP servers permit this, but eventually this behavior may change and cause problems.

There is a third type of object class called **abstract**. It is much like structural except that it must be inherited to be used. `top` is such a class.

Without getting into the specifics of each application there are some general structural object classes that are useful:

- `inetOrgPerson`: Defines a generic person, along with some contact information.
- `organizationalRole`: Much like a person, but it defines a generic role such as IT Helpdesk or Fire Warden.
- `organizationalUnit`: A generic container, may describe a department within a container or may be used to separate various parts of the LDAP tree such as groups, people, and servers
- `organization`: A company or other organization.
- `groupOfNames`: Stores one or more DNs referring to members of a group. Not necessarily useful for UNIX groups, but helpful for meeting invitations or other simple things.

Stick to these for your people and organizations and you will be safe. Most

extensions, such as authentication, use auxiliary attributes. When in doubt, consult the schema.

The final design consideration is the choice of DNs. Most branches are fairly easy because there is no chance of duplication. A UNIX group's name and ID is unique, so using `cn` as the RDN of a group is possible. What happens when you have two employees called Fred Smith? Because the DN must be unique, `cn=Fred Smith,ou=People,dc=example,dc=com` could be either of them. Either something else must be used, such as `employeeNumber`, or the RDN will have to be made from two different attributes separated by a plus sign (+). For example, `cn=Fred Smith+userid=123,ou=people,dc=example,dc=com` has an RDN made from two different attributes. Whatever you do, do it consistently!

Section 4. Schemas

This section covers material for topic 301.3 for the Senior Level Linux Professional (LPIC-3) exam 301. The topic has a weight of 3.

In this section, learn about:

- LDAP schema concepts
- How to create and modify schemas
- Attribute and object class syntax

Up until now, the schema has been mentioned several times but not fully explained. The schema is a collection of object classes and their attributes. A schema file contains one or more object classes and attributes in a text format the LDAP server can understand. You import the schema file into your LDAP server's configuration, and then use the object classes and attributes in your objects. If the available schemas don't fit your needs, you can create your own or extend an existing one.

LDAP schema concepts

Technically, a schema is a packaging mechanism for object classes and attributes. However, the grouping of object classes is not random. Schemas are generally organized along an application, such as a core, X.500 compatibility, UNIX network services, sendmail, and so on. If you have a need to integrate an application with LDAP, you generally have to add a schema to your LDAP server.

A more in-depth look at OpenLDAP configuration will be in a later tutorial in this series, but the way to add a schema is with `include /path/to/file.schema`. After restarting the server, the new schema will be available.

When the schema is loaded, you then apply the new object classes to the relevant objects. This can be done through an LDIF file or through the LDAP application program interface (API). Applying the object classes gives you more attributes to use.

Creating and modifying schemas

Schemas have a fairly simple format. Listing 6 shows the schema for the `inetOrgPerson` object class along with some of its attributes.

Listing 6. Part of the `inetOrgPerson` definition

```

attributetype ( 2.16.840.1.113730.3.1.241
    NAME 'displayName'
    DESC 'RFC2798: preferred name to be used when displaying entries'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    SINGLE-VALUE )

attributetype ( 0.9.2342.19200300.100.1.60
    NAME 'jpegPhoto'
    DESC 'RFC2798: a JPEG image'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.28 )

objectclass ( 2.16.840.1.113730.3.2.2
    NAME 'inetOrgPerson'
    DESC 'RFC2798: Internet Organizational Person'
    SUP organizationalPerson
    STRUCTURAL
    MAY (
        audio $ businessCategory $ carLicense $ departmentNumber $
        displayName $ employeeNumber $ employeeType $ givenName $
        homePhone $ homePostalAddress $ initials $ jpegPhoto $
        labeledURI $ mail $ manager $ mobile $ o $ pager $
        photo $ roomNumber $ secretary $ uid $ userCertificate $
        x500uniqueIdentifier $ preferredLanguage $
        userSMIMECertificate $ userPKCS12 )
    )

```

Line spacing is not important in schema files -- it is mostly there for human readability. The first definition is an `attributetype`, which means an attribute. The parentheses enclose the definition of the attribute. First comes a series of numbers, separated by periods, called the *object ID*, or *OID*, which is a globally unique number. The OID is also hierarchical, such that if you're assigned the 1.1 tree, you can create anything like 1.1.1 or 1.1.2.3.4.5.6 without having to register it.

Registering OIDs

You can't just pick any series of numbers for your own OID because

you don't know if the value you choose is, or will be, in use elsewhere. OIDs must be unique. Some servers let you specify textual OIDs (mycompany.1.2) which would be unique, just not necessarily compatible. Anything under the 1.1 namespace can be used locally but is not guaranteed to be unique.

The best solution is to register your own OID. There are many ways to do this, depending on if you are a country, a telephone carrier, or fall under other categories. Luckily, the Internet Assigned Numbers Authority (IANA) will freely give you your own branch under .1.3.6.4.1 if you ask.

Following the OID is a series of keywords, each of which may have a value after it. First the `NAME` of the attribute defines the name that humans will use, such as in the LDIF file or when retrieving the information through the LDAP API. Sometimes you might see the name in the form of `NAME ('foo' 'bar')`, which means that either `foo` or `bar` are acceptable. The server, however, considers the first to be the primary name of the attribute.

`DESC` provides a description of the attribute. This helps you understand the attribute if you're browsing the schema file. `EQUALITY`, `SUBSTR`, and `ORDERING` (not shown) require a *matching rule*. This defines how strings are compared, searched, and sorted, respectively. `caseIgnoreMatch` is a case-insensitive match, and `caseIgnoreSubstringsMatch` is also case insensitive. See the [Resources](#) section for Web sites that define all the standard matching rules. Like most things in LDAP, a server can define its own matching methods for its own attributes, so there are no comprehensive lists of matching rules.

The `SYNTAX` of the attribute defines the format of the data by referencing an OID. RFC 2252 lists the standard syntaxes and their OIDs. If the OID is immediately followed by a number in curly braces (`{}`), this represents the maximum length of the data. `1.3.6.1.4.1.1466.115.121.1.15` represents a `DirectoryString` that is a UTF-8 string.

Finally, the `SINGLE-VALUE` keyword has no arguments and specifies that only one instance of `displayName` is allowed.

The `jpegPhoto` attribute has a very short definition: just the OID, the name and description, and a syntax meaning a JPEG object, which is an encoded string of the binary data. It is not practical to search or sort a picture, and multiple photos can exist in a single record.

Defining an object class follows a similar method. The `objectclass` keyword starts the definition, followed by parentheses, the OID of the object class, and then the definitions. `NAME` and `DESC` are the same as before. `SUP` defines a superior object class, which is another way of saying that the object class being defined inherits from the object class specified by the `SUP` keyword. Thus, an `inetOrgPerson` carries the attributes of an `organizationalPerson`.

The `STRUCTURAL` keyword defines this as a structural object class, which can be considered the primary type of the object. Other options are `AUXILIARY`, which adds attributes to an existing object, and `ABSTRACT`, which is the same as structural but cannot be used directly. Abstract object classes must be inherited by another object class, which can then be used. The `top` object class is abstract. It is inherited by most other structural object classes, including `person`, which is the parent of `organizationalPerson`, which, in turn, is inherited by `inetOrgPerson`.

Two keywords, `MAY` and `MUST`, define the attributes that are allowed and mandatory, respectively, for records using that particular object class. For mandatory items, you may not save a record without all the items being defined. Each attribute is separated by a dollar sign (\$), even if the line continues on the next line.

It is not a good idea to modify structural object classes, or even existing, well-known, auxiliary object classes. Because these are well known, you may cause incompatibility issues in the future if your server is different. Usually the best solution is to define your own auxiliary object class, create a local schema, and apply it to your records. For instance, if you are a university and want to store student attributes, you might consider creating a student object class that is inherited from either `organizationalPerson` or `inetOrgPerson` and adding your own attributes. You could then create auxiliary object classes to add more attributes such as class schedules.

Understanding which schemas to use

After learning about how schemas are created, it is tempting to start fresh -- to create your own schema based on your environment. This would certainly take care of your present needs, but it could quite possibly make things more difficult in the long run as you add more functionality to your LDAP tree and integrate other systems. The best approach is to stick with standard object classes and attributes when you can and extend when you must.

OpenLDAP usually stores its schema files in `/etc/openldap/schema`, in files with a `.schema` extension. Table 3 lists the default schemas along with their purposes.

Table 3. Schemas that ship with OpenLDAP

File name	Purpose
<code>corba.schema</code>	Defines some object classes and attributes for handling Common Object Request Broker Architecture (CORBA) object references across multiple machines.
<code>core.schema</code>	Defines many common attributes and object classes. This schema is where you will find the <code>organizationalUnit</code> , <code>top</code> , <code>dcObject</code> , and

	<code>organizationalRole</code> . <code>core.schema</code> is the first place you should look if you need to find something.
<code>cosine.schema</code>	Attributes and object classes from the X.500 specifications. While there are some useful things in here, there are often better alternatives in others such as <code>core</code> and <code>inetorgperson</code> .
<code>dyngroup.schema</code>	An experimental set of objects used with Netscape Enterprise Server.
<code>inetorgperson.schema</code>	Defines the <code>inetOrgPerson</code> object (which extends objects from <code>core.schema</code>).
<code>java.schema</code>	Like <code>corba.schema</code> , this schema defines a series of object classes and attributes to handle the lookup of Java™ classes within an LDAP tree.
<code>misc.schema</code>	Implements some objects to handle mail lookups within the tree. It is best to consult your e-mail server's documentation to see which schema it uses.
<code>nis.schema</code>	This is the schema you use if you move authentication to LDAP. <code>nis.schema</code> defines <code>posixAccount</code> , which provides attributes for storing authentication data within the user's object. It also has the various map types to handle groups, networks, services, and other files that go into network-based authentication mechanisms such as the Network Information System (NIS).
<code>openldap.schema</code>	This is more for example purposes and shows some basic objects.
<code>ppolicy.schema</code>	A set of objects to implement password policies in LDAP, such as aging. Note that some of these are handled by the traditional UNIX shadow mechanisms and are already handled in <code>nis.schema</code> .

In addition, RFC 4519 explains common attributes. After finding the attributes you want, you can then look through the schema files to determine which files need to be included in your LDAP configuration and which object classes you must use for your records.

Section 5. Summary

In this tutorial you learned about LDAP concepts, architecture, and design. LDAP grew out of a need to connect to X.500 directories over IP in a simplified way. A directory presents data to you in a hierarchical manner, much like a tree. Within this tree are records that are identified by a distinguished name and have many attribute-value pairs, including one or more object classes that determine what data can be stored in the record.

LDAP itself refers to the protocol used to search and modify the tree. Practically though, the term LDAP is used for all components, such as LDAP server, LDAP data, or just "It's in LDAP".

Data in LDAP is often imported and exported with LDIF, which is a textual representation of the data. An LDIF file specifies a `changetype`, such as `add`, `delete`, `modrdn`, `moddn`, and `modify`. These operations let you add entries, delete entries, move data around in the tree, and change attributes of the data.

Designing the tree correctly is crucial to long-term viability of the LDAP server. A correct design means fewer change operations are needed, which leads to consistent data that can easily be found by other applications. By choosing common attributes, you ensure that other consumers of the LDAP data understand the meaning of the attributes and that fewer translations are required.

The LDAP schema dictates which attributes can be used in your server. Within the schema are definitions of the attributes, including OIDs to uniquely identify them, instructions on how to store and sort the data, and textual descriptions of what the attributes do. Object classes group attributes together and can be defined as structural, auxiliary, or abstract.

Structural object classes define the record, so a record may only have one structural object class. Auxiliary object classes add more attributes for specific purposes and can be added to any record. An abstract object class must be inherited and cannot be used directly.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- [RFC 1487](#), *X.500 Lightweight Directory Access Protocol*, gives you some insight into the development of LDAP and the history of X.500.
- [RFC 2247](#), *Using Domains in LDAP/X.500 Distinguished Names*, is a brief description of the `domainComponent` attribute and how to use it properly in your LDAP tree.
- [RFC 2252](#), *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*, lists the standard syntaxes for attributes, which can help you figure out what format a certain attribute is expecting.
- [RFC 2849](#), *The LDAP Data Interchange Format (LDIF) - Technical Specification*, describes the LDIF language. It uses Backus-Naur Form (BNF) to specify the language, which can be tricky to understand. [RFC 2234](#), *Augmented BNF for Syntax Specifications: ABNF*, might help you understand the various operators.
- [RFC 4511](#), *Lightweight Directory Access Protocol (LDAP): The Protocol*, is the latest draft of the LDAP protocol.
- [RFC 4519](#), *Lightweight Directory Access Protocol (LDAP): Schema for User Applications*, is an updated list of the commonly-used attributes; this list helps ensure you're using the same attributes everyone else is to describe the same data.
- The [OID descriptions for 2.5.13](#) link to detailed descriptions of how each matching rule (string comparison, substrings, and ordering) works.
- This [FAQ entry on object classes](#) gives details on some of trickier rules of dealing with object classes. Some of OpenLDAP's error messages are terse, especially when dealing with LDIF imports.
- The [overlay](#) framework in OpenLDAP is key because the meta-directory concept can be carried further than just tying together multiple LDAP servers. A request for a particular tree or OID can be directed to custom code that can call a script, read a database, or call an API. Another description is on the [Symas Corporation](#) Web site.
- "[Demystifying LDAP Data](#)" (O'Reilly, November 2006) explains object class

inheritance. It looks at the `inetOrgPerson` object class and describes structural and auxiliary object classes.

- [LDAP for Rocket Scientists](#) is an excellent open source guide, despite being a work in progress.
- In the [developerWorks Linux zone](#), find more resources for Linux developers, and scan our [most popular articles and tutorials](#).
- See all [Linux tips](#) and [Linux tutorials](#) on developerWorks.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [OpenLDAP](#) is a great choice if you're looking for an LDAP server.
- [phpLDAPadmin](#) is a Web-based LDAP administration tool.
- [Luma](#) is a good GUI to look at if that's more your style.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the [developerWorks community](#) through blogs, forums, podcasts, and community topics in our [new developerWorks spaces](#).

About the author

Sean A. Walberg

Sean Walberg has been working with Linux and UNIX systems since 1994 in academic, corporate, and Internet Service Provider environments. He has written extensively about systems administration over the past several years. You can contact him at sean@ertw.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 301 prep, Topic 302: Installation and development

Senior Level Linux Professional (LPIC-3)

Skill Level: Intermediate

[Sean Walberg](mailto:sean@ertw.com) (sean@ertw.com)
Network engineer
Freelance

04 Dec 2007

In this tutorial, Sean Walberg helps you prepare to take the Linux Professional Institute® Senior Level Linux Professional (LPIC-3) exam. In this second in a series of six tutorials, Sean walks you through installing and configuring a Lightweight Directory Access Protocol (LDAP) server, and writing some Perl scripts to access the data. By the end of this tutorial, you'll know about LDAP server installation, configuration, and programming.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *advanced level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102. To attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active advanced-level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the five junior, advanced, and senior certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. Table 1 lists the six topics and corresponding developerWorks tutorials for LPI exam 301.

Table 1. LPI exam 301: Tutorials and topics

LPI exam 301 topic	developerWorks tutorial	Tutorial summary
Topic 301	LPI exam 301 prep: Concepts, architecture, and design	Learn about LDAP concepts and architecture, how to design and implement an LDAP directory, and about schemas.
Topic 302	LPI exam 301 prep: Installation and development	(This tutorial) Learn how to install, configure, and use the OpenLDAP software. See the detailed objectives .
Topic 303	LPI exam 301 prep: Configuration	Coming soon.
Topic 304	LPI exam 301 prep: Usage	Coming soon.
Topic 305	LPI exam 301 prep: Integration and migration	Coming soon.
Topic 306	LPI exam 301 prep: Capacity planning	Coming soon.

To pass exam 301 (and attain certification level 3), you should:

- Have several years of experience in installing and maintaining Linux on a number of computers for various purposes
- Have integration experience with diverse technologies and operating systems
- Have professional experience as, or training to be, an enterprise-level Linux professional (including having experience as a part of another role)
- Know advanced and enterprise levels of Linux administration including installation, management, security, troubleshooting, and maintenance.
- Be able to use open source tools to measure capacity planning and troubleshoot resource problems
- Have professional experience using LDAP to integrate with UNIX® services and Microsoft® Windows® services, including Samba, Pluggable Authentication Modules (PAM), e-mail, and Active Directory
- Be able to plan, architect, design, build, and implement a full environment

using Samba and LDAP as well as measure the capacity planning and security of the services

- Be able create scripts in Bash or Perl or have knowledge of at least one system programming language (such as C)

The Linux Professional Institute doesn't endorse any third-party exam preparation material or techniques in particular.

About this tutorial

Welcome to "Installation and development," the second of six tutorials designed to prepare you for LPI exam 301. In this tutorial, you learn about LDAP server installation and configuration, and how to use Perl to access your new LDAP server.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weights.

Objectives

Table 2 shows the detailed objectives for this tutorial.

Table 2. Installation and development: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
302.1 Compiling and installing OpenLDAP	3	Compile and install OpenLDAP from source and from packages
302.2 Developing for LDAP with Perl/C++	1	Write basic Perl scripts to interact with an LDAP directory

Prerequisites

To get the most from this tutorial, you should have advanced knowledge of Linux and a working Linux system on which to practice the commands covered.

If your fundamental Linux skills are a bit rusty, you may want to first review the [tutorials for the LPIC-1 and LPIC-2 exams](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

System requirements

To follow along with the examples in these tutorials, you'll need a Linux workstation with the OpenLDAP package and support for PAM. Most modern distributions meet these requirements.

Section 2. Compiling and installing OpenLDAP

This section covers material for topic 302.1 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 3.

In this section, learn how to:

- Compile and configure OpenLDAP from source
- Understand OpenLDAP backend databases
- Manage OpenLDAP daemons
- Troubleshoot errors during installation

OpenLDAP is an open source application that implements an LDAP server and associated tools. Because it's open source, you can download the source code free of charge. The OpenLDAP project doesn't distribute binaries directly, but most major distributions package it themselves. In this tutorial, you learn how to install OpenLDAP from both source and packages.

Compiling from source

The first order of business is to download the latest version of OpenLDAP from the project site (see the [Resources](#) section for a download link). The project generally has two active versions available: One is a stable version, and the other is a test version. This tutorial was written using the stable versions 2.3.30 and 2.3.38. If you're following along, some of the directory names may be different depending on which version you have.

To extract the source code from the downloaded tarball, enter `tar -xzf openldap-stable-20070831.tgz`. This decompresses and untars the downloaded file into a directory. Change into the new directory with `cd openldap-2.3.38` (substituting your version of OpenLDAP as appropriate).

At this point, you're in the source directory. You must now configure the build environment for your system and then build the software. OpenLDAP uses a script called `configure` that performs these actions. Type `./configure --help` to see all the options available to you. Some define where the files are installed to (such as `--prefix`); others define the OpenLDAP features you wish to build. Listing 1 lists the features and their defaults.

Listing 1. Configuration options relating to OpenLDAP features

```

SLAPD (Standalone LDAP Daemon) Options:
--enable-slapd           enable building slapd [yes]
--enable-aci             enable per-object ACIs (experimental) [no]
--enable-cleartext      enable cleartext passwords [yes]
--enable-crypt          enable crypt(3) passwords [no]
--enable-lmpasswd       enable LAN Manager passwords [no]
--enable-spasswd        enable (Cyrus) SASL password verification [no]
--enable-modules        enable dynamic module support [no]
--enable-rewrite        enable DN rewriting in back-ldap and rwm overlay [auto]
--enable-reverselookups enable reverse lookups of client hostnames [no]
--enable-slapi          enable SLAPI support (experimental) [no]
--enable-slp            enable SLPv2 support [no]
--enable-wrappers       enable tcp wrapper support [no]

SLAPD Backend Options:
--enable-backends       enable all available backends no|yes|mod
--enable-bdb            enable Berkeley DB backend no|yes|mod [yes]
--enable-dnssrv        enable dnssrv backend no|yes|mod [no]
--enable-hdb           enable Hierarchical DB backend no|yes|mod [yes]
--enable-ldap          enable ldap backend no|yes|mod [no]
--enable-ldbm          enable ldbm backend no|yes|mod [no]
--enable-ldbm-api      use LDBM API auto|berkeley|bcompat|mdbm|gdbm [auto]
--enable-ldbm-type     use LDBM type auto|btree|hash [auto]
--enable-meta          enable metadirectory backend no|yes|mod [no]
--enable-monitor       enable monitor backend no|yes|mod [yes]
--enable-null          enable null backend no|yes|mod [no]
--enable-passwd        enable passwd backend no|yes|mod [no]
--enable-perl          enable perl backend no|yes|mod [no]
--enable-relay         enable relay backend no|yes|mod [yes]
--enable-shell         enable shell backend no|yes|mod [no]
--enable-sql           enable sql backend no|yes|mod [no]

SLAPD Overlay Options:
--enable-overlays       enable all available overlays no|yes|mod
--enable-accesslog     In-Directory Access Logging overlay no|yes|mod [no]
--enable-auditlog      Audit Logging overlay no|yes|mod [no]
--enable-denyop        Deny Operation overlay no|yes|mod [no]
--enable-dyngroup      Dynamic Group overlay no|yes|mod [no]
--enable-dynlist       Dynamic List overlay no|yes|mod [no]
--enable-lastmod       Last Modification overlay no|yes|mod [no]
--enable-ppolicy       Password Policy overlay no|yes|mod [no]
--enable-proxycache    Proxy Cache overlay no|yes|mod [no]
--enable-refint        Referential Integrity overlay no|yes|mod [no]
--enable-retcode       Return Code testing overlay no|yes|mod [no]
--enable-rwm           Rewrite/Remap overlay no|yes|mod [no]
--enable-syncprov       Syncrepl Provider overlay no|yes|mod [yes]
--enable-translucent   Translucent Proxy overlay no|yes|mod [no]
--enable-unique        Attribute Uniqueness overlay no|yes|mod [no]
--enable-valsort       Value Sorting overlay no|yes|mod [no]

SLURPD (Replication Daemon) Options:
--enable-slurpd        enable building slurpd [auto]

Optional Packages:
--with-PACKAGE[=ARG]   use PACKAGE [ARG=yes]

```

```

--without-PACKAGE      do not use PACKAGE (same as --with-PACKAGE=no)
--with-subdir=DIR      change default subdirectory used for installs
--with-cyrus-sasl      with Cyrus SASL support [auto]
--with-fetch           with fetch(3) URL support [auto]
--with-threads         with threads [auto]
--with-tls             with TLS/SSL support [auto]
--with-yielding-select with implicitly yielding select [auto]
--with-odbc            with specific ODBC support iodbc|unixodbc|auto [auto]

--with-gnu-ld          assume the C compiler uses GNU ld [default=no]
--with-pic             try to use only PIC/non-PIC objects [default=use
both]

--with-tags[=TAGS]    include additional configurations [automatic]

```

In Listing 1, you can see that many features are disabled by default, such as metadirectories and modules. In addition, many options are marked as "auto," which turns on features if the proper libraries are present on your system. Instead of relying on this automatic behavior, it's best to make a list of the required features and enable them. If you're missing any libraries, you'll get an error to this effect at compile time, rather than some time later.

Some configuration options can be passed either `no`, `yes`, or `mod`. `no` disables the option, `yes` causes the option to be statically linked to the final binary, and `mod` builds the option as a separate shared library. Shared libraries are loaded into the server at runtime (see "[Server parameters \(global\)](#)" below). By default, the modules are statically linked; that is, they are part of the binary and inseparable. If you wish to use dynamic modules, you will also need the `--enable-modules` option. The benefits of dynamic modules are that you can test various options without bloating your binary, and you can package the modules separately.

Listing 2 shows a configuration line, based on the configuration that ships in Fedora 7, that enables many helpful features. For the most part, the chosen options will enable features that will be required in later tutorials, such as `--enable-slurpd` and `--enable-multimaster` for replication, and `--enable-meta` for meta-directories. Other options enable various backends, such as `ldab`, `bdb`, `null`, and `monitor`.

Listing 2. A sample build configuration

```

./configure --enable-plugins --enable-modules --enable-slaped --enable-slurpd \
  --enable-multimaster --enable-bdb --enable-hdb --enable-ldap --enable-ldbm \
  --enable-ldbm-api=berkeley --enable-meta --enable-monitor --enable-null \
  --enable-shell --enable-sql=mod --disable-perl \
  --with-kerberos=k5only --enable-overlays=mod --prefix=/tmp/openldap

```

Listing 2 enables plug-ins and multiple backends, including Structured Query Language (SQL) based backends and Berkeley Database files. Backends are OpenLDAP's way of storing and retrieving data, and are examined in more detail under "[Backends and databases](#)," and in later tutorials.

Listing 2 also builds both the stand-alone daemon `slapd` and the replication daemon `slurpd`. Overlays, which allow easier customization of the backend data, are also enabled for testing. Because this is a test setup, the installation prefix has been changed to `/tmp/openldap`, so the resulting binaries end up in `/tmp/openldap/libexec`.

When you execute the `configure` script, it checks for the necessary libraries and then generates the build environment. If `configure` completes successfully, compile OpenLDAP with `make depend; make`.

After the code has compiled, you can install OpenLDAP with `make install`. This copies all the binaries, manpages, and libraries to their place in `/tmp/openldap`.

Installing from packages

If you were daunted by the previous section on compiling from source, you aren't alone. Compiling from source is time consuming and can be aggravating if you don't have the proper development libraries available. If your C development experience is limited or nonexistent, then you'll likely have trouble interpreting any build errors. Fortunately, most distributions package OpenLDAP as a set of binaries with a preset configuration. Usually these binaries have all the features you'll ever need.

RPM-based distributions

Fedora and CentOS use the `yum` tool to install RedHat packages (RPMs) from repositories. To find out which packages are available, use the `yum list` command, passing an optional regular expression that filters the list of packages returned. Listing 3 shows a search for all packages containing the term `openldap`.

Listing 3. Determining which packages are available through yum

```
# yum list \*openldap\*
Loading "installonlyn" plugin
Setting up repositories
Reading repository metadata in from local files
Installed Packages
openldap.i386                2.3.30-2.fc6           installed
openldap-clients.i386       2.3.30-2.fc6           installed
openldap-devel.i386         2.3.30-2.fc6           installed
openldap-servers.i386       2.3.30-2.fc6           installed
openldap-servers-sql.i386   2.3.30-2.fc6           installed
Available Packages
compat-openldap.i386         2.3.30_2.229-2.fc6     updates
```

In a large application such as OpenLDAP, the client and server tools are often split into two separate packages. In addition, you may find some compatibility libraries (to ensure applications linked against much older versions of the software still work). To install a package, use `yum install` with the name of the package, such as `yum install openldap-clients openldap-servers`; this downloads and installs

both the client and server packages, along with any needed dependencies.

For Red Hat Enterprise Linux, the command to search packages for `openldap` is `up2date --showall | grep openldap`. To install a package, supply the package names as arguments to `up2date`, such as `up2date openldap-clients openldap-servers`.

To make sure the OpenLDAP server starts on boot, use `chkconfig ldap on`.

Debian-based distributions

Debian-based distributions, such as Ubuntu, use the Advanced Packaging (APT) tools to install packages. First, to search for OpenLDAP packages, use `apt-cache search openldap`, as shown in Listing 4.

Listing 4. Listing the available OpenLDAP packages in Ubuntu Linux

```
notroot@ubuntu:~$ apt-cache search openldap
libldap2 - OpenLDAP libraries
libldap2-dev - OpenLDAP development libraries
python-ldap - A LDAP interface module for Python. [dummy package]
python-ldap-doc - Documentation for the Python LDAP interface module
python2.4-ldap - A LDAP interface module for Python 2.4
ldap-utils - OpenLDAP utilities
libldap-2.2-7 - OpenLDAP libraries
slapd - OpenLDAP server (slapd)
```

Listing 4 shows several packages available. The `slapd` package provides the server, and any dependencies will be resolved at install time. Run `sudo apt-get install slapd` to install the server. You may also include the `ldap-utils` package, which contains the command-line clients.

Configuring the software

Once you've installed OpenLDAP, you must configure it. For testing purposes, you need to specify only a few things; but for the real world (and the LPIC 3 exam), you must be well acquainted with the various options.

Two configuration files govern the behavior of OpenLDAP; both are in `/etc/openldap/` by default. The first is `ldap.conf`, which controls the global behavior of LDAP clients. The configuration file for all LDAP servers is called `slapd.conf`. Despite the name, `slapd.conf` also has the configuration for `slurpd`, the replication daemon. The focus of this article is on `slapd.conf`, specifically pertaining to the `slapd` daemon.

`slapd.conf` has an easy format: a single keyword followed by one or more arguments, subject to the following conditions:

- The keyword must start at column 0—that is, no spaces may exist in front of it.
- If an argument has spaces in it, the argument must be quoted with double quotes ("").
- If a line begins with a space, it's considered a continuation of the previous line.
- Keywords aren't case sensitive, but the arguments may be, depending on which keyword is used.

As with most UNIX® tools, the hash symbol (#) denotes a comment. Anything after the hash is ignored.

slapd.conf is divided into two sections: global options and backend database options. Although this ordering isn't enforced, you must be careful where you place your directives, because some directives alter the context in which subsequent directives are processed. For instance, if no `backend` or `database` keywords have been encountered, an option is considered global. Once a `database` directive is read, all further options apply to that database. This continues until another `database` directive is read, at which point the next commands apply to the new database.

Some of the global options will be covered in later tutorials in this 301 series, such as those dealing with access controls and replication. A description of the commonly used configuration directives follows.

Server parameters (global)

Several parameters limit the work that the `slapd` process can do, which prevents resource starvation. `conn_max_pending` accepts an integer that dictates how many anonymous requests can be pending at any given time. You'll learn about binding to the LDAP server in a later tutorial in this 301 series; simply put, you can make requests from the server by logging in as a user (an authenticated session) or without any credentials (anonymous session). Requests beyond the `conn_max_pending` limit are dropped by the server. Similarly, `conn_max_pending_auth` is the same as `conn_max_pending` but refers to authenticated sessions.

The `idletimeout` parameter (specified in seconds) tells `slapd` how long idle clients can be held before they should be disconnected. If this number is 0, no disconnections happen.

The `sizelimit` parameter limits the number of search results that can come back from a single query, and `timelimit` limits how long the server spends searching. These two parameters can take either an integer, the keyword `unlimited`, or more

complex hard and soft limits. This would allow you to set a default (soft) timeout or result-set size; but if a client requests a larger number of rows or a longer timeout, it can be accommodated up to the hard limit. For example, `sizelimit sizesoft=400 size.hard=1000` specifies that by default, 400 rows are returned. Clients can request that this limit be increased up to 1,000. This format can be applied to groups of users so that some people or applications can perform large searches, and others can perform only small searches

When a client performs a search on the tree, it usually specifies a node (called the *search base*, or *base*) from which the search should start—the Distinguished Names (DNs) of all the results have the search base in them. This allows for faster searching (because fewer nodes need to be searched) and easier client implementation (because searching only part of a tree is a simple but effective filter). If the client doesn't specify a base, the value of `defaultsearchbase` is used. This is a good parameter to set to avoid surprises with misconfigured clients down the road. Depending on the layout of your LDAP tree, you may wish to use either your users container or the root of the tree. (Trees and distinguished names are covered in the [previous tutorial](#).)

Three commands govern various features supported by your server, such as legacy support and security requirements by clients. These commands are `allow`, `disallow`, and `require`. Each command takes a series of whitespace keywords that enable, disable, or require a feature. The keywords are shown in Table 3.

Table 3. Keywords used with `allow`, `disallow`, and `require`

Command(s)	Keyword	Description	Default
<code>allow</code>	<code>bind_v2</code>	If set, allows legacy LDAPv2 clients to connect. The OpenLDAP documentation repeatedly points out that OpenLDAP doesn't truly support LDAPv2, so some requests may result in unexpected behavior.	Disallowed
<code>allow</code>	<code>bind_anon_cred</code>	Allows a client to bind with a password but no DN. If this option is allowed, then the client is allowed as an anonymous bind.	Disallowed
<code>allow</code>	<code>bind_anon_dn</code>	Allows a client to bind with a DN but no password, usually	Disallowed

		because the client is misconfigured. If this option is allowed, then the client is allowed as an anonymous bind.	
allow, disallow	update_anon	Allows an anonymous bind, which happens when a client connects to the LDAP server with no DN or password.	Allowed
disallow	bind_simple	Allows simple (unencrypted user names and passwords) authentication as opposed to a stronger method such as Simple Authentication and Security Layer (SASL).	Allowed
require	bind	Requires that the client bind to the directory with the bind operation before doing any other operations.	Not required
require	LDAPv3	Determines whether LDAPv3 is required. Note that this can conflict with <code>allow bind_v2</code> .	Not required
require	authc	Requires authentication, as opposed to an anonymous bind.	Not required
require	SASL	Requires that a SASL method be used to connect to the server	Not required
require	strong	Requires that a strong method of authentication be used. This can be either SASL or simple authentication over a protected method.	Not required

<code>require</code>	<code>none</code>	This option clears out all the requirements, usually if you're relaxing the requirements for a certain database by using this command in the database section of <code>slapd.conf</code> . If you wish to change the requirements for the database (as opposed to just clearing the list), you must use <code>none</code> before adding your new requirements, even if they have been required in the global section.	Not applicable
----------------------	-------------------	---	----------------

Even though certain types of login may be allowed by some commands from Table 3, the connections are still subject to access controls. For example, an anonymous bind may be granted read-only access to part of the tree. The nature of your application and the capabilities of your clients dictate how you allow or disallow various authentication methods.

If you wish to maintain a higher level of availability, then enable `gentlehup`. With this command enabled, `slapd` stops listening on the network when it receives a `SIGHUP` signal, but it doesn't drop any open connections. A new instance of `slapd` can then be started, usually with an updated configuration.

To get more verbose logging, adjust the value of `loglevel`. This command accepts an integer, multiple integers, or a series of keywords, which enable logging for a particular function. Consult the `slapd.conf` manpage for the full list of keywords and values. For example, connection tracing has a value of 8 and a keyword of `conns`, and synchronization has a value of 4096 and a keyword of `sync`. To enable logging of these two items `logging 5004, logging 8 4096, or logging conns sync` will achieve the same result.

If you compiled OpenLDAP from source, you may have enabled some modules. Alternatively, you may have downloaded extra modules from your package manager, such as the `openldap-server-sql` package, which includes the SQL backend module. The `modulepath` and `moduleload` options are used to load dynamic modules into `slapd`. `modulepath` specifies the directory (or list of directories) that contains the shared libraries, and each instance of `moduleload` specifies a module to load. It's not necessary to specify the module's version number or extension, because `slapd` looks for a shared library. For example, for a library

called `back_sql-2.3.so.0.2.18`, use `moduleload back_sql`. Alternatively, `moduleload` can be given the full path (without the version and extension) to the library, such as `moduleload /usr/share/openldap/back_sql`.

Some scripts expect the process id of a process to be held in a certain file. `pidfile` tells `slapd` where to write its process id.

Schema parameters

A handful of commands let you add schema items to your tree, either by including a schema file or by defining the object in `slapd.conf`. Recall from the [previous tutorial](#) that the schema provides the attributes and object classes that can be used by your LDAP tree.

To add a new schema file to your server, use the `include` command followed by the full path to the schema file (usually found in `/etc/openldap/schema`). If one schema makes reference to another (such as `inetOrgPerson` inheriting from `organizationalPerson`), you need to include all the necessary files in the proper order, with the base objects included first. OpenLDAP parses each schema file as it's included, so order of inclusion is important.

You can add new schema items directly through `slapd.conf` with the `attributetype` and `objectclass` commands for attributes and object classes, respectively. This is the same as putting the information in a schema file and including it with the `include` command. Similarly, you can define object identifiers (OIDs) with `objectidentifier`.

Backends and databases

Backends and databases are two separate but closely related concepts. A database represents part of a tree, such as `dc=ertw,dc=com`. A backend describes the method by which `slapd` retrieves the data. (The `dc=ertw,dc=com` tree has been the primary example in this series.)

In many cases, the backend is a file on disk (in some format; more on this later); or it can be a method to get data from another source, from a SQL database, to DNS, and even through a script. Each database is handled by one backend, and the same backend type can be used by multiple databases.

As noted earlier, `slapd.conf` starts with global directives. Backend mode then starts at the first instance of the `backend` directive. All directives in this backed mode apply to the particular backend being configured. Any options that were set globally apply to the backend, unless they're overridden at the backend level. Similarly, you configure databases with the `database` keyword. A database is tied to a backend type, which inherits any global or backend level configurations. You can override any options at the database level, too.

OpenLDAP splits the backends into three types:

1. Those that store data:
 - `bdb`—Uses the Berkeley database engine (such as Sleepycat, now owned by Oracle)
 - `hdb`—An improvement on `back-ldb`, which adds some indexing improvements
2. Those that proxy data:
 - `ldap`—Proxies another LDAP server
 - `meta`—Proxies several LDAP servers for different parts of the tree
 - `sql`—Returns data from a SQL database
3. Those that generate data:
 - `dnssrv`—Returns LDAP referrals based on data in DNS SRV records
 - `monitor`—Returns statistics from the LDAP server
 - `null`—A testing module; returns nothing
 - `passwd`—Returns data from the password file
 - `perl`—Returns data generated from a Perl script
 - `shell`—Returns data generated from a shell script

Configuration options are specific to each backend, and can be found in the relevant manpage (such as `slapd-bdb` for the `bdb` backend).

Databases represent the tree and its data. The `dc=ertw,dc=com` tree is an example of a database. All data under this DN would be stored in a similar fashion if it were part of the same database. It's also possible to have `ou=people,dc=ertw,dc=com` in one database, with anything else under `dc=ertw,dc=com` in another. Finally, an LDAP server can serve more than one tree, such as `dc=ertw,dc=com` and `dc=lpi,dc=org`. Each database has its own way of handling the request by way of its own backend.

Specify `database` followed by the database type to start database configuration mode. The commonly used form is the Berkeley database, so `database bdb` creates a BDB database. The next command you need is `suffix`, which specifies the root of the tree the database is serving.

`rootdn` and `rootpw` allow you to specify a user with all privileges (a *root user*) for the database. This user isn't even subject to access controls. The `rootdn` should be

within the specified suffix and may or may not have a password. If a `rootpw` is specified, this is used. Otherwise, the behavior is to look for the `rootdn`'s record in the tree and authenticate against the `userPassword` attribute. If no root user is specified, then all users are subject to the access controls configured.

If you specify `lastmod on`, OpenLDAP keeps several hidden attributes (called *operational attributes*), such as the name of the person who created the record and when it was modified. Some of these attributes are required for replication to work, so it's smart to leave `lastmod` enabled (which is the default). These operational attributes aren't shown to clients unless specifically requested.

You can further restrict what can be done to the database through the `restrict` command. This command takes parameters corresponding to LDAP operations, such as `add`, `bind`, `compare`, `delete`, `rename`, and `search`. To block users from deleting nodes in the tree, use `restrict delete`. If the tree contains users, but for some reason you don't want them to be able to bind to the tree, use `restrict bind`. Additionally, `read` and `write` are available to block any reading and writing to the tree, respectively, rather than having to spell out all the relevant operations. Alternatively, you can use the command `readonly` to make the database read only.

Different parts of the same tree can be handled by different databases. If properly configured, OpenLDAP glues all the parts together. The database containing the other is called the *superior database*; the database being contained is the *subordinate database*. First, define the subordinate database and add the `subordinate` command on a line of its own. Then, define the superior database. With this configuration, OpenLDAP can treat multiple databases as one, with some data stored locally and some pulled from other sources (a special case of this is when all the data is on remote LDAP servers, which is where a metadirectory is used). Note that if you define the superior database before the subordinate database, you'll get errors that you're trying to redefine part of your tree. Listing 5 shows the `dc=ertw,dc=com` tree split into a superior and a subordinate database.

Listing 5. Configuration for a subordinate and superior database

```
# Subordinate
database bdb
suffix "ou=people,dc=ertw, dc=com"
rootdn "cn=Sean Walberg,ou=people,dc=ertw,dc=com"
rootpw mysecret
directory /var/db/openldap/ertw-com-people
subordinate

# Superior
database bdb
suffix "dc=ertw, dc=com"
rootdn "cn=Sean Walberg,dc=ertw,dc=com"
rootpw mysecret
directory /var/db/openldap/ertw-com
```

Also note that two `rootdns` are configured. If you want to define a password, the `rootdn` must fall within the database. To build the tree, the second root account must be used to define the `dc=ertw,dc=com` entry, and the first root account defines the people organizational unit (OU) and any objects underneath it. Once users have been added, you can authenticate as a different user in order to get access to the whole tree.

If you're using the `bdb` backend, you also need to use the `directory` command to specify where the database files are stored. Each database instance needs a separate directory.

Setting up a new database is fairly simple, because there are only a few commands to worry about. Much of the complexity comes in when you try to tune the backend, which is the subject of the next tutorial in this 301 series.

Overlays

Overlays are an extension of the database. If you want to add a feature to a database, you can often add it as an overlay rather than forking the database code. For example, if you want all writes to be logged to a file, you can attach the `auditlog` overlay to the relevant database.

Overlays operate as a stack. After configuring the database, you specify one or more databases. Then, define each overlay with the `overlay` command, followed by the name of the overlay. Each overlay has its own configuration parameters.

If you've configured multiple overlays, they're run in the reverse order that you define them. The database is accessed only after all the overlays have run. After the database returns the data, the overlays are run again in the same order before `slapd` returns the data to the client.

At each step, an overlay can perform an action such as logging, it can modify the request or response, or it can stop processing.

Section 3. Developing for LDAP with Perl/C++

This section covers material for topic 302.2 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to:

- Use Perl's `Net::LDAP` module

- Write Perl scripts to bind, search, and modify directories
- Develop in C/C++

Although OpenLDAP includes command-line clients, it's often helpful to use LDAP information in your own scripts. Perl is a popular language for scripting. Perl has a module called `Net::LDAP` that is used to connect to and use an LDAP server.

Getting started

`Net::LDAP` doesn't ship with Perl, but your distribution may include it as a package. See ["Installing from packages"](#) for more information on searching for and installing packages.

If your distribution doesn't have the `Net::LDAP` package, then you can download it from the Comprehensive Perl Archive Network (CPAN). As root, run `perl -MCPAN -e "install Net::LDAP"`, which downloads and installs `Net::LDAP` and any dependencies.

Using Net::LDAP

Using `Net::LDAP` is fairly simple:

1. Create a new `Net::LDAP` object.
2. Bind to the desired server.
3. Perform your LDAP operations.

Create a new object

In typical Perl fashion, you must create an instance of the `Net::LDAP` module through the `new` method. All further operations will be on this instance. `new` requires, at a minimum, the name of the server you want to connect to. For example:

```
my $ldap = Net::LDAP->new('localhost') or die "$@";
```

Here, a new `Net::LDAP` object is created with the `new` method and is passed the string `localhost`. The result is assigned to the `$ldap` variable. If the function fails, the program exits and prints an error message describing the problem. `$@` is a Perl internal variable that contains the status of the last operation.

You can proceed to perform LDAP operations with the new `Net::LDAP` object. Each function returns a `Net::LDAP::Message` object that contains the status of

the operation, any error messages, and any data returned from the server.

Binding to the tree

The first operation you should do is to log in or *bind* to the tree. Listing 6 shows a bind operation and associated error checking.

Listing 6. Perl code to bind to the tree

```
my $message = $ldap->bind(  
    "cn=Sean Walberg,ou=people,dc=ertw,dc=com",  
    password=>"test" );  
  
if ($message->code() != 0) {  
    die $message->error();  
}
```

Listing 6 starts by calling the `bind` method of the previously created object. The first parameter to the function is the DN you're binding as. If you don't specify a DN, you bind anonymously. Further parameters are in the format of `key=>value`; the one you'll use most often is the password.

Each `Net::LDAP` method returns a `Net::LDAP::Message` object, which has the results of the function. The error code is retrieved through the `code` method. A code of 0 means success, so the code in Listing 6 exits the program with the error message if the result isn't 0. Note that the error is retrieved from `$message->error` rather than `$@`, like the earlier example. This is because the error isn't a Perl error; it's internal to `Net::LDAP`.

Once the bind is successful, you can do anything you want, subject to the server's access controls. To log out, call the `unbind` method.

Searching the tree

Searching is done through the `search` method. Like the `bind` method, you must pass some parameters and check the result of your query. However, the returned object now contains your data, so this must be parsed. With the `search` operation, the result is a `Net::LDAP::Search` object, which inherits all the methods from `Net::LDAP::Message` (such as `code` and `error`) and adds methods to help you parse the data. Listing 7 shows a search of the tree.

Listing 7. Searching the tree with search

```
$message = $ldap->search(base => "dc=ertw,dc=com", filter=> "(objectClass=*)");  
if ($message->code() != 0) {  
    print $message->error();  
} else {  
    foreach my $entry ($message->entries()) {  
        print $entry->dn() . " : ";  
    }  
}
```

```

        print join ", ", $entry->get_value("objectClass");
        print "\n";
    }
}

```

Listing 7 begins by calling the `search` method, passing two parameters: the base and a filter. The base tells the server where in the tree to begin searching. A complementary option, `scope`, tells the server how far to search:

- **base** —Only the base object
- **one** —Only the children of the base object (and not the base object itself)
- **sub** —The base object and all its children (the default)

The filter is a string describing the objects you're interested in. You can search on attributes and perform complex AND/OR queries. `objectClass=*` returns any object.

The result of the search is checked, and an error is printed if a problem happened. Because the script could still recover from an error, it just prints the error and continues, rather than exiting.

The `entries` function returns an array of `Net::LDAP::Entry` objects, each with a single result. First the entry's DN is printed, and then all the object classes. If you'd rather have a text version of the whole record, the `dump` method prints the entire entry in text format.

Adding a new entry

You add an entry to the tree through the `add` method. You must pass the function the DN of the entry you wish to add, along with the attributes. The attributes are an array of `key => value` pairs. The value can also be an array in the case of multiple instances of the same attribute. Listing 8 shows an entry being added to the tree.

Listing 8. Adding an entry using Net::LDAP

```

$message = $ldap->add(
    "cn=Fred Flintstone,ou=people,dc=ertw,dc=com",
    attr => [
        cn => "Fred Flintstone",
        sn => "Flintstone",
        objectclass => [ "organizationalPerson",
            "inetOrgPerson" ],
    ]
);

if ($message->code() != 0) {
    print $message->error();
}

```

The first parameter to `add` is either the DN or a `Net::LDAP::Entry` object. If the DN is passed, you must pass an arrayref through the `attr` method. Even though the `key => value` format is used as in a hashref, `Net::LDAP` is expecting an arrayref, so be careful!

More about Net::LDAP

`Net::LDAP` provides an interface to all the LDAP functions, such as `compare`, `delete`, and `moddn`. They're all used similarly to the previous examples and are fully documented in the `Net::LDAP` manpage.

All the examples shown operate in blocking mode, which means the function returns after the response has been received from the server. You can also operate in asynchronous mode, which involves giving a callback function that is called as packets are received.

By using `Net::LDAP`, you can use the data stored in your LDAP tree from within your scripts. Perl is already used in a wide variety of software, so the opportunities for integration are unlimited.

Developing in C/C++

Using the C libraries is more involved than the Perl libraries. The `ldap(3)` manpage contains a detailed description of how to use the library, and has pointers to the other manpages describing each function. To use the LDAP C libraries, your code must first include the `ldap.h` include file, such as with `#include <ldap.h>`. Your object files must then be linked with `libldap` using the `-lldap` option to the linker.

Section 4. Summary

In this tutorial, you learned about installing and configuring the OpenLDAP stand-alone server. When configuring `slapd`, use the `slapd.conf` file. You must take care to keep your global options at the top of the file and then progress to backend and database configurations, because `slapd` is dependent on the order of the directives. When in doubt, consult the `slapd.conf` manpage.

Perl code can make use of an LDAP server through the `Net::LDAP` module. First you create an object, and then you call methods of the object that correspond with the LDAP operation you want. Generally, you first `bind` and then perform your queries. It's important to check the results of your functions through the `code` and `error` functions.

Resources

Learn

- Review the previous tutorial in this 301 series, "[LPI exam 301 prep, Topic 301: Concepts, architecture, and design](#)" (developerWorks, October 2007).
- Take the developerWorks tutorial "[Linux Installation and Package Management](#)" (developerWorks, September 2005) to brush up on your package management commands.
- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Find answers to [What is a backend?](#) and [What is a database?](#) in the OpenLDAP FAQ.
- Learn more about [building overlays](#) in the developer's documentation. Overlays are a complex but powerful concept.
- Consult the [OpenLDAP Administrator's Guide](#) if the manpages for slapd.conf or your particular backend don't help. You might also find the [OpenLDAP FAQ](#) to be helpful.
- The online book [LDAP for Rocket Scientists](#) is excellent, despite being a work in progress.
- The [Perl-LDAP](#) page has lots of documentation and advice on using Net::LDAP.
- In the [developerWorks Linux zone](#), find more resources for Linux developers, and scan our [most popular articles and tutorials](#).
- See all [Linux tips](#) and [Linux tutorials](#) on developerWorks.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- Download [OpenLDAP](#).
- The [IBM Tivoli Directory Server](#) is a competing LDAP server that integrates well with other IBM products.
- [phpLDAPadmin](#) is a Web-based LDAP administration tool. If the GUI is more your style, [Luma](#) is a good one to look at.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software

for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the [developerWorks community](#) through blogs, forums, podcasts, and community topics in our [new developerWorks spaces](#).

About the author

Sean Walberg

Sean Walberg has been working with Linux and UNIX since 1994 in academic, corporate, and Internet service provider environments. He has written extensively about systems administration over the past several years.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 301 prep, Topic 302: Installation and development

Senior Level Linux Professional (LPIC-3)

Skill Level: Intermediate

[Sean Walberg](mailto:sean@ertw.com) (sean@ertw.com)
Network engineer
Freelance

04 Dec 2007

In this tutorial, Sean Walberg helps you prepare to take the Linux Professional Institute® Senior Level Linux Professional (LPIC-3) exam. In this second in a series of six tutorials, Sean walks you through installing and configuring a Lightweight Directory Access Protocol (LDAP) server, and writing some Perl scripts to access the data. By the end of this tutorial, you'll know about LDAP server installation, configuration, and programming.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *advanced level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102. To attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active advanced-level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the five junior, advanced, and senior certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. Table 1 lists the six topics and corresponding developerWorks tutorials for LPI exam 301.

Table 1. LPI exam 301: Tutorials and topics

LPI exam 301 topic	developerWorks tutorial	Tutorial summary
Topic 301	LPI exam 301 prep: Concepts, architecture, and design	Learn about LDAP concepts and architecture, how to design and implement an LDAP directory, and about schemas.
Topic 302	LPI exam 301 prep: Installation and development	(This tutorial) Learn how to install, configure, and use the OpenLDAP software. See the detailed objectives .
Topic 303	LPI exam 301 prep: Configuration	Coming soon.
Topic 304	LPI exam 301 prep: Usage	Coming soon.
Topic 305	LPI exam 301 prep: Integration and migration	Coming soon.
Topic 306	LPI exam 301 prep: Capacity planning	Coming soon.

To pass exam 301 (and attain certification level 3), you should:

- Have several years of experience in installing and maintaining Linux on a number of computers for various purposes
- Have integration experience with diverse technologies and operating systems
- Have professional experience as, or training to be, an enterprise-level Linux professional (including having experience as a part of another role)
- Know advanced and enterprise levels of Linux administration including installation, management, security, troubleshooting, and maintenance.
- Be able to use open source tools to measure capacity planning and troubleshoot resource problems
- Have professional experience using LDAP to integrate with UNIX® services and Microsoft® Windows® services, including Samba, Pluggable Authentication Modules (PAM), e-mail, and Active Directory
- Be able to plan, architect, design, build, and implement a full environment

using Samba and LDAP as well as measure the capacity planning and security of the services

- Be able create scripts in Bash or Perl or have knowledge of at least one system programming language (such as C)

The Linux Professional Institute doesn't endorse any third-party exam preparation material or techniques in particular.

About this tutorial

Welcome to "Installation and development," the second of six tutorials designed to prepare you for LPI exam 301. In this tutorial, you learn about LDAP server installation and configuration, and how to use Perl to access your new LDAP server.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weights.

Objectives

Table 2 shows the detailed objectives for this tutorial.

Table 2. Installation and development: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
302.1 Compiling and installing OpenLDAP	3	Compile and install OpenLDAP from source and from packages
302.2 Developing for LDAP with Perl/C++	1	Write basic Perl scripts to interact with an LDAP directory

Prerequisites

To get the most from this tutorial, you should have advanced knowledge of Linux and a working Linux system on which to practice the commands covered.

If your fundamental Linux skills are a bit rusty, you may want to first review the [tutorials for the LPIC-1 and LPIC-2 exams](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

System requirements

To follow along with the examples in these tutorials, you'll need a Linux workstation with the OpenLDAP package and support for PAM. Most modern distributions meet these requirements.

Section 2. Compiling and installing OpenLDAP

This section covers material for topic 302.1 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 3.

In this section, learn how to:

- Compile and configure OpenLDAP from source
- Understand OpenLDAP backend databases
- Manage OpenLDAP daemons
- Troubleshoot errors during installation

OpenLDAP is an open source application that implements an LDAP server and associated tools. Because it's open source, you can download the source code free of charge. The OpenLDAP project doesn't distribute binaries directly, but most major distributions package it themselves. In this tutorial, you learn how to install OpenLDAP from both source and packages.

Compiling from source

The first order of business is to download the latest version of OpenLDAP from the project site (see the [Resources](#) section for a download link). The project generally has two active versions available: One is a stable version, and the other is a test version. This tutorial was written using the stable versions 2.3.30 and 2.3.38. If you're following along, some of the directory names may be different depending on which version you have.

To extract the source code from the downloaded tarball, enter `tar -xzf openldap-stable-20070831.tgz`. This decompresses and untars the downloaded file into a directory. Change into the new directory with `cd openldap-2.3.38` (substituting your version of OpenLDAP as appropriate).

At this point, you're in the source directory. You must now configure the build environment for your system and then build the software. OpenLDAP uses a script called `configure` that performs these actions. Type `./configure --help` to see all the options available to you. Some define where the files are installed to (such as `--prefix`); others define the OpenLDAP features you wish to build. Listing 1 lists the features and their defaults.

Listing 1. Configuration options relating to OpenLDAP features

```

SLAPD (Standalone LDAP Daemon) Options:
--enable-slapd           enable building slapd [yes]
--enable-aci             enable per-object ACIs (experimental) [no]
--enable-cleartext      enable cleartext passwords [yes]
--enable-crypt          enable crypt(3) passwords [no]
--enable-ldapmanager    enable LAN Manager passwords [no]
--enable-sasldb         enable (Cyrus) SASL password verification [no]
--enable-modules        enable dynamic module support [no]
--enable-rewrite        enable DN rewriting in back-ldap and rwm overlay [auto]
--enable-reverselookup  enable reverse lookups of client hostnames [no]
--enable-slapi          enable SLAPI support (experimental) [no]
--enable-slp            enable SLPv2 support [no]
--enable-wrappers       enable tcp wrapper support [no]

SLAPD Backend Options:
--enable-backends       enable all available backends no|yes|mod
--enable-bdb            enable Berkeley DB backend no|yes|mod [yes]
--enable-dnssrv        enable dnssrv backend no|yes|mod [no]
--enable-hdb           enable Hierarchical DB backend no|yes|mod [yes]
--enable-ldap          enable ldap backend no|yes|mod [no]
--enable-ldbm          enable ldbm backend no|yes|mod [no]
--enable-ldbm-api      use LDBM API auto|berkeley|bcompat|mdbm|gdbm [auto]
--enable-ldbm-type     use LDBM type auto|btree|hash [auto]
--enable-meta          enable metadirectory backend no|yes|mod [no]
--enable-monitor       enable monitor backend no|yes|mod [yes]
--enable-null          enable null backend no|yes|mod [no]
--enable-passwd        enable passwd backend no|yes|mod [no]
--enable-perl          enable perl backend no|yes|mod [no]
--enable-relay         enable relay backend no|yes|mod [yes]
--enable-shell         enable shell backend no|yes|mod [no]
--enable-sql           enable sql backend no|yes|mod [no]

SLAPD Overlay Options:
--enable-overlays      enable all available overlays no|yes|mod
--enable-accesslog     In-Directory Access Logging overlay no|yes|mod [no]
--enable-auditlog      Audit Logging overlay no|yes|mod [no]
--enable-denyop        Deny Operation overlay no|yes|mod [no]
--enable-dyngroup      Dynamic Group overlay no|yes|mod [no]
--enable-dynlist       Dynamic List overlay no|yes|mod [no]
--enable-lastmod       Last Modification overlay no|yes|mod [no]
--enable-ppolicy       Password Policy overlay no|yes|mod [no]
--enable-proxycache    Proxy Cache overlay no|yes|mod [no]
--enable-refint        Referential Integrity overlay no|yes|mod [no]
--enable-retcode       Return Code testing overlay no|yes|mod [no]
--enable-rwm           Rewrite/Remap overlay no|yes|mod [no]
--enable-syncprov       Syncrepl Provider overlay no|yes|mod [yes]
--enable-translucent   Translucent Proxy overlay no|yes|mod [no]
--enable-unique        Attribute Uniqueness overlay no|yes|mod [no]
--enable-valsort       Value Sorting overlay no|yes|mod [no]

SLURPD (Replication Daemon) Options:
--enable-slurpd        enable building slurpd [auto]

Optional Packages:
--with-PACKAGE[=ARG]  use PACKAGE [ARG=yes]

```

```

--without-PACKAGE      do not use PACKAGE (same as --with-PACKAGE=no)
--with-subdir=DIR      change default subdirectory used for installs
--with-cyrus-sasl      with Cyrus SASL support [auto]
--with-fetch           with fetch(3) URL support [auto]
--with-threads         with threads [auto]
--with-tls             with TLS/SSL support [auto]
--with-yielding-select with implicitly yielding select [auto]
--with-odbc            with specific ODBC support iodbc|unixodbc|auto [auto]

--with-gnu-ld          assume the C compiler uses GNU ld [default=no]
--with-pic             try to use only PIC/non-PIC objects [default=use
both]

--with-tags[=TAGS]    include additional configurations [automatic]

```

In Listing 1, you can see that many features are disabled by default, such as metadirectories and modules. In addition, many options are marked as "auto," which turns on features if the proper libraries are present on your system. Instead of relying on this automatic behavior, it's best to make a list of the required features and enable them. If you're missing any libraries, you'll get an error to this effect at compile time, rather than some time later.

Some configuration options can be passed either `no`, `yes`, or `mod`. `no` disables the option, `yes` causes the option to be statically linked to the final binary, and `mod` builds the option as a separate shared library. Shared libraries are loaded into the server at runtime (see "[Server parameters \(global\)](#)" below). By default, the modules are statically linked; that is, they are part of the binary and inseparable. If you wish to use dynamic modules, you will also need the `--enable-modules` option. The benefits of dynamic modules are that you can test various options without bloating your binary, and you can package the modules separately.

Listing 2 shows a configuration line, based on the configuration that ships in Fedora 7, that enables many helpful features. For the most part, the chosen options will enable features that will be required in later tutorials, such as `--enable-slurpd` and `--enable-multimaster` for replication, and `--enable-meta` for meta-directories. Other options enable various backends, such as `ldab`, `bdb`, `null`, and `monitor`.

Listing 2. A sample build configuration

```

./configure --enable-plugins --enable-modules --enable-slaped --enable-slurpd \
  --enable-multimaster --enable-bdb --enable-hdb --enable-ldap --enable-ldbm \
  --enable-ldbm-api=berkeley --enable-meta --enable-monitor --enable-null \
  --enable-shell --enable-sql=mod --disable-perl \
  --with-kerberos=k5only --enable-overlays=mod --prefix=/tmp/openldap

```

Listing 2 enables plug-ins and multiple backends, including Structured Query Language (SQL) based backends and Berkeley Database files. Backends are OpenLDAP's way of storing and retrieving data, and are examined in more detail under "[Backends and databases](#)," and in later tutorials.

Listing 2 also builds both the stand-alone daemon `slapd` and the replication daemon `slurpd`. Overlays, which allow easier customization of the backend data, are also enabled for testing. Because this is a test setup, the installation prefix has been changed to `/tmp/openldap`, so the resulting binaries end up in `/tmp/openldap/libexec`.

When you execute the `configure` script, it checks for the necessary libraries and then generates the build environment. If `configure` completes successfully, compile OpenLDAP with `make depend; make`.

After the code has compiled, you can install OpenLDAP with `make install`. This copies all the binaries, manpages, and libraries to their place in `/tmp/openldap`.

Installing from packages

If you were daunted by the previous section on compiling from source, you aren't alone. Compiling from source is time consuming and can be aggravating if you don't have the proper development libraries available. If your C development experience is limited or nonexistent, then you'll likely have trouble interpreting any build errors. Fortunately, most distributions package OpenLDAP as a set of binaries with a preset configuration. Usually these binaries have all the features you'll ever need.

RPM-based distributions

Fedora and CentOS use the `yum` tool to install RedHat packages (RPMs) from repositories. To find out which packages are available, use the `yum list` command, passing an optional regular expression that filters the list of packages returned. Listing 3 shows a search for all packages containing the term `openldap`.

Listing 3. Determining which packages are available through yum

```
# yum list \*openldap\*
Loading "installonlyn" plugin
Setting up repositories
Reading repository metadata in from local files
Installed Packages
openldap.i386                2.3.30-2.fc6           installed
openldap-clients.i386       2.3.30-2.fc6           installed
openldap-devel.i386         2.3.30-2.fc6           installed
openldap-servers.i386       2.3.30-2.fc6           installed
openldap-servers-sql.i386   2.3.30-2.fc6           installed
Available Packages
compat-openldap.i386         2.3.30_2.229-2.fc6     updates
```

In a large application such as OpenLDAP, the client and server tools are often split into two separate packages. In addition, you may find some compatibility libraries (to ensure applications linked against much older versions of the software still work). To install a package, use `yum install` with the name of the package, such as `yum install openldap-clients openldap-servers`; this downloads and installs

both the client and server packages, along with any needed dependencies.

For Red Hat Enterprise Linux, the command to search packages for `openldap` is `up2date --showall | grep openldap`. To install a package, supply the package names as arguments to `up2date`, such as `up2date openldap-clients openldap-servers`.

To make sure the OpenLDAP server starts on boot, use `chkconfig ldap on`.

Debian-based distributions

Debian-based distributions, such as Ubuntu, use the Advanced Packaging (APT) tools to install packages. First, to search for OpenLDAP packages, use `apt-cache search openldap`, as shown in Listing 4.

Listing 4. Listing the available OpenLDAP packages in Ubuntu Linux

```
notroot@ubuntu:~$ apt-cache search openldap
libldap2 - OpenLDAP libraries
libldap2-dev - OpenLDAP development libraries
python-ldap - A LDAP interface module for Python. [dummy package]
python-ldap-doc - Documentation for the Python LDAP interface module
python2.4-ldap - A LDAP interface module for Python 2.4
ldap-utils - OpenLDAP utilities
libldap-2.2-7 - OpenLDAP libraries
slapd - OpenLDAP server (slapd)
```

Listing 4 shows several packages available. The `slapd` package provides the server, and any dependencies will be resolved at install time. Run `sudo apt-get install slapd` to install the server. You may also include the `ldap-utils` package, which contains the command-line clients.

Configuring the software

Once you've installed OpenLDAP, you must configure it. For testing purposes, you need to specify only a few things; but for the real world (and the LPIC 3 exam), you must be well acquainted with the various options.

Two configuration files govern the behavior of OpenLDAP; both are in `/etc/openldap/` by default. The first is `ldap.conf`, which controls the global behavior of LDAP clients. The configuration file for all LDAP servers is called `slapd.conf`. Despite the name, `slapd.conf` also has the configuration for `slurpd`, the replication daemon. The focus of this article is on `slapd.conf`, specifically pertaining to the `slapd` daemon.

`slapd.conf` has an easy format: a single keyword followed by one or more arguments, subject to the following conditions:

- The keyword must start at column 0—that is, no spaces may exist in front of it.
- If an argument has spaces in it, the argument must be quoted with double quotes ("").
- If a line begins with a space, it's considered a continuation of the previous line.
- Keywords aren't case sensitive, but the arguments may be, depending on which keyword is used.

As with most UNIX® tools, the hash symbol (#) denotes a comment. Anything after the hash is ignored.

slapd.conf is divided into two sections: global options and backend database options. Although this ordering isn't enforced, you must be careful where you place your directives, because some directives alter the context in which subsequent directives are processed. For instance, if no `backend` or `database` keywords have been encountered, an option is considered global. Once a `database` directive is read, all further options apply to that database. This continues until another `database` directive is read, at which point the next commands apply to the new database.

Some of the global options will be covered in later tutorials in this 301 series, such as those dealing with access controls and replication. A description of the commonly used configuration directives follows.

Server parameters (global)

Several parameters limit the work that the `slapd` process can do, which prevents resource starvation. `conn_max_pending` accepts an integer that dictates how many anonymous requests can be pending at any given time. You'll learn about binding to the LDAP server in a later tutorial in this 301 series; simply put, you can make requests from the server by logging in as a user (an authenticated session) or without any credentials (anonymous session). Requests beyond the `conn_max_pending` limit are dropped by the server. Similarly, `conn_max_pending_auth` is the same as `conn_max_pending` but refers to authenticated sessions.

The `idletimeout` parameter (specified in seconds) tells `slapd` how long idle clients can be held before they should be disconnected. If this number is 0, no disconnections happen.

The `sizelimit` parameter limits the number of search results that can come back from a single query, and `timelimit` limits how long the server spends searching. These two parameters can take either an integer, the keyword `unlimited`, or more

complex hard and soft limits. This would allow you to set a default (soft) timeout or result-set size; but if a client requests a larger number of rows or a longer timeout, it can be accommodated up to the hard limit. For example, `sizelimit sizesoft=400 size.hard=1000` specifies that by default, 400 rows are returned. Clients can request that this limit be increased up to 1,000. This format can be applied to groups of users so that some people or applications can perform large searches, and others can perform only small searches

When a client performs a search on the tree, it usually specifies a node (called the *search base*, or *base*) from which the search should start—the Distinguished Names (DNs) of all the results have the search base in them. This allows for faster searching (because fewer nodes need to be searched) and easier client implementation (because searching only part of a tree is a simple but effective filter). If the client doesn't specify a base, the value of `defaultsearchbase` is used. This is a good parameter to set to avoid surprises with misconfigured clients down the road. Depending on the layout of your LDAP tree, you may wish to use either your users container or the root of the tree. (Trees and distinguished names are covered in the [previous tutorial](#).)

Three commands govern various features supported by your server, such as legacy support and security requirements by clients. These commands are `allow`, `disallow`, and `require`. Each command takes a series of whitespace keywords that enable, disable, or require a feature. The keywords are shown in Table 3.

Table 3. Keywords used with `allow`, `disallow`, and `require`

Command(s)	Keyword	Description	Default
<code>allow</code>	<code>bind_v2</code>	If set, allows legacy LDAPv2 clients to connect. The OpenLDAP documentation repeatedly points out that OpenLDAP doesn't truly support LDAPv2, so some requests may result in unexpected behavior.	Disallowed
<code>allow</code>	<code>bind_anon_cred</code>	Allows a client to bind with a password but no DN. If this option is allowed, then the client is allowed as an anonymous bind.	Disallowed
<code>allow</code>	<code>bind_anon_dn</code>	Allows a client to bind with a DN but no password, usually	Disallowed

		because the client is misconfigured. If this option is allowed, then the client is allowed as an anonymous bind.	
allow, disallow	update_anon	Allows an anonymous bind, which happens when a client connects to the LDAP server with no DN or password.	Allowed
disallow	bind_simple	Allows simple (unencrypted user names and passwords) authentication as opposed to a stronger method such as Simple Authentication and Security Layer (SASL).	Allowed
require	bind	Requires that the client bind to the directory with the bind operation before doing any other operations.	Not required
require	LDAPv3	Determines whether LDAPv3 is required. Note that this can conflict with <code>allow bind_v2</code> .	Not required
require	authc	Requires authentication, as opposed to an anonymous bind.	Not required
require	SASL	Requires that a SASL method be used to connect to the server	Not required
require	strong	Requires that a strong method of authentication be used. This can be either SASL or simple authentication over a protected method.	Not required

<code>require</code>	<code>none</code>	This option clears out all the requirements, usually if you're relaxing the requirements for a certain database by using this command in the database section of <code>slapd.conf</code> . If you wish to change the requirements for the database (as opposed to just clearing the list), you must use <code>none</code> before adding your new requirements, even if they have been required in the global section.	Not applicable
----------------------	-------------------	---	----------------

Even though certain types of login may be allowed by some commands from Table 3, the connections are still subject to access controls. For example, an anonymous bind may be granted read-only access to part of the tree. The nature of your application and the capabilities of your clients dictate how you allow or disallow various authentication methods.

If you wish to maintain a higher level of availability, then enable `gentlehup`. With this command enabled, `slapd` stops listening on the network when it receives a `SIGHUP` signal, but it doesn't drop any open connections. A new instance of `slapd` can then be started, usually with an updated configuration.

To get more verbose logging, adjust the value of `loglevel`. This command accepts an integer, multiple integers, or a series of keywords, which enable logging for a particular function. Consult the `slapd.conf` manpage for the full list of keywords and values. For example, connection tracing has a value of 8 and a keyword of `conns`, and synchronization has a value of 4096 and a keyword of `sync`. To enable logging of these two items `logging 5004, logging 8 4096, or logging conns sync` will achieve the same result.

If you compiled OpenLDAP from source, you may have enabled some modules. Alternatively, you may have downloaded extra modules from your package manager, such as the `openldap-server-sql` package, which includes the SQL backend module. The `modulepath` and `moduleload` options are used to load dynamic modules into `slapd`. `modulepath` specifies the directory (or list of directories) that contains the shared libraries, and each instance of `moduleload` specifies a module to load. It's not necessary to specify the module's version number or extension, because `slapd` looks for a shared library. For example, for a library

called `back_sql-2.3.so.0.2.18`, use `moduleload back_sql`. Alternatively, `moduleload` can be given the full path (without the version and extension) to the library, such as `moduleload /usr/share/openldap/back_sql`.

Some scripts expect the process id of a process to be held in a certain file. `pidfile` tells `slapd` where to write its process id.

Schema parameters

A handful of commands let you add schema items to your tree, either by including a schema file or by defining the object in `slapd.conf`. Recall from the [previous tutorial](#) that the schema provides the attributes and object classes that can be used by your LDAP tree.

To add a new schema file to your server, use the `include` command followed by the full path to the schema file (usually found in `/etc/openldap/schema`). If one schema makes reference to another (such as `inetOrgPerson` inheriting from `organizationalPerson`), you need to include all the necessary files in the proper order, with the base objects included first. OpenLDAP parses each schema file as it's included, so order of inclusion is important.

You can add new schema items directly through `slapd.conf` with the `attributetype` and `objectclass` commands for attributes and object classes, respectively. This is the same as putting the information in a schema file and including it with the `include` command. Similarly, you can define object identifiers (OIDs) with `objectidentifier`.

Backends and databases

Backends and databases are two separate but closely related concepts. A database represents part of a tree, such as `dc=ertw,dc=com`. A backend describes the method by which `slapd` retrieves the data. (The `dc=ertw,dc=com` tree has been the primary example in this series.)

In many cases, the backend is a file on disk (in some format; more on this later); or it can be a method to get data from another source, from a SQL database, to DNS, and even through a script. Each database is handled by one backend, and the same backend type can be used by multiple databases.

As noted earlier, `slapd.conf` starts with global directives. Backend mode then starts at the first instance of the `backend` directive. All directives in this backed mode apply to the particular backend being configured. Any options that were set globally apply to the backend, unless they're overridden at the backend level. Similarly, you configure databases with the `database` keyword. A database is tied to a backend type, which inherits any global or backend level configurations. You can override any options at the database level, too.

OpenLDAP splits the backends into three types:

1. Those that store data:
 - `bdb`—Uses the Berkeley database engine (such as Sleepycat, now owned by Oracle)
 - `hdb`—An improvement on `back-ldb`, which adds some indexing improvements
2. Those that proxy data:
 - `ldap`—Proxies another LDAP server
 - `meta`—Proxies several LDAP servers for different parts of the tree
 - `sql`—Returns data from a SQL database
3. Those that generate data:
 - `dnssrv`—Returns LDAP referrals based on data in DNS SRV records
 - `monitor`—Returns statistics from the LDAP server
 - `null`—A testing module; returns nothing
 - `passwd`—Returns data from the password file
 - `perl`—Returns data generated from a Perl script
 - `shell`—Returns data generated from a shell script

Configuration options are specific to each backend, and can be found in the relevant manpage (such as `slapd-bdb` for the `bdb` backend).

Databases represent the tree and its data. The `dc=ertw,dc=com` tree is an example of a database. All data under this DN would be stored in a similar fashion if it were part of the same database. It's also possible to have `ou=people,dc=ertw,dc=com` in one database, with anything else under `dc=ertw,dc=com` in another. Finally, an LDAP server can serve more than one tree, such as `dc=ertw,dc=com` and `dc=lpi,dc=org`. Each database has its own way of handling the request by way of its own backend.

Specify `database` followed by the database type to start database configuration mode. The commonly used form is the Berkeley database, so `database bdb` creates a BDB database. The next command you need is `suffix`, which specifies the root of the tree the database is serving.

`rootdn` and `rootpw` allow you to specify a user with all privileges (a *root user*) for the database. This user isn't even subject to access controls. The `rootdn` should be

within the specified suffix and may or may not have a password. If a `rootpw` is specified, this is used. Otherwise, the behavior is to look for the `rootdn`'s record in the tree and authenticate against the `userPassword` attribute. If no root user is specified, then all users are subject to the access controls configured.

If you specify `lastmod on`, OpenLDAP keeps several hidden attributes (called *operational attributes*), such as the name of the person who created the record and when it was modified. Some of these attributes are required for replication to work, so it's smart to leave `lastmod` enabled (which is the default). These operational attributes aren't shown to clients unless specifically requested.

You can further restrict what can be done to the database through the `restrict` command. This command takes parameters corresponding to LDAP operations, such as `add`, `bind`, `compare`, `delete`, `rename`, and `search`. To block users from deleting nodes in the tree, use `restrict delete`. If the tree contains users, but for some reason you don't want them to be able to bind to the tree, use `restrict bind`. Additionally, `read` and `write` are available to block any reading and writing to the tree, respectively, rather than having to spell out all the relevant operations. Alternatively, you can use the command `readonly` to make the database read only.

Different parts of the same tree can be handled by different databases. If properly configured, OpenLDAP glues all the parts together. The database containing the other is called the *superior database*; the database being contained is the *subordinate database*. First, define the subordinate database and add the `subordinate` command on a line of its own. Then, define the superior database. With this configuration, OpenLDAP can treat multiple databases as one, with some data stored locally and some pulled from other sources (a special case of this is when all the data is on remote LDAP servers, which is where a metadirectory is used). Note that if you define the superior database before the subordinate database, you'll get errors that you're trying to redefine part of your tree. Listing 5 shows the `dc=ertw,dc=com` tree split into a superior and a subordinate database.

Listing 5. Configuration for a subordinate and superior database

```
# Subordinate
database bdb
suffix "ou=people,dc=ertw, dc=com"
rootdn "cn=Sean Walberg,ou=people,dc=ertw,dc=com"
rootpw mysecret
directory /var/db/openldap/ertw-com-people
subordinate

# Superior
database bdb
suffix "dc=ertw, dc=com"
rootdn "cn=Sean Walberg,dc=ertw,dc=com"
rootpw mysecret
directory /var/db/openldap/ertw-com
```

Also note that two `rootdns` are configured. If you want to define a password, the `rootdn` must fall within the database. To build the tree, the second root account must be used to define the `dc=ertw,dc=com` entry, and the first root account defines the people organizational unit (OU) and any objects underneath it. Once users have been added, you can authenticate as a different user in order to get access to the whole tree.

If you're using the `bdb` backend, you also need to use the `directory` command to specify where the database files are stored. Each database instance needs a separate directory.

Setting up a new database is fairly simple, because there are only a few commands to worry about. Much of the complexity comes in when you try to tune the backend, which is the subject of the next tutorial in this 301 series.

Overlays

Overlays are an extension of the database. If you want to add a feature to a database, you can often add it as an overlay rather than forking the database code. For example, if you want all writes to be logged to a file, you can attach the `auditlog` overlay to the relevant database.

Overlays operate as a stack. After configuring the database, you specify one or more databases. Then, define each overlay with the `overlay` command, followed by the name of the overlay. Each overlay has its own configuration parameters.

If you've configured multiple overlays, they're run in the reverse order that you define them. The database is accessed only after all the overlays have run. After the database returns the data, the overlays are run again in the same order before `slapd` returns the data to the client.

At each step, an overlay can perform an action such as logging, it can modify the request or response, or it can stop processing.

Section 3. Developing for LDAP with Perl/C++

This section covers material for topic 302.2 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to:

- Use Perl's `Net::LDAP` module

- Write Perl scripts to bind, search, and modify directories
- Develop in C/C++

Although OpenLDAP includes command-line clients, it's often helpful to use LDAP information in your own scripts. Perl is a popular language for scripting. Perl has a module called `Net::LDAP` that is used to connect to and use an LDAP server.

Getting started

`Net::LDAP` doesn't ship with Perl, but your distribution may include it as a package. See ["Installing from packages"](#) for more information on searching for and installing packages.

If your distribution doesn't have the `Net::LDAP` package, then you can download it from the Comprehensive Perl Archive Network (CPAN). As root, run `perl -MCPAN -e "install Net::LDAP"`, which downloads and installs `Net::LDAP` and any dependencies.

Using Net::LDAP

Using `Net::LDAP` is fairly simple:

1. Create a new `Net::LDAP` object.
2. Bind to the desired server.
3. Perform your LDAP operations.

Create a new object

In typical Perl fashion, you must create an instance of the `Net::LDAP` module through the `new` method. All further operations will be on this instance. `new` requires, at a minimum, the name of the server you want to connect to. For example:

```
my $ldap = Net::LDAP->new('localhost') or die "$@";
```

Here, a new `Net::LDAP` object is created with the `new` method and is passed the string `localhost`. The result is assigned to the `$ldap` variable. If the function fails, the program exits and prints an error message describing the problem. `$@` is a Perl internal variable that contains the status of the last operation.

You can proceed to perform LDAP operations with the new `Net::LDAP` object. Each function returns a `Net::LDAP::Message` object that contains the status of

the operation, any error messages, and any data returned from the server.

Binding to the tree

The first operation you should do is to log in or *bind* to the tree. Listing 6 shows a bind operation and associated error checking.

Listing 6. Perl code to bind to the tree

```
my $message = $ldap->bind(
    "cn=Sean Walberg,ou=people,dc=ertw,dc=com",
    password=>"test" );

if ($message->code() != 0) {
    die $message->error();
}
```

Listing 6 starts by calling the `bind` method of the previously created object. The first parameter to the function is the DN you're binding as. If you don't specify a DN, you bind anonymously. Further parameters are in the format of `key=>value`; the one you'll use most often is the password.

Each `Net::LDAP` method returns a `Net::LDAP::Message` object, which has the results of the function. The error code is retrieved through the `code` method. A code of 0 means success, so the code in Listing 6 exits the program with the error message if the result isn't 0. Note that the error is retrieved from `$message->error` rather than `$@`, like the earlier example. This is because the error isn't a Perl error; it's internal to `Net::LDAP`.

Once the bind is successful, you can do anything you want, subject to the server's access controls. To log out, call the `unbind` method.

Searching the tree

Searching is done through the `search` method. Like the `bind` method, you must pass some parameters and check the result of your query. However, the returned object now contains your data, so this must be parsed. With the `search` operation, the result is a `Net::LDAP::Search` object, which inherits all the methods from `Net::LDAP::Message` (such as `code` and `error`) and adds methods to help you parse the data. Listing 7 shows a search of the tree.

Listing 7. Searching the tree with search

```
$message = $ldap->search(base => "dc=ertw,dc=com", filter=> "(objectClass=*)");
if ($message->code() != 0) {
    print $message->error();
} else {
    foreach my $entry ($message->entries()) {
        print $entry->dn() . " ";
    }
}
```

```

        print join ", ", $entry->get_value("objectClass");
        print "\n";
    }
}

```

Listing 7 begins by calling the `search` method, passing two parameters: the base and a filter. The base tells the server where in the tree to begin searching. A complementary option, `scope`, tells the server how far to search:

- **base** —Only the base object
- **one** —Only the children of the base object (and not the base object itself)
- **sub** —The base object and all its children (the default)

The filter is a string describing the objects you're interested in. You can search on attributes and perform complex AND/OR queries. `objectClass=*` returns any object.

The result of the search is checked, and an error is printed if a problem happened. Because the script could still recover from an error, it just prints the error and continues, rather than exiting.

The `entries` function returns an array of `Net::LDAP::Entry` objects, each with a single result. First the entry's DN is printed, and then all the object classes. If you'd rather have a text version of the whole record, the `dump` method prints the entire entry in text format.

Adding a new entry

You add an entry to the tree through the `add` method. You must pass the function the DN of the entry you wish to add, along with the attributes. The attributes are an array of `key => value` pairs. The value can also be an array in the case of multiple instances of the same attribute. Listing 8 shows an entry being added to the tree.

Listing 8. Adding an entry using Net::LDAP

```

$message = $ldap->add(
    "cn=Fred Flintstone,ou=people,dc=ertw,dc=com",
    attr => [
        cn => "Fred Flintstone",
        sn => "Flintstone",
        objectclass => [ "organizationalPerson",
            "inetOrgPerson" ],
    ]
);

if ($message->code() != 0) {
    print $message->error();
}

```

The first parameter to `add` is either the DN or a `Net::LDAP::Entry` object. If the DN is passed, you must pass an arrayref through the `attr` method. Even though the `key => value` format is used as in a hashref, `Net::LDAP` is expecting an arrayref, so be careful!

More about Net::LDAP

`Net::LDAP` provides an interface to all the LDAP functions, such as `compare`, `delete`, and `moddn`. They're all used similarly to the previous examples and are fully documented in the `Net::LDAP` manpage.

All the examples shown operate in blocking mode, which means the function returns after the response has been received from the server. You can also operate in asynchronous mode, which involves giving a callback function that is called as packets are received.

By using `Net::LDAP`, you can use the data stored in your LDAP tree from within your scripts. Perl is already used in a wide variety of software, so the opportunities for integration are unlimited.

Developing in C/C++

Using the C libraries is more involved than the Perl libraries. The `ldap(3)` manpage contains a detailed description of how to use the library, and has pointers to the other manpages describing each function. To use the LDAP C libraries, your code must first include the `ldap.h` include file, such as with `#include <ldap.h>`. Your object files must then be linked with `libldap` using the `-lldap` option to the linker.

Section 4. Summary

In this tutorial, you learned about installing and configuring the OpenLDAP stand-alone server. When configuring `slapd`, use the `slapd.conf` file. You must take care to keep your global options at the top of the file and then progress to backend and database configurations, because `slapd` is dependent on the order of the directives. When in doubt, consult the `slapd.conf` manpage.

Perl code can make use of an LDAP server through the `Net::LDAP` module. First you create an object, and then you call methods of the object that correspond with the LDAP operation you want. Generally, you first `bind` and then perform your queries. It's important to check the results of your functions through the `code` and `error` functions.

Resources

Learn

- Review the previous tutorial in this 301 series, "[LPI exam 301 prep, Topic 301: Concepts, architecture, and design](#)" (developerWorks, October 2007).
- Take the developerWorks tutorial "[Linux Installation and Package Management](#)" (developerWorks, September 2005) to brush up on your package management commands.
- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Find answers to [What is a backend?](#) and [What is a database?](#) in the OpenLDAP FAQ.
- Learn more about [building overlays](#) in the developer's documentation. Overlays are a complex but powerful concept.
- Consult the [OpenLDAP Administrator's Guide](#) if the manpages for slapd.conf or your particular backend don't help. You might also find the [OpenLDAP FAQ](#) to be helpful.
- The online book [LDAP for Rocket Scientists](#) is excellent, despite being a work in progress.
- The [Perl-LDAP](#) page has lots of documentation and advice on using Net::LDAP.
- In the [developerWorks Linux zone](#), find more resources for Linux developers, and scan our [most popular articles and tutorials](#).
- See all [Linux tips](#) and [Linux tutorials](#) on developerWorks.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- Download [OpenLDAP](#).
- The [IBM Tivoli Directory Server](#) is a competing LDAP server that integrates well with other IBM products.
- [phpLDAPadmin](#) is a Web-based LDAP administration tool. If the GUI is more your style, [Luma](#) is a good one to look at.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software

for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the [developerWorks community](#) through blogs, forums, podcasts, and community topics in our [new developerWorks spaces](#).

About the author

Sean Walberg

Sean Walberg has been working with Linux and UNIX since 1994 in academic, corporate, and Internet service provider environments. He has written extensively about systems administration over the past several years.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 301 prep, Topic 303: Configuration

Senior Level Linux Professional (LPIC-3)

Skill Level: Intermediate

[Sean A. Walberg \(sean@ertw.com\)](mailto:sean@ertw.com)
Senior Network Engineer

04 Mar 2008

In this tutorial, Sean Walberg helps you prepare to take the Linux Professional Institute Senior Level Linux Professional (LPIC-3) exam. In this third in a [series of six tutorials](#), Sean walks you through configuring a Lightweight Directory Access Protocol (LDAP) server, including access control, security, and performance. By the end of this tutorial, you'll know about LDAP server configuration.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux® system administrators at three levels: *junior level* (also called "certification level 1"), *advanced level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102. To attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active advanced-level certification and pass exam 301 ("core"). You may also need to pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the five junior, advanced, and senior certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. Table 1 lists the six topics and

corresponding developerWorks tutorials for LPI exam 301.

Table 1. LPI exam 301: Tutorials and topics

LPI exam 301 topic	developerWorks tutorial	Tutorial summary
Topic 301	LPI exam 301 prep: Concepts, architecture, and design	Learn about LDAP concepts and architecture, how to design and implement an LDAP directory, and about schemas.
Topic 302	LPI exam 301 prep: Installation and development	Learn how to install, configure, and use the OpenLDAP software.
Topic 303	LPI exam 301 prep: Configuration	(This tutorial) Learn how to configure the OpenLDAP software in detail. See the detailed objectives .
Topic 304	LPI exam 301 prep: Usage	Coming soon.
Topic 305	LPI exam 301 prep: Integration and migration	Coming soon.
Topic 306	LPI exam 301 prep: Capacity planning	Coming soon.

To pass exam 301 (and attain certification level 3), the following should be true:

- You should have several years of experience with installing and maintaining Linux on a number of computers for various purposes.
- You should have integration experience with diverse technologies and operating systems.
- You should have professional experience as, or training to be, an enterprise-level Linux professional (including having experience as a part of another role).
- You should know advanced and enterprise levels of Linux administration including installation, management, security, troubleshooting, and maintenance.
- You should be able to use open source tools to measure capacity planning and troubleshoot resource problems.
- You should have professional experience using LDAP to integrate with UNIX® services and Microsoft® Windows® services, including Samba, Pluggable Authentication Modules (PAM), e-mail, and Active Directory.
- You should be able to plan, architect, design, build, and implement a full

environment using Samba and LDAP, as well as measure the capacity planning and security of the services.

- You should be able create scripts in Bash or Perl or have knowledge of at least one system programming language (such as C).

The Linux Professional Institute doesn't endorse any third-party exam preparation material or techniques in particular.

About this tutorial

Welcome to "Configuration," the third of six tutorials designed to prepare you for LPI exam 301. In this tutorial, you learn about LDAP server configuration, including access controls, security, replication, and database performance.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weights, as shown in Table 2.

Objectives

Table 2 lists the detailed objectives for this tutorial.

Table 2. Configuration: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
303.2 Access control lists in OpenLDAP	2	Plan and implement access control lists.
303.3 LDAP replication	5	Set up OpenLDAP to replicate data between multiple servers.
303.4 Securing the directory	4	Configure encrypted access to the LDAP server, and restrict access at the firewall level.
303.5 LDAP server performance tuning	2	Measure the performance of your LDAP server, and tune for maximum performance.
303.6 OpenLDAP daemon configuration	2	Understand the basic <code>slapd.conf</code> configuration directives, and become familiar with the basic <code>slapd</code> command-line options.

Prerequisites

To get the most from this tutorial, you should have advanced knowledge of Linux and a working Linux system on which to practice the commands covered.

If your fundamental Linux skills are a bit rusty, you may want to first review the [tutorials for the LPIC-1 and LPIC-2 exams](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

System requirements

To follow along with the examples in these tutorials, you need a Linux workstation with the OpenLDAP package and support for PAM. Most modern distributions meet these requirements.

Section 2. Access control lists in LDAP

This section covers material for topic 303.2 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 2.

In this section, learn how to:

- Plan LDAP access control lists
- Understand access control syntax
- Grant and revoke LDAP access permissions

A variety of data can be stored in an LDAP tree, including phone numbers, birth dates, and payroll information. Some of these may be public, and some may be accessible by only certain people. The same information may have different restrictions based on the user. For example, perhaps only the owner of the record and administrators can change a phone number, but everyone can read the number. Access control lists (ACLs) handle the configuration of these restrictions.

Planning LDAP access control lists

Before you start writing your configuration, you should determine what you want to achieve. Which parts of the tree contain sensitive information? Which attributes need to be protected, and from whom? How will the tree be used?

The components of an ACL

An ACL entry supplies three pieces of information:

1. **What** entries and attributes the ACL specifies
2. **Who** the ACL applies to
3. The level of **access** that is granted

Regular expressions

Regular expressions are used for matching text. You may have a general idea of what the text looks like, or know certain patterns, and you can build a regular expression to find what you're looking for. A regular expression, or *regex* for short, consists of literal matches and meta characters that match a variety of patterns.

A simple regex is `hello`, which matches any string of characters containing the pattern `hello`.

You may not know if the string is capitalized, so the set meta characters, `[]`, match one occurrence of any of the characters inside the set. So, `[Hh]ello` matches both `hello` and `Hello`.

The period matches any single character. `.ello` matches `Hello` and `hello`, but also `fellow` and `cello`. It doesn't, however, match `ello`, because the period has to match something.

The characters `?`, `*`, and `+` match zero or one of the preceding character, zero or more of the preceding character, and one or more of the preceding character, respectively. Thus, `hello+` matches `hello` and `helloooooo`, but not `hell`.

Regular expressions have many different options and allow you to efficiently pull patterns from text files. In the context of OpenLDAP, regular expressions are used to match parts of a DN to avoid having to hand-code hundreds of different possibilities.

When specifying the "what" clause, you can choose to filter on the distinguished name (DN) of the object, an LDAP-style query filter, a list of attributes, or a combination of the three. The simplest clause allows everything, but you can get much more restrictive. Filtering on DN lets you specify an exact match, such as `ou=People,dc=ertw,dc=com`, or a regular expression (see "[Regular expressions](#)"). The query filter can match a certain `objectClass` or other attributes. The attribute list is a comma-separated list of attribute names. A more complex matching criteria might be "All password entries under

```
ou=People,dc=ertw,dc=com who are administrators."
```

You have a great deal of flexibility when determining who the ACL applies to. Users are generally identified by the DN with which they bind to the tree, which is called the *bindDN*. Each LDAP entry can have a `userPassword` attribute that is used to authenticate that particular user. In some contexts, you can refer to the currently logged-in user as *self*, which is useful for allowing a user to edit his or her own details.

If a user doesn't bind, they're considered *anonymous*. By default, anonymous users can read the tree, so you must decide if you need to change this behavior. You can further segment your anonymous users (or any user, for that matter) by IP address or the method used to connect (such as plaintext or encrypted).

Once you've determined the what and the who, you must determine the level of access. Access can range from *none* up to *write* access. You may also specify that the user can authenticate against the entry but can't read; or you can give the user read, search, and compare access.

Regardless of how you configure your ACLs, any configured `rootDN` users have full control over their database. You can't change this, except by removing the `rootDN` configuration from `slapd.conf`.

Understanding access control syntax

The basic form of an ACL, expressed in Backus-Naur Form, is:

```
access to <what> [ by <who> [ <access> ] [ <control> ] ]+
```

Backus-Naur Form

Backus-Naur Form (BNF) is a way to describe grammars such as ACL syntax. It's often used in developing Internet protocols because BNF is terse and very precise.

In BNF notation, you have a left-hand item and a right-hand item separated by a `::=` symbol. This means the left-hand side can be substituted by items on the right-hand side. Items on the right-hand side that are enclosed in angle brackets (`<` and `>`) refer to another line of BNF, with the item in the angle brackets appearing on the left-hand side.

Items in square brackets (`[` and `]`) are optional. Vertical bars (`|`) indicate "one or the other"; and `+` and `*` mean "one or more of the preceding" and "zero or more of the preceding," respectively. Those familiar with regular expressions will recognize many of these items.

Looking at the BNF for an ACL, an ACL entry consists of the literal string "access to", followed by an item called "what" that is defined somewhere else. Following that are one or more lines of the form `by <who> [<access>] [<control>]`, where `who`, `access`, and `control` are defined elsewhere, and both `access` and `control` are optional.

We explore the missing grammar in the rest of this tutorial.

Describe the what

The *what* describes the attributes and entries that are to be enforced by the ACL. The BNF notation to do so is shown in Listing 1.

Listing 1. BNF description of the what part of an ACL

```
<what>          ::= * |
                 [dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
                 [filter=<ldapfilter>] [attrs=<attrlist>]
<basic-style>  ::= regex | exact
<scope-style> ::= base | one | subtree | children
<attrlist>    ::= <attr> [val[.<basic-style>]=<regex>]
                 | <attr> , <attrlist>
<attr>        ::= <attrname> | entry | children
```

Some of the elements in Listing 1 aren't defined here, such as DN and regex. The form of the distinguished name is already known, and regular expressions are best studied outside of BNF.

Listing 1 shows that a description of the what portion of the ACL can be either the asterisk character (*), which matches everything, or a combination of a description of the DN, an LDAP filter, and a list of attributes. The latter possibility can use one or more of three components, because they're individually enclosed in square brackets.

Listing 2 shows three what clauses that match the DN.

Listing 2. Three sample what clauses

```
dn.exact="ou=people,dc=ertw,dc=com"
dn.regex="ou=people,dc=ertw,dc=com$"
dn.regex="^cn=Sean.*,dc=com$"
```

The first example matches only the entry for `ou=people,dc=ertw,dc=com`. If this ACL is used, it doesn't match any children such as `cn=Sean Walberg,ou=people,dc=ertw,dc=com`, nor does it match the parent entry.

The second example is similar to the first, but it uses a regular expression and *anchors* the search string with the dollar-sign (\$) character. An anchor matches the position of the string rather than part of the string. The dollar sign matches the end of the string, so the second example matches anything ending in `ou=people,dc=ertw,dc=com`, which includes `cn=Sean Walberg,ou=people,dc=ertw,dc=com`. Note that without the anchor, the search string could be anywhere within the target, such as

```
ou=people,dc=ertw,dc=com,o=MegaCorp.
```

The third example from Listing 2 shows another anchor, `^`, which matches the beginning of the string. The third example also uses another regular expression, `.*`. The period matches any one character, and the asterisk matches zero or more of the preceding character. Thus, `.*` matches any string of zero or more characters. Put together, the third example matches any entry starting with `cn=Sean` and ending with `dc=com`.

You can also filter based on LDAP queries, the most helpful being a search on `objectClass`. For example, a `what` clause of `filter=(objectClass=posixAccount)` matches only entries with an `objectClass` of `posixAccount`. For a review of `objectClass`, see the first tutorial in this series, [LPI exam 301 prep: Concepts, architecture, and design](#).

The final option for the `what` clause is to specify attributes. The most common usage is to restrict who can see private attributes, especially passwords. Use `attrs=userPassword` to match the password attribute.

Once you've determined what entries and attributes are to be matched, you must then describe who the rule will apply to.

Describe the who

Access is applied to a user, based on the DN that was provided at the time the client bound to the tree. The DN is usually found on the tree, but it could also be the `rootDN` provided in `slapd.conf`.

Listing 3 shows the BNF for notation for the `who` part of the ACL.

Listing 3. BNF notation for matching the `who` part of an ACL

```
<who> ::= * | [anonymous | users | self[.<selfstyle>]
              | dn[.<basic-style>]=<regex> |
dn.<scope-style>=<DN>]
      [dnattr=<attrname>]
[group[/<objectclass>[/<attrname>][.<basic-style>]]=<regex>]
      [peername[.<peernamestyle>]=<peername>]
      [sockname[.<style>]=<sockname>]
      [domain[.<domainstyle>[,<modifier>]]=<domain>]
      [ssf=<n>]
      [transport_ssf=<n>]
      [tls_ssf=<n>]
      [sasl_ssf=<n>]

<style> ::= {exact|regex|expand}
<selfstyle> ::= {level{<n>}}
<dnstyle> ::= {{exact|base(object)}|regex
              |one(level)|sub(tree)|children|level{<n>}}
<groupstyle> ::= {exact|expand}
<peernamestyle> ::= {<style>|ip|path}
<domainstyle> ::= {exact|regex|sub(tree)}
<modifier> ::= {expand}
```

As in the what part of the ACL, an asterisk matches everything. To get more specific, you have many options. OpenLDAP defines three shortcuts named `anonymous`, `users`, and `self`. These shortcuts match unregistered users, authenticated users, and the currently logged-in user, respectively. The latter, `self`, is often used to allow the logged-in user to edit components of his or her own profile. This is based on an exact match of the DN; if you have a user's information split across different entries, the `self` keyword applies only to the entry the user bound with.

An interesting thing about the `self` keyword is that you can also make the ACL apply to parents or children of the user's entry with the `level` keyword. Using `self.level{1}` matches the user's entry and the parent entry, whereas `self.level{-1}` matches the user's entry and any directly attached children.

Still looking at the DN, you can perform regular-expression or exact matches with `dn.exact="DN"` and `dn.regex="regex"`, respectively. A later example will show how to use regular expressions to dynamically tie the what and the who together.

Arbitrary entries can be protected using the `dnattr` keyword, which also requires the name of an attribute. If the DN of the requester appears in the specified attribute of the target, the ACL is matched. For example, if you add `dnattr=manager` to your ACL and then add `manager: cn=Joe Blow,ou=people,dc=ertw,dc=com` to Fred Smith's entry, the ACL will match when Joe Blow accesses Fred Smith's entry.

The `group` keyword is similar to `dnattr`, except that the parameters refer to a group defined elsewhere in the tree rather than an attribute in the entry. By default, the group has an `objectClass` of `groupOfNames`, and the members are referenced in the `member` attribute.

Use the `peername`, `sockname`, and `domain` keywords to match attributes of the client connection. `peername` refers to the IP address of the client, such as `peernameip=127.0.0.1`. `sockname` is for connections over named pipes, which aren't commonly used. `domain` matches the hostname associated with the IP address, which can be easily spoofed.

The final set of options refers to the Security Strength Factor (SSF) of the connection, which is an OpenLDAP term for the connection's level of security. These options will become clearer when you're introduced to the security mechanisms used to connect to OpenLDAP, such as Transport Layer Security (TLS) and Simple Authentication and Security Layer (SASL).

All of the preceding items can be used together. For example, you could allow write access to the password field only to certain administrators coming from a certain IP address range with a certain level of encryption. You could also do far less, such as

requiring only a valid login, or even accepting everyone regardless of authentication.

Describe the access

Once you've determined who is accessing your tree and what they're trying to access, you must specify what level of access they have. Listing 4 shows the BNF notation for the access part of the ACL.

Listing 4. BNF notation describing the format of the access clause

```
<access> ::= [[real]self]{<level>|<priv>}
<level> ::= none|disclose|auth|compare|search|read|write
<priv> ::= {=|+|-}{w|r|s|c|x|d|0}+
```

When specifying access using the `level` format, each successive level includes the ones before it. That is, `read` access gives `search`, `compare`, `auth`, and `disclose` access. `none` and `disclose` both deny any access, except that some error messages that might disclose information about the contents of the tree are removed under `none` and allowed under `disclose`.

Alternatively, you can specify the level of access in terms of the LDAP operations permitted using the `priv` format. The options run opposite to the `level` format, such that `w` is for write and `0` is for none. When specifying access using the `priv` format, there is no implied progression as is the case with `level`. If you want to offer full access, you must do so with `wrscx`.

The `=/+/-` symbol before the letters denotes how the specified access is merged with the current access level if multiple rules apply. With `=`, all previously defined access is ignored, and the value to be used follows the equal sign. With `+` and `-`, access is added to or subtracted from the current level, respectively.

Understand control

By default, OpenLDAP takes a first-match approach to applying access lists. OpenLDAP finds the first ACL entry that matches the `what` clause and, within that entry, finds the first entry matching the `who` part. This is the same as putting the keyword `stop` after the access level is described. The other two options are `continue` and `break`. If you use `continue`, the current ACL entry is searched for the next line matching the `who` part. If you use `break`, processing of the current ACL entry stops, but OpenLDAP looks for the next ACL entry matching the `who` clause.

Pulling together the what, who, and access

Now that you've seen the three (four, if you count control) parts of the ACL, you can bring them together into a policy. Listing 5 shows a typical list of ACLs that allows

registered users to read the tree and lets users update their own passwords (but not read them).

Listing 5. A simple ACL setup

```
access to attrs=userPassword
  by self =xw
  by anonymous auth

access to *
  by self write
  by users read
```

The first clause matches anyone trying to access the `userPassword` field. The user is given write and authentication permission to their own entry, which is enforced through the equals sign. Anonymous users are given authentication permission. A user is anonymous as they're binding to the tree; therefore, anonymous users require the `auth` permission so they may log in to be a regular, privileged user.

If the information being requested isn't the password, the second ACL entry is consulted. Again, the user has full control over his or her own entry (except the `userPassword` field, by virtue of the first ACL entry), whereas all authenticated users have read access to the rest of the tree.

Listing 6 shows an ACL entry that uses regular expressions to tie the `what` and `who` clauses together.

Listing 6. Getting fancy with regular expressions

```
access to dn.regex="cn=([^\,]+),ou=addressbook,dc=ertw,dc=com"
  by dn.regex="cn=$1,ou=People,dc=ertw,dc=com" write
  by users read
```

Listing 6 allows users to edit their corresponding records under the `ou=addressbook,dc=ertw,dc=com` part of the tree. `[^\,]+` matches everything up to, but not including, a comma, and the parentheses save the matched text into `$1` for the first set of parentheses, `$2` for the next, and so forth up to and including `$9`. The `who` clause reuses the name of the user to determine who can access the entry. If the name of the user is the same as that of the entry being accessed, then the user is given write access. Failing that, authenticated users are given read access.

Practical considerations

It's wise to keep the more specific ACL entries at the top of the list because of the first-match behavior; otherwise, it's more likely that a previous ACL will cause the

one lower down the list to be ignored. This technique can also be used to grant and revoke access to a particular user. Simply put your specific ACL clause at the top of the list.

Keep your ACLs as simple as possible. Doing so reduces the possibility of error and also improves performance. ACLs must be parsed each time an operation is carried out against the tree.

Section 3. LDAP replication

This section covers material for topic 303.3 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 5.

In this section, you learn how to do the following:

- Understand replication concepts
- Configure OpenLDAP replication
- Execute and manage slurpd
- Analyze replication log files
- Understand replica hubs
- Configure LDAP referrals
- Configure LDAP sync replication

At some point, your needs may extend beyond one server. Your organization may rely on LDAP to the extent that the loss of your LDAP server is unacceptable, or your query volume may be high enough that you have to split your queries across multiple servers. It could even be a combination of both; but in any case, you need to use more than one server.

With multiple servers, you can partition your tree across different servers, but this leads to a decrease in reliability -- not to mention that it may be difficult to properly balance your queries. Ideally, each server has an identical copy of the tree. Any writes are propagated to the other servers in a timely fashion so that all the other servers are up to date. This is called *replication*.

This scenario, called *multi-master*, is complex because the data has no clear single owner. Most often, a master-slave relationship is formed, where one server takes on all the writes and sends them to the slaves. LDAP queries can be made against any

server. This can be extended to the *replica hub* scenario, where a single slave server replicates data from the master and, in turn, replicates data to several other slaves.

OpenLDAP provides two methods to achieve replication. The first is through *slurpd*, a separate daemon that watches for changes on the master and pushes the changes to the slaves. The second is using slapd's LDAP sync replication engine, otherwise known as *syncrepl*. The slurpd method is now considered obsolete; people are urged to use *syncrepl* instead. Both these methods are investigated in this section.

slurpd-based replication

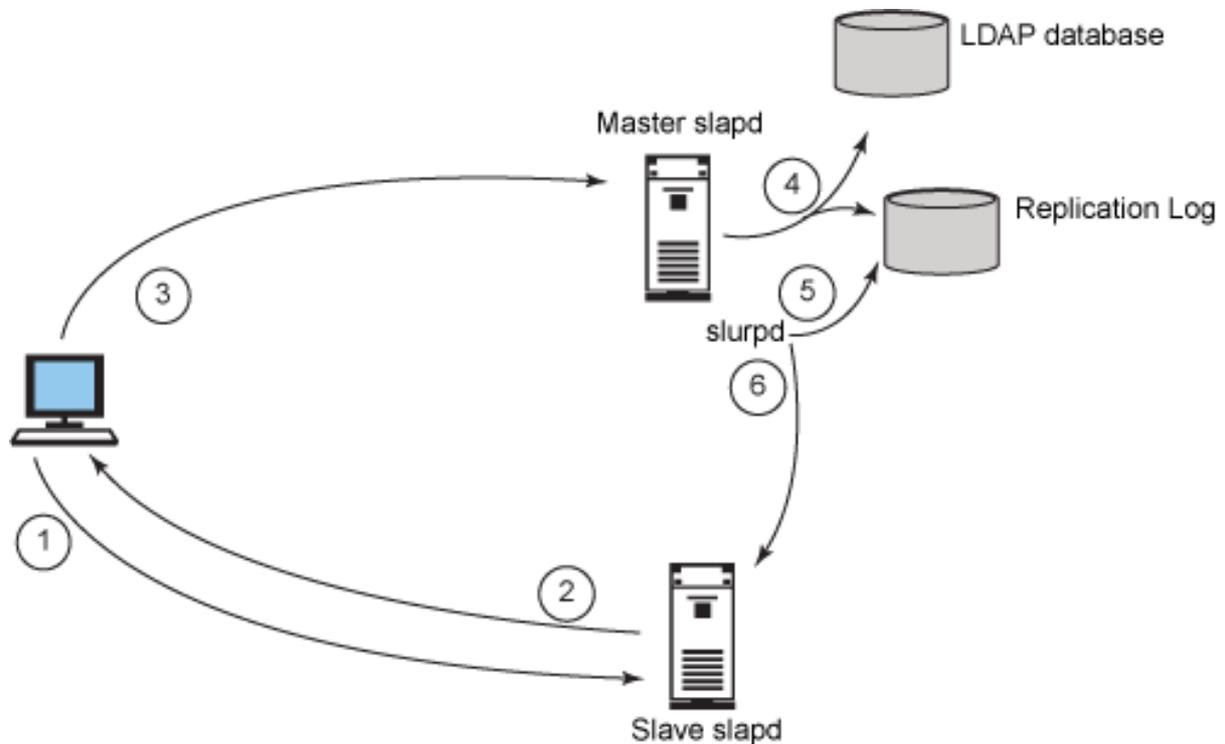
Slurpd-based replication is a push replication in that the master pushes any changes to the slaves. If a client attempts to update a slave, the slave is configured to send a *referral* back to the client, pointing the client to the master. The client is responsible for reissuing the request to the master. Slurpd is a standalone daemon that is configured from `slapd.conf`.

Data flow within the slurpd replication model

The master server is the server that handles all the writes from the clients and holds the authoritative source of data. Any changes to the master's tree are written to a *replication log*, which is monitored by slurpd. slurpd pushes the changes to all the slaves upon noticing a change in the replication log.

Figure 1 shows the typical behavior of slurpd.

Figure 1. Data flow in the slurpd replication model



The process is as follows:

1. The client sends an update request, which happens to be received by a slave.
2. The slave knows that writes can only come from its replication partner, and therefore it sends a referral back the client, pointing it to the master server.
3. The client reissues the update request to the master.
4. The master performs the update and writes the change to the replication log
5. slurpd, also running on the master, notices the change in the replication log.
6. slurpd sends the change to the slave.

In this way, slaves can be kept up to date with the master with little lag. If any interruptions happen, or an error occurs on a slave, slurpd always knows which slaves need which updates.

Configure slurpd

Configuring slurpd-based replication require the following steps:

1. Create a replica account that slurpd will use to authenticate against the slave replica.
2. Configure the master server with the name of the slave.
3. Configure the slave to be a replica, including any needed ACLs.
4. Copy the database from the mater to the slave.

Creating the replica is straightforward; the only requirement is that the account must have a password in the `userPassword` attribute. You may use the `inetOrgPerson` `objectClass` like most accounts belonging to people, or you can use a more generic `objectClass` such as `account` with the `simpleSecurityObject` auxiliary `objectClass` added. Recall from the first tutorial that structural `objectClasses` define the entry (and therefore you may use only one per entry), whereas auxiliary `objectClasses` add attributes to any entry regardless of the structural `objectClass`. Listing 7 shows the LDIF code to add a replica account.

Listing 7. LDIF code for a replica account

```
dn: uid=replica1,dc=ertw,dc=com
uid: replica1
userPassword: replica1
description: Account for replication to slave1
objectClass: simpleSecurityObject
objectClass: account
```

Listing 7 shows a sparse entry -- just a username, password, and description, which is adequate for replication. The description is optional, but it's recommended for documentation. Remember the password; you'll need it in the next step!

The master must now be configured to store all changes in a replication log, and a replica must be configured for slurpd to work. It's important to remember that slurpd reads its configuration from `slapd.conf`, and that slurpd pushes the updates to the slaves. This will help you remember where to configure replication and that the authentication credentials belong on the master. The slave can validate the credentials because the account is part of the tree. Listing 8 shows the configuration on the master to enable replication.

Listing 8. Configuration of master for slurpd replication

```
replica uri=ldap://slaveserver.ertw.com
suffix="dc=ertw,dc=com"
binddn="uid=replica1,dc=ertw,dc=com"
```

```
credentials="replica1"  
bindmethod=simple  
  
repllogfile /var/tmp/replicationlog
```

Configuration of replication happens in database mode, so be sure to have the `replica` command somewhere after your first database configuration. The `replica` takes several parameters of the form `key=value`. The `uri` key specifies the name or IP address of the slave in uniform resource identifier (URI) format, effectively the name of the slave prepended with `ldap://`.

After specifying the name of the slave, you can optionally configure the name of the database to replicate through the `suffix` option. By default, all databases are replicated. The final requirement is to provide authentication information so that `slurpd` can connect to the specified `uri`. For simple authentication, the `binddn`, `bindmethod`, and `credentials` (the `userPassword` you assigned earlier) are all you need.

The final piece of the puzzle is to tell `slapd` where to store its replication log. You need to provide a full path because relative filenames don't work. Don't worry about creating the file, because `slapd` will create it for you; but the path you specify must be writable by the user `slapd` and `slurpd` are running as.

On the slave server, you must configure the replication account and also tell the slave that it should return a referral back to the master for any writes. Listing 9 shows this configuration.

Listing 9. Configuration for the slave

```
updatedn uid=replica1,dc=ertw,dc=com  
updateref ldap://masterserver.ertw.com
```

The `updatedn` refers back to the account you created earlier on the master that `slurpd` will use to push the updates to the slaves. The `updateref` is another LDAP URI, this one pointing to the master. Listing 10 shows a client trying to update the slave after the previous configuration has been loaded, and receiving a referral.

Listing 10. A client receiving a referral after trying to update a slave

```
[root@slave openldap]# ldapadd -x -D cn=root,dc=ertw,dc=com -w mypass -f  
newaccount.ldif  
adding new entry "cn=David Walberg,ou=people,dc=ertw,dc=com"  
ldap_add: Referral (10)  
referrals:  
ldap://masterserver.ertw.com/cn=David%20Walberg,ou=People,dc=ertw,dc=com
```

The OpenLDAP command-line clients don't follow referrals, but other LDAP libraries

do. If you're using LDAP in a replicated environment, you should verify that your applications follow referrals correctly.

The final piece of the puzzle is to make sure the master and slave start with identical databases. To do so, follow these steps:

1. Shut down the master server.
2. Shut down the slave server.
3. Copy all the database files from the master to the slave.
4. Start both the master and the slave servers.
5. Start up slurpd.

Both servers must be shut down to copy the database, to ensure that no changes are made and that all data has been written to disk. It's vital that both servers start with the same data set; otherwise they may get out of sync later. slurpd-based replication essentially plays back all the transactions on the slave that happened on the master, so any differences can cause problems.

slurpd may automatically start up with slapd, depending on your distribution and start-up scripts. If it hasn't started automatically, start it by running `slurpd` at the command line.

At this point, replication should be running. Create an account on your master, and test. Also test that your slave sends back referrals if you try to update it.

Monitor replication

Understanding how to monitor replication is important because errors can happen that may cause data to get out of synchronization. The same skills in monitoring also help in debugging.

slurpd stores its own files in `/var/lib/ldap/replica` (this is separate from the replication log that is produced by slapd). In this directory are slurpd's own replication logs and any *reject files*. If slurpd tries to send an update to a slave that fails, the data is stored in a file named after the slave, with an extension of `.rej`. Inside the file is the LDIF code making up the entry along with the error returned by the slave, such as `ERROR: Already exists`. Listing 11 shows the contents of a `.rej` file with a different error.

Listing 11. A replication rejection file

```
ERROR: Invalid DN syntax: invalid DN
```

```
replica: slaveserver.ertw.com:389
time: 1203798375.0
dn: sendmailMTAKey=testing,ou=aliases,dc=ertw,dc=com
changetype: add
objectClass: sendmailMTAAliasObject
sendmailMTAAliasGrouping: aliases
sendmailMTACluster: external
sendmailMTAKey: testing
sendmailMTAAliasValue: testinglist@ertw.com
structuralObjectClass: sendmailMTAAliasObject
entryUUID: 5375b66c-7699-102c-822b-fbf5b7bc4860
creatorsName: cn=root,dc=ertw,dc=com
createTimestamp: 20080223202615Z
entryCSN: 20080223202615Z#000000#00#000000
modifiersName: cn=root,dc=ertw,dc=com
modifyTimestamp: 20080223202615Z
```

The rejection file in Listing 11 starts with a text error ("ERROR: Invalid DN syntax: invalid DN"), and the rest looks like LDIF. Note the first attribute is **replica**, which is the name of the replica that couldn't process the update, and the second attribute, **time**, is the time the error occurred (in UNIX timestamp format). The next few attributes come from the entry that was rejected.

The last 7 attributes are called **operational attributes**. They weren't part of the original change, but were added by the LDAP server for internal tracking. A universally unique identifier (UUID) was given to the entry, along with some information on when and who changed the record.

In Listing 11, the error most likely comes from a missing schema on the slave. The slave doesn't understand what the **sendmailMTAKey** attribute is, therefore the DN of the entry is invalid. The slave must have its schema updated before replication can continue.

To fix a rejected entry, you must evaluate the error and fix the underlying problem with the tree. Once you know the rejected entry will apply cleanly, use slurpd's *one-shot mode* to apply the update with `slurpd -r /path/to/rejection.rej -o`. The `-r` parameter tells slurpd to read from the given replication log, and `-o` causes slurpd to use one-shot mode, meaning it exits after processing the log instead of the default behavior of waiting for more entries to be added to the log.

If replication isn't working at all, the best approach is to start with the master and work your way to the slave. First, kill slurpd and make a change to the tree. Then, check to see if the replication log is growing; if it isn't, the master is set up incorrectly. Next, start up slurpd, and pass `-d 255` on the command line. This traces slurpd's actions as it processes the logs. Look for errors, especially related to opening files and access controls.

Finally, on the slave, use `loglevel auth sync` to check for any errors when replication is happening (slapd logs to syslog with the `local4` facility, so you may need to add `local4.* /var/log/slapd.log` to `/etc/syslog.conf`).

LDAP Sync replication

slurpd is a straightforward solution to the replication problem, but it has several shortcomings. Shutting down your master server so that you can synchronize a slave is inconvenient at best, and at worst it can affect service. The push architecture of slurpd can also be limiting. slurpd worked well for its time, but something better had to be created. RFC 4533 outlines the LDAP Content Synchronization Operation, which is implemented in OpenLDAP by the LDAP Sync replication engine, otherwise known as *syncrepl*.

syncrepl is built as an overlay that is inserted between the core of slapd and the backend database. All writes to the tree are tracked by the syncrepl engine rather than requiring a separate server instance. Other than the mechanics of the replication and the roles (described next), the concepts are similar to slurpd. Writes to a replica are refused, with a referral passed back to the master server.

syncrepl is initiated from the slave, which is now given the name *consumer*. The master role is called *provider*. In syncrepl, the consumer connects to the provider to get updates to the tree. In the most basic mode, called *refreshOnly*, the consumer receives all the changed entries since its last refresh, requests a cookie that keeps track of the last synchronized change, and then disconnects. On the next connection, the cookie is presented to the provider, which sends only the entries that changed since the last synchronization.

Another syncrepl mode, called *refreshAndPersist*, starts off like the *refreshOnly* operation; but instead of disconnecting, the consumer stays connected to receive any updates. Any changes that happen after the initial refresh are immediately sent over the connection to the consumer by the provider.

Configure syncrepl

Listing 12 shows the provider's configuration for both syncrepl modes (*refreshOnly* and *refreshAndPersist*).

Listing 12. Provider configuration for syncrepl

```
overlay syncprov
syncprov-checkpoint 100 10
syncprov-sessionlog 100
```

The first line of Listing 12 enables the syncprov overlay. Overlays must be configured against a database; therefore, this configuration must go after your database configuration line. The next two lines are optional, but they improve reliability. `syncprov-checkpoint 100 10` tells the server to store the value of `contextCSN` to disk every 100 write operations or every 10 minutes. `contextCSN`

is part of the cookie mentioned earlier that helps consumers pick up where they left off after the last replication cycle. `syncprov-sessionlog 100` logs write operations to disk, which again helps in the refresh cycle.

More details about configuring the provider are found in the `slapo-syncprov(5)` manpage.

Listing 13 shows the consumer side of the replication pair.

Listing 13. Consumer configuration for `syncrepl` in `refreshOnly` mode

```
updateref ldap://masterserver.ertw.com
syncrepl rid=1
  provider=ldap://masterserver.ertw.com
  type=refreshOnly
  interval=00:01:00:00
  searchbase="dc=ertw,dc=com"
  bindmethod=simple
  binddn="uid=replica1,dc=ertw,dc=com"
  credentials=replica1
```

Like the `replica` command from the `slurpd` configuration, the `syncrepl` command requires an `updateref`, information about the tree you're trying to replicate, and the authentication credentials you're going to use. This time, the credentials go on the consumer side and require enough access on the provider to read the part of the tree being replicated. The updates to the database on the consumer are run as the `rootdn`.

Items in Listing 13 specific to `syncrepl` are the `rid`, `provider`, `type`, and `interval`. The `rid` identifies this consumer to the master. The consumer must have a unique ID between 1 and 999. The `provider` is an LDAP URI pointing back to the provider. `type` specifies that you only want periodic synchronization through `refreshOnly`, and the `interval` is every hour. The `interval` is specified in `DD:hh:mm:ss` format.

Start the consumer with an empty database, and it will replicate its data from the provider and update every hour.

Making the transition to `refreshAndPersist` mode is simple. In Listing 13, remove the `interval`, and change the `type` to `refreshAndPersist`

`syncrepl` Filtering

It's worth noting that you don't have to replicate the whole LDAP tree. You can use the following commands to filter the data that is replicated.

Table 3. Commands for filtering replication traffic

Command	Description
---------	-------------

searchbase	A DN pointing to the node in the tree where replication will start. OpenLDAP will fill in any necessary parent nodes to make the tree complete.
scope	One of <code>sub</code> , <code>one</code> , or <code>base</code> . This determines how far down the tree, starting at the <code>searchbase</code> , that data is replicated. The default is <code>sub</code> , which covers the <code>searchbase</code> and all children
filter	A LDAP search filter, such as <code>(objectClass=inetOrgPerson)</code> that controls which records are replicated.
attrs	A list of attributes that will be copied over from the selected entries.

Like the other options for `syncrepl`, these options are entered in the form of `key=value`

Section 4.

This section covers material for topic 303.4 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 4.

In this section, you learn how to do the following:

- Secure the directory with SSL and TLS
- Configure and generate client / server certificates
- Understand firewall considerations
- Configure unauthenticated access methods
- Configure User / password authentication methods

Up to this point, all access to `slapd` has been over unencrypted channels using cleartext passwords. This is called *simple* authentication. This section looks at adding encryption to the client-server connection.

Use SSL and TLS to secure communications

You may be familiar with Secure Sockets Layer (SSL) and Transport Layer Security (TLS) as the protocols that secure Web transactions. Whenever you browse an https type URI, you're using SSL or TLS. TLS is an improvement on SSLv3 and in some cases is backward compatible to SSL. Because of their shared heritage and compatibility, they're often referred to collectively as SSL.

SSL makes use of X.509 certificates, which are a piece of data in a standard format that has been digitally signed by a trusted third party known as a Certificate Authority (CA). A valid digital signature means the data that was signed hasn't been tampered with. If the data being signed changes, even by one bit, then the signature won't validate. Independent parties, such as the client and the server, can validate signatures because they both start off by trusting the CA.

Within the server's certificate is information about the ownership of the server, including its name on the Internet. Thus you can be sure you're connecting to the right server because the name of the server you connected to exactly matches the name in the certificate, and you've trusted the CA to validate this before signing. The certificate also includes the server's public key, which can be used to encrypt data such that only the holder of the secret key can decrypt it.

Public and secret keys form the basis of *public key* or *asymmetric* cryptography. It's asymmetric because data encrypted by the public key can only be decrypted by the secret key, and data encrypted with the secret key can only be decrypted by the public key. For what you normally think of as encryption, such as keeping a message secret, the first case is used. The public key is made public, and the secret key is made secret. Because of the asymmetric behavior of the keys, the secret key can encrypt a message, and anyone with the public key can decrypt it. This is how digital signatures work.

After the client connects to the server and receives the server's certificate, the client can validate that the server name is correct, which prevents a *man in the middle attack*. The public key can be used to run through a protocol that ends up with the client and server agreeing on a shared secret that no one observing the conversation can determine. This secret is then used to encode the rest of the conversation between the client and the server, called *symmetric* cryptography because the same key both encrypts and decrypts the data. The divide between asymmetric and symmetric cryptography exists because the latter is orders of magnitude faster. Public key cryptography is used to authenticate and come up with a shared secret, and then symmetric key cryptography takes over.

To apply this all to OpenLDAP, you must create a certificate for the server and then configure the server to use it. This example uses a self-signed certificate rather than creating a Certificate Authority, which means the final certificate has been signed by itself. It doesn't provide the level of trust that a CA does, but it's adequate for testing. Listing 14 shows the generation of the key.

Listing 14. Generating the TLS key pair

```
[root@bob ssl]# openssl genrsa -out ldap.key 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
```

Listing 14 shows the key being generated with the `openssl genrsa` command. The key is 1024 bits long, which is currently considered adequate for public keys (note that using much larger values makes cryptographic operations much slower and may confuse some clients). Next, `openssl req` takes the public part of the freshly generated key pair, adds some location information, and packages the result -- a Certificate Signing Request (CSR) -- to be signed by a CA (see Listing 15).

Listing 15. Generating the Certificate Signing Request

```
[root@bob ssl]# openssl req -new -key ldap.key -out ldap.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:CA
State or Province Name (full name) [Berkshire]:Manitoba
Locality Name (eg, city) [Newbury]:Winnipeg
Organization Name (eg, company) [My Company Ltd]:ERTW
Organizational Unit Name (eg, section) []:Directory Services
Common Name (eg, your name or your server's hostname)
[:masterserver.ertw.com
Email Address []:sean@ertw.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

The generated file, `ldap.csr`, can be sent off to a CA (along with a hefty payment) to be signed. This is also the procedure you follow to generate a certificate for a Web server. If you do send this off for signing, make sure all the information you've provided is spelled correctly, that abbreviations are used only for the Country Name, and that the Common Name *exactly* matches the DNS name people will use to connect to your server.

Instead of getting a CA to sign the CSR, in this example you'll sign it yourself, as shown in Listing 16.

Listing 16. Signing the CSR

```
[root@bob ssl]# openssl x509 -req -days 1095 -in ldap.csr -signkey
```

```
ldap.key -out ldap.cert
Signature ok
subject=/C=CA/ST=Manitoba/L=Winnipeg/O=ERTW/OU=Directory Services/
CN=masterserver.ertw.com/emailAddress=sean@ertw.com
Getting Private key
```

Listing 16 signs the key with the `openssl x509` command. Using `-req` tells `openssl` that the input is a CSR. The validity of this certificate is 1095 days, or 3 years. Now you have `ldap.key` (the private key) and `ldap.cert` (the certificate and public key).

Before continuing, add a line to `/etc/openldap/ldap.conf` containing `TLS_REQCERT allow`. This tells the OpenLDAP client utilities to ignore the fact that they're seeing a self-signed certificate. Otherwise, the default settings deny the certificate as invalid.

Getting OpenLDAP to use the new key and certificate is easy. Assuming you stored the generated keys in `/etc/openldap/ssl/`, the lines in Listing 17 set up your server for TLS connections after you restart `slapd`.

Listing 17. Configuring slapd for SSL

```
TLSCertificateFile /etc/openldap/ssl/ldap.cert
TLSCertificateKeyFile /etc/openldap/ssl/ldap.key
```

The commands in Listing 17 point `slapd` to your certificate and private key. To test your new server, issue the `ldapwhoami -v -x -Z` command, which binds anonymously to your secure port. If you receive a "success" message, then everything is working correctly. Otherwise, the debugging information generated by `-v` will point you to the cause or any errors.

You can generate a client certificate, which is optional, the same way as the server certificate. Instead of Listing 17, use the `TLS_KEY` and `TLS_CERT` commands in `ldap.conf`, which set your client key and certificate, respectively. Client certificates are required only if you need to have the certificate itself identify the client.

Firewall considerations

LDAP uses TCP port 389, and LDAPS (LDAP over SSL) uses TCP port 636. If you have a firewall between your servers and your clients, these ports must be allowed through the firewall for connections to succeed. Clients always connect to servers, and, depending on your replication strategy, servers connect to other servers.

Linux iptables

If you have an `iptables`-based firewall on your LDAP server, you need to modify your rule set to allow the incoming connections. Generally, the commands in Listing 18

suffice.

Listing 18. Adding iptables rules to allow LDAP connections

```
iptables -A INPUT -p tcp --dport 389 -j ACCEPT
iptables -A INPUT -p tcp --dport 636 -j ACCEPT
```

Listing 18 works if your policy is simple. `-A INPUT` appends the rule to the `INPUT` table, where all incoming packets are checked. You may have to insert these rules at the top (use `-I INPUT` instead) or use your distribution's firewall tools to allow TCP ports 389 and optionally 636 if you need LDAPS connectivity.

If you're using your Linux firewall as a router, such that the clients are attached to another interface and the LDAP server is attached to another interface, you must use the `FORWARD` chain instead of `INPUT`. You may also want to specify the incoming interface with `-i`, such as `-i eth0` to indicate that only packets coming in `eth0` will be accepted. Once a packet has been accepted, the return packets are also accepted.

Protection through TCP Wrappers

One of the configuration options available when compiling OpenLDAP is `--enable-wrappers`, which links the resulting binaries against the TCP Wrappers libraries. The wrappers use two files, `/etc/hosts.allow` and `/etc/hosts.deny`, to permit or deny access to incoming clients.

First check to see if `slapd` uses TCP Wrappers with `ldd /usr/sbin/slapd | grep libwrap`. If anything is returned, then your binary is using TCP Wrappers. If not, you need to recompile with `--enable-wrappers` or use the `iptables` method shown earlier.

With wrappers support, you can deny everybody access by adding `slapd: ALL in /etc/hosts.deny`. You can then allow people in with `slapd: 192.168.1. ,127.0.0.1`, which lets anyone from the `192.168.1.0/24` network or the `localhost` connect. Note that connections denied through TCP Wrappers connect at first but are then disconnected automatically. Contrast this to a firewall, where the packet is dropped before it reaches `slapd`.

The format of `hosts.allow` and `hosts.deny` allows for many different ways to allow and deny connections; consult the `hosts_access(5)` manpage for all the details.

More on authentication

So far, the discussion of authentication has been limited to cleartext passwords defined in `slapd.conf` and simple authentication used between the client and the

server. Cleartext passwords can be solved with `slappasswd`. Enter `slappasswd` at the shell, and you're prompted for a password and then to verify that password. The output is a secure hash of the password, such as `{SSHA}oxmMsm9Ff/xVQ6zv+bgmMQjCUFL5x22+`. This hash is mathematically guaranteed not to be reversible, although given the hash, someone could guess repeatedly by trying various passwords and seeing if the hash is the same.

You've already experienced the anonymous bind, where no username or password has been provided, and the authenticated bind, where both the username and password are provided and valid. OpenLDAP also supports an unauthenticated bind, where the username is provided with no password. An unauthenticated bind is usually disabled unless you have `allow_bind_anon_cred` in your configuration. If allowed, an unauthenticated bind is considered anonymous.

The alternative to simple authentication is Simple Authentication and Security Layer (SASL), a framework for providing a plug-in architecture for authentication and encryption methods. A detailed look at SASL will appear in a forthcoming tutorial; in the meantime, SASL allows for different authentication methods from plaintext to Kerberos.

Earlier, when investigating ACLs, this tutorial mentioned that access can be influenced by the connection method. This is called the *Security Strength Factor* (SSF). An unencrypted session has an SSF of 0, and an encrypted session generally has an SSF corresponding to the key length. Thus, you can require an encrypted session for a particular ACL by adding `ssf=1` to the `who` clause of your ACL.

Section 5. LDAP server performance tuning

This section covers material for topic 303.5 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 2.

In this section, you learn how to do the following:

- Measure LDAP performance
- Tune software configuration to increase performance
- Understand indexes

OpenLDAP is a database. You provide it with a query or a task to run, and it finds the data and returns it to you. In order to do so as fast as possible, you must assign

resources to the places they will be best used, such as caching of frequently accessed data, and indexing databases.

Measure performance

Before you can try to make slapd run faster, you must be able to measure the current state. This may mean timing a certain operation in your application and looking for an improvement. It may also mean performing several queries yourself and calculating the average. The metric may not even be time based; it may be to reduce disk load on the LDAP server because the current configuration is causing too many reads and writes.

Either way, it's helpful to take several measurements of different metrics before and after any changes. Helpful commands are as follows:

- `vmstat` to show input/output (IO) statistics and CPU usage, notably the user time and wait time
- `iostat` to show more details about reads and writes to disk, along with the disk controller usage
- `ps` to show memory usage of the slapd process (not that using more memory is a bad thing, but it's important to make sure you don't run out of RAM)
- `time` for timing various command-line operations

Tune the daemon

Tuning always involves tradeoffs. Often, you increase the amount of resources (usually memory or disk) to a process to get the process to respond faster. Doing so decreases the resources that other processes can use. Similarly, if you make your process run faster, it often consumes more resources, such as CPU cycles or disk IO, that are unavailable to other processes.

Tradeoffs can also occur at the application level. By sacrificing some write performance, you're often able to drastically improve read performance. You can also make your application run faster by turning off various safety features such as transaction logging. Should you have a crash, you may end up restoring your database from a backup, but only you know if this is an acceptable tradeoff.

Most people use the Berkeley Database (BDB) backend. This backend is based on the Sleepycat Berkeley Database, now owned by Oracle, which is a fast embedded database. It doesn't support a query language; instead, it's based on hash-table lookups. Tuning this backend occurs in two places: one in `slapd.conf` and another in

a special file used by the BDB runtime.

slapd.conf configuration directives

The BDB database is linked against the binary that uses it, rather than being a standalone server like most SQL servers. As such, the application using the BDB database is responsible for some of the behavior of the database. The `slapd-bdb(5)` manpage describes all the directives that are controllable by slapd through `slapd.conf`; this tutorial covers only the most important.

Like many SQL backends, BDB databases write their changes to a transaction log to ensure reliability in the case of a failure; they also keep data in memory to save on disk writes. The operation that flushes all the in-memory data to disk and writes the transaction log out is called a *checkpoint*. The `checkpoint` command tells slapd how often to write out the data, in terms of kbytes of stored changes and minutes since the last checkpoint. Adding `checkpoint 128 15` to `slapd.conf` means that data will be flushed every 128KB of changes or at least every 15 minutes. By default, no checkpoint operations are performed which is the same as `checkpoint 0 0`.

Recently accessed entries can be cached in RAM for faster access. By default, 1000 entries are cached. To change this number, use `cachesize` along with the number of entries. The higher the `cachesize`, the more likely an entry is to be cached in RAM, but the more RAM is consumed by the slapd process. Your choice of value here depends on how many different entries are in your tree and the pattern of access. Make sure you have enough space in the cache to hold your commonly accessed items, such as the user list.

Similar to `cachesize` is `idlcachesize`, which has to do with how big the memory cache for indexes is. Your setting will depend on how many indexes you have configured (discussed later), but it's wise to start by making `idlcachesize` the same as `cachesize`.

Tune the BDB databases

As mentioned earlier, some of the tuning parameters for the BDB databases are handled in a separate file that is read by the BDB runtime and ignored by slapd. This file is called `DB_CONFIG`, and it lives in the same directory as your database. The most important parameter in that file is `set_cachesize`, which sets the internal BDB cache size, not the slapd entry cache. The format is `set_cachesize <GigaBytes> <Bytes> <Segments>;`, where `GigaBytes` and `Bytes` refer to the size of the cache (the two are added together), and `Segments` allows you to split the cache across separate memory blocks to get around 32-bit address limitations (both 0 and 1 have the same effect, allowing only a single memory segment). For a 1GB cache, use `set_cachesize 1 0 0`.

To determine the best BDB cache size, it's often easiest to look at the cache statistics of a working system and increase as needed. The command to look at the memory usage statistics of a BDB database is `db_stat -h /path/to/database -m`. The first 20 lines show the relevant details. If you have a high number of pages forced from the cache, or the number of pages found in the cache is low (< 95%), consider increasing the BDB cache size. On some distributions, `db_stat` may be called `slapd_db_stat` to separate it from the system BDB libraries and tools.

In addition to cache, you need to make sure the transaction logs are monitored. Set the path to the transaction logs with `set_lg_dir`. If you can put the transaction log on a different set of disk spindles than the database, you'll have much better performance.

Even though BDB is a simple database, it still needs to be able to lock files for writing. The default settings for the number of locks is usually large enough, but you should monitor the output of `db_stat -h /path/to/database -c` for any mention of hitting the maximum number of locks. In BDB, locks are split into three types (and reported separately): lockers, locks, and lock objects. The difference between them isn't important, but the maximum number of each is controlled through `set_lk_max_lockers`, `set_lk_max_locks`, and `set_lk_max_objects`, respectively.

Whenever you make changes to `DB_CONFIG`, you must restart `slapd`. Listing 19 shows a sample `DB_CONFIG` file based on the directives mentioned previously.

Listing 19. Sample `DB_CONFIG` file

```
# 256K cache
set_cachesize 0 268435456 0
set_lg_dir /var/log/openldap
set_lk_max_lockers 1000
set_lk_max_locks 1000
set_lk_max_objects 1000
```

Index your database

Most LDAP operations involve some sort of search on an attribute, such as a username, a phone number, or an email address. Without anything to help it, `slapd` must search each entry when performing a query. Adding an index to an attribute creates a file that lets `slapd` search much more quickly because the data in an index is stored in a way that allows for fast lookups. The tradeoff for an index is slower write speed and increased disk and memory usage. Therefore, it's best to index attributes that are searched on frequently.

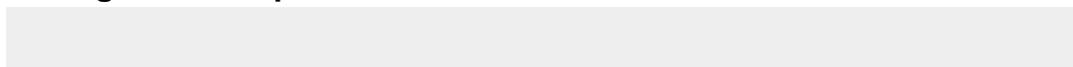
OpenLDAP supports several index types, depending on the type of search being performed. The index types are listed in Table 4.

Table 4. OpenLDAP index types

Type	keyword	Description	Search example
Presence	pres	Used for lookups that want to know whether an attribute exists.	uid=*
Equality	eq	Used for lookups that are looking for a specific value.	uid=42
Substring	sub	Used for lookups that are looking for a string somewhere within a value. Within this type, you can specify three other optimized types or use the generic sub type.	cn=Sean*
	subinitial	A substring index looking for a string at the beginning of a value.	cn=Sean*
	subany	A substring index looking for a string in the middle of a value.	cn=*jone*
	subfinal	A substring index looking for a string at the end of a value.	cn=*Smith
Approximate	approx	Used for sound-alike searches, where you want to find a value that sounds like your search string.	cn~=Jason

To apply an index to an attribute, use the syntax `index [attrlist] [indices]`, where `[attrlist]` is a comma-separated list of the attributes and `[indices]` is a comma-separated list of the index types from Table 4. You may have several index lines. Specifying `default` as the attribute list sets the type of indexes that are used when the list of indexes is empty. Consider the indexes defined in Listing 20.

Listing 20. A sample set of indexes



```
index default eq,sub
index entryUUID,objectClass eq
index cn,sn,mail eq,sub,subinitial,subany,subfinal
index userid,telephonenumber
index ou eq
```

Listing 20 first defines the default indexes as an equality and a basic substring match. The second line places an equality index on the `entryUUID` attribute (good for syncrepl performance) and the `objectClass` attribute (a common search). Line 3 places an equality index and all substring indexes on the `cn`, `sn`, and `mail` attributes because these fields often have different wildcard searches. The `userid` and `telephonenumber` attributes receive the default indexes because nothing more specific was entered. Finally, the `ou` attribute has an equality index.

After changing your index definitions, you must rebuild the indexes by stopping `slapd` and running `slapindex` as the `ldap` user (or, if you're running as root, make sure to reassign ownership of all the files in the database directory to the `ldap` user after running `slapindex`). Start up `slapd`, and your indexes are used.

Section 6.

This section covers material for topic 303.6 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 2.

In this section, you learn how to do the following:

- Understand `slapd.conf` configuration directives
- Understand `slapd.conf` database definitions
- Manage `slapd` and its command-line options
- Analyze `slapd` log files

The contents of `slapd.conf` have been largely covered earlier in this tutorial and in the previous tutorial. Of particular interest in this section are the command-line options and logging commands available to `slapd`.

Command-line options

The simplest way to start `slapd` is to run it without any arguments. `Slapd` then reads the default configuration file, forks to the background, and disassociates from the terminal.

Table 5 lists some helpful command-line arguments.

Table 5. slapd command-line arguments

Argument	Value	Description
-d	Integer	Runs slapd with extended debugging, and causes slapd to run in the foreground
-f	Filename	Specifies an alternate configuration file
-h	URL list	Specifies the addresses and ports that slapd listens on
-s	Sysloglevel	Specifies the syslog priority used for messages
-l	Integer	Specifies the local syslog facility (such as LOCAL4) used for messages
-u	Username	Runs slapd as the given user
-g	Groupname	Runs slapd in the given group

The URL list allows you to bind slapd to different interfaces. For example, `-h "ldap://127.0.0.1/ ldaps://"` causes slapd to listen on TCP port 389 (unencrypted LDAP) only on the loopback, and encrypted LDAP (TCP port 636) on all interfaces. You can even alter port numbers: for example, `ldap://:5565/` causes unencrypted LDAP to run on port 5565 of all interfaces.

Understand logging

Slapd uses the Unix syslog daemon for logging. By default, all messages are sent to the LOCAL4 facility, so you need at least `local4.* /var/log/openldap.log` in `syslog.conf` to capture the messages to `/var/log/openldap.log`. The `loglevel` command in `slapd.conf` then tells slapd what type of messages to log. Table 6 lists the possible message types.

Table 6. slapd message logging types

Keyword	Integer value	Description
trace	1	Trace function calls
packet	2	Debug packet handling
args	4	Heavy trace debugging

		(function args)
conns	8	Connection management
BER	16	Print out packets sent and received
filter	32	Search filter processing
config	64	Configuration file processing
ACL	128	ACL processing
stats	256	Stats log connections/operations/results
stats2	512	Stats log entries sent
shell	1024	Print communication with shell backends
parse	2048	Entry parsing
sync	16384	LDAPSync replication

You may use a space-separated list of keywords, integer values, or the sum of the integer values for the `loglevel` keyword. For example, `loglevel args ACL`, `loglevel 4 128`, and `loglevel 132` all enable debugging of function arguments and ACLs.

Section 7. Summary

In this tutorial, you learned about access control lists, replication, security, tuning, and more details about general configuration.

ACLs dictate who gets what access to what set of entries and attributes. You need to configure your ACLs using the form `access to <what> [by <who> [<access>] [<control>]]+`. You can use a variety of forms, including direct matches and regular expressions to specify the what. The who can also use matches and regular expressions, but it can also use keywords like `self`, `users`, and `anonymous`. The who clause can also look for things like the strength of the connection or the network the user is coming from.

Replication involves keeping a remote server up to date with the primary LDAP server. Two methods exist for replication: `slurpd` and `syncrepl`. In the `slurpd` model, a separate daemon runs on the master server and pushes all changes to the slaves. The slaves must start with a copy of the master's data; this requires downtime on the

master. In syncrepl, the provider (master) server runs an overlay to handle the replication tasks. The consumers (slaves) connect to the master and download any updates. If a consumer downloads updates on a periodic basis, it's said to be in refreshOnly mode. If the consumer downloads the updates and then stays connected, it's in refreshAndPersist mode, and it receives updates as they happen on the provider.

TLS and SSL let you encrypt communications between the client and server, and even replication traffic. You must generate server keys and have them signed by a CA for TLS to work. Regular LDAP traffic runs over TCP port 389, and encrypted LDAP traffic runs over TCP port 636, so you must have your firewalls configured accordingly.

Performance tuning involves assigning system resources to various caches and buffers and applying indexes on frequently searched columns. System resources are controlled in both the slapd.conf and DB_CONFIG files. Indexes can be equality, substring, presence, or approximate, depending on the type of search for which you're trying to optimize.

Most of slapd's behavior is controlled in slapd.conf, so there are only a few command-line parameters to control the addresses and ports that slapd listens on, the user it runs as, and some parameters about how it logs. What slapd logs is controlled by the `loglevel` directive in slapd.conf.

At this point, you have the skills to install, configure, and manage a functional OpenLDAP server, including security, replication, and performance tuning. The next two tutorials will focus on applications of LDAP, such as integrating LDAP with e-mail and authentication systems, and searching your tree from the command line.

Resources

Learn

- Review the previous tutorial in this 301 series, "[LPI exam 301 prep, Topic 302: Installation and development](#)" (developerWorks, December 2007), or [all tutorials in the 301 series](#).
- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Get the answer to [How do I determine the proper BDB database cache size?](#) in the OpenLDAP FAQ.
- Read about [Backus-Naur form](#) on Wikipedia for an in-depth understanding, including some examples. You may be familiar with BNF if you've looked into the LDAP Data Interchange Format (LDIF) or read some of the OpenLDAP manpages.
- The [OpenLDAP Administrator's Guide](#) has a chapter on [ACLs](#) that explains the syntax in detail. The `slapd.access(5)` manpage is a good companion to the Administrator's Guide.
- Also, see the chapters on [syncrepl](#) and [slurpd replication](#) in the [OpenLDAP Administrator's Guide](#). In particular, the guide has detailed descriptions of how both replication types work.
- [RFC 4533 \(The Lightweight Directory Access Protocol \(LDAP\) Content Synchronization Operation\)](#), spearheaded by the OpenLDAP Foundation and IBM, describes a method to synchronize LDAP servers more effectively than can be done with `slurpd`.
- [LDAP for Rocket Scientists](#), an online book, is excellent, despite being a work in progress.
- In the [developerWorks Linux zone](#), find more resources for Linux developers, and scan our [most popular articles and tutorials](#).
- See all [Linux tips](#) and [Linux tutorials](#) on developerWorks.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- The [Firewall Builder](#) utility makes the task of typing in iptables rules easy; it has a nice GUI and suite of tools to roll out updates to your firewalls.

- [OpenLDAP](#) is a great choice for an LDAP server.
- [phpLDAPadmin](#) is a Web-based LDAP administration tool. If the GUI is more your style, [Luma](#) is a good one to look at.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the [developerWorks community](#) through blogs, forums, podcasts, and community topics in our [new developerWorks spaces](#).

About the author

Sean A. Walberg

Sean Walberg has been working with Linux and UNIX since 1994 in academic, corporate, and Internet service provider environments. He has written extensively about systems administration over the past several years.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 301 prep, Topic 304: Usage

Senior Level Linux Professional (LPIC-3)

Skill Level: Intermediate

[Sean Walberg \(sean@ertw.com\)](mailto:sean@ertw.com)
Network engineer
Freelance

25 Mar 2008

In this tutorial, Sean Walberg helps you prepare to take the Linux Professional Institute Senior Level Linux Professional (LPIC-3) exam. In this fourth in a series of [series of six tutorials](#), Sean walks you through searching your LDAP tree and using the command-line tools. You'll also learn how to set up Microsoft Outlook to query your LDAP tree.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux® system administrators at three levels: *junior level* (also called "certification level 1"), *advanced level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102. To attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active advanced-level certification and pass exam 301 ("core"). You may also need to pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the five junior, advanced, and senior certification exams. Each exam covers several topics, and each topic has a

corresponding self-study tutorial on developerWorks. Table 1 lists the six topics and corresponding developerWorks tutorials for LPI exam 301.

Table 1. LPI exam 301: Tutorials and topics

LPI exam 301 topic	developerWorks tutorial	Tutorial summary
Topic 301	LPI exam 301 prep: Concepts, architecture, and design	Learn about LDAP concepts and architecture, how to design and implement an LDAP directory, and about schemas.
Topic 302	LPI exam 301 prep: Installation and development	Learn how to install, configure, and use the OpenLDAP software.
Topic 303	LPI exam 301 prep: Configuration	Learn how to configure the OpenLDAP software in detail.
Topic 304	LPI exam 301 prep: Usage	(This tutorial) Learn how to search the directory and use the OpenLDAP tools. See the detailed objectives .
Topic 305	LPI exam 301 prep: Integration and migration	Coming soon.
Topic 306	LPI exam 301 prep: Capacity planning	Coming soon.

To pass exam 301 (and attain certification level 3), the following should be true:

- You should have several years of experience with installing and maintaining Linux on a number of computers for various purposes.
- You should have integration experience with diverse technologies and operating systems.
- You should have professional experience as, or training to be, an enterprise-level Linux professional (including having experience as a part of another role).
- You should know advanced and enterprise levels of Linux administration including installation, management, security, troubleshooting, and maintenance.
- You should be able to use open source tools to measure capacity planning and troubleshoot resource problems.
- You should have professional experience using LDAP to integrate with UNIX® services and Microsoft® Windows® services, including Samba, Pluggable Authentication Modules (PAM), e-mail, and Active Directory.

- You should be able to plan, architect, design, build, and implement a full environment using Samba and LDAP as well as measure the capacity planning and security of the services.
- You should be able create scripts in Bash or Perl or have knowledge of at least one system programming language (such as C).

To continue preparing for certification level 3, see the [series developerWorks tutorials for LPI exam 301](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute doesn't endorse any third-party exam preparation material or techniques in particular.

About this tutorial

Welcome to "Configuration," the fourth of [six tutorials](#) designed to prepare you for LPI exam 301. In this tutorial, you learn how to search the directory, how to use the command-line tools for searching and administration, and how to configure third-party applications to use your LDAP tree as a Whitepages service.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weights.

Objectives

[Table 2](#) provides the detailed objectives for this tutorial.

Table 2. Configuration: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
304.1 Searching the directory	2	Use advanced options to search the LDAP directory
304.2 LDAP command-line tools	4	Use the various command-line tools to search, modify, and administer the LDAP server
304.3 Whitepages	1	Use your LDAP server as a Whitepages service for applications like Microsoft Outlook®

Prerequisites

To get the most benefit from this tutorial, you should have advanced knowledge of

Linux and a working Linux system on which to practice the commands covered.

If your fundamental Linux skills are a bit rusty, you may want to first review the [tutorials for the LPIC-1 and LPIC-2 exams](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

System requirements

To follow along with the examples in this tutorial, you'll need a Linux workstation with the OpenLDAP package and support for PAM. Most modern distributions meet these requirements.

Section 2. Searching the directory

This section covers material for topic 304.1 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 2.

In this section, learn how to:

- Use OpenLDAP search tools with basic options
- Use OpenLDAP search tools with advanced options
- Optimize LDAP search queries
- Use search filters and their syntax

The data in your tree is useful only if you can find entries when you need them. LDAP provides a powerful set of features that allow you to extract information from your tree.

The basics of search

To search a tree, you need four pieces of information:

1. Credentials on the server that holds the tree
2. A Distinguished Name (DN) on the tree to base your search on

3. A search scope
4. A search filter

You have already learned about server credentials in [previous tutorials](#) from this series. The credentials can be nothing, which results in an anonymous bind, or they can be the DN of an entry along with a password. Implicit in this is that the server recognizes these credentials as valid, and is willing to allow you to search!

The DN that you base your search on is called the *base DN*. All results will be either the base DN itself or its children. If your base DN is `ou=people,dc=ertw,dc=com`, then you might find `cn=SeanWalberg,ou=people,dc=ertw,dc=com`, but you won't find `cn=Users,ou=Groups,dc=ertw,dc=com`, because it lies outside the base DN you were trying to search.

The scope determines which entries under the base DN will be searched. You may want to limit the scope because of performance reasons, or because only certain children of the base DN contain the information you want. The default search scope, *subordinate* (usually abbreviated as *sub*), includes the base DN and all children. You can search only the base DN with a *base* scope, such as when you want to test to see if an entry exists. The search scope called *one* searches only the base DN's immediate children and excludes any grandchildren and the base DN itself. Figure 1 shows a tree and the entries that would be included in the three different search scopes.

Figure 1. Three different search scopes

The most powerful (and complex) part of searching is the search filter. The credentials, base DN, and scope limit which entries are to be searched, but it is the *query* that examines each record and returns only the ones that match your criteria.

Simple search filters

Search filters are enclosed in parentheses. Within the parentheses is one `attribute=value` pair. A simple search filter would be `(objectClass=inetOrgPerson)`, which will find any entries with an `objectClass` of `inetOrgPerson`. The attribute itself is not case sensitive, but the value may or may not be depending on how the attribute is defined in the schema. Recall from the first tutorial, [LPI exam 301 prep: Concepts, architecture, and design](#), that the schema defines the attributes and their properties. One property of the attribute is if comparisons are case sensitive or not.

Substring searches are performed with the asterisk (*) operator. Search for `(Sean*)` to match anything beginning with `Sean`. The asterisk can go anywhere in the string, such as `(*Walberg)` to find anything ending in `Walberg`, or even `S*Wa*berg` to find anything starting with `S`, ending in `berg`, and having `Wa`

somewhere in the middle. You might use this to find the author's name, not knowing if it is Sean or Shawn, or Walberg or Wahlberg.

The most generic form of the asterisk operator, `attribute=*` checks for the existence of the specified attribute. To find all the entries with defined e-mail addresses, you could use `(mail=*)`.

AND, OR, and NOT

You can perform logical AND and OR operations with the "&" and "|" operators respectively. LDAP search strings place the operator before the conditions, so you will see filters like those shown in Listing 1.

Listing 1. Sample search filters using AND and OR

```
(|(objectClass=inetOrgPerson)(objectClass=posixAccount))
(&(objectClass=*)(cn=Sean*)(ou=Engineering))
(&(|(objectClass=inetOrgPerson)(objectClass=posixAccount))(cn=Sean*))
```

The first search string in Listing 1 looks for anything with an `objectClass` of `inetOrgPerson` or `posixAccount`. Note that each component is still enclosed in parentheses, and that the OR operator (`|`) along with its two search options are also enclosed in another set of parentheses.

The second search string is similar, but starts with an AND operation instead of OR. Here, three different tests must be satisfied, and they all follow the ampersand in their own set of parentheses. The first clause, `objectClass=*`, matches anything with a defined `objectClass` (which should be everything, anyway). This search of all `objectClasses` is often used as a search filter when you want to match everything and are required to enter a filter. The second clause matches anything that starts with `Sean`.

The third search string shows both an AND and an OR used together in a filter that looks for anything with an `objectClass` of either `inetOrgPerson` or `posixAccount`, and a common name beginning with `Sean`.

The logical NOT is performed with the exclamation mark (`!`), much like the AND and OR. A logical NOT has only one argument so only one set of parentheses may follow the exclamation mark. Listing 2 shows some valid and invalid uses of NOT.

Listing 2. How to use, and how not to use, the logical NOT

```
(!cn=Sean)      # invalid, the ! applies to a filter inside ()
!(cn=Sean)     # valid
!(cn=Sean)(ou=Engineering) # invalid, only one filter can be negated
!(&cn=Sean*)(ou=Engineering)) # valid, negates the AND clause
```

In the fourth example of Listing 2, the negation is applied to an AND filter. Thus, that rule returns any entries that do not satisfy both of the AND clauses. Be careful when dealing with negation of composite filters, because the results are not always intuitive. The fourth example from Listing 2 will still return entries with an `ou` of `Engineering` if they don't have a common name starting with `Sean`. Both tests must pass for the record to be excluded.

Searching ranges

Often you need to search for a range of values. LDAP provides the `<=` and `>=` operators to query an attribute. Be careful that the equal sign (`=`) is included, because there are no `<` and `>` operators—you must also test for equality.

Not all integer attributes can be checked with the range operators. When in doubt, check the schema to make sure the attribute implements an ordering type through the `ORDERING` keyword. Recall from the first tutorial, [LPI exam 301 prep: Concepts, architecture, and design](#), that an attribute is defined in the schema, and part of the definition includes how the server should sort the attribute.

Searching for close matches

An LDAP directory is often used to store names, which leads to spelling problems. "Sean" can also be "Shawn" or "Shaun." LDAP provides a "sounds-like" operator, `~=`, which returns results that sound similar to the search string. For example, `(cn~=Shaun)` returns results that have a common name containing a word that sounds like "Shaun." Implicit in the sounds-like search is a substring search, such that `(cn=~Shaun)` will return results for `cn=Shawn Walberg`. The OpenLDAP implementation is not perfect, though; the same search will not return results for the "Sean" spelling.

Searching the DN

All the examples so far have focused on searching attributes, not searching on the distinguished name (DN) that identifies the record. Even though the leftmost component of the DN, the relative DN (RDN), appears as an attribute and is therefore searchable, the search filters presented so far will not look at the rest of the DN.

Searching the DN is done through a specific query filter requiring an exact match. The format is `attribute:dn:=value`, where the attribute is the component of the DN you want to search, and the value is the search string (no wildcards allowed). For example, `(ou:dn:=People)` would return all the entries that have `ou=People` in the DN, including the container object itself.

Altering the matchingRule

By default, most strings, such as the common name, are case insensitive. If you

want to override the matching rule, you can use a form similar to the DN search. A search such as `(ou:caseexactmatch:=people)` will match an organizational unit of "people", but not "People". Some common matching rules are:

- `caseIgnoreMatch` matches a string without regard for capitalization. Also ignores leading and trailing whitespace when matching.
- `caseExactMatch` is a string match that also requires similar capitalization between the two strings being searched.
- `octetStringMatch` is like a string match, but does not remove whitespaces, and therefore requires an exact, byte-for-byte, match.
- `telephoneNumberMatch` searches a telephone number, which has its own data type in LDAP.

You can also change the matching rule of a DN search by combining the DN search with the matching rule search. For example, `(ou:dn:caseexactmatch:=people)` searches for a DN containing the exact string "people".

These two types of searches, DN searches and matching rule searches, are also called *extensible searches*. They both require exact strings and do not allow wildcards.

Using ldapsearch

The command-line tool to search the tree is `ldapsearch`. This tool lets you bind to the directory in a variety of ways, execute one or more searches, and retrieve the data in LDIF format.

The default behavior of `ldapsearch` is:

- Attempt a Simple Authentication and Security Layer (SASL) authentication to the server
- Connect to the server at `ldap://localhost:389`
- Use `(objectClass=*)` as a search filter
- Read the search base from `/etc/openldap/ldap.conf`
- Perform a sub search; that is, include the search base and all children
- Return all user attributes, ignoring operational (internal usage) attributes
- Use extended LDAP Data Interchange Format (LDIF) for output

- Do not sort the output

Authenticating to the server

If you are not using SASL, then you need simple authentication using the `-x` parameter. By itself, `-x` performs an *anonymous bind*, which is a bind without any bind DN or password. Given the other defaults, `ldapsearch -x` will dump your entire tree, starting at the search base specified in `/etc/openldap/ldap.conf`. Listing 3 shows the usage of a simple anonymous search.

Listing 3. A simple anonymous search

```
$ ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# people, ertw.com
dn: ou=people,dc=ertw,dc=com
ou: people
description: All people in organization
objectClass: organizationalUnit
... output truncated ...
```

Listing 3 shows the header and first entry returned from a simple anonymous search. The first seven lines form the header, and in LDIF fashion, are commented starting with the hash mark (`#`). The first three lines identify the rest of the text as being extended LDIF and retrieved using LDAP version 3. The next line indicates that no base DN was specified and that a subtree search was used. The last two lines of text give the search filter which is everything and that all attributes were requested.

You may use the `-LLL` option to remove all the comments from your output.

After the header comes each entry; each entry starts with a header describing the entry and then the list of attributes, starting with the DN. Attributes are not sorted.

If you need to use a username and password to log in, use the `-D` and `-w` options to specify a bind DN and a password, respectively. For example, `ldapsearch -x D cn=root,dc=ertw,dc=com -w mypassword` will perform a simple authentication with the root DN username and password. You may also choose to type the password into a prompt that does not echo to the screen by using `-W` instead of `-w password`.

You may also connect to a different server by passing a Uniform Resource Identifier (URI) to the remote LDAP server using the `-H` option, such as `ldapsearch -x -H`

ldap://192.168.1.1/ to connect to an LDAP server at 192.168.1.1.

Performing searches

Append your search filter to your command line in order to perform a search. You will likely have to enclose the filter in quotes to protect special characters in the search string from being interpreted by the shell. Listing 4 shows a simple search on the common name.

Listing 4. A simple search from the command line

```
$ ldapsearch -LLL -x '(cn=Fred Smith)'  
dn: cn=Fred Smith,ou=people,dc=ertw,dc=com  
objectClass: inetOrgPerson  
sn: Smith  
cn: Fred Smith  
mail: fred@example.com
```

The search in Listing 4 uses the `-LLL` option to remove comments in the output and the `-x` option to force simple authentication. The final parameter is a search string that looks for Fred Smith's entry. Note that the parentheses are used around the search, and that single quotes are used both to protect the parentheses as being interpreted as a subshell invocation, and because the search string contains a space which would cause the "Smith" to be interpreted as a separate argument.

Listing 4 returned all of Fred Smith's attributes. It is a waste of both client and server resources to retrieve all values of a record if only one or two attributes are needed. Add the attributes you want to see to the end of the `ldapsearch` command line to only request those attributes. Listing 5 shows how the previous search looks if you only wanted Fred's e-mail address.

Listing 5. Requesting Fred Smith's e-mail address

```
$ ldapsearch -LLL -x '(cn=Fred Smith)' mail  
dn: cn=Fred Smith,ou=people,dc=ertw,dc=com  
mail: fred@example.com
```

The `mail` attribute is appended to the command line from Listing 4, and the result is the distinguished name of the record found, along with the requested attributes.

`ldapsearch` looks to `/etc/openldap/ldap.conf` for a line starting with `BASE` to determine the search base, and failing that, relies on the server's `defaultsearchbase` setting. The search base is the point on the tree where searches start from. Only entries that are children of the search base (and the search base itself) will be searched. Use the `-b` parameter to specify a different search base, such as `ldapsearch -x -b ou=groups,dc=ertw,dc=com` to search the groups container from the `ertw.com` tree.

Altering how data is returned

LDAP can store binary data such as pictures. The `jpegPhoto` attribute is the standard way to store a picture in the tree. If you retrieve the value of the attribute from the command line, you will find it is base64 encoded. The `-t` parameter is used to save any binary attributes into a temporary file. Listing 6 shows how to use this parameter.

Listing 6. Saving binary attributes on the file system

```
$ ldapsearch -LLL -x 'cn=joe*' jpegphoto | head
dn: cn=Joe Blow,ou=people,dc=ertw,dc=com
jpegPhoto:
/9j/4AAQSkZJRgABAQEASABIAAD//gAXQ3JlYXRlZCB3aXRoIFRoZSBHSU1Q/9sAQw
... output continues for 1300+ lines ...

$ ldapsearch -LLL -t -x '(cn=joe*)' jpegphoto
dn: cn=Joe Blow,ou=people,dc=ertw,dc=com
jpegPhoto:< file:///tmp/ldapsearch-jpegPhoto-VaIjkE

$ file /tmp/ldapsearch-jpegPhoto-VaIjkE
/tmp/ldapsearch-jpegPhoto-VaIjkE: JPEG image data, JFIF standard 1.01,
comment: \
"Created with The GIMP\377"
```

Listing 6 shows two searches for anyone with a name beginning with "Joe," and only retrieving the `jpegPhoto` attribute. The first try does not use the `-t` parameter, and therefore the value of `jpegPhoto` is shown on the console in base64 format. The usefulness of this is limited at the command line, so the second try specifies `-t` on the command line. This time the value of `jpegPhoto` is a URI to a file (you may change the directory with the `-T` option). Finally, the returned file is inspected, and indeed, it is the binary version of the picture that can be viewed.

By default, `ldapsearch` prints out results in the order they were received in from the server. You can sort the output with the `-s` parameter, passing it the name of the attribute you want to sort on. For sorting on multiple attributes, separate the attributes with a comma (,).

Section 3. LDAP command-line tools

This section covers material for topic 304.2 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 4.

In this section, learn how to:

- Use the `ldap*` tools to access and modify the directory
- Use the `slap*` tools to access and modify the directory

Several tools are provided with OpenLDAP to manipulate the directory and administer the server. You are already familiar with `ldapsearch`, which was covered in the [previous section](#). The commands beginning with `ldap` are for users of the tree, the commands beginning with `slap` are for administrators.

Tree manipulation tools

The commands in this section are for manipulating the tree, either by changing data or reading data. `ldapsearch` falls into this category, too. To use these commands, you need to authenticate to the server.

Idapadd and Idapmodify

These two commands are used to add and change entries in the tree. You may recall from the first tutorial, [LPI exam 301 prep: Concepts, architecture, and design](#) in the series that the LDAP Data Interchange Format (LDIF) can be used to add, change, and delete data from the tree. Listing 7 shows a sample of some LDIF to add an entry.

Listing 7. LDIF to add an entry to the tree

```
dn: cn=Sean Walberg,ou=people,dc=ertw,dc=com
objectclass: inetOrgPerson
cn: Sean Walberg
cn: Sean A. Walberg
sn: Walberg
homephone: 555-111-2222
```

Listing 7 begins with a description of the distinguished name of the entry. This entry will end up under the `ou=people,dc=ertw,dc=com` container, and have a relative distinguished name of `cn=Sean Walberg`, which is obtained by splitting the distinguished name (DN) after the first attribute/value pair. The entry has an `objectclass` of `inetOrgPerson`, which is a fairly generic type for any person belonging to an organization. Two variants of the common name follow, then a surname, and finally, a home phone number.

Implicit in Listing 7 is that this is an addition to the tree, as opposed to a change or deletion. Recall that LDIF files can specify the `changetype` keyword, which tells the reader what to do with the data.

The `ldapadd` command is used to process this LDIF file. If Listing 7 were stored as `"sean.ldif"`, then `ldapadd -x -D cn=root,dc=ertw,dc=com -w mypass -f`

`sean.ldif` would be one way to add the new entry to the tree. The `-x -D cn=root,dc=ertw,dc=com -w mypass` part of the command should be familiar from the earlier discussion of `ldapsearch`, as a way to authenticate to the tree with simple authentication and the all-powerful root DN. All the `ldap` commands in this section use the same parameters to authenticate to the tree, so you will see this form repeated.

`ldapadd` is implemented as a symbolic link to `ldapmodify`, and when called as `ldapadd` is interpreted as `ldapmodify -a`. The `-a` parameter tells `ldapmodify` to assume a default `changetype` of `add`, which is used to add new entries to the tree. When called as `ldapmodify`, the assumption is that the default `changetype` is `modify` operation.

`ldapadd` (and `ldapmodify`) is an efficient way of loading bulk data into a server without shutting it down. LDIF files can contain many operations, and often it is easier to generate LDIF from whatever other data source you are trying to import than to write custom code to parse the data source and add it directly through LDAP.

ldapdelete

`ldapdelete`, like the name implies, deletes an entry from the tree. All entries are uniquely identified in the tree by their DN; therefore, `ldapdelete` deletes entries by DN, and not by any other query.

Besides the authentication parameters already discussed, `ldapdelete` can take its list of DNs to delete either from the command line or from a file. To delete from the command line, simply append the DNs to your command line, such as `ldapdelete -x -D cn=root,dc=ertw,dc=com -w mypass "cn=Sean Walberg,ou=people,dc=ertw,dc=com"`. If you have many entries to delete, you can place the DNs, one per line, in a file, and point `ldapdelete` to that file with `-f filename`.

Note that you can also delete entries through LDIF and the `ldapadd/ldapmodify` commands. The `ldapdelete` command is more convenient in many cases, but is not the only way of deleting entries.

ldapmodrdn

The `ldapmodrdn` command changes the relative distinguished name of the object, that is, the first attribute/value pair in the DN. This effectively renames the entry within the current branch of the tree. Unlike the LDIF `moddn changetype`, this command can only rename the entry, and cannot move it to another spot on the tree.

Usage of this command is simple: give it the authentication credentials, the DN of the entry, and the new RDN. Listing 8 shows an account being renamed from "Joe Blow" to "Joseph Blow".

Listing 8. Renaming an entry

```
$ ldapmodrdn -x -D cn=root,dc=ertw,dc=com -w dirtysecret \  
    'cn=Joe Blow,ou=people,dc=ertw,dc=com' 'cn=Joseph Blow'  
$ ldapsearch -LLL -x '(cn=Joseph Blow)'  
dn: cn=Joseph Blow,ou=people,dc=ertw,dc=com  
objectClass: inetOrgPerson  
sn: Blow  
cn: Joe Blow  
cn: Joseph Blow
```

Note that the old RDN still appears as an attribute, that is, `cn: Joe Blow`. If you want the old RDN to be removed, add `-r` to your command line. This is the same as adding `deleteolddn: 1` to your LDIF code (which, curiously, is the default behavior for LDIF but not `ldapmodrdn`).

ldapcompare

`ldapcompare` allows you to compare a predetermined value to the value stored somewhere in the LDAP tree. An example will show how this works.

Listing 9. Using ldapcompare

```
$ ldapcompare -x "cn=Sean Walberg,ou=people,dc=ertw,dc=com" userPassword:mypassword  
TRUE  
$ ldapcompare -x "cn=Sean Walberg,ou=people,dc=ertw,dc=com" userPassword:badpassword  
FALSE
```

In Listing 9, the `ldapcompare` command is run. After the authentication parameters are taken care of, the final two parameters are the DN to check and the attribute and value to check against. The DN in both the examples above is the listing for "cn=Sean Walberg". The attributes being checked in both cases are the `userPassword` attribute. When the proper password is given, `ldapcompare` prints the string `TRUE` and an error code of 6. If the value given doesn't match what's in the entry, then `FALSE` is sent to the console, and an error code of 5 is returned. The `-z` option prevents anything from being printed; the caller is expected to use the error code to determine if the check was successful or not.

Even though the example in Listing 9 checked a password, any attribute can be used, including `objectClass`. If the attribute has multiple values, such as multiple common names or `objectClasses`, then the comparison is successful if one of them matches.

ldapwhoami

`ldapwhoami` allows you to test authentication to the LDAP server and to determine which DN you are authenticated against on the server. Simply call `ldapwhoami` with the normal authentication parameters, as shown in Listing 10.

Listing 10. A demonstration of ldapwhoami

```
$ ldapwhoami -x
anonymous
Result: Success (0)
$ ldapwhoami -x -D "cn=Sean Walberg,ou=people,dc=ertw,dc=com" -w
mypassword
dn:cn=Sean Walberg,ou=people,dc=ertw,dc=com
Result: Success (0)
$ ldapwhoami -x -D "cn=Sean Walberg,ou=people,dc=ertw,dc=com" -w badpass
ldap_bind: Invalid credentials (49)
```

The first example in Listing 10 shows a bind with no username or password. `Ldapwhoami` returns the string `anonymous` to indicate an anonymous bind, and also a status line indicating that the authentication was successful. The second example binds as a user's DN. This time the DN returned is the same one that was authenticated with. Finally, a bind attempt is made with invalid credentials. The result is an explanation of the problem.

`Ldapwhoami` is helpful for troubleshooting the configuration of the server, and also for manually verifying passwords. Access lists might get in the way of an `ldapsearch`, so using `ldapwhoami` instead can help you determine if the problem is credentials or access lists.

Administration tools

The commands beginning with `slap` are for administrators, and operate directly on the database files rather than through the LDAP protocol. As such, you will generally need to be root to use these commands, and in some cases, the server must also be shut down.

slapacl

`Slapacl` is a utility that lets the administrator test access lists against various combinations of bind DN, entry, and attribute. For instance, you would use `slapacl` to test to see what access a particular user has on another user's attributes. This command must be run as root because it is reading the database and configuration files directly rather than using LDAP.

The usage of `slapacl` is best described through an example. In Listing 11, the administrator is testing to see what access a user has on his own password before implementing an ACL, and then again after implementing an ACL that is supposed to limit the access to something more secure.

Listing 11. Using slapacl to determine the effect of an ACL change

```
# slapacl -D "cn=Sean Walberg,ou=people,dc=ertw,dc=com" \
```

```
    -b "cn=Sean Walberg,ou=People,dc=ertw,dc=com" userPassword
authcDN: "cn=sean walberg,ou=people,dc=ertw,dc=com"
userPassword: read(=rscxd)

... change slapd.conf ...

# slapacl -D "cn=Sean Walberg,ou=people,dc=ertw,dc=com" \
    -b "cn=Sean Walberg,ou=People,dc=ertw,dc=com" userPassword
authcDN: "cn=sean walberg,ou=people,dc=ertw,dc=com"
userPassword: =wx

# slapacl -D "cn=Joseph Blow,ou=people,dc=ertw,dc=com" \
    -b "cn=Sean Walberg,ou=People,dc=ertw,dc=com" userPassword
authcDN: "cn=joseph blow,ou=people,dc=ertw,dc=com"
userPassword: =0
```

Two pieces of information are mandatory for the `slapacl` command. The first is the bind DN, which is the DN of the user you are testing access for. The second piece is the DN of the entry you are testing against. The bind DN is specified with `-D`, and the target DN is set with `-b`. You can optionally limit the test to a single attribute by including it at the end (like the `userPassword` example in Listing 11). If you don't specify an attribute, you will receive results for each attribute in the entry.

In the first command from Listing 11, the administrator is testing the `cn=Sean Walberg` entry to see what access he has against his own password. The result is read access. Recall from the third tutorial in this series, [LPI exam 301 prep: Configuration](#), that users should be able to write and authenticate against their `userPassword` attribute, but not read it. After changing the ACLs, the test is performed again, and only the write and authenticate permissions are available. Finally, a test is performed to see what access Joseph Blow has on Sean Walberg's password; the result is that he has no access.

`Slapacl` is an effective way to test the results of ACL changes, and to debug ACL problems. It is particularly effective because it reads directly from the database and `slapd.conf`, so any changes made to `slapd.conf` are reflected in the output of `slapacl` and don't require a restart of `slapd`.

slapcat

`Slapcat` dumps the contents of the LDAP tree as LDIF to the standard output, or to a file if you use `-l filename`. You can optionally use the `-s` option to provide the starting DN, or `-a` to pass a query filter.

`Slapcat` operates directly on the database, and can be run while the server is still running. Only the `bdb` database types are supported.

slapadd

`Slapadd` is a bulk import tool that operates directly on the backend databases, which means `slapd` must be stopped to use this tool. It is designed to be used with the output from `slapcat`. `Slapadd` doesn't perform much validation on the input

data, so it is possible to end up with branches of the tree that are separated. This would happen if some container objects weren't imported.

The input to `slapadd` is an LDIF file, such as the one generated by `slapcat`. The `slapadd(8C)` manpage suggests using `ldapadd` instead because of the data validation provided by the online variant. The manpage also notes that the output of `slapcat` is not guaranteed to be ordered in a way that is compatible with `ldapadd` (the container objects may come after the children in the output, and hence would fail validation). Using any filters in `slapcat` may also cause important data to be missing. Therefore, you should use `slapadd` only with LDIF produced by `slapcat`, and use `ldapadd` for any other LDIF.

After shutting down your LDAP server, you can just run the `slapadd` command and pipe your LDAP output to the standard input. If you want to read from a file, use the `-l` option. As with `slapcat`, only the bdb database types are supported.

slappasswd

`Slappasswd` is used to generate hashed passwords to be stored in the directory, or in `slapd.conf`. A common use is to use a hashed password for the rootdn's account in `slapd.conf` so that anyone looking at the configuration file can not determine the password. `Slappasswd` will prompt you for a password to hash if you don't provide any parameters, as shown in Listing 12.

Listing 12. Using `slappasswd` to hash a password

```
$ slappasswd
New password:
Re-enter new password:
{SSHA}G8Ly2+t/HMHJ3OWWE7LN+GRmZJAweXoE
```

You may then copy the entire string to the `rootpw` line in `slapd.conf`. `Slapd` recognizes the format of the password and understands that `{SSHA}` means that what follows is a SHA1 hash. Anyone who reads `slapd.conf` will not learn the root password.

The hashes generated by `slappasswd` can also be used in LDIF files used with `ldapadd` and `ldapmodify`, which will allow you to store secure one-way hashes of your password instead of a less secure plaintext or base64-encoded version.

slapindex

You may recall `slapindex` from the third tutorial in this series, [LPI exam 301 prep: Configuration](#). After creating or changing an index with the `index` keyword in `slapd.conf`, you must rebuild your indexes, or `slapd` will return incorrect results. To rebuild your indexes, stop `slapd` and run `slapindex`. This may take a while depending on how many entries are in your databases, or as the manpage puts it,

"This command provides ample opportunity for the user to obtain and drink their favorite beverage."

slaptest

`Slapdtest` simply checks to see if your `slapd.conf` file is correct. This is helpful because if you were to restart `slapd` with a bad configuration file, it would fail to start up until you fixed the file. `Slaptest` lets you perform a sanity check on your configuration file before restarting.

Using `slaptest` is as simple as typing `slaptest`. If the `slapd.conf` is correct, you will see `config file testing succeeded`. Otherwise, you will receive an error explaining the problem.

`Slaptest` also checks for the existence of various files and directories necessary for operation. During testing however, the author was able to find some configuration file errors that passed `slaptest`, but would still cause `slapd` to fail.

Section 4. Whitepages

This section covers material for topic 304.3 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to do:

- Plan whitepages services
- Configure whitepages services
- Configure clients to retrieve data from whitepages services

A *whitepages service* allows e-mail clients to retrieve contact information from an LDAP database. By staying with common attribute names, such as those provided by the `inetOrgPerson` `objectClass`, you can get the most compatibility with e-mail clients. For example, both Microsoft Outlook and Evolution use the `mail` attribute to store the user's e-mail address, and the `givenName`, `displayName`, `cn`, and `sn` attributes to store various forms of the name.

Configuring e-mail clients for an LDAP directory

In theory, any mail client that supports LDAP can use your tree. You will need the following information configured in the client:

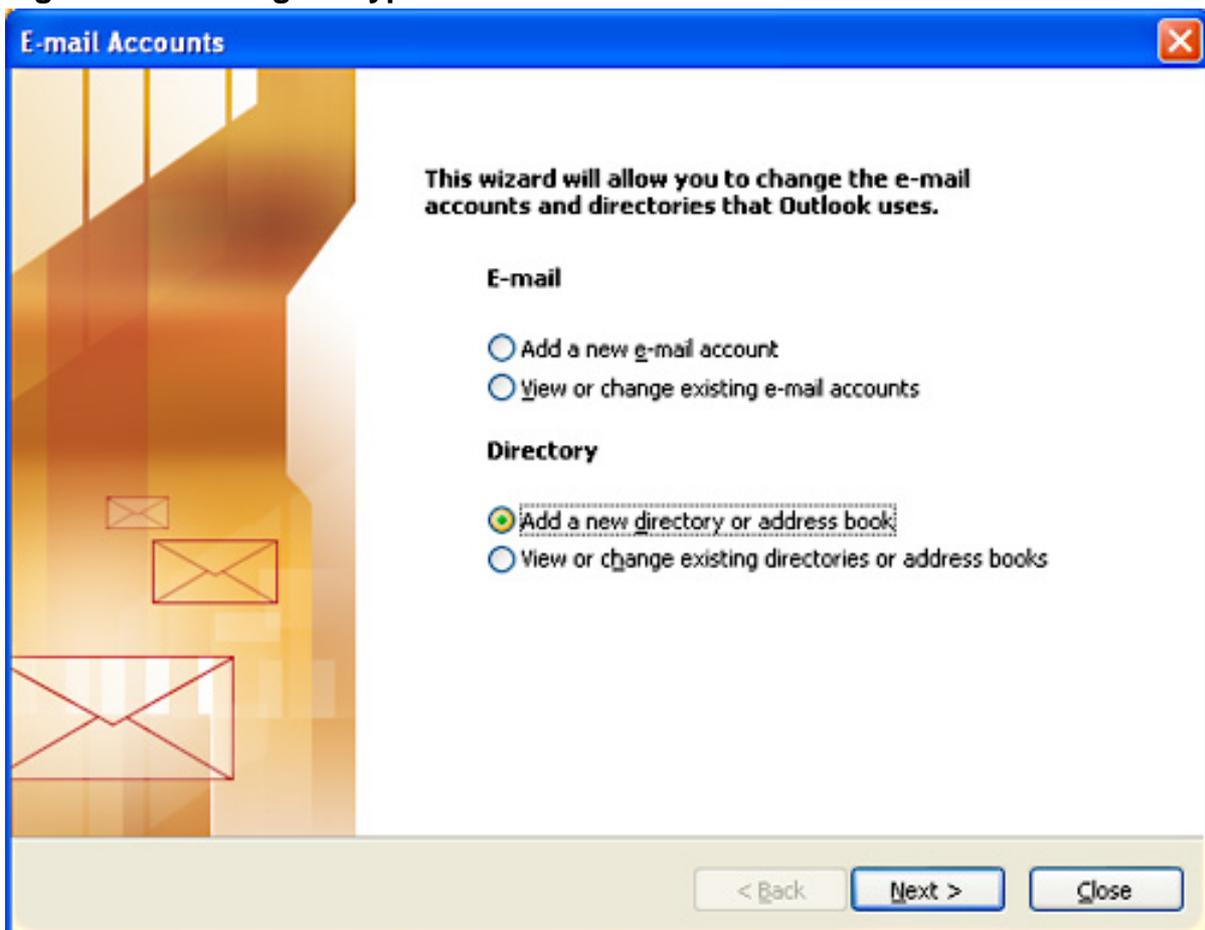
- The LDAP server's address or hostname
- Credentials to bind with, unless you are binding anonymously
- The base DN to search from
- A search filter, such as (mail=*), to weed out accounts without an e-mail address (optional)

Once you input the above information into your e-mail client, you should be able to search for contacts.

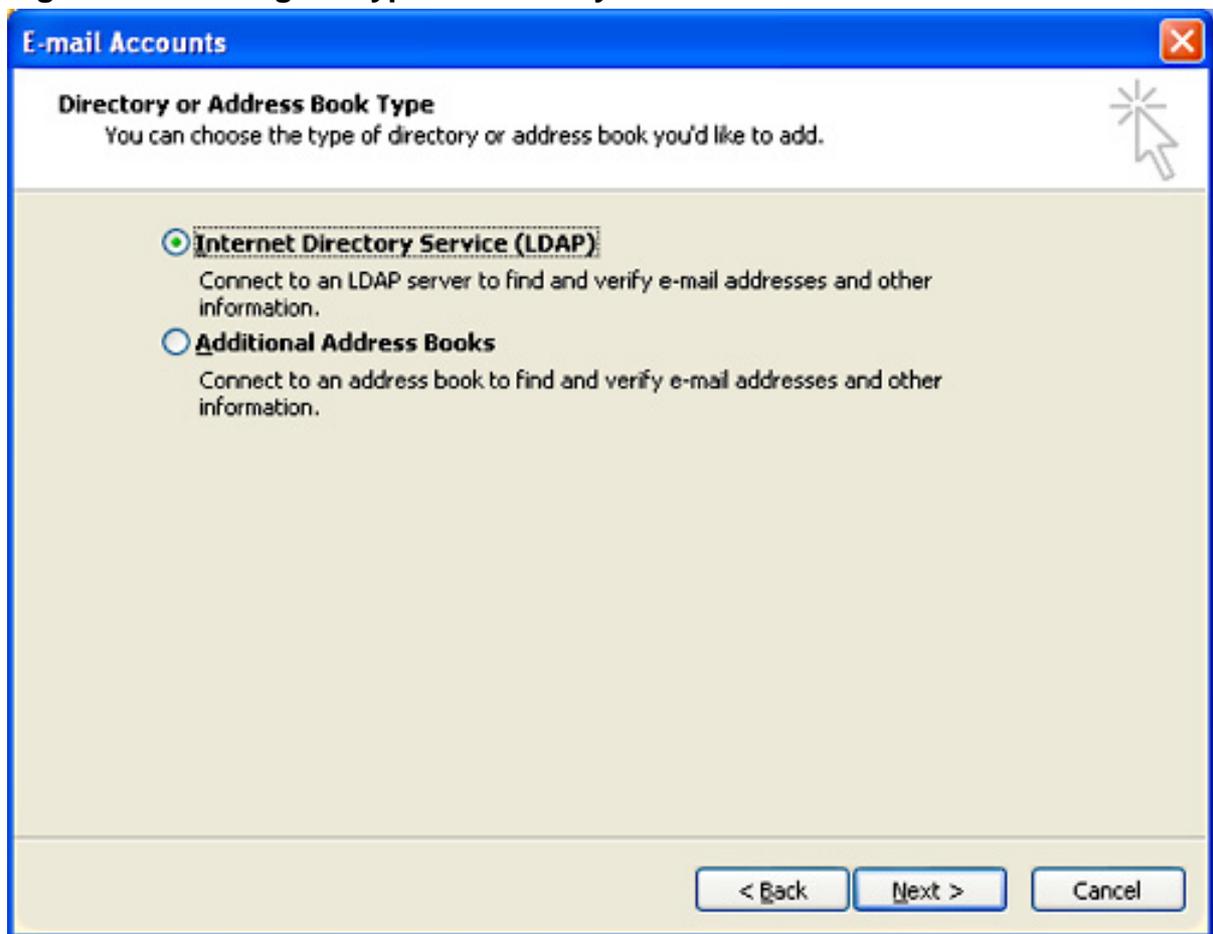
Configuring Microsoft Outlook for an LDAP directory

To configure Microsoft Outlook (tested on Outlook 2003), select **Tools > Email Accounts**. You will see a dialog similar to Figure 2.

Figure 2. Selecting the type of account to add



Select the option to add a new directory, and click **Next**. You will then see the dialog in Figure 3.

Figure 3. Selecting the type of directory to add

Select the option to add a new LDAP directory, and click **Next**. You will then see the dialog in Figure 4.

Figure 4. Specifying the LDAP server details

E-mail Accounts

Directory Service (LDAP) Settings
You can enter the required settings to access information in a directory service.

Server Information
Type the name of the directory server your Internet service provider or system administrator has given you.

Server Name: 192.168.1.1

Logon Information

This server requires me to log on

User Name: cn=seanwalberg,ou=people,dc=

Password: *****

Log on using Secure Password Authentication (SPA)

More Settings ...

< Back Next > Cancel

Enter the relevant details about your LDAP server in the dialog shown in Figure 4. The example shown uses user credentials to bind to the tree. You can use anonymous access if your server's configuration supports it.

After entering in the basic details, click **More Settings**, and you will be prompted for more information, as shown in Figure 5.

Figure 5. Adding advanced options to the LDAP server configuration

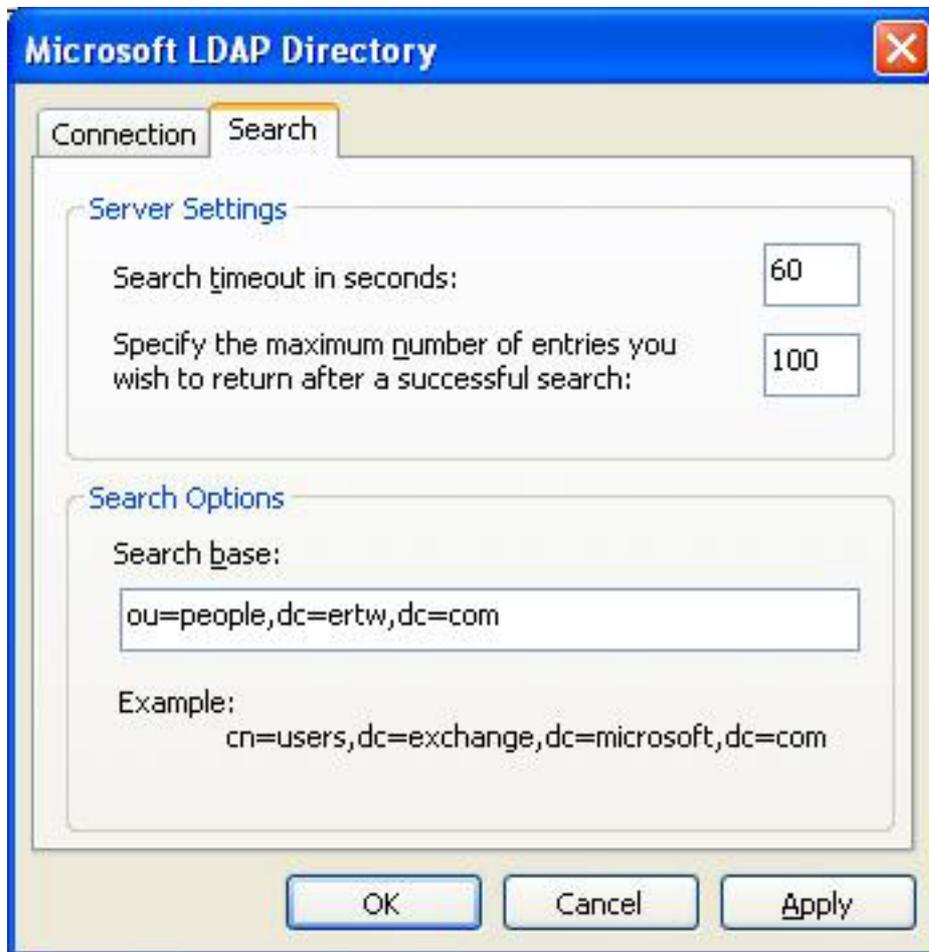


Figure 5 shows more options, the important one being the search base. Click **OK** after entering the search base, and you will be returned to the main Outlook screen.

You may now use the LDAP database wherever you are prompted to look for users by selecting the server name from the "Show Names From" field.

Section 5. Summary

In this tutorial you learned how to use your directory through searching and the command-line tools. You also learned how to configure e-mail clients to use the directory to store contact information.

Searching the LDAP tree requires you to build a query filter. The various operators that are used in a query are outlined in Table 3.

Table 3. LDAP Search Operators

Operator	Description	Example
=	Tests for equality	(cn=Walberg)
*	Tests for the existence of an attribute	(cn=*)
	Substring search	(sn=Walb*)
&	Logical AND	(&(condition1)(condition2))
	Logical OR	((condition1)(condition2))
!	Logical NOT	(!(mail=*))
~=	"Sounds-like" match	(cn~=Shawn)
<= and >=	Range match	(pagesPerMinute >= 20)

Several utilities are provided to use the directory, such as `ldapsearch` for searching, and `ldapadd` and `ldapmodify` for adding and changing data. The tools that start with `ldap` operate through the LDAP protocol and require credentials to log in to the server. The tools that start with `slap` are used by the administrator and operate directly on the database.

This tutorial and the previous ones in the [301 series](#) have focused on managing and working with an LDAP server. The next tutorial in the series will look at various applications including e-mail servers, and show how to use LDAP as the data source.

Resources

Learn

- Review the previous tutorial in this 301 series, "[LPI exam 301 prep, Topic 303: Configuration](#)" (developerWorks, March 2008), or [all tutorials in the 301 series](#).
- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In [RFC 4515 - The String Representation of LDAP Search Filters](#), find more info on search filters.
- Read "[MS Outlook: What LDAP Attributes Are Recognised?](#)" and "[MS Outlook: How Do LDAP Attributes Map to Address Book Fields?](#)" to see the results of reverse-engineering the attributes that are used by the Microsoft Outlook client.
- "[LDAP for Rocket Scientists](#)" is excellent, despite being a work in progress.
- Learn more about [matching rules](#).
- In the [developerWorks Linux zone](#), find more resources for Linux developers, and scan our [most popular articles and tutorials](#).
- See all [Linux tips](#) and [Linux tutorials](#) on developerWorks.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- The [Firewall Builder](#) utility makes the task of typing in iptables rules easy; it has a nice GUI and suite of tools to roll out updates to your firewalls.
- [OpenLDAP](#) is a great choice if you're looking for an LDAP server.
- [phpLDAPadmin](#) is a Web based LDAP administration tool. If a GUI is more your style, [Luma](#) is a good one to look at.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the [developerWorks community](#) through blogs, forums,

podcasts, and community topics in our [new developerWorks spaces](#).

About the author

Sean Walberg

Sean Walberg has been working with Linux and UNIX since 1994 in academic, corporate, and Internet service provider environments. He has written extensively about systems administration over the past several years.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 301 prep, Topic 305: Integration and migration

Senior Level Linux Professional (LPIC-3)

Skill Level: Intermediate

[Sean Walberg \(sean@ertw.com\)](mailto:sean@ertw.com)

Network engineer

Freelance

08 Apr 2008

In this tutorial, Sean Walberg helps you prepare to take the Linux® Professional Institute Senior Level Linux Professional (LPIC-3) exam. In this fifth in a [series of six tutorials](#), Sean walks you through integrating LDAP with your system's logins and applications. He also details the procedure to integrate your server into a foreign Microsoft® Active Directory.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *advanced level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102. To attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active advanced-level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the five junior, advanced, and senior certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. Table 1 lists the six topics and corresponding developerWorks tutorials for LPI exam 301.

Table 1. LPI exam 301: Tutorials and topics

LPI exam 301 topic	developerWorks tutorial	Tutorial summary
Topic 301	LPI exam 301 prep: Concepts, architecture, and design	Learn about LDAP concepts and architecture, how to design and implement an LDAP directory, and about schemas.
Topic 302	LPI exam 301 prep: Installation and development	Learn how to install, configure, and use the OpenLDAP software.
Topic 303	LPI exam 301 prep: Configuration	Learn how to configure the OpenLDAP software in detail.
Topic 304	LPI exam 301 prep: Usage	Learn how to search the directory and use the OpenLDAP tools.
Topic 305	LPI exam 301 prep: Integration and migration	(This tutorial) Learn how to use LDAP as the source of data for your systems and applications. See the detailed objectives .
Topic 306	LPI exam 301 prep: Capacity planning	Coming soon.

To pass exam 301 (and attain certification level 3), the following should be true:

- You should have several years of experience with installing and maintaining Linux on a number of computers for various purposes.
- You should have integration experience with diverse technologies and operating systems.
- You should have professional experience as, or training to be, an enterprise-level Linux professional (including having experience as a part of another role).
- You should know advanced and enterprise levels of Linux administration including installation, management, security, troubleshooting, and maintenance.
- You should be able to use open source tools to measure capacity planning and troubleshoot resource problems.

- You should have professional experience using LDAP to integrate with UNIX® services and Microsoft® Windows® services, including Samba, Pluggable Authentication Modules (PAM), e-mail, and Active Directory.
- You should be able to plan, architect, design, build, and implement a full environment using Samba and LDAP as well as measure the capacity planning and security of the services
- You should be able create scripts in Bash or Perl or have knowledge of at least one system programming language (such as C).

To continue preparing for certification level 3, see the [series developerWorks tutorials for LPI exam 301](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute doesn't endorse any third-party exam preparation material or techniques in particular.

About this tutorial

Welcome to "Integration and migration," the fifth of six tutorials designed to prepare you for LPI exam 301. In this tutorial, you'll learn all about integration of LDAP with authentication and other UNIX services.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weights.

Objectives

Table 2 provides the detailed objectives for this tutorial.

Table 2. Integration and migration: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
305.1 LDAP integration with PAM and NSS	2	Integrate the core system authentication with LDAP.
305.2 NIS to LDAP migration	1	Plan and implement a NIS migration strategy, including the deployment of a NIS to LDAP gateway.
305.3 Integrating LDAP with UNIX services	1	Use your LDAP server as the source of data for SSH, FTP, HTTP, and other services.
305.4 Integrating LDAP with Samba	1	Use your LDAP server as the source of data for Samba.

305.5 Integrating LDAP with Active Directory	2	Use your LDAP server alongside an Active Directory service.
305.6 Integrating LDAP with e-mail services	1	Integrate your e-mail services with your LDAP directory.

Prerequisites

To get the most from this tutorial, you should have advanced knowledge of Linux and a working Linux system on which to practice the commands covered.

If your fundamental Linux skills are a bit rusty, you may want to first review the [tutorials for the LPIC-1 and LPIC-2 exams](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

System requirements

To follow along with the examples in these tutorials, you'll need a Linux workstation with the OpenLDAP package and support for PAM. Most modern distributions meet these requirements.

Section 2. LDAP integration with PAM and NSS

This section covers material for topic 305.1 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 2.

In this section, learn how to:

- Configure NSS to retrieve information from LDAP
- Configure PAM to use LDAP for authentication
- Configure PAM modules in various UNIX environments

In traditional UNIX fashion, PAM and the Name Service Switch (NSS) facilities abstract various components of authentication and lookup from their implementation, which allows the administrator to change backend data stores without recompiling

any applications. For instance, moving from traditional `/etc/passwd`-based authentication to the Network Information Service (NIS) is transparent because NSS is implemented as part of the C library. Applications make use of the standard library calls such as `getpwent(3)` to look up users, but through some configuration magic, the data is redirected to another store like NIS.

PAM is a slightly different animal, because applications must be written specifically with PAM in mind. Administrators can use a rich set of libraries to customize the behavior of a PAM-aware application, such as requiring specific group membership and a login time in order to successfully authenticate.

PAM and NSS can work in tandem for user authentication. PAM-aware applications instruct PAM to check the user's credentials. The administrator can configure PAM to check the password through the NSS facility in addition to any other restrictions. PAM is used only for the password and shadow databases, not others like groups and hosts.

LDAP support for both PAM and NSS is provided by an open source package from PADL software.

Configure NSS to use LDAP

The NSS facility is implemented in the C library as a hook to traditional library calls to get information. The C library provides functions like `getpwent` to get user information and `gethostbyname(3)` for host information, which traditionally were implemented as lookups to `/etc/passwd` and `/etc/hosts`, respectively. The administrator can force hostname lookups to also use the Domain Name Service (DNS) by configuring NSS, meaning the application is unaware of the change.

Understand NSS

Table 3 outlines the databases that are handled by NSS. Most databases have a corresponding file in `/etc`, where the data is traditionally stored.

Table 3. NSS databases

Database name	Description
<code>aliases</code>	Mail aliases for sendmail, used to forward (alias) one local address to another address.
<code>ethers</code>	Maps ethernet addresses to IP addresses. Rarely seen anymore because of the availability of the Address Resolution Protocol (ARP).
<code>group</code>	Contains a list of groups and the users that belong to them.

hosts	Maps IP addresses to host names.
netgroup	Used to group servers together. Most often used for NIS and Network File System (NFS) security.
networks	A map of network names to numbers. Not often used because knowing the name of the network provides little value.
passwd	Stores user account information such as name, Userid, description, primary group, home directory, and sometimes a password.
protocols	Maps IP protocols to their name.
publickey	Used to distribute keys for NFS and NIS+.
rpc	Maps Remote Procedure Call (RPC) function names to numbers.
services	Maps TCP and UDP service names to the port number.
shadow	A protected, encrypted, password file. Usually the password field from /etc/passwd is stored in this file to keep it safe.

NSS is configured in /etc/nsswitchconf and contains one line per database from Table 3. Listing 1 shows a sample nsswitch.conf.

Listing 1. Sample nsswitch.conf

```
passwd:    files nis
shadow:   files nis
group:    files nis
hosts:    files nis dns
```

Listing 1 configures four maps: passwd, shadow, group, and hosts. The name of the map is followed by a colon (:) and then an ordered list of ways to access the data. The first three lines in Listing 1 are all the same: they first check the files for the requested information and then the NIS, sometimes known as the *Yellow Pages*. NIS is checked only if nothing is found in the files. The final line of the example checks the files (/etc/hosts), NIS, and then DNS for any hosts requests.

The methods available to be used in nsswitch.conf have a corresponding library in /lib that begins with libnss_. The functionality for files, for example, is found in /lib/libnss_files-2.5.so (the version number isn't important because it's resolved by the dynamic linker, ld-linux.so).

Introducing LDAP to NSS

After the previous discussion about dynamic libraries and the format of `nsswitch.conf`, it should come as no great surprise that LDAP integration with NSS is handled through a shared library called `libnss_ldap` and is referenced through the `ldap` keyword in `/etc/nsswitch.conf`. This shared library takes its configuration from `/etc/ldap.conf` (not to be confused with the OpenLDAP configuration file for the command-line clients, `/etc/openldap/ldap.conf`). Listing 2 shows a sample `ldap.conf`.

Listing 2. A sample `ldap.conf` to configure `libnss_ldap`

```
# Server IP address (or space-separated addresses)
host 192.168.1.138
# Search base
base dc=ertw,dc=com
# optional: bind credentials
binddn: cn=nssldap,dc=ertw,dc=com
bindpw: letmein
# If root is making the request, use this dn instead
# The password is stored in /etc/ldap.secret and only readable by root
rootbinddn cn=root,dc=ertw,dc=com
# Point the passwd, shadow, and group databases at a DN
# the ?one defines the scope
nss_base_passwd ou=People,dc=ertw,dc=com?one
nss_base_shadow ou=People,dc=ertw,dc=com?one
nss_base_group ou=Group,dc=ertw,dc=com?one
# Don't look for secondary groups for any of these users
nss_initgroups_ignoreusers
root,ldap,named,avahi,haldaemon,dbus,radvd,tomcat,radiusd
```

In addition to the content of `/etc/ldap.conf` shown in Listing 2, you also need to add the keyword `ldap` to the `passwd`, `shadow`, and `group` lines in `/etc/nsswitch.conf`. Always make sure to have `files` as the first entry; otherwise, you may find yourself waiting for downed servers to time out—or you may even be locked out of your system. (If you're locked out because of a problem with `nsswitch.conf`, boot to single-user mode, reset `nsswitch.conf` back to `files`, and then reboot.)

It's possible to use LDAP for all the databases, but the three listed here are the ones that are useful. The other maps rarely change and should be managed separately. The exception is the `hosts` database, which can use LDAP, although DNS is a much better choice.

Test it out

If you've got `nsswitch.conf` and `ldap.conf` configured properly, then you should be able to log in with an LDAP user, as long as the following attributes are available:

- `uid`: The login name
- `uidNumber`: The numeric userid
- `gidNumber`: The number primary groupid

- `homeDirectory`: The user's home directory
- `userPassword`: The user's password, encrypted with the `{crypt}` routine (use `slappasswd` to generate this)

These attributes and more are added through the `posixAccount` objectClass.

To test, try to log in as a user who is in your LDAP tree but not in the local password files. You can also use the `getent passwd` command to look at all the user entries NSS knows about. If `getent` works but the login doesn't, your `userPassword` attribute is likely incorrect.

If you've verified your configuration on the client, and NSS and LDAP still don't work together, enable `stats`-level logging on the OpenLDAP server and see if your queries are being seen by the server, and if they're being allowed.

Configure PAM to use LDAP

PAM is much like NSS in that it abstracts a set of library calls from the actual implementation. Unlike NSS, PAM doesn't replace existing UNIX calls; instead, it provides a set of new calls that applications can use.

Understand PAM

PAM is implemented as a library that applications use. Applications call this library to use the PAM management functions of checking authentication, account management, session management, and password management.

Checking authentication is the prime purpose of PAM. The application asks the PAM libraries whether the user is authenticated. The PAM libraries, in turn, follow the rules laid out by the systems administrator to prompt the user for a password or perform any number of other checks.

Account management is run after a user provides valid credentials and is responsible for checking to see if the login is allowed. A login may not be allowed at certain times or to certain applications.

Session management gives the application an opportunity to set up the environment after a successful login. It's often desirable to give the user logged into the console some extra permissions, such as the use of the local CDROM or other devices; this is done at the session-management level.

Finally, password management provides a flexible way to change passwords. As you'll soon see, this functionality lets users change their LDAP passwords through the familiar `passwd(1)` program. PAM password management also allows you to specify password-strength policies that operate independently of the password

backend.

To configure PAM for a service, you must create a file named after the service in `/etc/pam.d`, such as `/etc/pam.d/sshd` for the `sshd` service. This isn't a hard-and-fast rule, because the application specifies its own PAM service name. When in doubt, use the name of the binary, and check the logs for errors.

Each configuration file in `/etc/pam.d` specifies an ordered list of instructions for each of the PAM management functions. Each line in the file is of the form `function control module arguments`. The function is the management function, using keywords `auth`, `account`, `session`, and `password`.

The control specifies how the return value of the instruction being evaluated is to be used, and is one of the following keywords:

- `required` -- This check must succeed if the function is to succeed. If this check fails, then PAM will continue to check the rest of the instructions for the given function, but the results are meaningless.
- `requisite` -- This check must succeed if the function is to succeed. If this check fails, then PAM will stop checking the rest of the instructions and return a failure.
- `sufficient` -- If this check succeeds, processing stops and the function returns successfully, assuming no previous "required" elements have failed. If this check fails, the failure is ignored and processing continues.
- `optional` -- The results of the check are ignored.

The module and the arguments implement the check itself. The same module can implement one or more of the functions described, so you may see the same module listed several times. One module you'll see used often is `pam_stack`, which lets you call instruction stacks from other files. Listing 3 shows a PAM file that uses `pam_stack`.

Listing 3. Using `pam_stack` to call other instruction stacks

```
auth        required    pam_nologin.so
auth        required    pam_stack.so service=system-auth
account     required    pam_stack.so service=system-auth
session     required    pam_stack.so service=system-auth
password    required    pam_stack.so service=system-auth
```

Listing 3 shows the format of a PAM file. The `auth` function has two lines, both of which are required and therefore must succeed in order for a successful authentication to happen. The first `auth` line calls `pam_nologin`, whose job is to fail if a non-root user tries to log in when the `/etc/nologin` file exists. The next `auth` line calls the `pam_stack` module and passes it `service=system-auth`.

`pam_stack.so` then reads the contents of `/etc/pam.d/system-auth` and checks all the instructions under the `auth` function. If that returns a success, `pam_stack` returns a successful result back to the file in Listing 3.

The other three functions—`account`, `session`, and `password`—make reference only to `pam_stack` and the `system-auth` service. If the respective functions from `system-auth` return successfully, then the result is considered a success.

Many systems have a common set of routines for authentication, so `pam_stack` is used in most files, with the `system-auth` (or equivalent) containing all the interesting parts. For the rest of this section, the `system-auth` file will be the one used to inject LDAP into the PAM process.

Introducing LDAP to PAM

Both the NSS and PAM modules use `/etc/ldap.conf` for configuration, so if you're following along, you're halfway to having a working PAM-LDAP system. It's possible to use NSS and PAM together so that both PAM-aware and legacy applications can authenticate to LDAP. PAM provides some new features on top of NSS, including the following:

- Password changes by users
- More granular configuration of authentication requirements
- Support for more password encryption types
- Centralized administration of user accounts

Ensure that `pam_password md5` is in `/etc/ldap.conf`, and remove any other `pam_password` lines if they exist. This tells the `pam_ldap` library to hash the password with Message Digest 5 (MD5) locally before sending it to the LDAP server when changing passwords.

Edit your `/etc/pam.d/system-auth` (or equivalent) to add the references to `pam_ldap`, as shown in Listing 4. The line should go after any references to `pam_unix` (so that local accounts take precedence over LDAP accounts) but before any references to `pam_allow` and `pam_deny` (which provide a default allow or deny).

Listing 4. New system-auth that uses pam_ldap

```

auth          sufficient      pam_unix.so nullok try_first_pass
auth          sufficient      pam_ldap.so use_first_pass
auth          required       pam_deny.so

account       required       pam_unix.so broken_shadow
account       account       sufficient      pam_ldap.so
account       required       pam_permit.so

password      requisite     pam_cracklib.so try_first_pass retry=3

```

```

password    sufficient    pam_unix.so md5 shadow nullok try_first_pass
use_authtok
password  sufficient    pam_ldap.so use_authtok
password    required      pam_deny.so

session     required     pam_limits.so
session     required     pam_unix.so
session   optional     pam_ldap.so

```

The lines in **bold** show additions to the PAM configuration file. Note the addition of `broken_shadow` in the `account` function of `pam_unix`. This ensures that `pam_unix.so` doesn't return a failure if the user doesn't have a shadow entry (which it doesn't, because the account is in LDAP).

The `use_first_pass` option to the `auth` module of `pam_ldap` forces `pam_ldap.so` to use the password obtained from `pam_unix.so` rather than ask for a new password. `use_authtok` does a similar thing for the `password` function.

For authorization, the new configuration makes both UNIX passwords and LDAP passwords sufficient to log in: that is, the first one to succeed allows the user to log in. If neither returns success (either a failure, or "no such user"), then `pam_deny` causes a failure.

Test it out

Try to change a user's password through the `passwd` command, and then verify that the password was changed in the LDAP directory. Finally, ensure the user can still log in.

If you were able to get NSS working, PAM should also work. The biggest opportunity for error is mistyping the entries in the PAM configuration, putting the entries in the wrong file, or putting them in the wrong place in the file.

Section 3. NIS to LDAP migration

This section covers material for topic 305.2 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to:

- Analyze NIS structure prior to migration to LDAP
- Analyze NIS structure prior to integration with LDAP

- Automate NIS-to-LDAP migration
- Create a NIS-to-LDAP gateway

NIS is the traditional method of central authentication for UNIX machines. NIS is simple to set up and works well. Despite being more complex, LDAP authentication is superior to NIS in several ways:

- LDAP is more secure than NIS because you can encrypt the traffic and lock down the database.
- LDAP can store more than just authentication data, whereas NIS is limited.
- LDAP is accessible by more clients than is NIS.

You can choose to replace NIS with LDAP or use both simultaneously. When you use them together, LDAP is the canonical data source, and the NIS server uses data from LDAP instead of local files. This is a good approach for a longer-term migration or for supporting legacy operating systems that won't work with LDAP.

Approach 1: Migrate to LDAP

The general approach to migrating from NIS to LDAP is as follows:

1. Determine which NIS databases need to be replaced.
2. Load the NIS data into LDAP.
3. Reconfigure the clients to use LDAP instead of NIS.

For the time between the start of step 2 and the end of step 3, you have two active databases with no connections. Any changes, such as adding a user or changing a user's password, must be done on both databases; otherwise, your data may become inconsistent. You may elect to put a freeze on all changes or go with an integration strategy as shown in the next section.

Analyze your existing NIS structure

Before performing any migration, you must determine which databases are being hosted by NIS. Log in to the NIS master server, and look in the database directory. On most systems, the files are stored in `/var/yp/`, in a directory named after the domain name. Listing 5 shows the files in a typical NIS server's database directory.

Listing 5. Determining which databases are served by NIS

```
# ls /var/yp/`domainname`  
group.bygid  
group.byname  
hosts.byaddr  
hosts.byname  
mail.aliases  
netid.byname  
passwd.byname  
passwdbyuid  
protocols.byname  
protocols.bynumber  
rpc.byname  
rpc.bynumber  
services.byname  
services.byservicename  
ypservers
```

Listing 5 uses the `domainname` command to display the domain name. When placed inside backticks (```), the result of this command is inserted in the command line. With the exception of the `ypservers` file, all the other files in this directory represent a NIS database. Gather the list of unique database names to determine which databases need to be moved to LDAP. NIS stores the same data with different lookup keys, such as by name and UID in the case of the password file; in this case, they both represent the password database. Some aren't obvious: for example, `mail.aliases` is the aliases table. If in doubt, look in `/var/yp/Makefile` to determine the source of the database.

After looking at the server, you may wish to examine some of your NIS clients to determine which maps they're using. To do so, look for the `nis` keyword in `/etc/nsswitch.conf`. You'll probably find that your server is storing more maps than are being used.

Use the migration tools

The most popular tools to migrate NIS data to LDAP are provided by PADL software, the makers of `pam_ldap`, `nss_ldap`, and the NIS-LDAP gateway discussed later. Chances are, your distribution includes the files; otherwise, you can find links to the tools in the [Resources](#) section. The PADL migration tools can take data from local files, NIS, or NIS+ and dump them into your LDAP server.

Before using the PADL tools, you must have your LDAP server up and running with no data. The tools will generate all the entries necessary, and you want to avoid duplication.

The migration tools consist of a set of shell and perl scripts. On RedHat systems, the scripts are part of the `openldap-servers` package and are found in `/usr/share/openldap/migration` directory. Debian users will want the `migrationtools` package. Look for a file called `migrate_base.pl`, or download the latest version from PADL.

These scripts take data from a variety of sources, convert it to LDIF, and then add it to your server. Data is added with the `ldapadd` command in online mode and through `slapadd` in offline mode, so you'll need administrative credentials for the former, and you'll need to have your LDAP process stopped for the latter.

Before getting started, you'll find it helpful to set some environment variables to set up the base domain name (DN) of the tree and your root DN. Listing 6 shows the bash commands to prepare for migration of the `ertw.com` domain.

Listing 6. Setting environment variables in preparation for an LDAP migration

```
export LDAP_BASEDN="dc=ertw,dc=com"
export LDAP_BINDDN="cn=root,dc=ertw,dc=com"
export LDAP_DEFAULT_MAIL_DOMAIN=ertw.com
```

The first line in Listing 6 is the base DN of the LDAP tree, which will be used to generate all the DNs later. The second line is your root DN. You need the password only if you're using online mode. The final line of Listing 6 sets the default domain name for e-mail addresses. Some of the tools won't prompt you for this information, so setting it now will prevent aggravation later on.

The tools are split into two categories. The files in the first category have names starting with `migrate_all_`. The second category includes the remaining files, which have names beginning with `migrate_` followed by the name of a file or database. The scripts in the first category are used to gather the data together; the second category is used to convert the native format into LDIF.

You now have two options. You can use one of the `migrate_all_` scripts, which will automatically grab all the common databases from your chosen location (NIS, files, NIS+m, and so on); or you can grab only the relevant data yourself and use the individual migration scripts to convert the data into LDIF. The first approach, when it works, is easier. Listing 7 shows the use of `migrate_all_nis_online.sh` to migrate all the data from NIS into LDAP in online mode.

Listing 7. Migrating data from NIS to LDAP using the `migrate_all_nis_online.sh` script

```
[root@server1 migration]# ./migrate_all_nis_online.sh
Enter the NIS domain to import from (optional):
No such map networks.byaddr. Reason: Internal NIS error
Enter the hostname of your LDAP server [ldap]: localhost
Enter the credentials to bind with: mypassword
Do you wish to generate a DUAConfigProfile [yes|no]? no

Importing into dc=ertw,dc=com...

Creating naming context entries...
Migrating groups...
Migrating hosts...
Migrating networks...
```

```
Migrating users...
Migrating protocols...
Migrating rpcs...
Migrating services...
Migrating netgroups...
Migrating netgroups (by user)...
sh: /etc/netgroup: No such file or directory
Migrating netgroups (by host)...
sh: /etc/netgroup: No such file or directory
adding new entry "dc=ertw,dc=com"

Importing into LDAP...
adding new entry "ou=Hosts,dc=ertw,dc=com"
..... output omitted ...
adding new entry "cn=rquotad,ou=Rpc,dc=ertw,dc=com"

adding new entry "cn=rquotad,ou=Rpc,dc=ertw,dc=com"
ldap_add: Already exists (68)

/usr/bin/ldapadd: returned non-zero exit status: saving failed LDIF to
/tmp/nis.ldif.X17515
```

Listing 7 starts by running the `migrate_all_nis_onlinesh` script, which grabs data from NIS, converts it to LDIF, and then uses `ldapadd` to import the data. The first query from the script is the NIS domain; you can press **Enter** for the default NIS domain on the system. The script then imports the NIS data (on this system, a nonfatal error is printed because the networks map isn't used). The script prompts for information on the LDAP server, such as the hostname and the password (the bind DN and base DN were learned through the environment variables you entered in Listing 6). You should choose not to import a `DUAConfigProfile` unless you have a schema that supports it, which is unlikely.

If, at this point, you start getting errors about invalid DN syntax, be sure you've imported the `nis.schema` file inside `slapd.conf`.

If your schema is correct, the script will import data into your LDAP tree. It's likely that the script may die with an error such as the one seen at the end of Listing 7. Because of the way data is stored in NIS, you may have duplicate entries in some databases. This is fine in NIS but not in LDAP. There are a few solutions to this problem, depending on your needs:

- Edit the LDIF file (`/tmp/nis.ldif.X17515` in this case) to remove the duplicates, and then delete your LDAP database and import the file.
- Tell `ldapadd` to ignore errors with the `-c` option. `export LDAPADD="/usr/bin/ldapadd -c"` will do this. (Note that the script will still report an error, but the data will have been imported.)
- Edit `migrate_all_nis_online.sh` to set the value of `ETC_SERVICES`, `ETC_PROTOCOLS`, and `ETC_RPC` to `/dev/null` instead of a temporary file. Doing so skips processing the database. (Note that some of the `migrate_all_` scripts can be overridden by environment variables, but not

the NIS variant.)

- Skip `migrate_all_nis_online.sh`, and migrate by hand.

The first three options are self explanatory and effective as long as you're comfortable with the results (such as not having protocols, RPC, and services in LDAP for the third option). The fourth option needs some explanation.

If all you want to do is move groups and users over to LDAP, you can just as easily copy the files yourself and generate the LDIF using the other scripts provided, and using `ypcat` to get the data out of NIS. Listing 8 shows the process.

Listing 8. Migrating groups and users by hand

```
[root@server1 migration]# ypcat passwd > /tmp/passwd.tmp
[root@server1 migration]# ypcat group > /tmp/group.tmp
[root@server1 migration]# ./migrate_base.pl > /tmp/ldif
[root@server1 migration]# ./migrate_passwd.pl /tmp/passwd.tmp >> /tmp/ldif
[root@server1 migration]# ./migrate_group.pl /tmp/group.tmp >> /tmp/ldif
[root@server1 migration]# ldapadd -x -D "cn=root,dc=ertw,dc=com" \
-w "mypassword" -f /tmp/ldif
adding new entry "dc=ertw,dc=com"
adding new entry "ou=Hosts,dc=ertw,dc=com"
..... output omitted ...
```

The first two lines of Listing 8 use `ypcat` to get the data from NIS into a file in `/tmp`. The next three lines generate LDIF. `migrate_base` generates some basic entries in the tree, and the next two lines convert the password and group files to LDIF. Note the use of the append operator (`>>`), so the resulting file will contain the output of all three migration scripts. Finally, you call `ldapadd` to import the data.

Whichever way you go, perform some basic searches to make sure you can see the data. Be sure you can see the password hashes (use the root DN for this, because it's possible you have an access control list preventing passwords from being seen).

At this point, you have your NIS data in LDAP. Until all your NIS clients are moved over, all changes to NIS must be replicated to LDAP and vice versa.

Move the clients and verify results

Moving the clients is a simple matter of setting up NSS and PAM on the client. The [previous section](#) covered this in detail. In brief, you populate `/etc/ldap.conf` with your server information and edit `/etc/nsswitch.conf` to replace `nis` with `ldap`. If you're setting up PAM, then you need to edit the relevant files in `/etc/pam.d` to add references to `pam_ldap.so`.

Test your clients by logging in to them as a regular user and running the `getent` commands on the databases you moved to LDAP.

Approach 2: Integrate with LDAP

The second approach calls for the coexistence of NIS and LDAP. This can be helpful if you have clients that don't speak LDAP (either by not having a native LDAP module or by not supporting PAM), or if you want to spread out your transition over a longer period of time. The approach for a NIS/LDAP coexistence is similar to the first strategy:

1. Determine which NIS databases are in use.
2. Load the NIS data into LDAP.
3. Replace your NIS servers with `ypldapd`.
4. Reconfigure the clients that will be using LDAP.

The clients that will continue to use NIS need no changes because `ypldapd` is a fully functional NIS server. The only difference between it and the standard `ypserv` that comes with your operating system is that `ypldapd` gets its data from LDAP instead of local files.

The first two steps are the same as the first approach, so you begin at step 3.

Replace your NIS servers with `ypldapd`

`ypldapd` is a NIS server daemon that gets its information from LDAP instead of the database files in `/var/yp`. It's commercial software from PADL, but you can get a 30-day trial license by e-mailing PADL (see the [Resources](#)). Installation of `ypldapd` is a simple process:

1. Untar the software to `/opt/ypldapd`.
2. Copy the license to `/opt/ypldapd/etc/padlock.ldif`.
3. Edit the configuration file, `/opt/ypldapd/etc/ypldapd.conf`
4. Stop your existing NIS server.
5. Start up `ypldapd`.

First, run `mkdir -p /opt/ypldapd` as root to make the `ypldapd` directory (and `/opt`, if it doesn't already exist). Change into this directory (`cd /opt/ypldapd`), and untar the `ypldapd` distribution with `tar -xzf /tmp/ypldapd_linux-i386.tar.gz`. This places the `ypldapd` files in the

proper directory.

You'll have been given a license file, which you'll place in `/opt/ypldapd/etc/padlock.ldif`. If you're copying it from an e-mail, then make sure your e-mail client didn't wrap long lines: the key should be four lines long with a series of `attribute:value` pairs.

`ypldapd`'s configuration file is in `/opt/ypldapd/etc/ypldapd.conf`. There is a file called `ypldapd.conf.sample` that you can copy to start with. As with the other utilities you've seen so far, you need to provide information about your LDAP server. Listing 9 shows a simple `ypldapd.conf`.

Listing 9. Sample `ypldapd.conf`

```
# The NIS domain name
ypdomain ertw
# The LDAP server and base DN
ldaphost localhost
basedn dc=ertw,dc=com
# Credentials... The user must be able to read the userPassword attribute
binddn cn=ypldapd,dc=ertw,dc=com
bindcred mypassword
# The map of NIS databases to DNs (relative to basedn)
# If you used the migration tools then you shouldn't have to change anything
namingcontexts namingcontexts.conf
# Should ypldapd cache data?
caching on
# Cache lifetime, in minutes
cache_dump_interval 15
# Should passwords be hidden?
hide_passwords off
# How many ypldapd servers can be running at a given time?
maxchildren 5
```

With `ypldapd.conf` in place, you can shut down all instances of `ypserv` and then run `sbin/ypldapd`, which starts `ypldapd` in the background.

Move the clients and verify results

To test your new NIS server, run `ypwhich`, which tells you which NIS server you're bound to. If you get an error, make sure that no other instances of `ypserv` are running and that only one `ypldapd` is running. Then, try to fetch a map by typing `ypcat passwd` (this assumes your server was also running a client).

Clients that will be staying with NIS should also be able to run `ypwhich` and `ypcat` against the new server. For clients that will be moving to LDAP, see the previous set of instructions for the migration.

Section 4. Integrate LDAP with UNIX services

This section covers material for topic 305.3 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to:

- Integrate SSH with LDAP
- Integrate FTP with LDAP
- Integrate HTTP with LDAP
- Integrate FreeRADIUS with LDAP
- Integrate print services with LDAP

Most applications will work correctly with LDAP if you've configured NSS and PAM. Some applications need to be told to use PAM, or provide additional functionality by accessing LDAP correctly. This section focuses on the common UNIX daemons and how they support LDAP integration.

Integrate SSH with LDAP

The OpenSSH distribution integrates with LDAP through PAM, as long as the functionality was compiled in. To check, run `ldd /usr/sbin/sshd | grep pam` to see if the PAM shared libraries are linked. If not, you must recompile `sshd` with `--with-pam`.

To use PAM, be sure you have a PAM configuration file named `/etc/pam.d/sshd` if one doesn't exist already. Listing 10 shows a sample PAM file that makes use of the `system-auth` stack.

Listing 10. A sample `/etc/pam.d/system-auth`

```
auth        required      pam_stack.so service=system-auth
account     required      pam_stack.so service=system-auth
password    required      pam_stack.so service=system-auth
session     required      pam_stack.so service=system-auth
```

With the PAM configuration file in place, you can configure `sshd` to work with PAM. In `/etc/ssh/sshd_config`, add `UsePAM yes`, and restart `sshd`.

Integrate FTP with LDAP

Many FTP daemons are available, and it isn't clear which ones apply to the LPIC 3 exam.

The easiest integration method is to rely on NSS integration. When the FTP server performs a password lookup, the NSS facility uses LDAP.

In modern times, though, FTP servers are likely to be built with PAM support. In these cases, you create your PAM configuration file in `/etc/pam.d`. This file is usually called `ftp`, but it can be overridden depending on the software and distribution. For example, RedHat packages the `vsftpd` daemon to use `/etc/pam.d/vsftpd` instead of the default `/etc/pam.d/ftp`.

Once the `ftp` daemon has found its PAM configuration file, it processes it just like any other PAM client. The configuration in Listing 10 is enough to get started. You may also consider using `pam_listfile.so item=user sense=deny file=/etc/ftpusers onerr=succeed` and `pam_shells` in the `auth` phase to limit the users who can log in and the valid shells, like the legacy FTP servers did.

Integrate HTTP with LDAP

The Apache Web server has modules that handle basic HTTP authentication with an LDAP backend instead of the traditional `htpasswd` file-based backend. This is provided through the `mod_authnz_ldap` and `mod_ldap` modules. The first module provides the mechanisms to use LDAP information to authenticate a Web user, whereas `mod_ldap` provides an interface for `mod_authnz_ldap` (or any future LDAP-based module) to access LDAP, including connection pooling and caching.

The instructions in this section refer to Apache 2.2. If you're using Apache 2.0, the `mod_auth_ldap` module is used instead of `mod_authnz_ldap`. Configuration of these two modules is similar.

Both `mod_ldap` and `mod_authnz_ldap` are part of the Apache distribution. If you compile your Web server by hand, you need to add `--enable-authnz-ldap` `--enable-ldap` to your `configure` command. If you use your distribution's version of Apache, then install the appropriate module (for Red Hat distributions, the modules are part of the core `httpd` package).

When a user makes a request to a protected resource, Apache returns an error code 401 (unauthorized). At this point, the Web browser should prompt the user for a username and password. The Web browser then reissues the request with this information encoded in an `Authorization` header. If the username and password are accepted by the Web server, the page is served to the client; otherwise, the

server returns another 401.

Apache, when configured to check passwords against LDAP, first binds to the server as a predefined user and performs a lookup on the user to find the DN. The server then rebinds as the user with the provided password. If the server can successfully bind as this user, the authentication is considered successful.

After a successful authentication, the server can be configured to perform additional authorization tasks, such as checking against the DN or attribute, or testing to see if the user passes a search filter. If any of these tests are configured, then the test must pass for the authorization to pass.

Configuration of `mod_authnz_ldap` is similar to the standard authentication method using text files. Listing 11 shows the simplest case of LDAP authentication with no authorization.

Listing 11. Apache configuration for LDAP authentication

```
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so

<Location /protected>
  AuthType basic
  AuthName ProtectedByLDAP
  AuthBasicProvider ldap

  AuthLDAPUrl ldap://192.168.1138/ou=People,dc=ertw,dc=com?uid
  # Anon bind for first phase
  #AuthLDAPBindDN
  #AuthLDAPBindPassword

  AuthzLDAPAuthoritative off
  require valid-user
</Location>
```

The first two lines of Listing 11 load the required modules into the Web server. The rest of the configuration is enclosed in a `Location` container, meaning it applies only to requests beginning with `/protected`. The configuration first declares Basic authentication and a name of `ProtectedByLDAP`. The Web browser shows the name to the user. The `AuthBasicProvider` line tells Apache that authentication is provided through LDAP.

Listing 11 continues with `AuthLDAPUrl`, which points Apache to the LDAP server. The form of the parameter is

`ldap://host:port/basedn?attribute?scope?filter`. `host` and `port` define the LDAP server, and `basedn` is the base DN from which the initial search is performed. `attribute` refers to the attribute that will be searched along with the username during the initial search (the default is `uid`). `scope` is either `one` or `sub` to correspond with one level or all children. `filter` is an optional filter that will be logically ANDed with the search for the given user/attribute combination.

The example in Listing 11 has `AuthLDAPBindDN` and `AuthLDAPBindPassword` commented out, which results in an anonymous bind. If you want to specify a user here, you may. Either way, the user performing the initial bind must be able to search on the attribute provided in the `AuthLDAPUrl` command.

The final two lines disable authorization by allowing any validated user. `AuthzLDAPAuthoritative off` means that a later module can allow the access even if LDAP denies authorization (but not authentication). `require valid-user` comes from another module, so this deferral is required. Instead of these two lines, you can use LDAP-related ones, such as checking for group membership or an LDAP attribute. Listing 12 shows part of the configuration from Listing 11, but it restricts access to people with the `ou=Engineering` attribute and value.

Listing 12. Restricting access to a particular OU

```
AuthzLDAPAuthoritative on
require ldap-filter ou=engineering
```

Two things are notable in Listing 12. First, `AuthzLDAPAuthoritative` is now on (this is the default) because your requirement can be handled by the LDAP module. Second, the `ldap-filter` doesn't include parentheses. Apache uses the given LDAP filter and also performs a logical AND with the `uid` (or whichever attribute you specified in the `AuthLDAPUrl` command) and builds the search filter with a simple string insertion. If you have extra quotes or parentheses in your filter, the resulting query becomes invalid. The authentication fails, and a log message is printed to the server's `error_log`.

Integrate FreeRADIUS with LDAP

FreeRADIUS is an open source Remote Authentication Dial In User Service (RADIUS) server that is often used for authentication of dial-up or other network devices. Clients use RADIUS to authenticate users, and the RADIUS server in turn uses LDAP to find its information.

You can integrate PAM and FreeRADIUS two ways: by using PAM, or by enabling native LDAP support through the `rlm_ldap` module. The choice depends on how you plan to use RADIUS. If all you need is authentication, or you don't wish to modify your LDAP schema, then use PAM. If you need to use RADIUS attributes, then it's easier to configure the LDAP module and store the attributes in LDAP (RADIUS allows the server to send configuration details to the device asking for authentication, which lets you provide different services to different users).

For PAM mode, ensure that you have PAM set up for LDAP like the other systems. The PAM configuration file for FreeRADIUS is `/etc/pam.d/radiusd`. Starting from the default configuration files that come with FreeRADIUS, uncomment the `pam` keyword

in the `authenticate` section of `radiusd.conf`. Next, edit the `users` file and look for `DEFAULT Auth-Type = System`. Change the `System` keyword to `PAM`. Restart `radiusd`, and you're done.

The native LDAP module, `rlm_ldap`, is more complex. First, you must have FreeRADIUS installed on your system, built with the `rlm_ldap` module (`--enable-ldap`). FreeRADIUS is built like most other packages and so won't be covered here. Your Linux distribution, if it includes FreeRADIUS, likely includes the LDAP module.

FreeRADIUS includes an LDAP schema in a file called `openldap.schema`. Copy this to `/etc/openldap/schema/freeradius.schema`, and import it into OpenLDAP through the `include` directive in `slapd.conf`. The schema provides several attributes and two objectClasses. One of the objectClasses is `radiusprofile`; it's used for any users who will be authenticated with RADIUS. `radiusprofile` is an auxiliary objectClass and therefore can go on any entry. `radiusObjectProfile` is a structural objectClass used to create containers of radius profiles; it isn't necessary for operation.

Next, edit the default `users` file as in the PAM example, but instead of changing the default method to PAM, comment out that entire section. This file controls how users are authenticated and authorized. Removing the default method is enough to allow the LDAP module to take over and handle user authentication and authorization.

`radiusd.conf` needs more work. In both the `authenticate` and `authorize` sections, uncomment the `ldap` keyword that enables LDAP authentication and authorization. You must also find a section that looks like `Auth-Type LDAP { ldap }` and uncomment that. Finally, uncomment the `ldap { ... }` section, and enter your server's address, base DN, and optional authentication information. Like other software you've seen, the initial bind performs the lookup of the user's DN; then, a second bind is made as that user to confirm the password and retrieve the attributes. Therefore, the user you initially bind as (or anonymous if you have no configured user) must be able to perform searches on the `uid` attribute, and users must be able to read their own attributes.

Users who need to be authenticated by LDAP must use the `radiusProfile` objectClass and have a `dialupAccess` attribute with some value in it, such as "yes". With more advanced configurations, you can use the value to apply different settings, but for basic purposes, the attribute can take any value.

FreeRADIUS is an extremely robust RADIUS server, and a great deal of configuration can be required to get it to do what you need. The two configurations shown here focus only on what is needed to get LDAP working.

Integrate CUPS with LDAP

The Common UNIX Printing System (CUPS) is the currently favored printing daemon because of its ease of configuration, support for the Internet Printing Protocol (IPP), and backward compatibility with the traditional `lpr` tools. CUPS supports PAM, but it must be told how and when to authenticate.

First edit `/etc/pam.d/cups` so that it supports LDAP. Next, in `/etc/cups/cupsd.conf`, create for your printers a container that requires authentication, such as in Listing 13.

Listing 13. A printers container that requires authentication

```
<Location /printers>  
    AuthType Basic  
</Location>
```

Listing 13 shows a configuration that requires Basic authentication for any URL beginning with `/printers`. The CUPS configuration is almost identical to that of Apache, so this configuration should remind you of Listing 11. However, CUPS is using PAM instead of a native LDAP module, so no LDAP configuration is necessary. CUPS uses PAM for authentication because that is how it's configured. Now, when you try to browse a URL under `/printers`, which includes printing to a printer, you're prompted for a password. Listing 14 shows such a prompt.

Listing 14. Verifying that CUPS is working with LDAP

```
[sean@bob LPIC-III_5]$ lpr index.xml  
Password for sean on localhost? mypassword  
[sean@bob LPIC-III_5]$
```

If the password were incorrect or PAM wasn't working, then Listing 14 would have reprompted for a password. PAM was successful, though, so the document was printed and the user was returned to the shell prompt.

Section 5. Integrate LDAP with Samba

This section covers material for topic 305.4 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to:

- Migrate from `smbpasswd` to LDAP

- Understand the OpenLDAP Samba schema
- Understand LDAP as a Samba password backend

Samba is the UNIX community's way of integrating with Microsoft Windows networks. With this software, you can share files with Microsoft networks (both client and server) and make your UNIX computer appear as a Windows computer to the other Windows clients.

Understand Samba authentication

Samba's goal is integration with Windows networks, so it must use the authentication mechanisms that Windows uses. If you're authenticating against a Windows server, this is fine, but often the Samba server is the repository for the credentials. Thus, two copies of password hashes are needed—one for the traditional UNIX passwords and another for the Microsoft hashes.

Microsoft passwords are similar to UNIX passwords in that they're hashes of the real password. A *hash function* is a one-way function that accepts a variable-length input (such as a password) and outputs a fixed-length hash (string). It's impossible to take the hash and recover the original password, although you could try billions of different inputs in hopes that the resulting hash will match.

Two different password hashes are stored for Microsoft passwords: the LANManager hash and the Windows NT hash. The first isn't as secure as the second because several things are done to the password before hashing that reduce the number of possible outputs. The Windows NT hash was designed to overcome these limitations. Even though both hashes are stored, you can choose to disable the LANManager support if all your clients support NT hashes (available on Windows NT SP 3 and above).

Samba has traditionally stored the password hashes in the `smbpasswd` file and uses tools like `smbpasswd` to manage the password file by the same name. This can easily be moved to LDAP so that multiple Samba servers can authenticate without needing to use Primary Domain Controllers or other Microsoft infrastructure. Storing the data in LDAP also reduces duplication of information across your network.

Understand the Samba schema

NT passwords are different from UNIX passwords and can't be stored in the `userPassword` attribute. Therefore, the LDAP schema must be extended to store the password hashes and other pieces of information that a Microsoft device expects to be available.

The schema file is distributed with the Samba suite as `samba.schema`. Copy this file to `/etc/openldap/schema`, and use the `include` directive in `slapd.conf` to make it a part of your server's schema.

`samba.schema` introduces several new `objectClasses`, which are explained in Table 4.

Table 4. objectClasses in samba.schema

objectClass	Description
<code>sambaSamAccount</code>	Provides the information needed for an account (computer, user, and so on) in an NT environment.
<code>sambaGroupMapping</code>	Maps a UNIX group to a Windows group.
<code>sambaTrustPassword</code>	Provides authentication information about trust relationships between domains.
<code>sambaDomain</code>	Stores information about the domain in the LDAP tree. You'll find one of these added automatically to your LDAP tree after you set up Samba/LDAP.

Configure Samba for LDAP

Configuring Samba for LDAP involves editing `smb.conf` to set up the LDAP data source and then manipulating your users' LDAP entries to make them aware of the new Samba attributes.

In your `smb.conf`, you'll find a line like `passwd backend = tdbsam`, which represents the `smbpasswd` file storage mechanism. Replace this with the code in Listing 15, modified for your environment.

Listing 15. Using the `ldapsam` password storage

```
# ldapsam requires the uri to the LDAP server
passwd backend = ldapsam:ldap://192.168.1.138/
# A user in your LDAP server that can read and write the new attributes
# The password will be entered later
ldap admin dn = cn=root,dc=ertw,dc=com
# Same as search base
ldap suffix = dc=ertw,dc=com
# OUs for users/computers/groups
ldap user suffix = ou=People
ldap machine suffix = ou=Computers
ldap group suffix = ou=Group
```

Once you've set up `smb.conf`, restart Samba and execute `smbpasswd -W`. You're prompted for the password for the LDAP admin DN you entered in `smb.conf`. At this

point, Samba will use LDAP data to authenticate users.

Manage Samba users in LDAP

Users must be set up with the `sambaSamAccount` objectClass before they can use Samba, which includes setting the password hashes and assigning a security identifier (SID) to the user. This is easily handled by the `smbpasswd` utility, which traditionally added users to the `smbpasswd` file. `smbpasswd` will manage an LDAP user if `smb.conf` is configured to use LDAP, such as in Listing 15.

To set up a new user, first be sure the user's account is set up with the `posixAccount` objectClass and a `uid` attribute, which should already be there if the user logs in through LDAP and PAM or NSS. Next, run `smbpasswd -a username` to modify the user's LDAP entry, which includes setting the Samba password. Listing 16 shows a typical user's entry after being set up for Samba.

Listing 16. A Samba user's entry

```
dn: cn=Jim Joe,ou=people,dc=ertw,dc=com
givenName: Jim
sn: Joe
cn: Jim Joe
uid: jjoe
uidNumber: 1000
sambaSID: S-1-5-21-2287037134-1443008385-640796334-
userPassword:: e01ENX1yTDBZMjB6QytGenQ3MlZQek1TazJBPT0=
sambaLMPassword: 5BFAFBEBFB6A0942AAD3B435B51404EE
sambaNTPassword: AC8E657F83DF82BEEA5D43BDAF7800CC
loginShell: /bin/bash
gidNumber: 4
homeDirectory: /home/a
sambaAcctFlags: [U]
objectClass: inetOrgPerson
objectClass: sambaSamAccount
objectClass: posixAccount
objectClass: top
```

The bold lines from Listing 16 were added by `smbpasswd`. Starting from the top, a SID is added to the account. Using `smbpasswd` frees you from needing to calculate this, because `smbpasswd` figures out what SID to use. Next, the LanManager and NT password hashes are stored. The `sambaAcctFlags` is used to store some attributes of the entry. Possible values of this flag are as follows:

- N: No password required
- D: Account disabled
- H: Home directory required
- T: Temporary duplicate of other account

- U: Regular user account
- M: MNS (Majority Node Set cluster) logon user account
- W: Workstation Trust Account
- S: Server Trust Account
- L: Automatic locking
- X: Password doesn't expire
- I: Domain Trust Account

Finally, the `sambaSamAccount` objectClass enables all these attributes.

In addition to those described here, you can set many other options to carry more Windows-specific information. Consult the `pdbedit` manpage to learn about reading and modifying Samba user information from the command line. Samba can act as a Windows Primary Domain Controller (PDC), and the extra information is necessary for Windows clients to function correctly.

Password synchronization

Now that two sets of passwords exist (`userPassword` and the two Samba hashes), you must find a way to keep the passwords in sync with each other. If a user changes his or her Samba password, either from the command line or from a Windows client, the UNIX password should change. Likewise, if a user changes the UNIX password, the Samba password should change.

The first case is the easiest. Add `ldap password sync = yes` to the `[global]` section of `smb.conf`, and restart Samba. Any further password changes will change both the Samba and `userPassword` hashes.

Getting the Samba passwords changed when a user changes his or her password through the UNIX `passwd` command requires PAM. Samba comes with `mod_smbpasswd`, which is used to authenticate and change passwords through the Samba system. For now, there is no need to authenticate passwords, so only the `password` function will be used. Listing 17 shows part of a PAM configuration file that, when used, changes both UNIX and Samba passwords in LDAP.

Listing 17. A PAM password stack to change both UNIX and Samba passwords

```
password      requisite      pam_cracklib.so try_first_pass retry=3
password      optional      pam_smbpass.so use_authtok use_first_pass
password      sufficient   pam_unix.so md5 shadow nullok try_first_pass
use_authtok   sufficient   pam_ldap.so use_authtok
```

```
password    required    pam_deny.so
```

In Listing 17, the added line is shown in bold. The `pam_smbpass` module is listed as optional so that if a user isn't configured as a Samba user, that step will fall through. The Samba password change is before the UNIX and LDAP password changes, because these two are marked as `sufficient`, meaning the first one to succeed stops processing.

With Listing 17 in place, a user changing his or her password from the command line will also change the Samba password.

Migrate existing users to LDAP

When you move to an LDAP backend, you're likely to have existing users in a file-based password mechanism that you need to migrate. The `pdbedit` utility can copy accounts from one place to another to make this job easy.

Listing 18 shows the use of `pdbedit` to migrate users. The `-i` parameter sets the source of the data, and the `-e` parameter sets the destination. Before running the `pdbedit` command, you should have the `ldapsam` database set up in `smb.conf`.

Listing 18. Migrating users from `tdbsam` to `ldapsam`

```
[root@server1 ~]# pdbedit -e ldapsam -i tdbsam
Importing account for fred...ok
Importing account for jsmith...ok
```

If you're using the older `smbpasswd` password backend, use `smbpasswd` instead of `tdbsam`.

Section 6. Integrate LDAP with Active Directory

This section covers material for topic 305.5 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 2.

In this section, learn about:

- Kerberos integration with LDAP
- Cross-platform authentication

- Single sign-on concepts
- Integration and compatibility limitations between OpenLDAP and Active Directory

Microsoft Windows can be found in almost every company; there's a good chance that your environment already makes use of Active Directory, Microsoft's enterprise directory service. Active Directory is based on two open protocols: LDAP and Kerberos. By understanding these protocols and configuring the Linux system appropriately, your Linux box can authenticate against the enterprise directory and facilitate *single sign on* (SSO). This means you log in to your machine once, and your credentials are good throughout the network.

Understand Kerberos

Kerberos, named after the three-headed dog from Hades in Greek mythology, is a protocol that allows users and servers to prove their identity to each other over an untrusted network. It was developed at the Massachusetts Institute of Technology (MIT) for use on their network and has since found its place in many other networks. Microsoft chose to use Kerberos as part of Windows 2000's Active Directory.

Kerberos V is the currently developed stream, although you may run into Kerberos IV at times. Kerberos V provides backward compatibility for systems still using Kerberos IV.

The Kerberos protocol

Kerberos is a protocol that allows a service to authenticate the identity of a user without needing to see a password. This is achieved by having a mutually trusted server, called the Authentication Service (AS). The AS shares a secret with each user and service. The secret is used to protect information between the AS and the other end of the conversation; it even lets the AS give the user a message (called a *ticket*) destined for someone else. In this latter case, users can't read the ticket because they don't have the shared secret.

All clients and servers form a Kerberos *realm*, which is much like an NIS domain or, in some respects, the base DN of an LDAP tree. The realm defines all the devices and people that authenticate to a common set of Kerberos servers. Generally, the realm is the DNS zone of the organization written in uppercase, such as ERTW.COM. For the purposes of Kerberos, the clients are the computers that get tickets from a Kerberos server. The servers are the devices that provide the Kerberos services of granting tickets.

Everything that is authenticated in the Kerberos realm has a corresponding Kerberos *principal* that identifies it and is associated with the password or shared secret.

When a user connects to a server, they're really connecting to a service running on the server. Each service is treated separately and must be registered as a principal with the Kerberos server. A service's principal is of the form `servicename/servername@REALM`, whereas a user's principal is of the form `user@REALM`.

The Kerberos protocol is shown in Figure 1.

Figure 1. The Kerberos protocol

The Kerberos protocol can be viewed as having two distinct phases: the initial login of the user to the realm, and the authenticating of the user to the service. The magic of Kerberos is that the initial login happens only once; the subsequent service authentication can happen many times to many servers.

The first phase of Kerberos starts with the user asking the Kerberos server (specifically, a component called the Key Distribution Center [KDC]) for a *Ticket Granting Ticket* (TGT), which will be used later to request service. The KDC generates a TGT, encrypts it with the user's password, and sends it back to the user.

The TGT has been likened to a visitor pass in a company. You show your identification to the security guard (KDC) and are granted a visitor pass that's valid for one day. This process allows you to keep your own ID secure and also limits the company's exposure to a stolen visitor pass. The TGT expires in a short period of time, usually around 8 hours.

In the second phase, the user decides he or she needs access to a service. The user sends a request to the Kerberos server's Ticket Granting Service (TGS) component, which includes the TGT and the name of the service (the principal). The TGS checks to see if the TGT is still valid and then issues a ticket that has been encrypted with the shared secret *of the service*. Finally, the user presents this ticket to the service. If the service can successfully decrypt the ticket, then the service knows the Kerberos system approved the request. No passwords ever crossed the network.

Kerberos thwarts replay attacks, where an attacker captures a ticket and uses it again, by imposing limited lifetimes on tickets and including timestamps in the encrypted ticket. A ticket for a service may be valid for 5 minutes, so the service has to remember just 5 minutes' worth of tickets to know if a ticket was replayed. All clocks must be synchronized for this to succeed.

Where does LDAP fit in?

Kerberos provides only an authentication framework, much like the PAM system does. User information isn't stored in the Kerberos database.

Kerberos secrets can be stored in the LDAP database or they can be left separate. The choice is up to the implementation of Kerberos. In either case, LDAP is used to

store the user information such as home directory and personal information.

You must keep your Kerberos database secure regardless of where it's stored. The Kerberos keys are like passwords: they can be stolen and used to generate TGTs and tickets. Most guides strongly recommend keeping your Kerberos server on its own device and protecting it as much as possible.

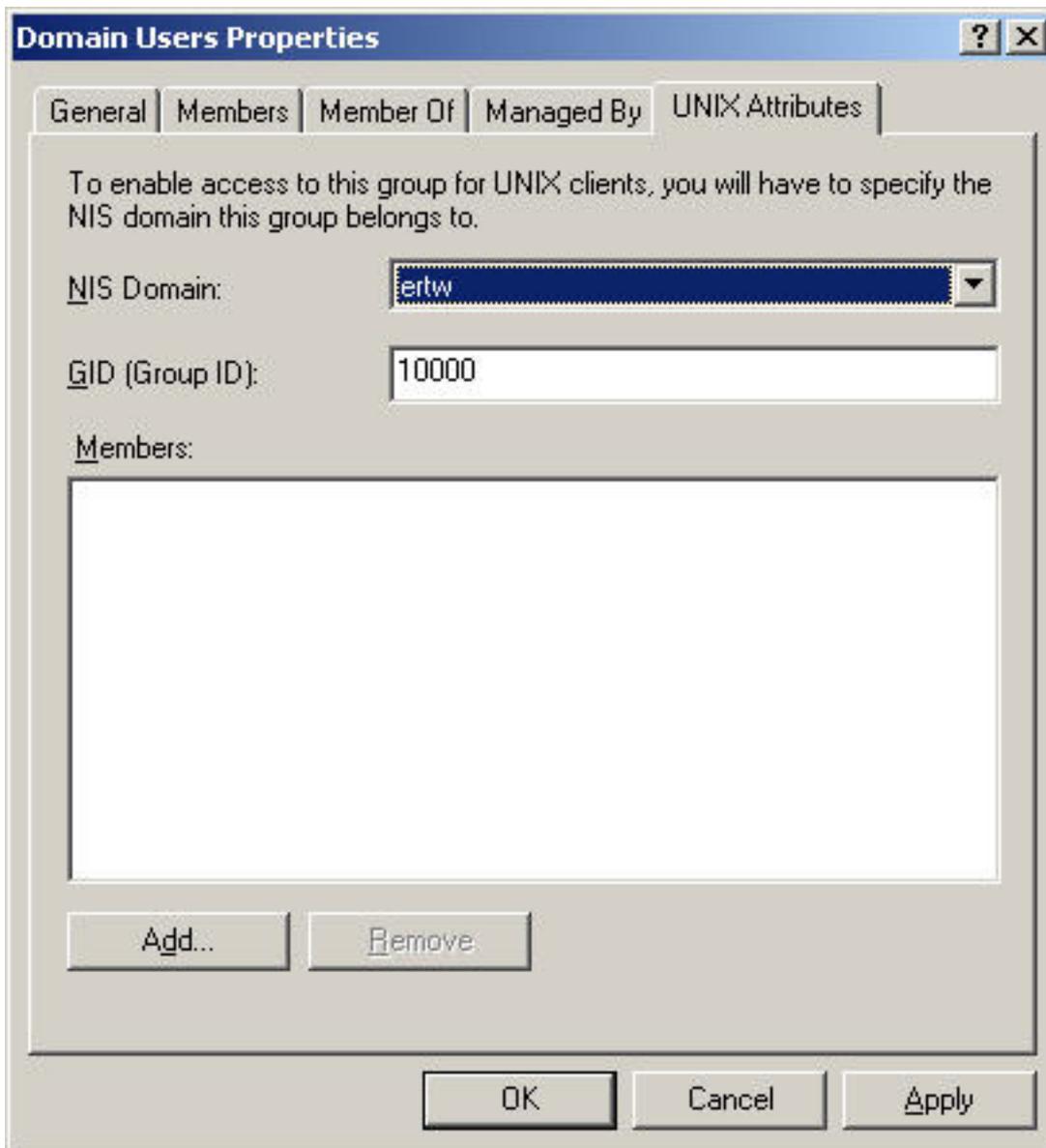
Configure Microsoft Active Directory for your Linux guests

Active Directory uses implementations of Kerberos and LDAP that are compatible with those shipped with Linux. Microsoft has extended Kerberos to support Windows-specific attributes, but this doesn't prevent UNIX users from using it (see the [Resources](#) for Microsoft's documentation on the subject).

The Active Directory schema must be extended to support some of the UNIX attributes, which is easily done in Windows 2003 Server. Go to the Control Panel of your Domain Controller, and choose **Add or Remove Programs > Add/Remove Windows Components**. From the Active Directory Services component, choose the Identity Management for UNIX subcomponent. (If you have an earlier version of Windows, this component is sometimes called the Server for NIS.) Install this software, and the LDAP schema will be extended; your user dialogs will also include a UNIX Attributes tab, which will be used soon.

From the Active Directory Users and Computers application, edit the Domain Users security group. Note the new tab, **UNIX Attributes**. Assign a group and NIS domain to your Domain Users group, as shown in Figure 2. Doing so allows the group to be seen by the UNIX systems. This group will be the user's primary group.

Figure 2. Assigning UNIX attributes to a group



Still in the Users container, find a user whom you want to use on your UNIX servers. Find the **UNIX Attributes** tab for this user, and assign the standard UNIX attributes to them. Figure 3 shows a sample user.

Figure 3. The UNIX attributes of a user

Sean Walberg Properties [?] [X]

Member Of | Dial-in | Environment | Sessions
General | Address | Account | Profile | Telephones | Organization
Remote control | Terminal Services Profile | COM+ | UNIX Attributes

To enable access to this user for UNIX clients, you will have to specify the NIS domain this user belongs to.

NIS Domain: ertw

UID: 10000

Login Shell: /bin/sh

Home Directory: /home/sean

Pimary group name/GID: Domain Users

OK Cancel Apply

The user in Figure 3 has been assigned a primary group, a home directory, a shell, and a userid.

Next, you must create a service account that allows access to your LDAP tree, because anonymous access is disabled by default. Use the following configuration for this user:

- **Name:** LDAP service account (or your choice)
- **User logon name:** ldap (or your choice)

- **Password:** Your choice
- **User can't change password:** Selected
- **Password never expires:** Selected
- **Primary group:** Domain Guests

The service account should only be a member of Domain Guests. From the **Member Of** tab, add the Domain Guests group to the account, highlight it in the list of groups, then click the **Set Primary Group** button. With the group changed, you can remove the Domain Users group from the profile.

If your security policy prohibits the password options, you'll have to adjust your Linux configuration (described next) each time the password changes. Note that LDAP is used here only for directory information and not passwords, so the requirement to change passwords is lessened.

Configure Linux

The Linux side of the equation involves three steps. First, you set up directory access through `/etc/ldap.conf`. Next, you configure PAM for Kerberos authentication. Finally, configure Samba to use Active Directory information for authentication, and join it to the domain.

Before you start, you must ensure that your Linux machine is using your Microsoft server for both DNS and network time. Your Linux server must also have a host record in the Microsoft DNS zone for your domain.

Configure LDAP

LDAP is configured as it was [earlier](#), except that some mapping is required from UNIX attributes to Microsoft attributes. Listing 19 shows `/etc/ldap.conf` configured to access a Microsoft LDAP directory using the user account set up previously.

Listing 19. Configuring `ldap.conf` to use a Microsoft directory

```
# Information about the directory
uri ldap://192.168.1.151
binddn ldap@ertw.com
bindpw ldap
ssl no
base dc=ertw,dc=com

# Map attributes
nss_map_objectclass posixAccount user
nss_map_objectclass shadowAccount user
nss_map_attribute uid sAMAccountName
nss_map_attribute homeDirectory unixHomeDirectory
nss_map_attribute shadowLastChange pwdLastSet
```

```
nss_map_objectclass posixGroup group
nss_map_attribute uniqueMember member
pam_login_attribute sAMAccountName
pam_filter objectclass=User
pam_password ad
```

The configuration shown in Listing 19 first points the module to the Microsoft LDAP server using the credentials set up earlier. The attributes are mapped from the UNIX name to the Microsoft name, such as using `sAMAccountName` for the `userid`.

Finally, add `ldap winbind` to the `passwd`, `group`, and `shadow` sections of `/etc/nsswitchconf` (leave files in there). This lets your system pull directory information from LDAP and Samba (the latter to be configured later).

After this step, you can run `getent passwd` to see the LDAP users. Note that you must set the UNIX attributes for the user in Active Directory for the user to show up in the list.

Configure Kerberos

Kerberos is configured through PAM and the `/etc/krb5.conf` file. If you're using your Microsoft DNS server for your DNS, then you only need to specify your realm, because the server will be learned automatically from DNS. Listing 20 shows the contents of `/etc/krb5.conf`

Listing 20. `/etc/krb5.conf` for the ERTW.COM realm

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = ERTW.COM
dns_lookup_realm = false
dns_lookup_kdc = true
ticket_lifetime = 24h
forwardable = yes

[realms]
ERTW.COM = {
    default_domain = ertw.com
}

[domain_realm]
.ertw.com = ERTW.COM
ertw.com = ERTW.COM

[appdefaults]
pam = {
    debug = false
    ticket_lifetime = 36000
    renew_lifetime = 36000
    forwardable = true
    krb4_convert = false
}
```

krb5.conf is divided into sections, with the name of the section enclosed in square brackets. The logging section specifies the paths of various logfiles. The libdefaults sections configure the Kerberos libraries: in particular, the `dns_lookup_kdc` tells the library to look for server records (SRV) in DNS to find the KDC. The record looks like `_kerberos._tcp.ERTWCOM.`, and the response is the name of a server and the port to contact.

The realms section defines the realms and the associated DNS zones. The `domain_realm` section does the reverse: it allows a host to determine its realm based on its fully qualified domain name (FQDN). Finally, the `appdefaults` section is for the applications using Kerberos; in this case, PAM has been configured with some default options.

In practice, there is little to configure in `krb5.conf` because the default configuration file has all the required elements. All you have to do is substitute your realm and domain name where appropriate. You can also use your system's Kerberos configuration utility, such as `authconfig`.

The configuration of PAM is just like the previous configurations of LDAP and `smbpasswd`. You insert a call to the Kerberos PAM library where appropriate. Listing 21 shows part of the Fedora `system-auth` file after Kerberos has been configured.

Listing 21. system-auth file after configuring Kerberos

```

auth          sufficient      pam_unix.so nullok try_first_pass
auth          sufficient      pam_krb5.so use_first_pass
auth          required        pam_deny.so

account       required        pam_unixso broken_shadow
account       [default=bad success=ok user_unknown=ignore] pam_krb5.so
account       required        pam_permit.so

password     sufficient      pam_unix.so md5 shadow nullok try_first_pass
use_authtok
password     sufficient      pam_krb5.so use_authtok
password     required        pam_deny.so

session      required        pam_unix.so
session      optional        pam_krb5.so

```

Kerberos has been added directly after the UNIX password check in the authorization phase as a sufficient item. This means that if a UNIX password is found, Kerberos isn't consulted. If no UNIX password is found, then Kerberos is consulted. If Kerberos fails, the stack fails. If no user is found, then control passes to the `pam_deny` module, which causes a failure.

The account phase uses an alternative syntax to the one you've seen so far. Each PAM module can return different options, such as "success" or "no such user". The square brackets allow the administrator to take a different action based on each possible return code. Listing 21 implements a policy that says if `pam_krb5` returns a

successful result, then continue processing. If the user is unknown, then ignore the module completely. Anything else is considered a failure. This behavior is close to the `required` keyword, with the exception that an unknown user doesn't cause a failure. Consult the `pam.conf(5)` manpage for more details on this syntax, including the options.

The password and session phases include the module in the stack with no special options.

At this point, you should be able to log in to your Linux server using Active Directory credentials. The next step configures Samba and further secures the connection by creating a computer account for the server.

Configure Samba and join the domain

For Samba's configuration, you must first remove your existing password backend from `smb.conf`, and any of the `tdb` files in `/etc/samba` and `/var/cache/samba`. Listing 22 shows the directives you must add to the `[global]` section of `smb.conf` to allow Samba to use AD.

Listing 22. Samba configuration for AD integration

```
# Active directory security
security = ads
realm = ERTW.COM
use kerberos keytab = yes

# Identity mapping
idmap backend = ad
ldap idmap suffix = dc=ertw,dc=com

# LDAP configuration
ldap admindn = cn=ldap,cn=users,dc=ertw,dc=com
ldap suffix = dc=ertw,dc=com

# Winbind
winbind use default domain = yes
winbind nested groups = yes
```

Listing 22 starts by specifying that ADS security mode is used (remote Active Directory server), along with the Kerberos realm. The second section configures idmapping, which is a feature that maps remote Microsoft SIDs to local UNIX ids. This configuration specifies that Active Directory is the source of the information. The mapping is taken care of on the Microsoft side, because you've already entered the IDs in the UNIX Attributes tab of the users and groups. Your server just has to pull this information from LDAP.

The LDAP configuration is familiar; it sets the DN for the LDAP connection to the `ldap` user created earlier. Note that the container is `cn=users` instead of the `ou=people` that has been used so far. The password is entered through

smbpasswd.

The last two lines enable Winbind, which is an implementation of some of the Microsoft Remote Procedure Calls (see [Resources](#) for more information on Winbind). It lets you get more information out of your Active Directory server, rather than only the groups and users for which you've added UNIX attributes.

After smb.conf is configured, start the Samba and winbind services.

The final steps in the Samba configuration are to set the `adminidn` password and to join your domain. Listing 23 shows the computer joining the domain.

Listing 23. Setting the `adminidn` password and joining the domain

```
[root@server1 ~]# smbpasswd -W
Setting stored password for "cn=ldap,cn=users,dc=ertw,dc=com" in secrets.tdb
New SMB password: ldap
Retype new SMB password: ldap

[root@server1 ~]# net ads join -U administrator
administrator's password: mypassword
Using short domain name -- ERTW0
Joined 'SERVER1' to realm 'ERTW.COM'
```

Test it out

You should be able to log in to your server with Active Directory credentials and browse file shares from a remote computer without having to log in. Some helpful commands to test are:

- `net ads testjoin`: Tests the computer account
- `wbinfo -u`: Shows a list of Active Directory users and tests winbind
- `klist`: After you log in through Kerberos, shows your TGT and any service tickets that have been issued
- `smbclient -k -L '\\SERVERNAME'`: Shows a list of shares from SERVERNAME, using a Kerberos login

Section 7. Integrate LDAP with e-mail services

This section covers material for topic 305.6 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to:

- Plan LDAP schema structure for e-mail services
- Create e-mail attributes in LDAP
- Integrate Postfix with LDAP
- Integrate sendmail with LDAP

sendmail and Postfix are two of the more popular mail transport agents (MTAs) in use. The job of the MTA is to receive messages from the systems and send them to be delivered to the end user or to the next hop MTA. MTAs also take messages from users and find the remote MTA that is capable of delivering the message.

Both sendmail and Postfix rely on various maps—key/value pairs that are normally held in flat files or hash databases like BDB. This type of lookup is also a good fit for LDAP. The advantages of LDAP are that many hosts can share the same configuration, and that it's easier to develop tools to manage the data in LDAP rather than in flat files that must then be rebuilt into hash tables. The overhead of LDAP versus disk reads should not be that onerous, especially if the LDAP tree is properly indexed.

Configure sendmail

The sendmail MTA is a complex creature, and adding LDAP to the mix only increases the complexity. You can do just about anything with sendmail because it's almost infinitely configurable. The downside is that things that should be simple tend to be more complicated than necessary.

Understand that sendmail is a program that interprets a language often called *cf* in order to process mail. Cf is a language made for easy parsing by sendmail, not humans. Fortunately, humans can use a language called M4, which has a much simpler syntax, to generate the resulting cf code.

sendmail maps

Many cf operations involve looking up information in *maps*, which are series of key-value pairs. Each map has a particular purpose, such as the `aliases` map for mail aliases and the `mailertable` map for static routing of e-mail. The map is a two-column entity; lookups are performed on the left-hand side (LHS), and the corresponding value from the right-hand side (RHS) is returned.

The concept of a map doesn't translate directly into LDAP. A single-key lookup in a sendmail map may return only one RHS entry (the RHS can have multiple values, but only one instance of the key may exist). sendmail works with this by defining a

schema that allows key-value pairs to be stored in LDAP. Furthermore, sendmail translates each map request into an LDAP query filter that is designed to return a set of attributes from a single entry. You can choose to use the sendmail schema, or you can modify the filters to work with the data in your tree.

To get started, add the `misc.schema` schema (it comes with OpenLDAP) to your server's schema. This implements LDAP-based mail routing. Then, add `sendmail.schema` from the sendmail distribution, which lets you store maps in LDAP.

Configure LDAP mail routing

Some organizations have multiple mail servers to handle all their users, either because of geographical constraints or for managing capacity on the servers. In this case, a user's mailbox might be on a server called `wpgertw.com` but the user's e-mail address might be `sean@ertw.com`. LDAP mail routing allows any sendmail server to receive the message, perform an LDAP lookup, and rewrite the address to the internal version. It's then a simple matter to change the destination server by changing LDAP. In any case, the user's e-mail address stays the same.

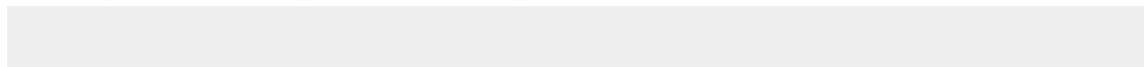
The `misc.schema` implements an Internet draft for LDAP routing. This schema provides the `inetLocalMailRecipient` objectClass with the following attributes:

- `mailLocalAddress`: An attribute that defines someone's e-mail address as it's seen by someone outside the organization
- `mailRoutingAddress`: An attribute that defines the internal address of a user, which usually includes the server that holds the user's mailbox
- `mailHost`: An attribute that defines the server that handles this user's e-mail

The host information for the user can be held in either `mailRoutingAddress` or `mailHost`. For example, a `mailRoutingAddress` of `sean@wpg.ertw.com` with a `mailHost` of `mx.ertw.com` seems like a contradiction. If the host is set, the mail will be delivered there regardless of the routing address. The address will still be rewritten to the routing address if that attribute exists. In the example of `sean@wpg.ertw.com`, the address in the envelope will be rewritten to `sean@wpg.ertw.com`, but the message will be delivered to `mx.ertw.com`.

Listing 24 shows the M4 code that enables LDAP routing. This should go in `/etc/mail/sendmail.mc`; then, you must rebuild your `sendmail.cf`. Usually this means going into `/etc/mail/` and running `make`; or it may mean running `m4 sendmail.mc > sendmail.cf`.

Listing 24. Enabling LDAP routing in sendmail.mc



```
define(`confLDAP_DEFAULT_SPEC', `-h localhost -b dc=ertw,dc=com')
FEATURE(`ldap_routing')
LDAPROUTE_DOMAIN(`ertw.com')
```

The first line sets the default arguments for the internal LDAP client: the host and the search base. The second line enables the LDAP routing feature, and the third enables the ertw.com domain for LDAP routing.

The duplication of the `mailLocalAddress` attribute from `inetLocalMailRecipient` and the `mail` attribute from `inetOrgPerson` is worth looking at. `sendmail` lets you override the searches it uses internally by passing extra arguments to the `ldap_routing` feature. The first argument is the filter used to find the `mailHost` attribute, and the second is used to find `mailRoutingAddress`. Therefore, `FEATURE(`ldap_routing', `ldap -l -T<TMPF> -v mailHost -k (&(objectClass=inetLocalMailRecipient)(mail=%0))', `ldap -l -T<TMPF> -v mailRoutingAddress -k (&(objectClass=inetLocalMailRecipient)(mail=%0))')` enables `sendmail` to use the `mail` attribute instead of `mailLocalAddress`. The search filter is specified with the `-k` switch, and the attribute to return with `-v`. The rest of the arguments are standard for `sendmail`.

Configure aliases

`sendmail` implements LDAP aliases as a series of entries in the LDAP tree, keyed on an attribute called `sendmailMTAKey` using an objectClass of `sendmailMTAAliasObject`. You may wish to keep the aliases in their own container. Listing 25 shows the LDIF for a `sendmail` alias that takes mail for `exec@ertw.com` and sends it to `hair@ertwcom` and `teeth@ertw.com`.

Listing 25. Alias for `exec@ertw.com`

```
dn: sendmailMTAKey=execs,ou=aliases,dc=ertw,dc=com
objectClass: sendmailMTAAliasObject
sendmailMTACluster: external
sendmailMTAAliasGrouping: aliases
sendmailMTAKey: execs
sendmailMTAAliasValue: hair@ertwcom
sendmailMTAAliasValue: teeth@ertw.com
```

The first attribute, `sendmailMTACluster`, defines the servers that can use this alias. You must also define the cluster name in the `sendmail.mc` file, such as `define(`confLDAP_CLUSTER', `external')`. This cluster is used as part of the search filter, so if you forget to define it, your aliases will never be used. The alternative to defining a cluster is to set `sendmailMTAHost`, which makes the entry apply only to a particular host.

`sendmailMTAAliasGrouping` must be `aliases`; this is part of the search filter. The key refers to the name of the alias; finally, you have one or more values that are

the targets.

The final step is to configure sendmail to use LDAP for the aliases file with the `define(`ALIAS_FILE', `ldap:') M4` directive. In general, anywhere you're asked for a file in `sendmail.mc`, you can put `ldap:`, and the map will be referenced in LDAP. The `sendmailMTAAliasGrouping` then becomes the name of the map.

Configure Postfix

Postfix is designed to be simpler than sendmail but to remain compatible with sendmail. The concept of maps is still around, but instead of fitting the maps into a schema, you must define your own query filters that use your own attributes.

For most maps, you specify the target as `ldap:/path/to/config.cf`, with `config.cf` being a configuration file that defines the LDAP server, the query, and the attributes that form the response. For example, the `local_recipient_maps` directive specifies how Postfix will map e-mail addresses to local accounts. Specify `local_recipient_maps = $aliases, ldap:/etc/postfix/localrecipients.cf` to first check the aliases database (to come later) and then the regular address attached to a user's entry. Listing 26 shows the contents of `localrecipients.cf`.

Listing 26. The local recipients LDAP lookup

```
# LDAP server info
server_host = ldap://localhost
search_base = ou=people,dc=ertw,dc=com

# %s is the e-mail address...
query_filter = mail=%s

# the uid tells the account that gets the delivery
result_attribute = uid
```

Listing 26 specifies the local LDAP server and the People OU. Postfix consults the search filter and replaces the `%s` with the e-mail address. Thus, an e-mail for `fred@ertw.com` will result in a search for `(mail=fred@ertw.com)` in the People OU. The `uid` attribute is used to determine the mailbox. To test, you can run `postmap -q fred@ertw.com ldap:/etc/postfix/localrecipients.cf`, which runs the given e-mail address through the `localrecipients.cf` configuration file (note that NSS must be configured to return details about the fred account).

Section 8. Summary

In this tutorial, you learned how to integrate LDAP with your current systems. NSS provides an easy way for core UNIX tools to make use of LDAP by redirecting the standard C library calls to the backend of your choice. PAM is yet another abstraction; it allows you to change the way applications authenticate in a granular fashion as long as the application is PAM aware. PAM also has hooks for account restrictions and password changes. The PAM files live in `/etc/pam.d`.

Migrating from NIS to LDAP involves planning which databases need to be moved and then running some utilities to extract the data and convert it to LDIF. If you still must support NIS in your environment, PADL has written a NIS server called `ypldapd` that translates between NIS and LDAP by presenting a NIS interface to applications, and that reads the data from LDAP.

Many applications are PAM aware, which means your migration to LDAP is as simple as changing a few files in `/etc/pam.d`. Some applications, like Apache, speak LDAP directly. Configuring Apache for LDAP involves using the `mod_authnz_ldap` module and specifying search filters that help Apache find the users in the tree.

Samba provides Windows services on a UNIX platform. You can configure Samba to use LDAP data or even to use Kerberos data to talk directly to Windows. In the latter case, LDAP is still used for directory information, and Kerberos is used for authentication.

E-mail is a natural fit for LDAP because of its similarity to a phone book. Both sendmail and Postfix allow maps to be served from LDAP.

This concludes the look at directory services for the LPIC 3 exam. The next and final tutorial in the series will focus on monitoring and predicting the performance of your Linux servers.

Resources

Learn

- Review the previous tutorial in this 301 series, "[LPI exam 301 prep, Topic 304: Usage](#)" (developerWorks, March 2008), or [all tutorials in the 301 series](#).
- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- New to Kerberos? Start with this [explanation in the form of a play](#), read the [Moron's Guide to Kerberos](#), and then read the [Kerberos FAQ](#).
- Read Microsoft's documentation on [Kerberos operation](#) and [Kerberos troubleshooting](#).
- This PowerPoint presentation on [how Kerberos works](#) (ppt) includes play-by-play animations of the packets and messages. It's geared toward Windows administrators, but if you ignore the Windows-specific details, you'll get an excellent description of the protocol.
- These guides on [Using Samba and Kerberos](#) and [Native LDAP, native Kerberos, and Windows Server AD Services and schema for cross-platform identity management](#) provide a step-by-step method for integrating Linux services into Active Directory.
- The Samba documentation on [IDMAP](#) shows you how Samba maps between Microsoft SIDs and UNIX userids.
- This article on [Winbind](#) shows you an alternative way to integrate Samba and AD without using Kerberos.
- [mod_authnz_ldap](#) gives you all the details on Apache and LDAP configuration.
- [How HTTP authentication works](#) is covered in Wikipedia.
- Read the FreeRADIUS documentation on the [LDAP module](#). If you're having a hard time finding the schema, try [RADIUS-LDAPv3.schema.gz](#).
- Read the [Postfix LDAP Howto](#) and the man page for `ldap_table(5)` before you start with Postfix and LDAP.
- The [sendmail configuration guide](#) describes all the ways LDAP can be used, including the way the search filters are generated from the configuration file. This [post about an alternative way to build sendmail aliases](#) is enlightening, not only because it's simpler than the normal way, but also because it gives you a peek behind the scenes.

- This [online LDAP book](#) is an excellent work in progress.
- In the [developerWorks Linux zone](#), find more resources for Linux developers, and scan our [most popular articles and tutorials](#).
- See all [Linux tips](#) and [Linux tutorials](#) on developerWorks.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Firewall Builder](#) simplifies the task of typing in iptables rules with a nice GUI and suite of tools to roll out updates to your firewalls.
- Download [pam_ldap](#) and [nss_ldap](#) if your distribution does not include the PADL PAM and NSS LDAP libraries.
- Download [ypldapd](#) if you're going to follow along with the NIS-LDAP gateway demonstration. The license is good for 30 days.
- Download the [LDAP migration tools](#) from PADL's site.
- Microsoft has developed [gssMonger](#) for verifying Kerberos authentication interoperability between Windows and other platforms.
- [OpenLDAP](#) is a great choice for an LDAP server.
- [phpLDAPadmin](#) is a Web-based LDAP administration tool. If the GUI is more your style, [Luma](#) is a good one to look at.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the [developerWorks community](#) through blogs, forums, podcasts, and community topics in our [new developerWorks spaces](#).

About the author

Sean Walberg

Sean Walberg has been working with Linux and UNIX since 1994 in academic, corporate, and Internet service provider environments. He has written extensively about systems administration over the past several years.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

LPI exam 301 prep, Topic 306: Capacity planning

Senior Level Linux Professional (LPIC-3) exam study guide

Skill Level: Intermediate

[Sean Walberg \(sean@ertw.com\)](mailto:sean@ertw.com)
Senior Network Engineer
Freelance

15 Apr 2008

In this tutorial, Sean Walberg helps you prepare to take the Linux Professional Institute Senior Level Linux® Professional (LPIC-3) exam. In this last in a [series of six tutorials](#), Sean walks you through monitoring your system resources, troubleshooting resource problems, and analyzing system capacity.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *advanced level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102. To attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active advanced-level certification and pass exam 301 ("core"). You may also need to pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the five junior, advanced, and senior certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. Table 1 lists the six topics and

corresponding developerWorks tutorials for LPI exam 301.

Table 1. LPI exam 301: Tutorials and topics

LPI exam 301 topic	developerWorks tutorial	Tutorial summary
Topic 301	LPI exam 301 prep: Concepts, architecture, and design	Learn about LDAP concepts and architecture, how to design and implement an LDAP directory, and about schemas.
Topic 302	LPI exam 301 prep: Installation and development	Learn how to install, configure, and use the OpenLDAP software.
Topic 303	LPI exam 301 prep: Configuration	Learn how to configure the OpenLDAP software in detail.
Topic 304	LPI exam 301 prep: Usage	Learn how to search the directory and use the OpenLDAP tools.
Topic 305	LPI exam 301 prep: Integration and migration	Learn how to use LDAP as the source of data for your systems and applications.
Topic 306	LPI exam 301 prep: Capacity planning	(This tutorial.) Measure resources, troubleshoot resource problems, and plan for future growth. See the detailed objectives .

To pass exam 301 (and attain certification level 3), the following should be true:

- You should have several years experience with installing and maintaining Linux on a number of computers for various purposes.
- You should have integration experience with diverse technologies and operating systems.
- You should have professional experience as, or training to be, an enterprise-level Linux professional (including having experience as a part of another role).
- You should know advanced and enterprise levels of Linux administration including installation, management, security, troubleshooting, and maintenance.
- You should be able to use open source tools to measure capacity planning and troubleshoot resource problems.
- You should have professional experience using LDAP to integrate with UNIX® services and Microsoft® Windows® services, including Samba,

Pluggable Authentication Modules (PAM), e-mail, and Active Directory.

- You should be able to plan, architect, design, build, and implement a full environment using Samba and LDAP as well as measure the capacity planning and security of the services.
- You should be able create scripts in Bash or Perl or have knowledge of at least one system programming language (such as C).

To continue preparing for certification level 3, see the [series developerWorks tutorials for LPI exam 301](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute doesn't endorse any third-party exam preparation material or techniques in particular.

About this tutorial

Welcome to "Capacity planning," the last of six tutorials designed to prepare you for LPI exam 301. In this tutorial, you'll learn all about measuring UNIX resources, analyzing requirements, and predicting future resource requirements.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weights.

Objectives

Table 2 shows the detailed objectives for this tutorial.

Table 2. Capacity planning: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
306.1 Measure resource usage	4	Measure hardware and network usage.
306.2 Troubleshoot resource problems	4	Identify and troubleshoot resource problems.
306.3 Analyze demand	2	Identify the capacity demands of your software.
306.4 Predict future resource needs	1	Plan for the future by trending usage and predicting when your applications will need more resources.

Prerequisites

To get the most from this tutorial, you should have advanced knowledge of Linux and a working Linux system on which to practice the commands covered.

If your fundamental Linux skills are a bit rusty, you may want to first review the [tutorials for the LPIC-1 and LPIC-2 exams](#).

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

System requirements

To follow along with the examples in these tutorials, you'll need a Linux workstation with the OpenLDAP package and support for PAM. Most modern distributions meet these requirements.

Section 2. Measure resource usage

This section covers material for topic 306.1 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 4.

In this section, learn how to:

- Measure CPU usage
- Measure memory usage
- Measure disk I/O
- Measure network I/O
- Measure firewalling and routing throughput
- Map client bandwidth usage

A computer relies on hardware resources: central processing unit (CPU), memory, disk, and network. You measure these resources to get an idea of how the computer is doing at the present moment and where any trouble spots may lurk. Looking at these measurements over a period of time, such as a few months, gives you some interesting history. It's often possible to extrapolate these readings to the future,

which helps you predict when one of the resources will run out. Alternatively, you can develop a mathematical model of your system, using historical information to validate the model, which you can then use to more accurately predict future usage.

Servers always require more than one hardware resource to complete a task. A task may require disk access to retrieve data, and memory to store it while the CPU processes it. If one of the resources is constrained, performance suffers. The CPU can't process information until it's read from disk; nor can the information be stored if memory is full. These concepts are related. As memory fills up, the operating system starts swapping other memory to disk. Memory is also taken away from buffers, which are used to speed up disk activity.

Understanding resources

Before measurements are useful, you must understand what you're measuring. Then you can begin to draw useful information about your system: current information, history, or future predictions.

CPU

The computer's CPU performs all the calculations an application needs, causes commands to be issued to disks and other peripherals, and takes care of running the operating system kernel. Only one task runs on the CPU at a time, whether the task is running the kernel or a single application. The current task can be interrupted by a hardware signal called an *interrupt*. Interrupts are triggered by external events, such as a network packet being received; or internal events, such as the system clock (called a *tick* in Linux). When an interrupt happens, the current running process is suspended, and a routine is run to determine what the system should do next.

When the currently running process has exceeded its allotted time, the kernel can swap it out for another process using a procedure called a *context switch*. A process can be switched out before its allotted time if the process issues any I/O commands such as a read to disk. The computer is so much faster than the disk that the CPU can run other tasks while waiting for the suspended process's disk request to return.

When talking about the CPU of a Linux system, you should be concerned with several factors. The first is the percentage of time the CPU is idle compared to the time it's doing work (in reality, the CPU is always doing *something*—it's considered idle if no tasks are waiting to be executed). The CPU is running at maximum when the idle percentage is zero. The non-idle part of the CPU is split into system and user time, where *system time* refers to the time spent in the kernel, and *user time* is the time spent doing work requested by the user. The idle time is split into the time the kernel is idle because it has nothing to do, and the time it's idle because it's waiting on some bit of I/O.

Measuring these counters is tricky because getting an accurate number would require the CPU to spend all of its time determining what it's doing! The kernel checks the current status (system, user, iowait, idle) about 100 times per second and uses these measurements to calculate the percentages.

Another metric that Linux uses to convey CPU usage is the *load average*. This metric doesn't tie directly to the CPU utilization; it represents an exponential weighting of the number of tasks in the kernel's run queue for the past minute, 5 minutes, and 15 minutes. This metric is investigated more closely later.

Other things to consider about the kernel are the interrupt load and the context switches. There are no upper bounds on these figures, but the more interrupts and context switches are performed, the less time the CPU has to do the user's work.

Memory

The system has two types of memory: real memory and swap space. *Real memory* refers to the sticks of RAM on the motherboard. *Swap space* is a temporary holding spot that is used when the system tries to allocate more RAM than physically exists. In this situation, pages of RAM are swapped to disk to free up space for the current allocation. The data is swapped back to RAM when the data is needed again.

RAM can be used by applications, by the system, or not at all. The system uses RAM in two ways: as a buffer for raw disk blocks (incoming or outgoing) and as a file cache. The sizes of buffers and cache are dynamic so that memory can be given back to applications if needed. This is why most people see their Linux systems as having no free memory: the system has allocated the unused memory for buffers and cache.

Swap memory resides on disk. A lot of swapping slows things down and is a sign that the system is out of RAM.

Disk

Disk is where long-term data is stored, such as on a hard drive, a flash disk, or tape (collectively referred to as *block devices*). One exception is a RAM disk, which behaves like a block device but resides in RAM; this data is gone when the system shuts down. The most prevalent form of disk is the hard drive, so the discussion of disk in this tutorial focuses on this medium.

Two categories of measurements are used to describe disk: space and speed. The *free space* on a disk refers to the number of bytes on the disk that are available for use. The *overhead* on a disk includes any space used by the file system or that is otherwise unavailable for use. Keep in mind that most manufacturers report disks in terms of 1,000,000,000-byte gigabytes, whereas your operating system uses the base 2 value of 1,073,741,824; this results in a 93% "loss" for the consumer. This isn't overhead, but if you don't account for it, your calculations will be incorrect.

The second metric of a disk is *speed*, which measures how fast data is returned from the disk. When the CPU issues a request, several things must come together to get the data back to the CPU:

1. The kernel puts the request in a queue, where it waits for its turn to be sent to disk (wait time).
2. The command is sent to the disk controller.
3. The disk seeks the disk heads to the required block (seek time).
4. The disk heads read the data from disk.
5. The data is returned to the CPU.

Each of these steps is measured differently, or sometimes not at all. The *service time* encompasses the last three steps and represents how long a request takes to service once the request has been issued. The *wait time* represents the entire procedure, which includes the time in queue and the service time.

One bit of optimization the kernel performs is to reorder and merge requests in the queue from step 1 to minimize the number of disk seeks. This is called an *elevator*, and several different algorithms have been used over the years.

Network

Linux plays two broad roles with respect to the network: a client, where packets are sent and received by applications on the server; and a router (or firewall, or bridge). Packets are received on one interface and sent out on another (perhaps after some filtering or inspection has happened).

Networks are most often measured in terms of bits per second (or kilobits, megabits, or gigabits) and in packets per second. Measuring packets per second is often less useful because computers have a fixed overhead per packet, resulting in poorer throughput at smaller packet sizes. Don't confuse the speed of the network card (100Mbit/sec, or gigabit) with the expected speed of data transfers from or through the machine. Several outside factors come into play, including latency and the remote side of the connection, not to mention tuning on the server.

Queues

Queues don't fit well with the other resources, but they appear so often in performance monitoring that must be mentioned. A *queue* is a line in which requests wait until they're processed. The kernel uses queues in a variety of ways, from the run queue that holds the list of processes to be run, to disk queues, network queues, and hardware queues. Generally, a queue refers to a spot in memory that the kernel

uses to keep track of a particular set of tasks, but it can also refer to a piece of memory on a hardware component that is managed by the hardware.

Queues appear two ways in performance tuning. First, when too much work comes into a queue, any new work is lost. For example, if too many packets come into a network interface, some get dropped (in networking circles, this is caused a *tail drop*). Second, if a queue is being used excessively (or, sometimes, not enough), then another component isn't performing as well as necessary. A large number of processes in the run queue on a frequent basis may mean the CPU is overloaded.

Measure performance

Several tools are available to measure performance on a Linux box. Some of these tools measure CPU, disk, memory, and network directly, and others show indicators such as queue usage, process creation, and errors. Some tools show instantaneous values, and some show values that have been averaged over a period of time. It's equally important to understand both how the measurement was taken and what is being measured.

vmstat

`vmstat` is a helpful tool for showing the most-often-used performance metrics in real time. The most important thing to understand about `vmstat` is that the first display it produces represents the average values since the system was booted, which you should generally ignore. Specify a repeat time (in seconds) at the command line to have `vmstat` repeatedly report the information using current data. Listing 1 shows the output of `vmstat 5`.

Listing 1. The output of `vmstat 5`

```
# vmstat 5
procs -----memory----- ---swap-- -----io----- --system--
-----cpu-----
 r  b   swpd   free   buff  cache   si   so   bi   bo   in   cs us sy
id wa st
0  3  17780  10304  18108 586076   0   0  2779  332   1   1  3  4
76 17  0
1  2  17780  10088  19796 556172   0   0  7803  3940 2257 4093 25 28
14 34  0
0  2  17780   9568  19848 577496   0   0 18060  1217 1610  910  0  3
48 49  0
0  0  17780  51696  20804 582396   0   0  9237  3490 1736  839  0  3
55 41  0
```

Listing 1 shows the output of the `vmstat` command, with measurements taken every 5 seconds. The first line of values represents the average since the system booted, so it should be ignored. The first two columns refer to processes. The number under the `r` heading is the number of processes in the run queue at the time

of the measurement. Processes in the run queue are waiting on the CPU. The next column is the number of processes *blocked on I/O*, meaning they're sleeping until some piece of I/O is returned, and they can't be interrupted.

The columns under the **memory** heading are instantaneous measurements about the system memory and are in kilobytes (1024 bytes). **swpd** is the amount of memory that has been swapped to disk. **free** is the amount of free memory that isn't used by applications, buffers, or cache. Don't be surprised if this number is low (see the discussion on [free](#) for more information about what free memory really is). **buff** and **cache** indicate the amount of memory devoted to buffers and cache. Buffers store raw disk blocks, and cache stores files.

The first two categories are instantaneous measurements. It's possible that for a brief period, all free memory was consumed but returned before the next interval. The rest of the values are averaged over the sampling period.

swap is the average amount of memory swapped in from disk (**si**) and out to disk (**so**) per second; it's reported in kilobytes. **io** is the number of disk blocks per second read in from all block devices and sent out to block devices.

The **system** category describes the number of interrupts per second (**in**) and context switches (**cs**) per second. Interrupts come from devices (such as a network card signaling the kernel that a packet is waiting) and the system timer. In some kernels, the system timer fires 1,000 times per second, so this number can be quite high.

The final category of measurements shows what's going on with the CPU, reported as a percent of total CPU time. These five values should add to 100. **us** is the average time the CPU spent on user tasks over the sampling period, and **sy** is the average time the CPU spent on system tasks. **id** is the time the CPU was idle, and **wa** is the time the CPU was waiting for I/O (Listing 1 was taken from a heavily I/O bound system, you can see that 34-49% of the CPU's time is spent waiting for data to return from disk). The final value, **st** (the *steal* time) is for servers running a hypervisor and virtual machines. It refers to the percentage of time the hypervisor could have run a virtual machine but had something else to do.

As you can see from [Listing 1](#), `vmstat` provides a wealth of information across a broad spectrum of metrics. If something is going on while you're logged in, `vmstat` is an excellent way to narrow down the source.

`vmstat` can also show some interesting information about disk usage on a per-device basis, which can shed more light on the swap and io categories from Listing 1. The `-d` parameter reports some disk statistics, including the total number of reads and writes, on a per-disk basis. Listing 2 shows part of the output from `vmstat -d 5` (with unused devices filtered out).

Listing 2. Using vmstat to show disk usage

```

disk- -----reads----- writes-----
-----IO-----
      total merged sectors      ms  total merged sectors      ms      cur
sec
hda   186212  28646 3721794  737428 246503 4549745 38981340 8456728
0     2583
hdd   181471  27062 3582080  789856 246450 4549829 38981624 8855516
0     2652
    
```

Each disk is displayed on a separate line, and the output is broken down into reads and writes. Reads and writes are further split into the total numbers of requests issued, how many requests were merged in the disk elevator, the number of sectors read from or written to, and the total service time. All these numbers are counters, so they will increase until the next reboot, as opposed to the average values seen without the `-d` option.

The final group of measurements in Listing 2, under the **IO** heading, show the current number of I/O operations in progress for the disk and the total number of seconds spent in I/O since boot.

Listing 2 shows that read volume is similar across the two disks and that write volume is almost identical. These two disks happen to form a software mirror, so this behavior is expected. The information from Listing 2 can be used to indicate slower disks or disks with higher usage than others.

iostat

Closely tied to the `vmstat -d` example from Listing 2 is `iostat`. This command provides details about disk usage on a per-device basis. `iostat` improves on `vmstat -d` by giving more details. Just as with `vmstat`, you can pass a number to `iostat` that indicates the refresh interval. Also, as with `vmstat`, the first output represents the values since the system was started and therefore is usually ignored. Listing 3 shows the output of `iostat` with 5-second intervals.

Listing 3. Output of the iostat command

```

$ iostat 5
Linux 2.6.20-1.3002.fc6xen (bob.ertw.com)      02/08/2008

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.85    0.13   0.35   0.75   0.01   97.90

Device:            tps    Blk_read/s    Blk_wrtn/s    Blk_read    Blk_wrtn
hda                 1.86      15.24         13351         4740568     41539964
hdd                 1.85      14.69         133.51        4570088     41540256
    
```

The first part of every measurement interval shows the CPU usage, which is also shown by `vmstat`. However, two decimal places are presented here. The second

part of the output shows all the block devices on the system (to limit the number of devices shown, pass the names of the devices on the command line, such as `iostat 5 hda sda`). The first column, **tps**, represents the transfers per second to the device after the requests have been merged by the elevator. The sizes of the transfers aren't specified. The last four columns deal in 512-byte blocks and show the blocks read per second, written per second, total blocks read, and total blocks written, respectively. If you'd rather see values reported in kilobytes or megabytes, use `-k` or `-m`, respectively. The `-p` option displays details down to the partition level, if you need that data.

You can get a great deal more information by using the `-x` parameter, which is shown in Listing 4. Listing 4 also limits the output to one drive. The formatting was adjusted to fit page-width constraints.

Listing 4. Extended information from iostat

```
# iostat -x 5 hda
..... CPU information removed ...
Device:      rrqm/s   wrqm/s   r/s     w/s     rsec/s   wsec/s  avgrq-sz
hda          16669.31  1.49    756.93  1.49    139287.13  27.72   18369
              avgqu-sz   await   svctm   %util
              1.58     208     1.28    96.83
```

The first six values are concerned with reads and writes per second. **rrqm/s** and **wrqm/s** refer to the number of read and write requests that were merged. By contrast, **r/s** and **w/s** represent the number of reads and writes sent to disk. Therefore, the percentage of disk requests merged is $16669 / (16669 + 757) = 95\%$. **rsec/s** and **wsec/s** show the read and write rate in terms of sectors per second.

The next four columns display information about disk queues and times. **avgrq-sz** is the average size of requests issued to the device (in sectors). **avgqu-sz** is the average length of the disk queue over the measurement interval. The **await** is the average wait time (in milliseconds), which represents the average time a request takes from being sent to the kernel to the time it's returned. **svctm** is the average service time (in milliseconds), which is the time a disk request takes from when it's out of the queues and sent to disk to the time it's returned.

The final value, **%util**, is the percentage of time the system was performing I/O on that device, also referred to as the *saturation*. The 96.83% reported in Listing 4 shows that the disk was almost at capacity during that time.

mpstat

`mpstat` reports detailed information about the CPU (or all CPUs in a multiprocessor machine). Much of this information is reported by `iostat` and `vmstat` in some form, but `mpstat` provides data for all processors separately. Listing 5 shows `mpstat` with 5-second measurement intervals. Unlike with `iostat` and `vmstat`,

you shouldn't ignore the first line.

Listing 5. Showing CPU information with mpstat

```
# mpstat -P 0 5
Linux 2.620-1.3002.fc6xen (bob.ertw.com) 02/09/2008

09:45:23 PM CPU %user %nice %sys %iowait %irq %soft %steal
%idle intr/s
09:45:25 PM 0 77.61 21.89 0.00 0.00 0.50 0.00 0.00
0.00 155.22
09:45:27 PM 0 68.16 30.85 1.00 0.00 0.00 0.00 0.00
0.00 154.73
```

The addition of `-P 0` specifies that the first CPU (starting at 0) should be shown. You can also specify `-P ALL` for all CPUs separately. The fields returned by `mpstat` are as follows:

- **%user**: The percentage of time spent in user tasks, excluding the nice tasks
- **%nice**: The percentage of time spent in nice (lower priority) user tasks
- **%sys**: The percentage of time spent in kernel tasks
- **%iowait**: The percentage of time spent waiting for I/O while idle
- **%irq**: The percentage of time servicing hardware interrupts
- **%soft**: The percentage of time spent in software interrupts
- **%steal**: The percentage of time the hypervisor stole from a virtual machine
- **intr/s**: The average number of interrupts per second

pstree

Understanding which processes spawned another process is helpful when you're tracking down resource usage. One way to find this is to use the output of `ps -ef` and use the parent processid to work your way back to PID 1 (`init`). You can also use `ps -efjH`, which sorts the output into a parent-child tree, and include CPU time usage.

A utility called `pstree` prints the process tree in a more graphical format and also rolls multiple instances of the same process into one line. Listing 6 shows the output of `pstree` after it's passed the PID of the Postfix daemon.

Listing 6. pstree output

```
[root@sergeant ~]# pstree 7988
master###anvil
  ##cleanup
  ##local
  ##pickup
  ##proxymap
  ##qmgr
  ##2*[smtpd]
  ##2*[trivial-rewrite]
```

The master process, cleverly called **master**, has spawned several other processes such as **anvil**, **cleanup**, and **local**. The last two lines of the output are of the format $N*[something]$, where *something* is the name of a process, and N is the number of children by that name. If *something* was enclosed in curly brackets ({}), that would indicate N threads running (`ps` doesn't normally show threads unless you use `-L`).

w, uptime, and top

These utilities are grouped together because they're the first utilities people tend to reach for when investigating a problem. Listing 7 shows the output of the `w` command.

Listing 7. Output of the w command

```
# w
12:14:15 up 33 days, 15:09,  2 users,  load average: 0.06, 0.12, 0.09
USER      TTY      FROM          LOGIN@      IDLE        JCPU      PCPU  WHAT
root      tty2     -             17Jan08     18days     0.29s     0.04s  login --
root
root      pts/0    bob           Sat22       0.00s      0.57s     0.56s  -bash
```

The first line of `w` provides the bulk of the information. The first part, "12:14:15 up 33 days, 15:09" gives the current time, followed by the uptime of 33 days, 15 hours, and 9 minutes. The second part, "2 users", gives the number of logged-in users. The final part is the load average, given as a 1-minute, 5-minute, and 15-minute averaging.

The load average is a weighted average of the number of processes in the run queue over a given period of time. The higher the load average, the more processes are trying to contend for the CPUs. Load averages aren't normalized to the number of CPUs, meaning that the load average and the number of CPUs aren't related.

To use the load average, you must also understand the weighting. The load average is updated every 5 seconds, with the older information playing a less prominent role in the calculation. If your system were to go immediately from 0 processes in the run queue to 1 process, a graph of the 1-minute load average over the next minute would be not a straight line but a curve that rises quickly at first and then tapers off until the 60-second mark. For a more detailed look at how the load average calculation is done, see the [Resources](#) section.

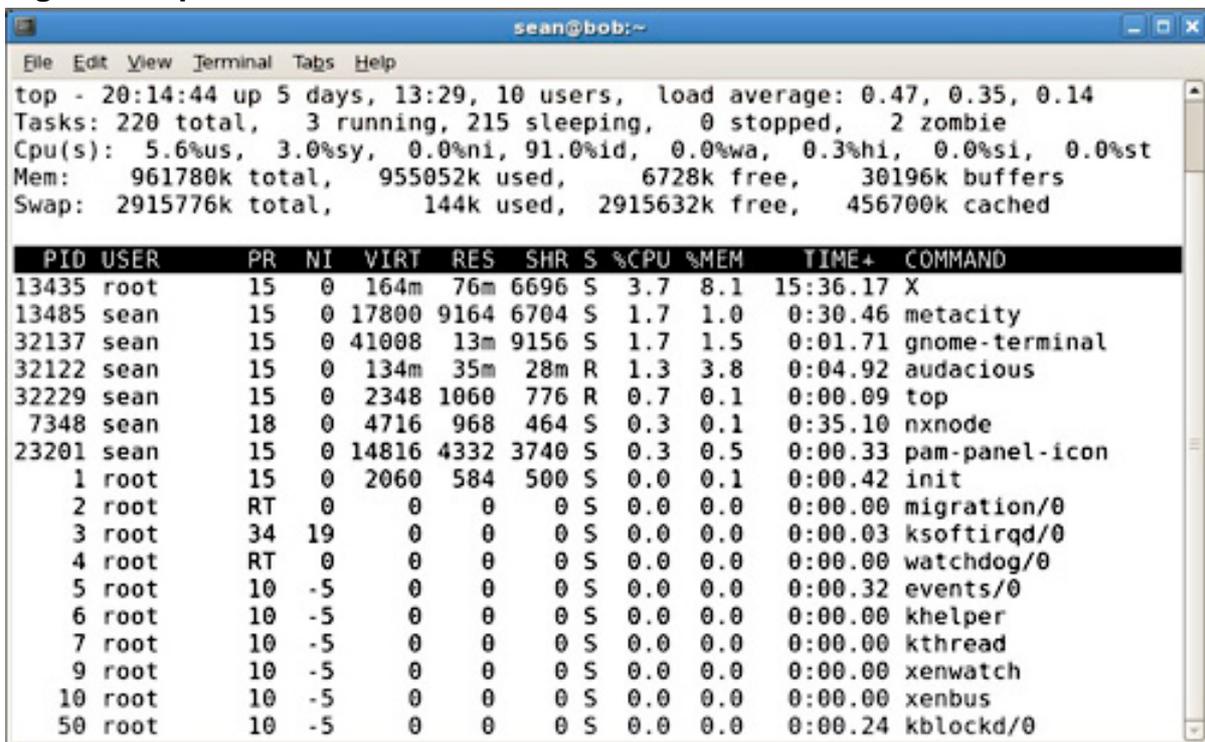
The practical application of weighting the load average is that variations in the actual load at the time of measurement are smoothed out; but the current state is reflected more in the numbers, especially the 1-minute average.

After the first line comes a list of logged-in users, including their login time, their location, and information about CPU usage. The first user, root, is logged in from **tty2** (a local console) and has been idle for 18 days. The second user is again root, but logged in over the network and currently at the shell. The **JCPU** and **PCPU** columns give you an idea of how much CPU time the user has used; the first column includes jobs in the past, whereas **PCPU** is for the process the user is currently using.

The output of `uptime` shows exactly the same first line of `w`, but no information about the users. In practical terms, `w` is the more helpful of the two because of the additional information about users and because it's shorter to type!

Another popular command is `top`, which shows a continuously updating list of the top processes (sorted by either memory or CPU usage), in addition to some other metrics. Figure 1 shows a screenshot of `top` in action.

Figure 1. top in action



The first line shows the same thing as `uptime`, such as the uptime and load average. The second line is the number of processes. *Running* processes are in the run queue; *sleeping* processes are waiting for something to wake them up. A *stopped* process has been paused, likely because it's being traced or debugged. A

zombie process has exited, but the parent process hasn't acknowledged the death.

Something doesn't add up

$VIRT = RES + SWAP$, which means that $SWAP = VIRT - RES$. Looking at PID 13435, you can see that VIRT is 164m, and RES is 76m, meaning SWAP must be 88m. However, the swap stats from the top of the screen indicate that only 144K of swap are used! This can be verified by using the `f` key when inside `top` and enabling more fields, such as swap.

As it turns out, swap means more than just pages swapped to disk. An application's binary and libraries need not stay in memory the whole time. The kernel can mark some of the memory pages as unnecessary at the time; but because the binary is at a known location on disk, there is no need to use the swap file. This is still counted as swap because the part of the code isn't resident. In addition, memory can be mapped to a disk file by the application. Because the whole size of the application (VIRT) includes the mapped memory, but it isn't resident (RES), it's counted as swap.

The third line gives the CPU usage: in order, you see user, system, niced, idle, I/O wait, hardware interrupt, software interrupt, and steal time. These numbers are percentages of time in the last measurement interval (3 seconds by default).

The last two lines of the top section show memory statistics. The first gives information about real memory; in [Figure 1](#), you can see the system has 961,780K of RAM (after the kernel overhead). All but 6,728K have been used, with around 30MB of buffers and 456M of cache (cache is displayed at the end of the second line). The second line displays the swap usage: the system has almost 3G of swap, with only 144K used.

The rest of the screen shows information about the currently running processes. `top` displays as many processes as possible to fill the size of the window. Each process gets its own line, and the list is updated every measurement interval with the tasks that used the most CPU at the top. The columns are as follows:

- **PID:** The processid of the process
- **USER:** The effective username of the process (if the program uses `setuid(2)` to change the user, the new user is displayed)
- **PR:** The priority of the task, used by the kernel to determine which process gets the CPU first
- **NI:** The nice level of the task, set by the system administrator to influence which processes get the CPU first
- **VIRT:** The size of the process's virtual image, which is the sum of the space used by RAM (the resident size) and the size of the data in swap

(swapped size)

- **RES:** The resident size of the process, which is the amount of real RAM used by your process
- **SHR:** The amount of memory shared by the application, such as SysV shared memory or dynamic libraries (*.so)
- **S:** The state, such as sleeping, running, or zombie
- **%CPU:** The percentage of CPU used in the last measurement interval
- **%MEM:** The percentage of RAM (excluding swap) used when last measured
- **TIME+:** The time in minutes:seconds:hundredths used by the process
- **COMMAND:** The name of the command running

`top` provides a fast way to see which processes are using the most CPU and also gives you a good dashboard view of the system's CPU and memory. You can have `top` sort by memory usage by typing **M** in the `top` display.

free

After you've seen `top`, you should understand `free` right away. Listing 8 shows the output of `free`, using the `-m` option to report all values in megabytes.

Listing 8. Using the free command

```
# free -m
              total        used        free      shared    buffers
cached
Mem:           939          904           34           0          107
310
-/+ buffers/cache:      486          452
Swap:          2847           0          2847
```

`free` lays out the memory usage in a few directions. The first line shows the same information you saw for `top`. The second line represents used and free memory without considering buffers and cache. In Listing 8, 452M of memory is free for an application to use; this memory will come from the free memory (34M), buffers (107M), or cache (310M).

The final line shows the same swap statistics as `top`.

Show network statistics

Getting network statistics is less direct than for CPU, memory, and disk. The primary method is to read counters from `/proc/net/dev`, which reports transfers per interface on both a packet and byte basis. If you want a per-second value, you must calculate

it yourself by dividing the difference between two successive measurements by the interval. Alternatively, you can use a tool like `bwm` to automate the collection and display of total bandwidth. Figure 2 shows `bwm` in action.

Figure 2. Using the `bwm` command

```

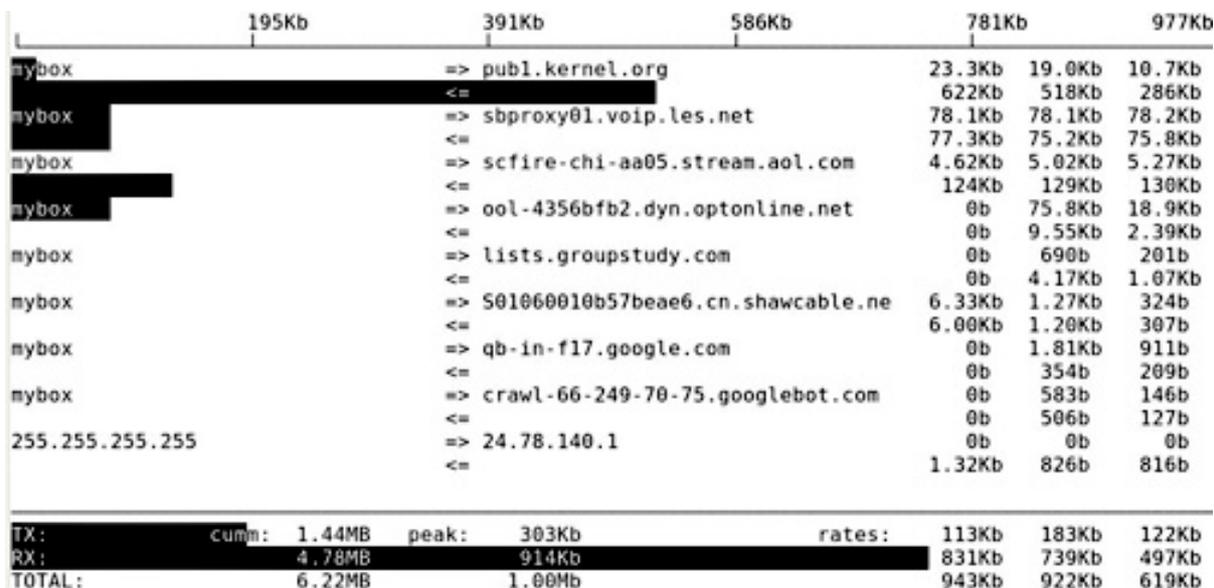
bwm-ng v0.5 (probing every 0.500s), press 'h' for help
input: libstatgrab type: rate
-
  iface          Rx          Tx          Total
-----
  lo:            0.00 KB/s   0.00 KB/s   0.00 KB/s
  peth0:        20.02 KB/s   1.00 KB/s  21.02 KB/s
  virbr0:       0.00 KB/s   0.00 KB/s   0.00 KB/s
  vif0.0:       1.00 KB/s  20.02 KB/s  21.02 KB/s
  eth0:        20.02 KB/s   1.00 KB/s  21.02 KB/s
  xenbr0:       0.00 KB/s   0.00 KB/s   0.00 KB/s
-----
  total:       41.04 KB/s  22.02 KB/s  63.07 KB/s

```

`bwm` shows interface usage in a variety of ways. Figure 2 shows the instantaneous rate every half second, although 30-second averages, maximum bandwidth, and counting bytes are available. You can see from Figure 2 that `eth0` is receiving about 20K/sec of traffic, which seems to be coming from `vif0.0`. If bytes per second aren't what you're looking for, you can cycle between bits, packets, and errors with the `u` key.

To get more details about which hosts are responsible for the traffic, you need `iftop`, which provides a `top`-like interface for your network traffic. The kernel doesn't provide this information directly, so `iftop` uses the `pcap` library to inspect packets on the wire, which requires root privileges. Figure 3 shows `iftop` in action, when attached to the `eth2` device.

Figure 3. The output of `iftop -i eth2`



iftop shows the top talkers on your network. By default, each conversation takes two lines: one for the sending half and the other for the receiving half. Looking at the first conversation from mybox to publ.kernel.org, the top row shows the traffic sent from mybox, and the second line shows the traffic received by mybox. The numbers to the right indicate the average traffic over the last 2 seconds, 10 seconds, and 40 seconds, respectively. You can also see a black bar overlaying the hostnames, which is a visual indication of the 10-second average (the scale is displayed at the top of the screen).

Looking more closely at Figure 3, the first transfer is probably a download because of the large amount of traffic received (averaging around half a megabit per second over the last 10 seconds) compared to the small amount of upload traffic. The second talker has an equal amount of traffic, which has been relatively steady at around 75-78k/sec. This is a G.711 voice call through les.net, my VoIP provider. The third transfer shows a 128K download with a small upload: it was an Internet radio stream.

The choice of interface you attach to is important. Figure 3 uses the outside interface on a firewall, which sees all packets after they have passed through IP masquerading. This causes the internal address to be lost. Using a different interface, such as an internal interface, would preserve this information.

sar

sar is the topic of an entire article (see the [Resources](#) section). sar measures dozens of key metrics every 10 minutes and provides a method to retrieve the measurements. You can use the preceding tools to determine "what is going on now?"; sar answers, "what happened this week?" Note that sar purges its data to keep only the last 7 days of data.

You must configure data collection by adding two lines to root's crontab. Listing 9 shows a typical crontab for `sar`.

Listing 9. root's crontab for sar data collection

```
# Collect measurements at 10-minute intervals
0,10,20,30,40,50 * * * * /usr/lib/sa/sa1 -d 1 1
# Create daily reports and purge old files
0 0 * * * /usr/lib/sa/sa2 -A
```

The first line executes the `sa1` command to collect data every 10 minutes; this command runs `sadc` to do the actual collection. This job is self-contained: it knows which file to write to and needs no other configuration. The second line calls `sa2` at midnight to purge older data files and collect the day's data into a readable text file.

It's worth checking to see how your system runs `sar` before you rely on the data. Some systems disable collection of disk statistics; to fix this, you must add `-d` to the call to `sa1` (Listing 9 has this added).

With some data collected, you can now run `sar` without any options to see the day's CPU usage. Listing 10 shows part of the output.

Listing 10. Sample output from sar

```
[root@bob cron.d]# sar | head
Linux 2.6.20-1.3002.fc6xen (bob.ertw.com) 02/11/2008

12:00:01 AM      CPU      %user      %nice      %system      %iowait      %steal
%idle
12:10:01 AM      all       0.18       0.00       0.18       3.67       0.01
95.97
12:20:01 AM      all       0.08       0.00       0.04       0.02       0.01
99.85
12:30:01 AM      all       0.11       0.00       0.03       0.02       0.01
99.82
12:40:01 AM      all       0.12       0.00       0.02       0.02       0.01
99.83
12:50:01 AM      all       0.11       0.00       0.03       0.05       0.01
99.81
01:00:01 AM      all       0.12       0.00       0.02       0.02       0.01
99.83
01:10:01 AM      all       0.11       0.00       0.02       0.03       0.01
99.83
```

The numbers shown in Listing 10 should be familiar by now: they're the various CPU counters shown by `top`, `vmstat`, and `mpstat`. You can view much more information by using one or more of the command-line parameters shown in Table 3.

Table 3. A synopsis of sar options

Option	Example	Description
<code>-A</code>	<code>sar -A</code>	Displays <i>everything</i> . Unless you're dumping

		this result to a text file, you probably don't need it. If you do need it, this process is run nightly as part of <code>sa2</code> anyway.
<code>-b</code>	<code>sar -b</code>	Shows transactions and blocks sent to, and read from, block devices, much like <code>iostat</code> .
<code>-B</code>	<code>sar -B</code>	Shows paging (swap) statistics such as those reported by <code>vmstat</code> .
<code>-d</code>	<code>sar -d</code>	Shows disk activity much like <code>iostat -x</code> , which includes wait and service times, and queue length.
<code>-n</code>	<code>sar -n DEV</code> or <code>sar -n NFS</code>	Shows interface activity (like <code>bwm</code>) when using <code>DEV</code> , or NFS client statistics when using the <code>NFS</code> keyword (use the <code>NFSD</code> keyword for the NFS server daemon stats). The <code>EDEV</code> keyword shows error information from the network cards.
<code>-q</code>	<code>sar -q</code>	Shows information about the run queue and total process list sizes, and load averages, such as those reported by <code>vmstat</code> and <code>uptime</code> .
<code>-r</code>	<code>sar -r</code>	Shows information about memory, swap, cache, and buffer usage (like <code>free</code>).
<code>-f</code>	<code>sar -f /var/log/sa/sa11</code>	Reads information from a different file. Files are named after the day of the month.
<code>-s</code>	<code>sar -s 08:59:00</code>	Starts displaying information at the first measurement after the given time. If you specify <code>09:00:00</code> , the first measurement will be at <code>09:10</code> , so subtract a minute from the time you want.

<code>-e</code>	<code>sar -e 10:01:00</code>	Specifies the cutoff for displaying measurements. You should add a minute to the time you want, to make sure you get it.
-----------------	------------------------------	--

You may also combine several parameters to get more than one report, or a different file with a starting and an ending time.

df

Your hard disks are a finite resource. If a partition runs out of space, be prepared for problems. The `df` command shows the disk-space situation. Listing 11 shows the output of `df -h`, which forces output to be in a more friendly format.

Listing 11. Checking disk space usage with `df -h`

```
$ df -h
Filesystem                Size      Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol100
                          225G    169G    44G   80% /
/dev/hda1                  99M      30M    64M   32% /boot
tmpfs                      474M          0   474M    0% /dev/shm
```

Listing 11 shows a single file system on the root that is 225G in size, with 44G free. The `/boot` partition is 99M with 64M free. `tmpfs` is a special file system and doesn't refer to any particular device. If a partition is full, you see no available space and 100% usage.

Section 3. Troubleshoot resource problems

This section covers material for topic 306.2 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 4.

In this section, learn how to:

- Match/correlate system symptoms with likely problems
- Identify bottlenecks in a system

The previous section showed how to display different performance counters within the Linux system. It's now time to apply these commands to solving resource-related

problems in your systems.

Troubleshooting methodology

Lay out your strategy for problem solving before getting into the details of resource constraints and your system. The many strategies for problem solving boil down to four steps:

1. Identify the symptoms.
2. Determine the root cause.
3. Implement a fix.
4. Evaluate the results.

Identify the symptoms

The first step to solving a problem is to identify the symptoms. An example of a symptom is "e-mail is slow" or "I'm out of disk space." The symptoms are causing your users to complain, and those users won't be happy until the symptoms go away. Don't confuse the symptoms with the problem, though: most often, the problem is different than what your users are reporting, although the problem is causing the symptoms.

Once you've collected the symptoms, try to quantify the complaint and clarify the conditions under which it occurs. Rather than the e-mail system being slow, you may learn that e-mails used to be received within seconds of being sent but now take hours. And a user who is out of disk space has to be doing something, such as saving a file or processing a batch job.

This final step of quantifying the complaint has two purposes. The first is to let you reproduce the problem, which will help later in determining when the problem has been solved. The second purpose is to get more details from the user that will help you determine the root cause. After learning that a job that once took 5 minutes to execute now takes an hour, you might inquire about the nature of the job. This might lead you to learn that it pulls information from a database on another server, which you must include in the scope of your search for the root cause.

Determine the root cause

Determining the root cause involves using the commands learned in the [Measure resource usage](#) section to find the cause of the problem. To do so, you must investigate the resources, such as CPU, memory, disk, and network. Ideally, you'll be able to collect data while the problem is occurring with real-time tools like

`vmstat`, `iostat`, and `top`. If not, something that produces historical information such as `sar` may have to do.

If the problem is resource related, then one of two results will appear: one (or more) of your resources will be at 100% utilization, and the cause should be obvious; or nothing is obviously being overused.

In the second case, you should refer to a *baseline*. The baseline is a set of reference data you can use to compare what's "normal" to what you're seeing. Your baseline might be a series of graphs, or archived `sar` reports showing normal activity. The baseline will also be helpful later when you learn about predicting growth.

As you use the administration tools, you'll start to develop a picture of the problem's root cause. It may be that your mail server is stuck on a message, causing it to stop processing other messages. Or perhaps a batch job is consuming all the CPU on the system.

At this point, you must be careful that you have properly identified the root cause of the problem. An application that generates a huge logfile may have caused you to run out of space. If you identify the logfile as the root cause and decided to delete it, the application may still fill up the disk at some point in the future.

Implement a fix

You'll often have several ways to fix a problem. Take, for instance, a batch job that is consuming all CPU resources. If you kill the job, then the user running the job will probably lose his or her work, although the other users will have their server back. You may decide to renice the process to give other processes more time on the CPU. This is usually a judgment call, depending on the needs of the business and the urgency of the situation.

Evaluate the results

Once you've implemented your solution, you must go back and check to see if the problem was solved. Are e-mails delivered near-instantly? Are users able to log in? If not, then you must step back and look at the root cause again, which will lead to another fix that must be evaluated. If your fix failed, you must also check to see that you didn't make things worse!

After the problem is fixed, determine if you need to take any longer-term actions. Do you have to consider a bigger disk, or moving a user's batch job to another host? Unexplained processes on a machine may prompt you to do a more in-depth security check of the server to make sure it hasn't been exploited.

Compound problems

Some performance problems are obvious. A user complains that something is running slowly: you log in and fire up `top`. You see a process unnecessarily hogging the CPU: you kill it, and the system returns to normal. After showering you with praise, your boss gives you a raise and the rest of the day off. (OK, maybe the last part is made up.)

What happens when your problem isn't obvious? Sometimes problems are caused by more than one thing, or a symptom is caused by something that may seem unrelated at first.

The swap spiral

Memory is fast, and you probably have lots of it in your system. But sometimes an application needs more memory than the system has, or a handful of processes combined end up using more memory than the system has. In this case, virtual memory is used. The kernel allocates a spot on disk and swaps the resident memory pages to disk so that the active application can use it. When the memory on disk is needed, it's brought back to RAM, optionally swapping out some other memory to disk to make room.

The problem with that process is that disk is slow. If you briefly dip into swap, you may not notice. But when the system starts aggressively swapping memory to disk in order to satisfy a growing demand for memory, you've got problems. You'll find your disk I/O skyrocketing, and it will seem that the system isn't responding. In fact, the system probably isn't responding, because your applications are waiting for their memory to be transferred from disk to RAM.

UNIX admins call this the *swap spiral* (or sometimes, more grimly, the *swap death spiral*). Eventually, the system grinds to a halt as the disks are running at capacity trying to swap memory in and out. If your swap device is on the same physical disk as data, things get even worse. Once your application makes it onto the CPU and issues an I/O request, it has to wait longer while the swap activity is serviced.

The obvious symptom of the swap spiral is absurdly long waits to do anything, even getting the uptime. You also see a high load average, because many processes are in the run queue due to the backed-up system. To differentiate the problem from a high-CPU problem, you can check `top` to see if processes are using the CPU heavily, or you can check `vmstat` to see if there is a lot of swap activity. The solution is usually to start killing off processes until the system returns to order, although depending on the nature of the problem, you may be able to wait it out.

Out of disk space

Applications aren't required to check for errors. Many applications go through life assuming that every disk access executes perfectly and quickly. A disk volume that fills up often causes applications to behave in weird ways. For example, an

application may consume all available CPU as it tries to do an operation over and over without realizing it's not going to work. You can use the `strace` command to see what the application is doing (if it's using system calls).

Other times, applications simply stop working. A Web application may return blank pages if it can't access its database.

Logging in and checking your available disk (with `du`) is the quickest way to see if disk space is the culprit.

Blocked on I/O

When a process requests some form of I/O, the kernel puts the process to sleep until the I/O request returns. If something happens with the disk (sometimes as part of a swap spiral, disk failure, or network failure on networked file systems), many applications are put to sleep at the same time.

A process that's put to sleep can be put into an interruptible sleep or an uninterpretable sleep. The former can be killed by a signal, but the second can't. Running `ps aux` shows the state. Listing 12 shows one process in uninterpretable sleep and another in interruptible sleep.

Listing 12. Two processes in a sleep state

```
apache  26575  0.2 19.6 132572 50104 ?          S    Feb13   3:43
/usr/sbin/httpd
root    8381  57.8  0.2   3844   532 pts/1    D    20:46   0:37 dd
```

The first process in Listing 12, `httpd`, is in an interruptible sleep state, indicated by the letter `S` just after the question mark. The second process, `dd`, is in an uninterpretable sleep state. Uninterpretable sleeps are most often associated with hard disk accesses, whereas interruptible sleeps are for operations that take comparably longer to execute, such as NFS and socket operations.

If you find a high load average (meaning a lot of processes in the run queue) and a lot of processes in an uninterpretable sleep state, then you may have a problem with hard drive I/O, either because the device is failing or because you're trying to get too many reads/writes out of the drive at a time.

Section 4. Analyze demand

This section covers material for topic 306.3 for the Senior Level Linux Professional

(LPIC-3) exam 301. This topic has a weight of 2.

In this section, learn how to:

- Identify capacity demands
- Detail capacity needs of programs
- Determine CPU/memory needs of programs
- Assemble program needs into a complete analysis

Fixing immediate problems is a key task for the system admin. Another task involves analyzing how systems are currently performing in the hope that you can foresee resource constraints and address them before they become a problem. This section looks at analyzing the current demand, and the next section builds on that to predict future usage.

You can use two approaches to analyze current demand: measure the current demand over a period of time (like a baseline), or model the system and come up with a set of parameters that makes the model reflect current behavior. The first approach is easier and reasonably good. The second is more accurate but requires a lot of work. The real benefit of modeling comes when you need to predict future behavior. When you have a model of your system, you can change certain parameters to match growth projections and see how performance will change.

In practice, both of these approaches are used together. In some cases, it's too difficult to model a particular system, so measurements are the only basis on which to base demand and growth projections. Measurements are still required to generate models.

Model system behavior

The activity in a computer can be modeled as a series of *queues*. A queue is a construct where requests come in one end and are held until a resource is available. Once the resource is available, the task is executed and exits the queue.

Multiple queues can be attached together to form a bigger system. A disk can be modeled as a queue where requests come in to a buffer. When the request is ready to be serviced, it's passed to the disk. This request generally comes from the CPU, which is a single resource with multiple tasks contending for the use of the CPU. The study of queues and their applications is called *queuing theory*.

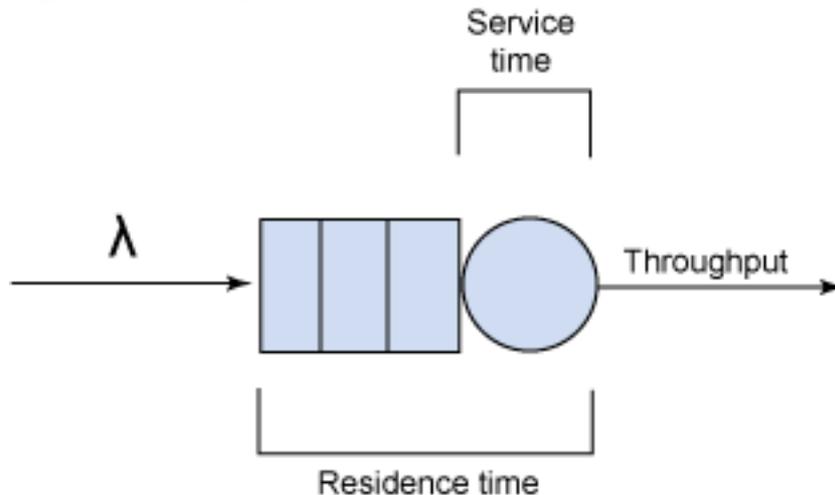
The book *Analyzing Computer System Performance with Perl::PDQ* (see [Resources](#) for a link) introduces queuing theory and shows how to model a computer system as a series of queues. It further describes a C library called PDQ and an associated

Perl interface that lets you define and solve the queues to give performance estimates. You can then estimate the result of changes to the system by changing parameters.

Introducing queues

Figure 4 shows a single queue. A request comes in from the left and enters the queue. As requests are processed by the circle, they leave the queue. The blocks to the left of the circle represent the queued objects.

Figure 4. A simple queue



The queue's behavior is measured in terms of times, rates, and sizes. The *arrival rate* is denoted as *Lambda* (λ) and is usually expressed in terms of items per second. You can determine λ by observing your system over a reasonable period of time and counting the arrivals. A reasonable amount of time is defined to be at least 100 times the *service time*, which is the length of time that the request is processed. The *residence time* is the total time a request spends in the queue, including the time it takes to be processed.

The arrival rate describes the rate at which items enter the queue, and the *throughput* defines the rate at which the items leave. In a more complex system, the *nodal throughput* defines the throughput of a single queuing node, and the *system throughput* refers to the system as a whole.

The size of the buffer doesn't matter in most cases because it will have a finite and predictable size as long as the following conditions hold true:

- The buffer is big enough to handle the queued objects.
- The queue doesn't grow unbounded.

The second constraint is the most important. If a queue can dispatch requests at the

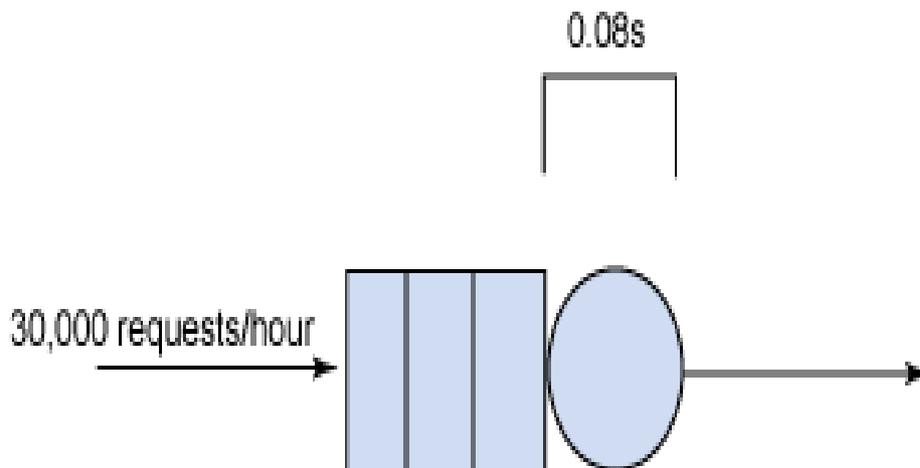
rate of one request per second, but requests come in more often than one per second, then the queue will grow unbounded. In reality, the arrival rate will fluctuate, but performance analysis is concerned with the *steady state*, so averages are used. Perhaps at one point, 10 requests per second come in, and at other times no requests come in. As long as the average is less than one per second, then the queue will have a finite length. If the average arrival rate exceeds the rate at which requests are dispatched, then the queue length will continue to grow and never reach a steady state.

The queue in Figure 4 is called an *open queue* because an unlimited population of requests is arriving, and they don't necessarily come back after they're processed. A *closed queue* feeds back to the input; there is a finite number of requests in the system. Once the requests have been processed, they go back to the arrival queue.

The classic example of a queue is a grocery store. The number of people entering the line divided by the measurement period is the arrival rate. The number of people leaving the line divided by the measurement period is the throughput. The average time it takes a cashier to process a customer is the service time. The average time a customer waits in line, plus the service time, is the residence time.

To move into the PDQ realm, consider the following scenario. A Web service sees 30,000 requests over the course of 1 hour. Through some tracing of an unloaded system, the service time is found to be 0.08 seconds. Figure 5 shows this drawn as a queue.

Figure 5. The Web service modeled as a queue



What information can PDQ provide? Listing 13 shows the required PDQ program and its output.

Listing 13. A PDQ program and its output

```
#!/usr/bin/perl
use strict;
```

```

use pdq;
# Observations
my $arrivals = 30000; # requests
my $period = 3600; # seconds
my $serviceTime = 0.08; # seconds

# Derived
my $arrivalRate = $arrivals / $period;
my $throughput = 1 / $serviceTime;
# Sanity check -- make sure arrival rate < throughput
if ($arrivalRate > $throughput) {
    die "Arrival rate $arrivalRate > throughput $throughput";
}

# Create the PDQ model and define some units

pdq::Init("Web Service");
pdq::SetWUnit("Requests");
pdq::SetTUnit("Seconds");
# The queuing node
pdq::CreateNode("webservice", $pdq::CEN, $pdq::FCFS);

# The circuit
pdq::CreateOpen("system", $arrivalRate);

# Set the service demand

pdq::SetDemand("webservice", "system", $serviceTime);

# Run the report
pdq::Solve($pdq::CANON);
pdq::Report();

```

..... output ..

```

*****
***** Pretty Damn Quick REPORT *****
*****
*** of : Sat Feb 16 11:24:54 2008 ***
*** for: Web Service ***
*** Ver: PDQ Analyzer v4.2 20070228***
*****
*****

```

```

*****
***** PDQ Model INPUTS *****
*****

```

Node	Sched	Resource	Workload	Class	Demand
CEN	FCFS	webservice	system	TRANS	0.0800

Queueing Circuit Totals:

```

Streams: 1
Nodes: 1

```

WORKLOAD Parameters

Source	per Sec	Demand
system	8.3333	0.0800

```

*****
***** PDQ Model OUTPUTS *****
*****

Solution Method: CANON

***** SYSTEM Performance *****

Metric          Value      Unit
-----
Workload: "system"
Mean Throughput 8.3333   Requests/Seconds
Response Time   0.2400   Seconds

Bounds Analysis:
Max Demand     12.5000  Requests/Seconds
Max Throughput 12.5000  Requests/Seconds

***** RESOURCE Performance *****

Metric          Resource  Work      Value  Unit
-----
Throughput      webservice system    8.3333 Requests/Seconds
Utilization     webservice system    66.6667 Percent
Queue Length    webservice system    2.0000 Requests
Residence Time  webservice system    0.2400 Seconds
    
```

Listing 13 begins with the UNIX "shebang" line that defines the interpreter for the rest of the program. The first two lines of Perl code call for the use of the PDQ module and the strict module. PDQ offers the PDQ functions, whereas strict is a module that enforces good Perl programming behavior.

The next section of Listing 13 defines variables associated with the observations of the system. Given this information, the section that follows the observations calculates the arrival rate and the throughput. The latter is the inverse of the service time—if you can serve one request in *N* seconds, then you can serve 1/*N* requests per second.

Installing PDQ
 You can download the PDQ tarball from the author's Web site (see the [Resources](#)). Unpack it in a temporary directory with `tar -xzf pdq.tar.gz`, and change into the newly created directory with `cd pdq42`. Then, run `make` to compile the C code and the Perl module. Finally, `cd perl5` and then run `./setup.sh` to finish building the Perl module and install it in your system directory.

The sanity test checks to make sure the queue length is bounded. Most of the PDQ functions flag an error anyway, but the author of the module recommends an explicit

check. If more requests per second come in than are leaving, then the program dies with an error.

The rest of the program calls the PDQ functions directly. First, the module is initialized with the title of the model. Then, the time unit and work units are set so the reports show information the way you expect.

Each queue is created with the `CreateNode` function. In Listing 13, a queue called `webservice` is created (the names are tags to help you understand the final report) that is of type `CEN` (a queuing center, as opposed to a delay node that doesn't do any work). The queue is a standard first in, first out (FIFO) queue that PDQ calls a *first-come first-served queue*.

Next, `CreateOpen` is called to define the *circuit* (a collection of queues). The arrival rate to the circuit has already been calculated. Finally, the demand for the queue is set with `SetDemand`. `SetDemand` defines the time taken to complete a particular workload (a queue within a circuit).

The circuit is finally solved with the `Solve` function and reported with the `Report` function. Note that PDQ takes your model, turns it into a series of equations, and then solves them. PDQ doesn't simulate the model in any way.

Interpreting the output is straightforward. The report first starts with a header and a summary of the model. The **WORKLOAD Parameters** section provides more interesting information. The circuit's service time is 0.08 seconds, which was defined. The per second rate is the input rate.

The **SYSTEM performance** section calculates performance of the system as a whole. The circuit managed to keep up with the input rate of 8.3333 requests per second. The response time, which includes the 0.08 seconds of service time, and the time spent in queue was 0.24 seconds (more on this later). The maximum performance of the circuit was deemed to be 12.5 requests per second.

Looking closely at the queue, you can see that it's 66.6667% used. The average queue length is two requests. This means that a request coming in can expect to have two requests queued ahead of it, plus the request being executed. At 0.08 seconds per request, the average wait is then the 0.24 seconds reported earlier.

This model could be extended to show the components of the Web service. Rather than a single queue representing the Web service, you might have a queue to process the request, a queue to access a database, and a queue to package the response. The system performance should stay the same if your model is valid, but then you'll have insight into the inner workings of the Web service. From there, you can play "what if" and model a faster database or more Web servers to see what sort of improvement in response you get. The individual resource numbers will tell you if a particular queue is the bottleneck, and how much room you have to grow.

Listing 13 is a basic example of using the PDQ libraries. Read *Analyzing Computer System Performance with Perl::PDQ* (see [Resources](#) for a link) to learn how to build more complex models.

Section 5. Predict future resource needs

This section covers material for topic 306.4 for the Senior Level Linux Professional (LPIC-3) exam 301. This topic has a weight of 1.

In this section, learn how to:

- Predict capacity break point of a configuration
- Observe growth rate of capacity usage
- Graph the trend of capacity usage

The [previous section](#) introduced the PDQ library and a sample report. The report showed the calculated values for the utilization and maximum load of a queue and the system as a whole. You can use the same method to predict the break point of a configuration. You can also use graphs to show the growth of a system over time and predict when it will reach capacity.

More on PDQ

Listing 14 shows the same Web service as [Listing 13](#), but it's broken into two queues: one representing the CPU time on the Web server for processing the request and response, and one showing the time waiting for the database request to return.

Listing 14. A new PDQ program for the sample Web service

```
#!/usr/bin/perl
use strict;
use pdq;
# Observations
my $arrivals = 30000; # requests
my $period = 3600; # seconds

# Derived
my $arrivalRate = $arrivals / $period;

# Create the PDQ model and define some units
pdq::Init("Web Service");
```

```

pdq::SetWUnit("Requests");
pdq::SetTUnit("Seconds");

# The queuing nodes
pdq::CreateNode("dblookup", $pdq::CEN, $pdq::FCFS);
pdq::CreateNode("process", $pdq::CEN, $pdq::FCFS);

# The circuit
pdq::CreateOpen("system", $arrivalRate);

# Set the service demand
pdq::SetDemand("dblookup", "system", 0.05);
pdq::SetDemand("process", "system", 0.03);

# Solve
pdq::Solve($pdq::CANON);
pdq::Report();

```

The code in Listing 14 adds another queue to the system. The total service time is still 0.08 seconds, comprising 0.05 seconds for a database lookup and 0.03 seconds for CPU processing. Listing 15 shows the generated report.

Listing 15. The PDQ report from Listing 14

```

*****
***** Pretty Damn Quick REPORT *****
*****
*** of : Sun Feb 17 11:35:35 2008 ***
*** for: Web Service ***
*** Ver: PDQ Analyzer v4.2 20070228***
*****
*****
***** PDQ Model INPUTS *****
*****

Node Sched Resource Workload Class Demand
---- -
CEN FCFS dblookup system TRANS 0.0500
CEN FCFS process system TRANS 0.0300

Queueing Circuit Totals:
Streams: 1
Nodes: 2

WORKLOAD Parameters
Source per Sec Demand
---- -
system 8.3333 0.0800

```

```

*****
***** PDQ Model OUTPUTS *****
*****

Solution Method: CANON

***** SYSTEM Performance *****

Metric          Value      Unit
-----
Workload: "system"
Mean Throughput 8.3333    Requests/Seconds
Response Time   0.1257    Seconds

Bounds Analysis:
Max Demand     20.0000   Requests/Seconds
Max Throughput 20.0000   Requests/Seconds

***** RESOURCE Performance *****

Metric          Resource   Work      Value      Unit
-----
Throughput      dblookup  system    8.3333     Requests/Seconds
Utilization     dblookup  system    41.6667    Percent
Queue Length    dblookup  system    0.7143     Requests
Residence Time  dblookup  system    0.0857     Seconds

Throughput      process   system    8.3333     Requests/Seconds
Utilization     process   system    25.0000    Percent
Queue Length    process   system    0.3333     Requests
Residence Time  process   system    0.0400     Seconds
    
```

Look at the output side of the report, and note that the average response time has decreased from Listing 13, and the maximum requests per second has gone from 12.5 to 20. This is because the new model allows for *pipelining*. While a request is being dispatched to the database, another request can be processed by the CPU. In the older model, this wasn't possible to calculate because only one queue was used.

More important, you can see that the database is 42% utilized and the CPU is only 25% utilized. Thus the database will be the first to hit capacity as the system falls under higher load.

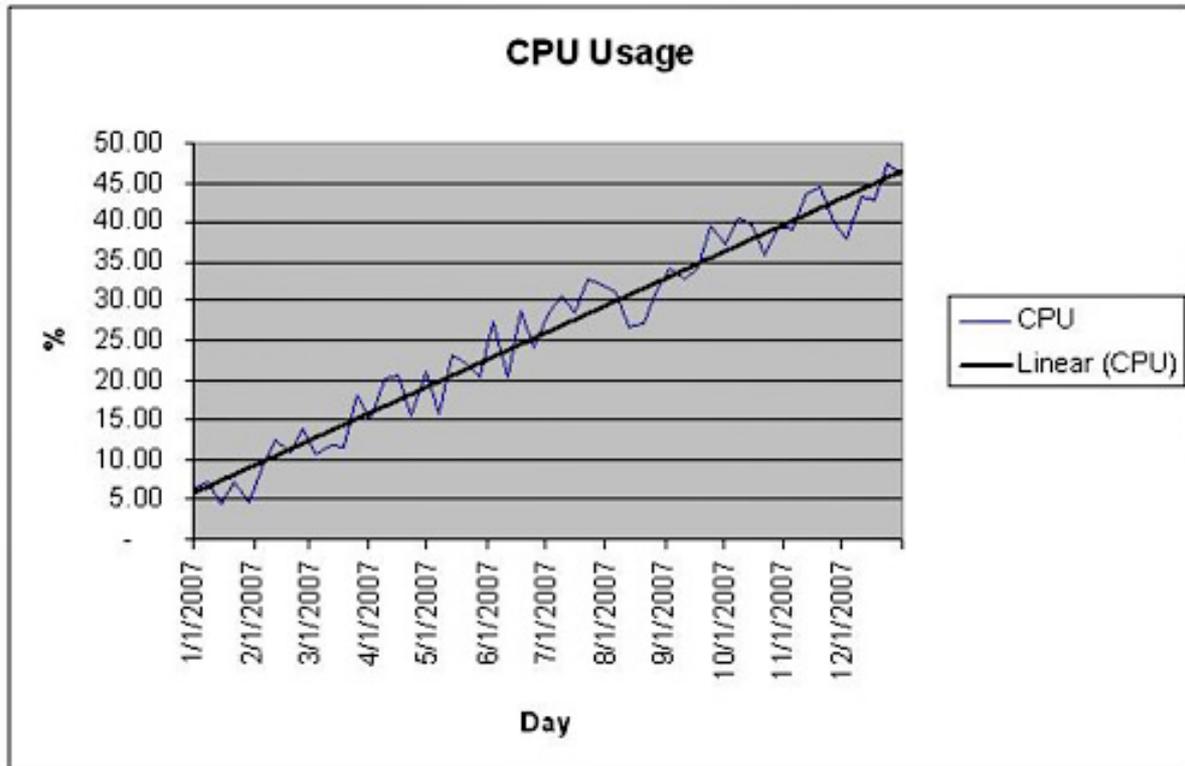
Change the arrivals to be 60,000 over the course of an hour, and you'll find that the average response time increases to 0.36 seconds, with the database hitting 83% utilization. You'll also see that out of the 0.36 second, .30 is spent waiting on the database. Thus, your time would be better spent speeding up database access.

You may define maximum capacity in different ways. At 20 requests per second (from the top of the report), the system is at 100% capacity. You may also choose to define your capacity in terms of average response time. At roughly 15 requests per second, the response time will exceed a quarter of a second. If your goal is to keep response to under 0.25 second, your system will hit capacity at that point even though you still have room to grow on the hardware.

Use graphs for analysis

Graphs are an excellent way to show historical information. You can look at the graph over a long period of time, such as 6 months to a year, and get an idea of your growth rate. Figure 6 represents the CPU usage of an application server over the course of a year. The average daily usage measurements were brought into a spreadsheet and graphed. A trendline was also added to show the growth.

Figure 6. Graphing the CPU usage of a server



From this graph, you can project the future usage (assuming growth stays constant). Growth on the server in Figure 6 is approximately 10% every 3 months. The effects of queuing are more pronounced at higher utilization, so you may find you need to upgrade prior to reaching 100% usage.

How to graph

The spreadsheet method doesn't scale well to many servers with many different measurements. One method takes the output from `sar` and passes it through a graphing tool like `GNUplot`. You may also look at the graphing tools available, many of which are open source. The open source category includes a series of tools based on the `RRDTool` package.

`RRDTool` is a series of programs and libraries that put data into a round-robin

database (RRD). An RRD continually archives data as it comes in, so you can have hourly data for the past year and 5-minute averages for the week. This gives you a database that never grows and that constantly prunes old data. RRDTOol also comes with tools to make graphs.

See the [Resources](#) section for several good graphing tools.

What to graph

You should graph any information that is important to your service, and anything you could potentially use to make decisions. Graphs also play a secondary role by helping you find out what happened in the past, so you may end up graphing items like fan speeds. Normally, though, you'll focus your attention on graphing CPU, memory, disk, and network stats. If possible, graph response times from services. Not only will this help you make better decisions based on what your users will expect, but the information will also help you if you develop any models of your system.

Section 6. Summary

In this tutorial, you learned about measuring and analyzing performance. You also learned to use your measurements to troubleshoot problems.

Linux provides a wealth of information regarding the health of the system. Tools like `vmstat`, `iostat`, and `ps` provide real-time information. Tools like `sar` provide longer-term information. Remember that when you're using `vmstat` and `iostat`, the first value reported isn't real-time!

When troubleshooting a system, you should first try to identify the symptoms of the problem, both to help you understand the problem and to know when it has been solved. Then, measure system resources while the problem is ongoing (if possible) to determine the source of the problem. Once you've identified a fix, implement it and then evaluate the results.

The PDQ Perl module allows you to solve queuing problems. After replacing your system with a series of queues, you can write a Perl script using the PDQ functions. You can then use this model to calculate demand based on current usage and on future predicted usage.

Both models and graphs can be used to predict growth. Ideally, you should use both methods and compare the results.

This concludes the series on preparing for the LPIC 3 exam. If you'll be writing the exam, I wish you success and hope this series is helpful to you.

Resources

Learn

- Review the previous tutorial in this 301 series, "[LPI exam 301 prep, Topic 305: Integration and migration](#)" (developerWorks, April 2008), or [all tutorials in the 301 series](#).
- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Check out [UNIX Load Average Part 1: How It Works](#) and [UNIX Load Average Part 2: Not Your Average Average](#), by Neil Gunther, who runs the [Performance Dynamics Company](#) and has written [several other articles](#) and books entitled *Analyzing Computer System Performance with Perl::PDQ* and *Guerrilla Capacity Planning: A Tactical Approach to Planning for Highly Scalable Applications and Services*.
- Sean's article "[Easy system monitoring with SAR](#)" (developerWorks, February 2006) introduces SAR. It was written with Solaris in mind, so the command-line parameters are somewhat different, but the theory is the same.
- Sean's tutorial "[Expose Web performance problems with the RRDTOol](#)" (developerWorks, March 2006) shows you how to monitor the performance of a Web site and graph the results with RRDTOol.
- The [sysstat FAQ](#) is a good place to go for answers about SAR and friends.
- In the [developerWorks Linux zone](#), find more resources for Linux developers, including [Linux tutorials](#), as well as [our readers' favorite Linux articles and tutorials](#) over the last month.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- Get the [source for PDQ](#), along with instructions for installing it on various operating systems.
- [RRDTOol](#), the basis of most open source network monitoring systems, is a spinoff of the famous [Multi Router Traffic Grapher](#) package. RRDTOol can be used as part of another package or in your own scripts.
- [Cacti](#) is one of the best open source monitoring packages around. It's made primarily for network devices but has been extended to perform a variety of systems tasks. Cacti has an active user community that is always willing to help

in the forums.

- [ZABBIX](#) is another open source systems-monitoring package worth investigating.
- Check out the [Tivoli](#) area of developerWorks for more information on IBM's enterprise systems, network, security, and application management tools. In particular, the [Tivoli Composite Application Management](#) solution helps you increase the performance and availability of today's business-critical composite applications, including portal and SOA-based technologies.
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content.](#)
- Get involved in the [developerWorks community](#) through our developer blogs, forums, podcasts, and community topics in our new [developerWorks spaces](#).

About the author

Sean Walberg

Sean Walberg has been working with Linux and UNIX since 1994 in academic, corporate, and Internet service provider environments. He has written extensively about systems administration over the past several years.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.