Praise for Previous Editions of *A Practical Guide to Fedora™ and Red Hat® Enterprise Linux®*

"Since I'm in an educational environment, I found the content of Sobell's book to be right on target and very helpful for anyone managing Linux in the enterprise. His style of writing is very clear. He builds up to the chapter exercises, which I find to be relevant to real-world scenarios a user or admin would encounter. An IT/IS student would find this book a valuable complement to their education. The vast amount of information is extremely well balanced and Sobell manages to present the content without complicated asides and meandering prose. This is a 'must have' for anyone managing Linux systems in a networked environment or anyone running a Linux server. I would also highly recommend it to an experienced computer user who is moving to the Linux platform."

—Mary Norbury
IT Director
Barbara Davis Center
University of Colorado at Denver
from a review posted on slashdot.org

"I had the chance to use your UNIX books when I when was in college years ago at Cal Poly, San Luis Obispo, CA. I have to say that your books are among the best! They're quality books that teach the theoretical aspects and applications of the operating system."

—Benton Chan
IS Engineer

"The book has more than lived up to my expectations from the many reviews I read, even though it targets FC2. I have found something very rare with your book: It doesn't read like the standard technical text, it reads more like a story. It's a pleasure to read and hard to put down. Did I say that?! :-)"

—David Hopkins
Business Process Architect

"Thanks for your work and for the book you wrote. There are really few books that can help people to become more efficient administrators of different workstations. We hope (in Russia) that you will continue bringing us a new level of understanding of Linux/UNIX systems."

—Anton Petukhov

"Mark Sobell has written a book as approachable as it is authoritative."

—*Jeffrey Bianchine*
Advocate, Author, Journalist

"Excellent reference book, well suited for the sysadmin of a Linux cluster, or the owner of a PC contemplating installing a recent stable Linux. Don't be put off by the daunting heft of the book. Sobell has striven to be as inclusive as possible, in trying to anticipate your system administration needs."

—*Wes Boudville*
Inventor

"*A Practical Guide to Red Hat® Linux®* is a brilliant book. Thank you Mark Sobell."

—*C. Pozrikidis*
University of California at San Diego

"This book presents the best overview of the Linux operating system that I have found. . . . [It] should be very helpful and understandable no matter what the reader's background: traditional UNIX user, new Linux devotee, or even Windows user. Each topic is presented in a clear, complete fashion and very few assumptions are made about what the reader knows. . . . The book is extremely useful as a reference, as it contains a 70-page glossary of terms and is very well indexed. It is organized in such a way that the reader can focus on simple tasks without having to wade through more advanced topics until they are ready."

—*Cam Marshall*
Marshall Information Service LLC
Member of Front Range UNIX
Users Group [FRUUG]
Boulder, Colorado

"Conclusively, this is THE book to get if you are a new Linux user and you just got into RH/Fedora world. There's no other book that discusses so many different topics and in such depth."

—*Eugenia Loli-Queru*
Editor in Chief
OSNews.com

Praise for Other Books by Mark G. Sobell

"This book is a very useful tool for anyone who wants to 'look under the hood' so to speak, and really start putting the power of Linux to work. What I find particularly frustrating about man pages is that they never include examples. Sobell, on the other hand, outlines very clearly what the command does and then gives several common, easy-to-understand examples that make it a breeze to start shell programming on one's own. As with Sobell's other works, this is simple, straightforward, and easy to read. It's a great book and will stay on the shelf at easy arm's reach for a long time."

—Ray Bartlett
Travel Writer

"Overall I found this book to be quite excellent, and it has earned a spot on the very front of my bookshelf. It covers the real 'guts' of Linux—the command line and its utilities—and does so very well. Its strongest points are the outstanding use of examples, and the Command Reference section. Highly recommended for Linux users of all skill levels. Well done to Mark Sobell and Prentice Hall for this outstanding book!"

—Dan Clough
Electronics Engineer and
Slackware Linux User

"Totally unlike most Linux books, this book avoids discussing everything via GUI and jumps right into making the power of the command line your friend."

—Bjorn Tipling
Software Engineer
ask.com

"This book is the best distro-agnostic, foundational Linux reference I've ever seen, out of dozens of Linux-related books I've read. Finding this book was a real stroke of luck. If you want to really understand how to get things done at the command line, where the power and flexibility of free UNIX-like OSes really live, this book is among the best tools you'll find toward that end."

—Chad Perrin
Writer, TechRepublic

"I currently own one of your books, *A Practical Guide to Linux®*. I believe this book is one of the most comprehensive and, as the title says, practical guides to Linux I have ever read. I consider myself a novice and I come back to this book over and over again."

—Albert J. Nguyen

"Thank you for writing a book to help me get away from Windows XP and to never touch Windows Vista. The book is great; I am learning a lot of new concepts and commands. Linux is definitely getting easier to use."

—James Moritz

"I am so impressed by how Mark Sobell can approach a complex topic in such an understandable manner. His command examples are especially useful in providing a novice (or even an advanced) administrator with a cookbook on how to accomplish real-world tasks on Linux. He is truly an inspired technical writer!"

—George Vish II
Senior Education Consultant
Hewlett-Packard Company

"Overall, I think it's a great, comprehensive Ubuntu book that'll be a valuable resource for people of all technical levels."

—John Dong
Ubuntu Forum Council Member
Backports Team Leader

"The JumpStart sections really offer a quick way to get things up and running, allowing you to dig into the details of the book later."

—Scott Mann
Aztek Networks

"I would so love to be able to use this book to teach a class about not just Ubuntu or Linux but about computers in general. It is thorough and well written with good illustrations that explain important concepts for computer usage."

—Nathan Eckenrode
New York Local Community Team

"Ubuntu is gaining popularity at the rate alcohol did during Prohibition, and it's great to see a well-known author write a book on the latest and greatest version. Not only does it contain Ubuntu-specific information, but it also touches on general computer-related topics, which will help the average computer user to better understand what's going on in the background. Great work, Mark!"

—Daniel R. Arfsten
Pro/ENGINEER Drafter/Designer

"I read a lot of Linux technical information every day, but I'm rarely impressed by tech books. I usually prefer online information sources instead. Mark Sobell's books are a notable exception. They're clearly written, technically accurate, comprehensive, and actually enjoyable to read."

—Matthew Miller
Senior Systems Analyst/Administrator
BU Linux Project
Boston University Office
of Information Technology

"This is well written, clear, comprehensive information for the Linux user of any type, whether trying Ubuntu on for the first time and wanting to know a little about it, or using the book as a very good reference when doing something more complicated like setting up a server. This book's value goes well beyond its purchase price and it'll make a great addition to the Linux section of your bookshelf."

—Linc Fessenden
Host of The LinuxLink TechShow
tllts.org

"The author has done a very good job at clarifying such a detail-oriented operating system. I have extensive Unix and Windows experience and this text does an excellent job at bridging the gaps between Linux, Windows, and Unix. I highly recommend this book to both 'newbs' and experienced users. Great job!"

—Mark Polczynski
Information Technology Consultant

"When I first started working with Linux just a short 10 years or so ago, it was a little more difficult than now to get going. . . . Now, someone new to the community has a vast array of resources available on the web, or if they are inclined to begin with Ubuntu, they can literally find almost every single thing they will need in the single volume of Mark Sobell's *A Practical Guide to Ubuntu Linux®*.

"I'm sure this sounds a bit like hyperbole. Everything a person would need to know? Obviously not everything, but this book, weighing in at just under 1200 pages, covers so much so thoroughly that there won't be much left out. From install to admin, networking, security, shell scripting, package management, and a host of other topics, it is all there. GUI and command line tools are covered. There is not really any wasted space or fluff, just a huge amount of information. There are screen shots when appropriate but they do not take up an inordinate amount of space. This book is information-dense."

—JR Peck
Editor
GeekBook.org

"I have been wanting to make the jump to Linux but did not have the guts to do so—until I saw your familiarly titled *A Practical Guide to Red Hat® Linux®* at the bookstore. I picked up a copy and am eagerly looking forward to regaining my freedom."

—Carmine Stoffo
Machine and Process Designer
to pharmaceutical industry

"I am currently reading *A Practical Guide to Red Hat® Linux®* and am finally understanding the true power of the command line. I am new to Linux and your book is a treasure."

—Juan Gonzalez

"Overall, *A Practical Guide to Ubuntu Linux®* by Mark G. Sobell provides all of the information a beginner to intermediate user of Linux would need to be productive. The inclusion of the Live DVD of the Gutsy Gibbon release of Ubuntu makes it easy for the user to test-drive Linux without affecting his installed OS. I have no doubts that you will consider this book money well spent."

—Ray Lodato
Slashdot contributor
www.slashdot.org

Excerpts of Chapters From

A Practical Guide to Fedora™ and Red Hat® Enterprise Linux®

FIFTH EDITION

Mark G. Sobell

ISBN: 978-0-13-706088-7

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

blank

3

Step-by-Step Installation

In This Chapter

Chapter 2 covered planning the installation of Fedora/RHEL: determining the requirements; performing an upgrade versus a clean installation; planning the layout of the hard disk; obtaining the files you need for the installation, including how to download and burn CD/DVD ISO images; and collecting information about the system. This chapter focuses on installing Fedora/RHEL. Frequently the installation is quite simple, especially if you have done a good job of planning. Sometimes you may run into a problem or have a special circumstance; this chapter gives you tools to use in these cases. Read as much of this chapter as you need to; once you have installed Fedora/RHEL, continue with Chapter 4, which covers getting started using the Fedora/RHEL desktop. If you install a textual (command line) system, refer to Chapter 5.

Figure 3-1 Live session, automatic boot screen

RUNNING A FEDORA LIVE SESSION

As discussed in Chapter 2, a live session is a Linux session that you run on a computer without installing Linux on the computer. When you reboot after a live session, the computer is untouched. If you are running Windows, after a live session Windows boots the way it did before the live session. If you choose, you can install Fedora from a live session. Red Hat Enterprise Linux does not offer live sessions.

A live session gives you a chance to preview Fedora without installing it. Boot from the live CD to begin a live session and work with Fedora as explained in Chapter 4. When you are finished, remove the CD and reboot the system. The system will then boot as it did before the live session took place.

Because a live session does not write to the hard disk (other than using a swap partition, if one is available), none of the work you save will be available once you reboot. You can use a USB flash drive, Webmail, or another method to transfer files you want to preserve to another system.

BOOTING THE SYSTEM

Before Fedora can display the desktop of a live session or install itself on a hard disk, the Linux operating system must be read into memory (booted). This process can take a few minutes on older, slower systems and systems with minimal RAM (memory).

Figure 3-2 The Fedora Live Welcome menu

In most cases, you can boot Fedora to run a live session that displays a desktop without doing anything after you boot from a live CD. To begin, insert the live CD (the standard GNOME **Fedora Desktop Live Media**) into the CD drive and turn on or reset the system. Refer to "BIOS setup" on page 26 if the system does not boot from the CD. Refer to "Modifying Boot Parameters (Options)" on page 68 if Fedora does not boot or displays an error message.

A few moments after you start the system, Fedora displays a screen that says **Automatic boot in 10 seconds** and counts down from 10 to 1 (Figure 3-1). Next the system displays a graphical screen showing a progress bar.

Checking the CD The first time you use a CD, it is a good idea to check it for defects. To do so, interrupt the automatic boot by pressing a key such as the SPACE bar while Fedora is counting down. Fedora displays the Welcome menu (Figure 3-2). Use the DOWN ARROW key to highlight the **Verify and Boot** line and press RETURN (the mouse will not work yet). Fedora displays a progress bar as it verifies the contents of the CD; nothing happens for a while. If the CD is good, the system boots.

Memory test Selecting **Memory Test** from the Welcome menu runs memtest86+, a GPL-licensed, stand-alone memory test utility for x86-based computers. Press **C** to configure the test; press ESCAPE to exit and reboot. See www.memtest.org for more information.

GNOME If you are installing from Fedora Desktop Live Media (what this book refers to as the *live CD*), you are installing the GNOME desktop manager. When you boot from this CD, Fedora displays a login screen for a few seconds, automatically logs

Figure 3-3 A GNOME Live desktop

in as the user named **liveuser**, and displays the GNOME desktop (Figure 3-3). To speed up this process, you can click the button labeled **Log In** when Fedora displays the login screen.

KDE If you are installing from Fedora KDE Live Media, you are installing the KDE desktop manager. When you boot from this disk, Fedora next displays a KDE startup screen and then the KDE desktop—there is no need to log in.

optional

SEEING WHAT IS GOING ON

If you are curious and want to see what Fedora is doing as it boots from a live CD, remove **quiet**, which controls kernel messages, and **rhgb** (Red Hat graphical boot), which controls messages from the graphical installer, from the boot parameters. See Figure 3-13 on page 68; the list of parameters on the screen will be different from those in the figure. With the Fedora Live Welcome menu displayed (Figure 3-2), press TAB to display the boot command-line parameters. Use the BACK ARROW key to back up over—but not remove—any words to the right of **quiet**. Press BACKSPACE or DEL to back up over and erase **quiet** and **rhgb** from the boot command line. Press RETURN. Now as Fedora boots, it displays information about what it is doing. Text scrolls on the screen, although sometimes too rapidly to read. When you boot Fedora from a DVD and when you boot RHEL, this information is displayed by default: You do not have to change the command line.

INSTALLING FEDORA/RHEL

You can install Fedora/RHEL from a live session (preceding section; *FEDORA* only) or from the install DVD (*RHEL+FEDORA*). Installing from a live session is simpler but does not give you the flexibility that installing from the install DVD does. For example, you cannot select the language the installer uses, nor can you choose which software packages you want to install when you install from a live session.

Check to see what is on the hard disk before installing Fedora/RHEL

caution Unless you are certain the hard disk you are installing Fedora/RHEL on has nothing on it (it is a new disk) or you are sure the disk holds no information of value, it is a good idea to examine the contents of the disk before you start the installation. You can use palimpsest (page 78) from a live session for this purpose.

The install DVD holds many of the software packages that Fedora/RHEL supports. You can install whichever packages you like from this DVD without connecting to the Internet. However, without an Internet connection, you will not be able to update the software on the system.

The live CD holds a limited set of software packages. Once you install from this CD, you must connect to the Internet to update the software on the system and download and install additional packages.

To begin most installations, insert the live CD or the install DVD into the CD/DVD drive and turn on or reset the system. For hard disk and network-based installations, you can use the first installation CD, the Net Boot CD, the install DVD, or a USB flash drive.

INSTALLING FROM A LIVE SESSION

Bring up a live GNOME session as explained on page 52. Double-click (left button) the object labeled **Install to Hard Drive** (Figure 3-3) to begin installing Linux. Continue reading at "The Anaconda Installer" on page 57.

INSTALLING/UPGRADING FROM THE INSTALL DVD

FEDORA To install/upgrade Fedora from the install DVD, insert this DVD into the DVD drive and turn on or reset the system. After a few moments, Fedora displays the Welcome to Fedora menu (Figure 3-4, next page) and a message that says **Automatic boot in 60 seconds**.

Press a key, such as the SPACE bar, within 60 seconds to stop the countdown and display the message **Press [TAB] to edit options** as shown in Figure 3-4. If you do not press a key, after 60 seconds Fedora begins a graphical install/upgrade. Refer to "BIOS setup" on page 26 if the system does not boot from the DVD. Refer to "Modifying Boot Parameters (Options)" on page 68 if Fedora/RHEL does not boot or displays an error message.

Figure 3-4 The install DVD Welcome menu

The Welcome menu has the following selections:

Install or upgrade an existing system — Installs a graphical Fedora/RHEL system using the graphical installer.

Install system with basic video driver — Installs a graphical Fedora/RHEL system using the graphical installer. Fedora/RHEL does not attempt to determine the type of display attached to the system; it uses a basic video driver that works with most displays. Choose this selection if the previous selection fails just after the Disc Found screen (page 57).

Rescue installed system — Brings up Fedora/RHEL but does not install it. After detecting the system's disks and partitions, the system enters rescue mode and allows you to mount an existing Linux filesystem. For more information refer to "Rescue Mode" on page 411.

Boot from local drive — Boots the system from the hard disk. This selection frequently has the same effect as booting the system without the CD/DVD (depending on how the BIOS [page 26] is set up).

STARTING THE INSTALLATION

Make a selection from the Welcome menu and press RETURN to boot the system. Text scrolls by as the system boots.

RHEL The process of installing Red Hat Enterprise Linux is similar to that of installing Fedora. The biggest difference relates to the initial screen the two systems display. While *FEDORA* displays a menu, *RHEL* displays a **boot:** prompt. Follow the instructions on the screen for installing *RHEL* in graphical or textual mode. To bring the system up

in Rescue mode (page 411), enter **linux rescue** and press RETURN. Most parameters you enter at the **boot:** prompt begin with the word **linux**. You can use all the parameters discussed in "Modifying Boot Parameters (Options)" on page 68, but they must be preceded by the word **linux**. Press the function keys listed at the bottom of the screen for more information.

THE DISC FOUND SCREEN

The first screen the install DVD installation process displays is the pseudographical Disc Found screen. Because it is not a true graphical screen, the mouse does not work. Instead, you must use the TAB or ARROW keys to highlight different choices and then press RETURN to select the highlighted choice. This screen allows you to test as many installation CD/DVDs as you like. Choose **OK** to test the media or **Skip** to bypass the test. See the following caution box.

Test install DVDs

caution Many people download ISO image files from the Web and then burn disks using these files. It is possible for data to become corrupted while fetching an ISO image; it is also possible for a transient error to occur while writing an image to recordable media. When you boot Fedora/RHEL from an install DVD, Anaconda displays the Disc Found screen before starting the installation. From this screen, you can verify that the install DVD does not contain any errors. Testing the DVD takes a few minutes but can save you hours of aggravation if the installation fails due to bad media.

A DVD may fail the media test if the software that was used to burn the disk did not include padding. If a DVD fails the media test, try booting with the **nodma** parameter. See page 68 for information on adding parameters to the boot command line.

If the DVD passes the media test when you boot the system with the **nodma** parameter, the DVD is good; reboot the system without this parameter before installing Fedora/RHEL. If you install Linux after having booted with this parameter, the kernel will be set up to always use this parameter. As a consequence, the installation and operation of the system may be slow.

THE ANACONDA INSTALLER

Anaconda, which is written in Python and C, identifies the hardware, builds the filesystems, and installs or upgrades the Fedora/RHEL operating system. Anaconda can run in textual or graphical (default) interactive mode or in batch mode (see "Using the Kickstart Configurator" on page 82).

Exactly which screens Anaconda displays depends on whether you are installing Fedora from a live session or from the install DVD, whether you are installing Red Hat Enterprise Linux, and which parameters you specify on the boot command line. With some exceptions—most notably if you are running a textual installation—Anaconda probes the video card and monitor, and starts a native X server.

While it is running, Anaconda opens the virtual consoles (page 137) shown in Table 3-1. You can display a virtual console by pressing CONTROL-ALT-F*x*, where *x* is the virtual console number and F*x* is the function key that corresponds to the virtual console number.

Table 3-1 Virtual console assignments during installation

	Information displayed during installation	
Virtual console	**Install DVD**	**Live CD**
1	Installation dialog	Installation dialog
2	Shell	Login prompt (log in as **liveuser**)
3	Installation log	Installation log
4	System messages	Login prompt (log in as **liveuser**)
5	X server output	Login prompt (log in as **liveuser**)
6	GUI interactive installation screen[a]	Login prompt (log in as **liveuser**)
7	GUI interactive installation screen[a]	GUI interactive installation

a. The GUI appears on virtual console 6 or 7.

At any time during the installation, you can switch to virtual console 2 (CONTROL-ALT-F2) and give commands to see what is going on. Do not give any commands that change any part of the installation process. To switch back to the graphical installation screen, press CONTROL-ALT-F6 or CONTROL-ALT-F7.

USING ANACONDA

Anaconda displays a button labeled **Next** at the lower-right corner of each installation screen and a button labeled **Back** next to it on most screens. When you have completed the entries on an installation screen, click **Next** or press F12; from a textual installation, press the TAB key until the **Next** button is highlighted and then press RETURN. Select **Back** to return to the previous screen.

ANACONDA SCREENS

Anaconda displays different screens depending on which commands you give and which choices you make. During a graphical installation, Anaconda starts, loads drivers, and probes for the devices it will use during installation. After probing, it starts the X server. This section describes the screens that Anaconda displays during a default installation and explains the choices you can make on each of them.

Figure 3-5 The logo screen

Logo — Anaconda displays the logo screen (Figure 3-5) after it obtains enough information to start the X Window System. There is nothing for you to do on this screen. Click **Next**.

Language — Select the language you want to use for the installation. This language is not necessarily the same language the installed system will display.

Installation number (*RHEL*) — RHEL asks if you want to provide an installation number or skip this step. See www.redhat.com/support/resources/faqs/installation_numbers for more information.

Keyboard — Select the type of keyboard attached to the system.

Error processing drive — Anaconda displays this warning if the hard disk has not been used before. The dialog box says the drive may need to be initialized. When you initialize a drive, all data on the drive is lost. Click **Re-initialize drive** if it is a new drive or if you do not need the data on the drive. Anaconda initializes the hard disk immediately.

Hostname — Fedora asks you to specify the name of the system. RHEL asks for this information on the Network Configuration screen.

Time zone — The time zone screen allows you to specify the time zone where the system is located (Figure 2-1, page 29). Use the scroll wheel on the mouse or the slider to the left of the map to zoom in or out on the selected portion of the map, drag the horizontal and vertical *thumbs* (page 1111) to position the map in the window, and then click a city in the local system's time zone. Alternatively, you can scroll through the drop-down list and highlight the appropriate selection. Remove the tick from the check box labeled **System clock uses UTC** if the system clock is not set to *UTC* (page 1114). Click **Next**.

Root password — Enter and confirm the password for the **root** user (Superuser). See page 405 for more information on **root** privileges. Click **Next**. If you enter a password that is not

Figure 3-6 The Install or Upgrade screen

very secure, Anaconda displays a dialog box with the words **Weak password**; click **Cancel** or **Use Anyway**, as appropriate.

Install or Upgrade

(This choice is not available from the live CD.) Anaconda displays the Install or Upgrade screen (Figure 3-6) only if it detects a version of Fedora/RHEL on the hard disk that it can upgrade. Anaconda gives you the choice of upgrading the existing installation or overwriting the existing installation with a new one. Refer to "Upgrading an Existing Fedora/RHEL System Versus Installing a Fresh Copy" on page 29 for help in making this selection. Select one of the entries and click **Next**.

Disk Partitioning

The Disk Partitioning screen (Figure 3-7) allows you to specify partition information and to select the drives you want to install Fedora/RHEL on (assuming the system has more than one drive). Specify which drives you want to install Linux on in the frame labeled **Select the drive(s) to use for this installation**. Anaconda presents the following choices in the drop-down list near the top of the screen; click the arrow button at the right end of the list and then click the choice you want:

- **Use entire drive**—Deletes all data on the hard disk and creates a default layout on the entire hard disk, as though you were working with a new hard disk.
- **Replace existing Linux system**—Removes all Linux partitions, deleting the data on those partitions and creating a default layout in place of one or more of the removed partitions. If there is only a Linux system on the hard disk, this choice is the same as the previous one.
- **Shrink current system**—Shrinks the partitions that are in use by the operating system that is already installed on the hard disk. This choice

Figure 3-7 The Disk Partitioning screen

creates a default layout in the space it has recovered from the installed operating system.

- **Use free space**—Installs Fedora/RHEL in the *free space* (page 30) on the disk. This choice does not work if there is not enough free space.
- **Create custom layout**—Does not alter hard disk partitions. This choice causes Anaconda to run Disk Druid (page 71) so you can preserve those partitions you want to keep and overwrite other partitions. It is a good choice for installing Fedora/RHEL over an existing system where you want to keep **/home**, for example, but want a clean installation and not an upgrade.

Default layout

The default layout the first four choices create includes two logical volumes (swap and root [**/**]) and one standard partition (**/boot**). With this setup, most of the space on the disk is assigned to the root partition. For information on the Logical Volume Manager, see page 38.

Put a tick in the check box labeled **Encrypt system** to encrypt the filesystems you are creating. If you are installing on more than one disk, you can select which drive the system boots from. See the tip on page 35.

Disk Druid

Anaconda runs Disk Druid only if you put a tick in the check box labeled **Review and modify partitioning layout** or if you select **Create custom layout** from the drop-down list as described earlier. You can use Disk Druid to verify and modify the layout before it is written to the hard disk. For more information refer to "Using Disk Druid to Partition the Disk" on page 71.

Warning

Anaconda displays a warning if you are removing or formatting partitions. Click **Yes**, **Format**, or **Write changes to disk** to proceed.

Figure 3-8 The Boot Loader Configuration screen

Boot Loader Configuration

Anaconda displays the Boot Loader Configuration screen (Figure 3-8) only when you put a tick in the check box labeled **Review and modify partitioning layout** or select **Create custom layout** from the drop-down list in the Partition the Disk screen. By default, Anaconda installs the grub boot loader (page 551). If you do not want to install a boot loader, remove the tick from the check box labeled **Install boot loader on /dev/*xxx***. To change the device the boot loader is installed on, click **Change device**. When you install Fedora/RHEL on a machine that already runs another operating system, Anaconda frequently recognizes the other operating system and sets up grub so you can boot from either operating system. Refer to "Setting Up a Dual-Boot System" on page 82. To manually add other operating systems to grub's list of bootable systems, click **Add** and specify a label and device to boot from. For a more secure system, specify a boot loader password.

A live CD begins copying files at this point. See "Beginning Installation" on page 65.

Enable Network Interface

(This window is displayed by the Net Install CD only). This window allows you to specify a network interface. See page 64 for more information.

Network Configuration (*RHEL*)

The RHEL Network Configuration screen, which allows you to specify network configuration information, has three parts: Network Devices, Hostname, and Miscellaneous Settings. If you are using DHCP to set up the network interface, you do not need to change anything on this screen.

The Network Devices frame lists the network devices found by the installer. Normally you want network devices to become active when the system boots. Remove the tick from the check box at the left of a device if you do *not* want that device to become active when the system boots.

To configure a network device manually (not using DHCP), highlight the device and click **Edit** to the right of the list of devices. Anaconda displays the Edit Interface window. To set up IPv4 networking manually, click the radio button labeled **Manual**

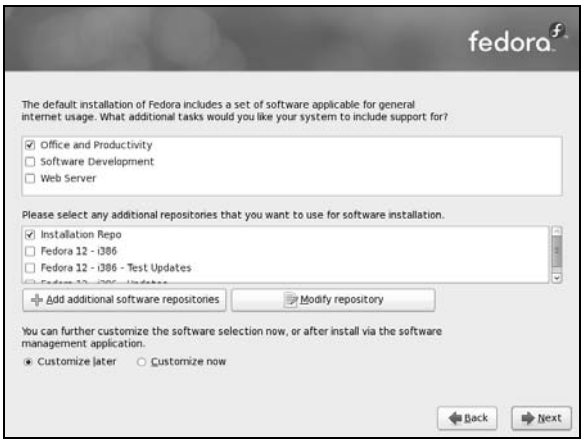

Figure 3-9 The Software Selection screen

configuration under **Enable IPv4 support** and enter the IP address and netmask of the system in the appropriate boxes. You can also set up or disable IPv6 networking on this screen. Click **OK**.

If you are not using DHCP, click the radio button labeled **manually** under **Set the hostname** in the network configuration screen and enter the name of the system. When you turn off DHCP configuration in the Network Devices frame, Anaconda allows you to specify a gateway address and one or more DNS (nameserver) addresses. You do not have to specify more than one DNS address, although it can be useful to have two in case the first nameserver stops working. Click **Next**.

Software Selection

(This screen does not appear when you install from a live CD.) As the Software Selection screen explains, by default Anaconda installs a basic Fedora system, including software that allows you to use the Internet. See Figure 3-9. Near the top of the screen are three check boxes that you can put ticks in to select categories of software to install: **Office and Productivity** (*FEDORA* only; selected by default), **Software Development**, and **Web Server**.

Fedora/RHEL software is kept in repositories (see Chapter 13). In the middle of the screen are check boxes (*FEDORA* only) you can put ticks in to select repositories that hold the following items:

- **Installation Repo**—Indicates Anaconda is to install from the repository included on the installation medium.
- **Fedora 12 - *xxx***—Indicates Anaconda is to use the online Fedora 12 repository. The ***xxx*** indicates the system architecture (e.g., i386).
- **Fedora 12 - *xxx* - Updates**—Indicates Anaconda is to use the online Fedora 12 Updates repository. The ***xxx*** indicates the system architecture (e.g., i386).

Figure 3-10 The Enable Network Interface window

Selecting either of the last two choices gives you more software packages to choose from later in the installation process if you decide to customize the software selection during installation.

Enable Network Interface

When you put a tick in either of the last two check boxes, Anaconda displays the Enable Network Interface window (Figure 3-10). By default, the check box labeled **Use dynamic IP configuration (DHCP)** has a tick in it. If the system is connected to the local network, and if that network is using DHCP, click **OK** and Anaconda will set up the network connection automatically. Otherwise, you must select the appropriate network from the Interface drop-down list, remove the tick from the check box labeled **Use dynamic IP configuration (DHCP)**, and specify an IP address for the local system, a gateway address, and a nameserver address. Your network administrator or ISP should be able to provide this information. Click **OK**; the Enable Network Interface window closes.

Below the repository selection frame in the Software Selection screen are buttons labeled **Add additional software repositories** and **Modify repository**. See Chapter 13 for information on software repositories.

Toward the bottom of the screen are two radio buttons:

- **Customize later**—Installs the default packages plus those required to perform the tasks selected from the list at the top of this screen.
- **Customize now**—Displays the package selection screen (discussed in the next section) after you click **Next** on this screen so you can select specific categories of software and package groups you want to install. If you want to set up servers as described in Part V of this book, select **Customize now** and install them in the next step.

In most cases it is a good idea to customize the software selection before installation. Regardless of which software groups and packages you select now, you can

Figure 3-11 The package selection screen

change which software groups and packages are installed on a system any time after the system is up and running (as long as the system can connect to the Internet).

Package selection

If you selected **Customize now**, Anaconda displays a package selection screen that contains two adjacent frames near the top of the screen (Figure 3-11). If you added repositories in addition to the Installation repo, this screen will display more choices. Select a software category from the frame on the left and package groups from the frame on the right. Each package group comprises many software packages, some mandatory (the base packages) and some optional.

For example, to install KDE, which is not installed by default, click **Desktop Environments** in the left frame. Anaconda highlights your selection and displays a list of desktop environments you can install in the right frame. Put a tick in the check box labeled **KDE (K Desktop Environment)**; Anaconda highlights KDE, displays information about KDE in the frame toward the bottom of the window, displays the number of optional packages that are selected, and activates the button labeled **Optional packages**. Click this button to select which optional packages you want to install in addition to the base packages. To get started, accept the default optional packages. If you will be running servers on the system, click **Servers** on the left and select the servers you want to install from the list on the right. Select other package categories in the same manner. When you are done, click **Next**; Anaconda begins writing to the hard disk.

BEGINNING INSTALLATION

After going through some preliminary steps, Anaconda installs Fedora/RHEL based on your choices in the preceding screens, placing a log of the installation in **/root/install.log** and a Kickstart file (page 82) in **/root/anaconda-ks.cfg**. To change the way you set up Fedora/RHEL, you can press CONTROL-ALT-DEL to reboot the system and start over. If you reboot the system, you will lose all the work you did up to this point.

Figure 3-12 The Welcome screen

Installing Fedora/RHEL can take a while. The amount of time depends on the hardware you are installing the operating system on and the number of software packages you are installing.

Installation Complete

When Anaconda is finished, it tells you that the installation is complete. An installation from a live CD ejects the CD. If you are using another installation technique, you must remove the CD/DVD (if that is the medium you installed from). Click **Reboot**.

FIRSTBOOT: WHEN YOU REBOOT

When the system reboots, it is running Fedora/RHEL. The first time it boots, Fedora/RHEL runs Firstboot, which asks a few questions before allowing you to log in.

Welcome

There is nothing to do on the Welcome screen (Figure 3-12). Click **Forward**.

License Information

After the Welcome screen, Firstboot displays the License Information screen. If you understand the license information, click **Forward**.

Firewall (*RHEL*)

Next RHEL gives you the opportunity to set up a very basic *firewall* (page 1082). First select **Enabled** or **Disabled** from the drop-down list labeled **Firewall**. If you enable the firewall, select which services the firewall will pass through to the system. These services are the ones the system is providing by means of servers you set up. For example, you do not need to enable WWW to browse the Web using Firefox; you need to enable WWW only if you want to set up an Apache (HTTP) Web server. Select **Secure WWW (HTTPS)**, which is used for secure browser connections, to allow secure HTTP to pass through the firewall. Click the triangle to the left of **Other ports** to open a frame in which you can add and remove additional protocols and ports that the firewall will pass. Use the buttons labeled **Add** and **Remove** to manipulate this list.

For more information on setting up a firewall, refer to "JumpStart: Building a Firewall Using system-config-firewall" on page 824. Chapter 25 on iptables has information on how to build a more complete and functional firewall. Click **Forward**.

SELinux (*RHEL*) SELinux (Security Enhanced Linux) enforces security policies that limit what a user or a program can do. On this screen RHEL allows you to choose one of two policies: **Enforcing** or **Permissive**. Alternatively, you can disable SELinux. If you enable SELinux, you can modify its policy. The policy defaults to Enforcing, which prevents any user or program from doing anything that is not permitted by the policy. If you will never want to use SELinux, disable it. If you do not want to use it now but may want to do so in the future, establish a Permissive policy—it issues warnings but does not enforce the policy. It can take a lot of time to turn on SELinux on a system where it has been disabled. For more information refer to "SELinux" on page 414. Click **Forward**.

Create User The next screen allows you to set up a user account. For more information refer to "Configuring User and Group Accounts" on page 556.

Date and Time The next screen allows you to set the system date and time. Running the Network Time Protocol (NTP) causes the system clock to reset itself periodically from a clock on the Internet. If the system is connected to the Internet, you can enable NTP by putting a tick in the check box labeled **Synchronize date and time over the network**. Click **Forward**.

When the Date and Time screen closes, the installation is complete. You can now use the system and set it up as you desire. For example, you may want to customize the desktop (as explained in Chapters 4 and 8) or set up servers (as discussed in Part V of this book).

INITIALIZING DATABASES AND UPDATING THE SYSTEM

Updating the **whatis** database ensures that the whatis (page 168) and apropos (page 167) utilities will work properly. Similarly, updating the **locate** database ensures that locate will work properly. (The locate utility indexes and allows you to search for files on the system quickly and securely.) Instead of updating these databases when you install the system, you can wait for cron (page 565) to run them, but be aware that whatis, apropos, and locate will not work for a while. The best way to update these databases is via the cron scripts that run daily. Working with **root** privileges (Superuser; page 405), give the following commands:

```
# /etc/cron.daily/makewhatis.cron
# /etc/cron.daily/mlocate.cron
```

These utilities run for several minutes and may complain about not being able to find a file or two. When the system displays a prompt, the **whatis** and **locate** databases are up-to-date.

Figure 3-13 The Welcome screen displaying boot parameters (options)

INSTALLATION TASKS

This section details some common tasks you may need to perform during or after installation. It covers modifying the boot parameters, using Disk Druid to partition the disk during installation, using palimpsest to view and modify partitions, using logical volumes (LVs) to facilitate disk partitioning, using Kickstart to automate installation, and setting up a system that will boot either Windows or Linux (a dual-boot system).

MODIFYING BOOT PARAMETERS (OPTIONS)

FEDORA To modify boot parameters, you must interrupt the automatic boot process by pressing a key such as the SPACE bar while Fedora is counting down when you first boot from a live CD (page 52) or install DVD (page 55). When you press a key, Fedora displays the Welcome menu (Figure 3-2 on page 53 or Figure 3-4 on page 56). Use the ARROW keys to highlight the selection you want *before* proceeding (page 56). With the desired selection highlighted, press the TAB key to display the boot command-line parameters (Figure 3-13).

RHEL RHEL presents a **boot:** prompt in place of the boot parameters line Fedora displays when you press TAB. You can enter any of the parameters described in this section in response to the **boot:** prompt; however, you must precede these parameters with the word **linux**. (See the examples in the next paragraphs.)

RHEL+FEDORA Type a SPACE before you enter any parameters. You can specify multiple parameters separated by SPACEs. Press RETURN to boot the system. For more information on boot parameters, refer to www.kernel.org/doc/Documentation/kernel-parameters.txt, www.kernel.org/pub/linux/kernel/people/gregkh/lkn/lkn_pdf/ch09.pdf, or the Web page at fedoraproject.org/wiki/Anaconda/Options. Alternatively, you can use Google to search on **linux boot parameters**.

What to do if the installation does not work

tip On some hardware, the installation may pause for as long as ten minutes. Before experimenting with other fixes, try waiting for a while. If the installation hangs, try booting with one or more of the boot parameters described in this section. Try running the installer in pseudographical (textual) mode.

Following are some of the parameters you can add to the boot command line. If you encounter problems with the display during installation, supply the **nofb** parameter, which turns off video memory. If you are installing from a medium other than a DVD—that is, if you are installing from files on the local hard disk or from files on another system using FTP, NFS, or HTTP—supply the **askmethod** or **method** parameter.

Many of these parameters can be combined. For example, to install Linux in text mode using a terminal running at 115,200 baud, no parity, 8 bits, connected to the first serial device, supply the following parameters (the **,115200n8** is optional). The first line shows the parameters you enter while booting Fedora. The second line shows the parameters, including **linux**, you enter in response to the **boot:** prompt while booting RHEL.

```
text console=ttyS0,115200n8 FEDORA
boot: linux text console=ttyS0,115200n8 RHEL
```

The next set of parameters installs Fedora/RHEL on a monitor with a resolution of 1024 × 768, without probing for any devices. The installation program asks you to specify the source of the installation data (CD, DVD, FTP site, or other) and requests a video driver.

```
resolution=1024x768 noprobe askmethod FEDORA
boot: linux resolution=1024x768 noprobe askmethod RHEL
```

noacpi Disables ACPI (Advanced Configuration and Power Interface). This parameter is useful for systems that do not support ACPI or that have problems with their ACPI implementation. The default is to enable ACPI. Also **acpi=off**.

noapic Disables APIC (Advanced Programmable Interrupt Controller). The default is to enable APIC.

noapm Disables APM (Advanced Power Management). The default is to enable APM. Also **apm=off**.

askmethod Presents a choice of installation sources: local CD/DVD or hard disk, or over a network using NFS, FTP, or HTTP (first installation CD, Net Boot CD, and install DVD only).

- **Local CD/DVD**—Displays the Disc Found screen, which allows you to test the installation media (the same as if you had not entered any boot parameters).
- **Hard drive**—Prompts for the partition and directory that contain the installation tree or the ISO image of the install DVD. Do not include the name of the mount point when you specify the name of the directory. For example, if the ISO images are in the **/home/sam/FC12** directory and **/dev/sda6** holds the partition that is normally mounted on **/home**, you would specify the partition as **/dev/sda6** and the directory as **sam/FC12** (no leading slash).
- The next two selections display the Configure TCP/IP screen from which you can select DHCP or Manual configuration. Manual configuration requires you to enter the system's IP address and netmask as well as the IP addresses of the default gateway and primary nameserver.
  - **NFS directory**—Displays the NFS Setup screen, which requires you to enter the NFS server name and the name of the directory that contains the installation tree or the ISO image of the install DVD. Enter the server's IP address and the *name* of the exported directory, not its device name. The remote (server) system must export (page 739) the directory hierarchy that holds the installation tree or the ISO image of the install DVD.
  - **URL**—Displays the URL Setup screen, which requires you to enter the URL of the directory that contains the installation tree or the ISO image of the install DVD.

nodma Turns off direct memory access (DMA) for all disk controllers. This parameter may make buggy controllers (or controllers with buggy drivers) more reliable, but also causes them to perform very slowly because the connected devices have to run in PIO mode instead of DMA mode. It may facilitate testing CD/DVDs that were not written correctly. For more information refer to "The Disc Found Screen" on page 57.

nofb (**no framebuffer**) Turns off the framebuffer (video memory). This option is useful if problems arise when the graphical phase of the installation starts.

irqpoll Changes the way the kernel handles interrupts.

ks=***URI*** Specifies the location of a Kickstart (page 82) file to use to control the installation process. The ***URI*** is the pathname or network location of the Kickstart file.

nolapic Disables local APIC. The default is to enable local APIC.

lowres Runs the installation program at a resolution of 640 × 480 pixels. See also **resolution**.

mem=*xxx*M Overrides the detected memory size. Replace *xxx* with the number of megabytes of RAM in the computer.

method=*URI* Specifies an installation method and location without prompting as **askmethod** does. For example, you can use the following parameter to start installing from the specified server:

```
method=ftp://download.fedora.redhat.com/pub/fedora/linux/releases/12/Fedora/i386/os
```

noprobe Disables hardware probing for all devices, including network interface cards (NICs), graphics cards, and the monitor. This option forces you to select devices from a list. You must know exactly which cards or chips the system uses when you use this parameter. Use **noprobe** when probing causes the installation to hang or otherwise fail. This parameter allows you to supply arguments for each device driver you specify.

rescue Puts the system in rescue mode; see page 411 for details.

resolution=*WxH* Specifies the resolution of the monitor you are using for a graphical installation. For example, **resolution=1024x768** specifies a monitor with a resolution of 1024 × 768 pixels.

text Installs Linux in pseudographical (page 28) mode. Although the images on the screen appear to be graphical, they are composed entirely of text characters.

vnc Installs Linux via a VNC (virtual network computing) remote desktop session. After providing an IP address, you can control the installation remotely using a VNC client from a remote computer. You can download a VNC client, which runs on several platforms, from www.realvnc.com. Use yum (page 500) to install the **vnc** software package to run a VNC client on a Fedora/RHEL system.

vncpassword=*passwd* Enables a password for a VNC connection. This option requires that you also use the **vnc** option.

PARTITIONING THE DISK

See page 30 for a discussion of partitions and setup of the hard disk.

USING DISK DRUID TO PARTITION THE DISK

Disk Druid, a graphical disk-partitioning program that can add, delete, and modify partitions on a hard disk, is part of the Fedora/RHEL installation system. You can use Disk Druid only while you are installing a system; it cannot be run on its own. You can use palimpsest (page 78), parted (page 568), or fdisk to manipulate partitions and system-config-lvm to work with LVs after you install Fedora/RHEL. As explained earlier, if you want a basic set of partitions, you can allow Anaconda to partition the hard disk automatically.

Anaconda runs Disk Druid when you put a tick in the check box labeled **Review and modify partitioning layout** or when you select **Create custom layout** in the Disk Partitioning screen (Figure 3-7, page 61).

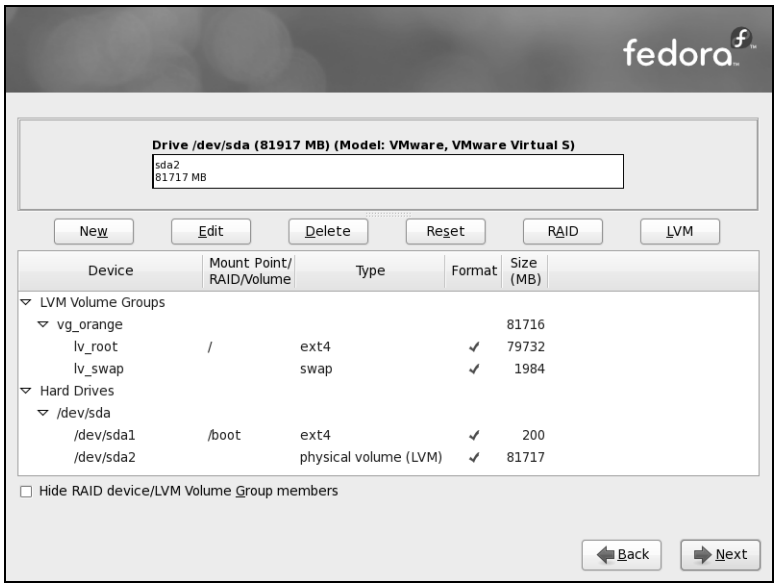

Figure 3-14 Disk Druid: main screen, default layout

Clone and RAID

Disk Druid includes Clone, a tool that copies the partitioning scheme from a single drive to other drives. Clone is useful for making multiple copies of a RAID partition/drive when you are creating a large RAID array of identical partitions or identically partitioned drives. Click the **RAID** button to access the Clone tool, which is active only when at least one unallocated RAID partition exists. For more information on RAID, see page 37.

Default layout

Figure 3-14 shows the Disk Druid main screen as it appears when you have chosen the default layout for the hard disk (see "Disk Partitioning" on page 60). This screen has three sections (from top to bottom): a graphical representation of the disk drives showing how each is partitioned, a row of buttons, and a table listing one partition or LV per line. In the figure, the graphical depiction of the **/boot** partition is so small that its descriptive text is covered by the other partition.

The following buttons appear near the top of the screen:

- **New**—Adds a new partition to the disk (page 76)
- **Edit**—Edits the highlighted partition or LV (both on page 73)
- **Delete**—Deletes the highlighted partition or LV
- **Reset**—Cancels the changes you have made and causes the Disk Druid table to revert so it matches the layout of the disk

- **RAID**—Enables you to create software RAID partitions and to join two or more RAID partitions into a RAID device (page 37)
- **LVM**—Enables you to create physical volumes (PVs), which you can then use to create LVs (page 38)

The Disk Druid table contains the following columns:

- **Device**—The name of the device in the **/dev** directory (for example, **/dev/sda1** or the name of the LV).
- **Mount Point/RAID/Volume**—Specifies where the partition will be mounted when the system is brought up (for example, **/usr**). It is also used to specify the RAID device or LVM volume the partition is part of.
- **Type**—The type of the partition, such as **ext4**, **swap**, or **physical volume (LVM)**.
- **Format**—A tick in this column indicates the partition will be formatted as part of the installation process. All data on the partition will be lost.
- **Size (MB)**—The size of the partition or LV in megabytes.
- **Start**—The number of the block the partition starts on. (*RHEL*)
- **End**—The number of the block the partition ends on. (*RHEL*)

At the bottom of the screen is a check box that allows you to hide RAID device and LVM volume group members. Do not put a tick in this check box if you want to see all information about the disk drives.

blank

8

Linux GUIs: X and GNOME

This chapter covers the Linux graphical user interface (GUI). It continues where Chapter 4 left off, going into more detail about the X Window System, the basis for the Linux GUI. It presents a brief history of GNOME and KDE and discusses some of the problems and benefits of having two major Linux desktop environments. The section on the Nautilus File Browser covers the View and Side panes, the control bars, the menubar, and the Spatial view. The final section explores some GNOME utilities, including Terminal, the GNOME terminal emulator.

In This Chapter

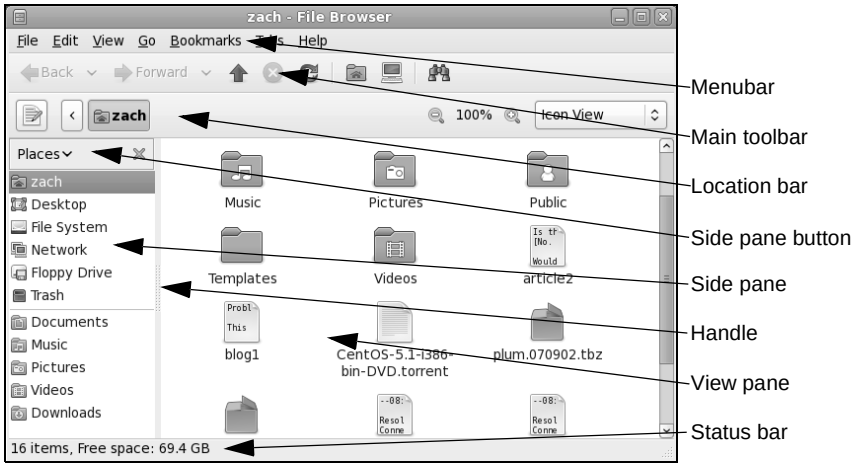

Figure 8-2 A Nautilus File Browser window displaying icons

THE NAUTILUS FILE BROWSER WINDOW

"Using Nautilus to Work with Files" on page 98 presented an introduction to using Nautilus. This section discusses the Nautilus File Browser window in more depth.

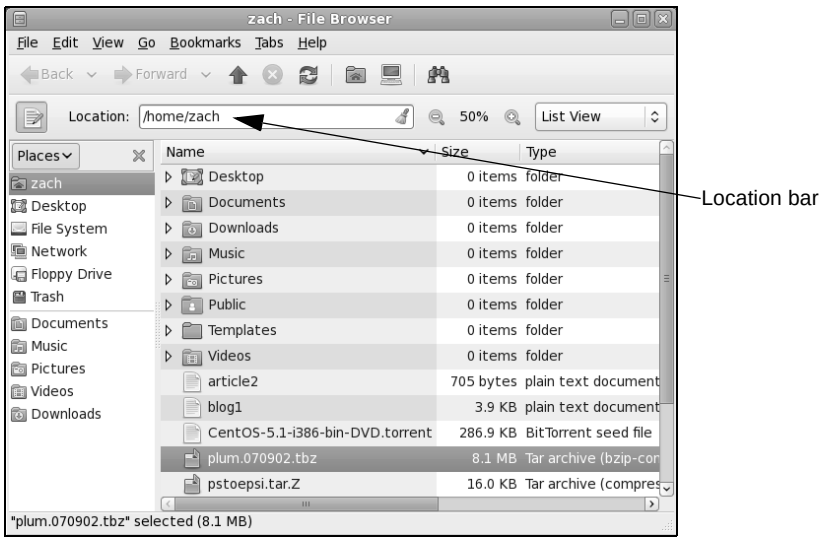

Figure 8-3 A Nautilus File Browser window displaying a List view and a textual location bar

Turn off Spatial view; turn on File Browser windows

tip To make the Nautilus windows on the desktop you are working on correspond to the figures in this book, you must turn off Spatial view (page 270) and turn on File Browser windows. For more information refer to "The Two Faces of Nautilus" on page 99.

Figure 8-2 shows a File Browser window with a Side pane (sometimes called a *side-bar*), View pane, menubar, toolbar, location bar, and status bar. To display your home folder in a File Browser window, select **Main menu: Places⇨Home Folder.**

THE VIEW PANE

The View pane displays icons or a list of filenames. Select the view you prefer from the drop-down list at the right end of the location bar. Figure 8-2 shows an Icon view and Figure 8-3 shows a List view. A Compact view is also available. Objects in the View pane behave exactly as objects on the desktop do. See the sections starting on page 92 for information on working with objects.

You can cut/copy and paste objects within a single View pane, between View panes, or between a View pane and the desktop. The Object context menu (right-click) has cut, copy, and paste selections. Alternatively, you can use the clipboard (page 116) to cut/copy and paste objects.

THE SIDE PANE

The Side pane augments the information Nautilus displays in the View pane. Press F9 or click the **X** at the top of the Side pane to close it. You can display the Side pane by pressing F9 or by selecting **File Browser menubar: View⇨Side Pane**. To change the horizontal size of the Side pane, drag the handle (Figure 8-2) on its right side.

Nautilus can open a terminal emulator

tip When you install the **nautilus-open-terminal** package (see page 501 for instructions) and log out and log back in, Nautilus presents an Open in Terminal selection in context menus where appropriate. For example, with this package installed, when you right-click a folder (directory) object and select **Open in Terminal**, Nautilus opens a terminal emulator with that directory as the working directory (page 192).

The Side pane can display six types of information. The button at its top controls which type it displays. This button is initially labeled **Places**; click it to display the Side pane drop-down list, which has the selections described next.

Places Places lists folders. Double-click one of these folders to display that folder in the View pane. You can open a directory in a new File Browser window by right-clicking the directory in Places and selecting **Open in New Window**. Under Fedora you can right-click and select **Open in New Tab** to open the directory in a new tab.

Places contains two parts: The list above the divider is static and holds your home directory, your desktop, the filesystem, the network, a CD-ROM drive (when it contains a disk), unmounted filesystems (if present), and the trash. The list below the divider holds bookmarks. Add a bookmark by displaying the directory you want to bookmark in the View pane and pressing CONTROL-D or by selecting **File Browser menubar: Bookmarks⇨Add Bookmark**. Remove a bookmark by selecting **File Browser menubar: Bookmarks⇨Edit Bookmarks** or by right-clicking the bookmark and selecting **Remove**. You can also use **Edit Bookmarks** to reorder bookmarks.

Information Information presents information about the folder displayed by the View pane.

Tree Tree presents an expandable tree view of your home folder, and each mounted filesystem. Each directory in the tree has a triangle to its left. Click a triangle that points right to expand a directory; click a triangle that points down to close a directory. Click a directory in the tree to display that directory in the View pane. Double-click a directory to expand it in the Side pane and display it in the View pane.

History History displays a chronological list of the folders that have been displayed in the View pane, with the most recently displayed folder at the top. Double-click a folder in this list to display it in the View pane.

Notes Notes provides a place to keep notes about the folder displayed in the View pane.

Emblems Similar to the Emblems tab in the Object Properties window (page 120), Emblems allows you to drag emblems from the Side pane and drop them on objects in the View pane. Drag and drop the **Erase** emblem to erase emblems associated with an object. You cannot erase emblems that Fedora/RHEL places on objects, such as locked and link emblems.

CONTROL BARS

This section discusses the four control bars that initially appear in a File Browser window: the status bar, menubar, Main toolbar, and location bar (Figure 8-2). From

File Browser menubar: View, you can choose which of these bars to display—except for the menubar, which Nautilus always displays.

Menubar The menubar appears at the top of the File Browser window and displays a menu when you click one of its selections. Which menu selections Nautilus displays depend on what the View pane is displaying and which objects are selected. The next section describes the menubar in detail.

Main toolbar The Main toolbar appears below the menubar and holds navigation tool icons: Back, Forward, Up, Stop, Reload, Home, Computer, and Search. If the Main toolbar is too short to hold all icons, Nautilus displays a button with a triangle pointing down at the right end of the toolbar. Click this button to display a drop-down list of the remaining icons.

Location bar Below the Main toolbar is the location bar, which displays the name of the directory that appears in the View pane. It can display this name in two formats: iconic (using buttons) and textual (using a text box). Press CONTROL-L to switch to textual format, and click the pencil-and-paper icon at the left of this bar to switch between iconic and textual formats.

In iconic format, each button represents a directory in a pathname (page 193). The View pane displays the directory of the depressed (darker) button. Click one of these buttons to display that directory. If the leftmost button holds a triangle that points to the left, Nautilus is not displaying buttons for all the directories in the absolute (full) pathname; click the button with a triangle in it to display more directory buttons.

In textual format, the text box displays the absolute pathname of the displayed directory. To have Nautilus display another directory, enter the pathname of the directory and press RETURN.

The location bar also holds the magnification selector and the View drop-down list. To change the magnification of the display in the View pane, click the plus or minus sign in a magnifying glass on either side of the magnification percentage. Right-click the magnification percentage itself to return to the default magnification. Left-click the magnification percentage to display a drop-down list of magnifications. Click the button found at the right side of the right-hand magnifying glass to choose whether to view files as icons, as a list, or in compact format.

Status bar If no items are selected, the status bar, at the bottom of the window, indicates how many items are displayed in the View pane. If the directory you are viewing is on the local system, it also tells you how much free space is available on the device that holds the directory displayed by the View pane. If an item is selected, the status bar displays the name of the item and its size.

MENUBAR

The Nautilus File Browser menubar controls which information the File Browser displays and how it displays that information. Many of the menu selections duplicate controls found elsewhere in the File Browser window. This section highlights

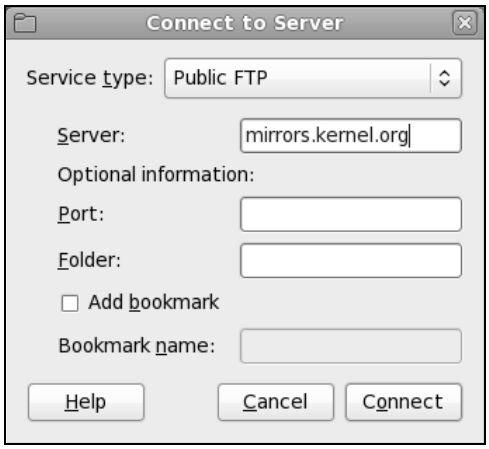

Figure 8-4 The Connect to Server window

some of the selections on the menubar; click **Help** on the menubar and select **Contents** for more information. The menubar holds the menus described next.

File The several Open selections and the Property selection of File work with the highlighted object(s) in the View pane. If no objects are highlighted, these selections are grayed out or absent. Selecting **Connect to Server** (also available from **Main menu: Places**) displays the Connect to Server window (Figure 8-4). This window presents a **Service type** drop-down list that allows you to select FTP, SSH, Windows, or other types of servers. Enter the URL of the server in the text box labeled **Server**. For an FTP connection, do not enter the **ftp://** part of the URL. Fill in the optional information as appropriate. Click **Connect**. If the server requires authentication, Nautilus displays a window in which you can enter a username and password. Nautilus opens a window displaying a directory on the server and an object, named for the URL you specified, on the desktop. After you close the window, you can open the object to connect to and display a directory on the server.

Edit Many of the Edit selections work with highlighted object(s) in the View pane; if no objects are highlighted, these selections are grayed out or absent. This section discusses three selections from Edit: Compress, Backgrounds and Emblems, and Preferences.

The **Edit⇨Compress** selection creates a single archive file comprising the selected objects. This selection opens a Compress window (Figure 8-5) that allows you to specify the name and location of the archive. The drop-down list to the right of the text box labeled **Filename** allows you to specify a filename extension that determines the type of archive this tool creates. For example, **.tar.gz** creates a tar (page 163) file compressed by gzip (page 163) and **.tar.bz2** creates a tar file compressed by bzip2 (page 162). Click the triangle to the left of Other Objects to specify a password for and/or to encrypt the archive (available only with certain types of archives). You can also split the archive into several files (volumes).

Figure 8-5 The Compress window

The **Edit⇨Backgrounds and Emblems** selection has three buttons on the left: Patterns, Colors, and Emblems. Click **Patterns** to display many pattern objects on the right side of the window. Drag and drop one of these objects on the View pane of a File Browser window to change the background of all File Browser View panes. Drag and drop the Reset object to reset the background to its default color and pattern (usually white). The Colors button works the same way as the Patterns button. The Emblems button works the same way as the Emblems tab in the Side pane (page 266).

The **Edit⇨Preferences** selection displays the File Management Preferences window (Figure 8-6). This window has six tabs that control the appearance and behavior of File Browser windows.

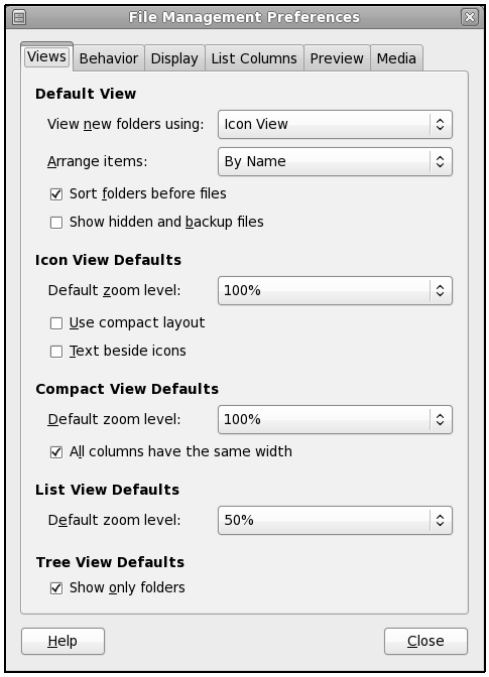

Figure 8-6 The File Management Preferences window, Views tab

The **Views** tab sets several defaults, including which view the File Browser displays (Icon, List, or Compact view), the arrangement of the objects, the default zoom level, and default settings for the Compact view.

Delete versus Move to Trash

The **Behavior** tab controls how many clicks it takes to open an object and what Nautilus does when it opens an executable text object (script). For more confident users, this tab has an option that includes a Delete selection in addition to the Move to Trash selection on several menus. The Delete selection immediately removes the selected object instead of moving it to the **Trash** folder. This tab also holds the check box labeled **Always open in browser window** that is described under "The Two Faces of Nautilus" on page 99.

The **Display** tab specifies which information Nautilus includes in object (icon) captions. The three drop-down lists specify the order in which Nautilus displays information as you increase the zoom level of the View pane. This tab also specifies the date format Nautilus uses.

The **List Columns** tab specifies which columns Nautilus displays, and in what order it displays them, in the View pane when you select **List View**.

The **Preview** tab controls when Nautilus displays or plays previews of files (Always, Local Files Only, Never).

The **Media** tab specifies which action Nautilus takes when you insert media such as a CD/DVD, or connect devices such as a flash drive, to the system.

View Click the **Main Toolbar**, **Side Pane**, **Location Bar**, and **Statusbar** selections in the View submenu to display or remove these elements from the window. The **Show Hidden Files** selection displays in the View pane those files with hidden filenames (page 192).

Go The Go selections display various folders in the View pane.

Bookmarks Bookmarks appear at the bottom of this menu and in the Side pane under Places. The Bookmarks selections are explained under "Places" on page 266.

Tabs The Tabs selections work with tabs in the Nautilus window (*FEDORA*).

Help The Help selections display local information about Nautilus.

optional

THE NAUTILUS SPATIAL VIEW

Nautilus gives you two ways to work with files: the traditional File Browser view described in the previous section and the innovative Spatial view shown in Figure 8-7. By default, Fedora/RHEL display the Spatial view. Other than in this section, this book describes the more traditional File Browser window. See "The Two Faces of Nautilus" on page 99 for instructions on how to turn off the Spatial view and turn on the File Browser.

The Nautilus Spatial (as in "having the nature of space") view has many powerful features but may take some getting used to. It always provides one window per folder. By default, when you open a folder, Nautilus displays a new window.

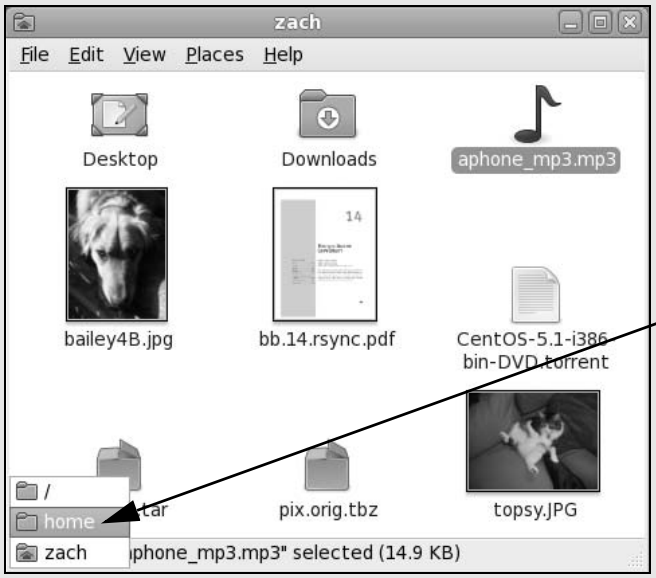

Figure 8-7 The Nautilus Spatial view

To open a Spatial view of your home directory, double-click the Home icon on the desktop and experiment as you read this section. If you double-click the Desktop icon in the Spatial view, Nautilus opens a new window that displays the **Desktop** folder.

A Spatial view can display icons, a list of filenames, or a compact view. To select your preferred format, click **View** on the menubar and choose **Icons**, **List**, or **Compact**. To create files to experiment with, right-click in the window (not on an icon) to display the Nautilus context menu and select **Create Folder** or **Create Document**.

Use SHIFT to close the current window as you open another window

tip If you hold the SHIFT key down when you double-click to open a new window, Nautilus closes the current window as it opens the new one. This behavior may be more familiar and can help keep the desktop from becoming overly cluttered. If you do not want to use the keyboard, you can achieve the same result by double-clicking the middle mouse button.

Window memory

Move the window by dragging the titlebar. The Spatial view has *window memory*—that is, the next time you open that folder, Nautilus opens it at the same size and in the same location. Even the scrollbar will be in the same position.

Parent-folders button

The key to closing the current window and returning to the window of the parent directory is the **Parent-folders** button (Figure 8-7). Click this button to display the Parent-folders pop-up menu. Select the directory you want to open from this menu. Nautilus then displays in a Spatial view the directory you specified.

From a Spatial view, you can open a folder in a traditional view by right-clicking the folder and selecting **Browse Folder**.

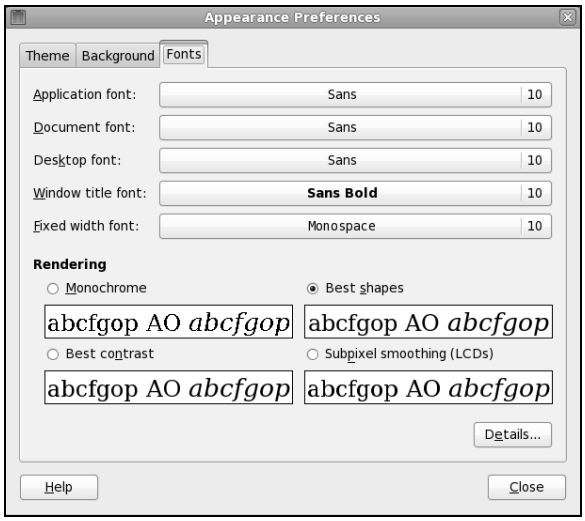

Figure 8-8 The Appearance Preferences window, Fonts tab

GNOME UTILITIES

GNOME comes with numerous utilities that can make your work with the desktop easier and more productive. This section covers several tools that are integral to the use of GNOME.

FONT PREFERENCES (*FEDORA*)

The Fonts tab of the Appearance Preferences window (Figure 8-8) enables you to change the font GNOME uses for applications, documents, the desktop, window titles, and terminal emulators (fixed width). To display this window, select **Main menu: System⇨Preferences⇨Appearance** or enter **gnome-appearance-properties** on a command line. Click the **Fonts** tab. Click one of the five font bars in the upper part of the window to display the Pick a Font window (discussed next).

Examine the four sample boxes in the lower part of the window and select the one in which the letters look the best. Subpixel smoothing is usually best for LCD monitors. Click **Details** to refine the font rendering further, again picking the box in each section in which the letters look the best.

PICK A FONT WINDOW (*FEDORA*)

The Pick a Font window (Figure 8-9) appears when you need to choose a font; see the previous section. From this window you can select a font family, a style, and a size. A preview of your choice appears in the Preview frame in the lower part of the window. Click **OK** when you are satisfied with your choice.

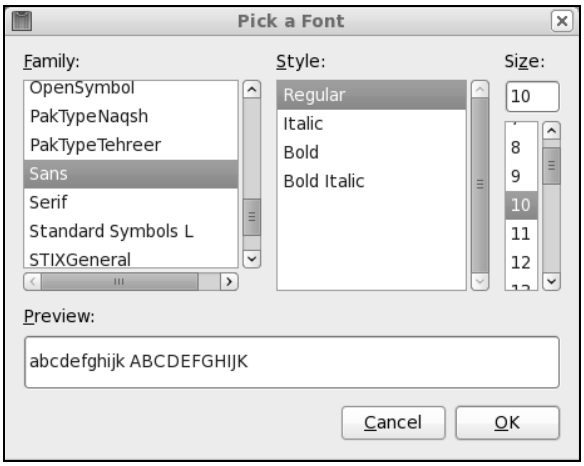

Figure 8-9 The Pick a Font window

PICK A COLOR WINDOW

The Pick a Color window (Figure 8-10) appears when you need to specify a color, such as when you specify a solid color for the desktop background (page 107) or a panel. To specify a color for a panel, right-click the panel to display its context menu, select **Properties**, click the **Background** tab, click the radio button labeled **Solid color**, and click within the box labeled **Color**. GNOME displays the Pick a Color window.

When the Pick a Color window opens, the bar below the color circle displays the current color. Click the desired color on the color ring, and click/drag the lightness of that color in the triangle. As you change the color, the right end of the bar below the color circle previews the color you are selecting, while the left end continues to display the current color. You can also use the eyedropper to pick up a color from the workspace: Click the eyedropper, and then click the resulting eyedropper mouse pointer on the color you want to select. The color you choose appears in the bar. Click **OK** when you are satisfied with the color you have specified.

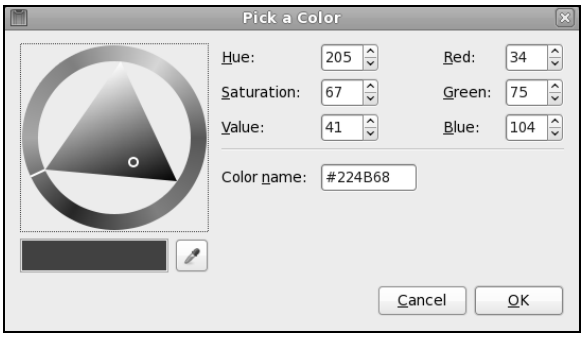

Figure 8-10 The Pick a Color window

blank

11

System Administration: Core Concepts

The job of a system administrator is to keep one or more systems in a useful and convenient state for users. On a Linux system, the administrator and user may both be you, with you and the computer being separated by only a few feet. Or the system administrator may be halfway around the world, supporting a network of systems, with you being simply one of thousands of users. A system administrator can be one person who works part-time taking care of a system and perhaps is also a user of the system. Or the administrator can be several people, all working full-time to keep many systems running.

In This Chapter

blank

SECURING A SERVER

You may secure a server either by using TCP wrappers or by setting up a chroot jail.

TCP WRAPPERS: CLIENT/SERVER SECURITY (hosts.allow AND hosts.deny)

When you open a local system to access from remote systems, you must ensure that the following criteria are met:

- Open the local system only to systems you want to allow to access it.
- Allow each remote system to access only the data you want it to access.
- Allow each remote system to access data only in the appropriate manner (readonly, read/write, write only).

As part of the client/server model, TCP wrappers, which can be used for any daemon that is linked against **libwrap.so**, rely on the **/etc/hosts.allow** and **/etc/hosts.deny** files as the basis of a simple access control language. This access control language defines rules that selectively allow clients to access server daemons on a local system based on the client's address and the daemon the client tries to access.

Each line in the **hosts.allow** and **hosts.deny** files has the following format:

daemon_list : client_list [: command]

where ***daemon_list*** is a comma-separated list of one or more server daemons (such as **rpcbind**, **vsftpd**, or **sshd**), ***client_list*** is a comma-separated list of one or more clients (see Table 11-3, "Specifying a client," on page 442), and the optional ***command***

is the command that is executed when a client from ***client_list*** tries to access a server daemon from ***daemon_list***.

When a client requests a connection with a local server, the **hosts.allow** and **hosts.deny** files are consulted in the following manner until a match is found:

1. If the daemon/client pair matches a line in **hosts.allow**, access is granted.
2. If the daemon/client pair matches a line in **hosts.deny**, access is denied.
3. If there is no match in either the **hosts.allow** or **hosts.deny** files, access is granted.

The first match determines whether the client is allowed to access the server. When either **hosts.allow** or **hosts.deny** does not exist, it is as though that file was empty. Although it is not recommended, you can allow access to all daemons for all clients by removing both files.

Examples For a more secure system, put the following line in **hosts.deny** to block all access:

```
$ cat /etc/hosts.deny
...
ALL : ALL : echo '%c tried to connect to %d and was blocked' >> /var/log/tcpwrappers.log
```

This line prevents any client from connecting to any service, unless specifically permitted in **hosts.allow**. When this rule is matched, it adds a line to the file named **/var/log/tcpwrappers.log**. The **%c** expands to client information and the **%d** expands to the name of the daemon the client attempted to connect to.

With the preceding **hosts.deny** file in place, you can include lines in **hosts.allow** that explicitly allow access to certain services and systems. For example, the following **hosts.allow** file allows anyone to connect to the OpenSSH daemon (ssh, scp, sftp) but allows telnet connections only from the same network as the local system and users on the 192.168. subnet:

```
$ cat /etc/hosts.allow
sshd : ALL
in.telnet : LOCAL
in.telnet : 192.168.* 127.0.0.1
...
```

The first line allows connection from any system (ALL) to **sshd**. The second line allows connection from any system in the same domain as the server (LOCAL). The third line matches any system whose IP address starts with **192.168.** as well as the local system.

SETTING UP A chroot JAIL

On early UNIX systems, the root directory was a fixed point in the filesystem. On modern UNIX variants, including Linux, you can define the root directory on a per-process basis. The chroot utility allows you to run a process with a root directory other than **/**.

The root directory appears at the top of the directory hierarchy and has no parent: A process cannot access any files above the root directory (because they do not exist). If, for example, you run a program (process) and specify its root directory as **/home/sam/jail**, the program would have no concept of any files in **/home/sam** or above: **jail** is the program's root directory and is labeled **/** (not **jail**).

By creating an artificial root directory, frequently called a (chroot) jail, you prevent a program from accessing or modifying—possibly maliciously—files outside the directory hierarchy starting at its root. You must set up a chroot jail properly to increase security: If you do not set up the chroot jail correctly, you can actually make it easier for a malicious user to gain access to a system than if there were no chroot jail.

USING chroot

Creating a chroot jail is simple: Working as **root**, give the command **/usr/sbin/chroot** ***directory***. The ***directory*** becomes the root directory and the process attempts to run the default shell. Working with **root** privileges from the **/home/sam** directory, give the following command to set up a chroot jail in the (existing) **/home/sam/jail** directory:

```
# /usr/sbin/chroot /home/sam/jail
/usr/sbin/chroot: cannot run command '/bin/bash': No such file or directory
```

This example sets up a chroot jail, but when it attempts to run the bash shell, the operation fails. Once the jail is set up, the directory that was named **jail** takes on the name of the root directory, **/**, so chroot cannot find the file identified by the pathname **/bin/bash**. In this situation the chroot jail is working but is not useful.

Getting a chroot jail to work the way you want is a bit more complicated. To have the preceding example run bash in a chroot jail, you need to create a **bin** directory in **jail** (**/home/sam/jail/bin**) and copy **/bin/bash** to this directory. Because the bash binary is dynamically linked to shared libraries, you need to copy these libraries into **jail** as well. The libraries go in **lib**.

The next example creates the necessary directories, copies **bash**, uses ldd to display the shared library dependencies of bash, and copies the necessary libraries into **lib**. The **linux-gate.so.1** file is a dynamically shared object (DSO) provided by the kernel to speed system calls; you do not need to copy it to the **lib** directory.

```
$ pwd
/home/sam/jail
$ mkdir bin lib
$ cp /bin/bash bin
$ ldd bin/bash
        linux-gate.so.1 =>  (0x0089c000)
        libtinfo.so.5 => /lib/libtinfo.so.5 (0x00cdb000)
        libdl.so.2 => /lib/libdl.so.2 (0x00b1b000)
        libc.so.6 => /lib/libc.so.6 (0x009cb000)
        /lib/ld-linux.so.2 (0x009ae000)
$ cp /lib/{libtinfo.so.5,libdl.so.2,libc.so.6,ld-linux.so.2} lib
```

Now that everything is set up, you can start the chroot jail again. Although all of the setup can be done by an ordinary user, you have to run chroot as Superuser:

```
$ su
Password:
# /usr/sbin/chroot .
bash-3.2# pwd
/
bash-3.2# ls
bash: ls: command not found
bash-3.2#
```

This time the chroot finds and starts **bash**, which displays its default prompt (**bash-3.2#**). The pwd command works because it is a shell builtin (page 247). However, bash cannot find the ls utility (it is not in the chroot jail). You can copy **/bin/ls** and its libraries into the jail if you want users in the jail to be able to use ls.

To set up a useful chroot jail, first determine which utilities the users of the chroot jail will need. Then copy the appropriate binaries and their libraries into the jail. Alternatively, you can build static copies of the binaries and put them in the jail without installing separate libraries. (The statically linked binaries are considerably larger than their dynamic counterparts. The base system with bash and the core utilities exceeds 50 megabytes.) You can find the source code for most of the common utilities in the **bash** and **coreutils** SRPMS (source rpm) packages.

Whichever technique you choose, you must put a copy of su in the jail. The su command is required to run programs while working as a user other than **root**. Because **root** can break out of a chroot jail, it is imperative that you run a program in the chroot jail as a user other than **root**.

The dynamic version of su distributed by Fedora/RHEL requires PAM and will not work within a jail. You need to build a copy of su from the source to use in a jail. By default, any copy of su you build does not require PAM. Refer to "GNU Configure and Build System" on page 513 for instructions on how to build packages such as **coreutils** (which includes su).

To use su, you must copy the relevant lines from the **/etc/passwd** and **/etc/shadow** files into files with the same names in the **etc** directory inside the jail.

Keeping multiple chroot jails

tip If you plan to deploy multiple chroot jails, it is a good idea to keep a clean copy of the **bin** and **lib** files somewhere other than in one of the active jails.

RUNNING A SERVICE IN A chroot JAIL

Running a shell inside a jail has limited usefulness. In reality, you are more likely to need to run a specific service inside the jail. To run a service inside a jail, you must make sure all files needed by that service are inside the jail. The format of a command to start a service in a chroot jail is

```
# /usr/sbin/chroot jailpath /bin/su user daemonname &
```

where *jailpath* is the pathname of the jail directory, *user* is the username that runs the daemon, and *daemonname* is the path (inside the jail) of the daemon that provides the service.

Some servers are already set up to take advantage of chroot jails. For example, you can set up DNS so that **named** runs in a jail (page 804), and the **vsftpd** FTP server can automatically start chroot jails for clients (page 658).

SECURITY CONSIDERATIONS

Some services need to be run as **root**, but they release their **root** privileges once started (Procmail and **vsftpd** are examples). If you are running such a service, you do not need to put su inside the jail.

A process run as **root** could potentially escape from a chroot jail. For this reason, you should always su to another user before starting a program running inside the jail. Also, be careful about which setuid (page 205) binaries you allow inside a jail—a security hole in one of them could compromise the security of the jail. In addition, make sure the user cannot access executable files that he uploads.

DHCP: CONFIGURES HOSTS

Instead of storing network configuration information in local files on each system, DHCP (Dynamic Host Configuration Protocol) enables client systems to retrieve network configuration information each time they connect to the network. A DHCP server assigns IP addresses from a pool of addresses to clients as needed. Assigned addresses are typically temporary, but need not be.

This technique has several advantages over storing network configuration information in local files:

- A new user can set up an Internet connection without having to deal with IP addresses, netmasks, DNS addresses, and other technical details. An experienced user can set up a connection more quickly.
- DHCP facilitates assignment and management of IP addresses and related network information by centralizing the process on a server. A system administrator can configure new systems, including laptops that connect to the network from different locations, to use DHCP; DHCP then assigns IP addresses only when each system connects to the network. The pool of IP addresses is managed as a group on the DHCP server.
- IP addresses can be used by more than one system, reducing the total number of IP addresses needed. This conservation of addresses is important because the Internet is quickly running out of IPv4 addresses. Although a particular IP address can be used by only one system at a time, many end-user systems require addresses only occasionally, when they connect to the Internet. By reusing IP addresses, DHCP lengthens the life of the IPv4 protocol. DHCP applies to IPv4 only, as IPv6 forces systems to configure their IP addresses automatically (called autoconfiguration) when they connect to a network (page 373).

DHCP is particularly useful for administrators who are responsible for maintaining a large number of systems because individual systems no longer need to store unique configuration information. With DHCP, the administrator can set up a master system and deploy new systems with a copy of the master's hard disk. In educational establishments and other open access facilities, the hard disk image may be stored on a shared drive, with each workstation automatically restoring itself to pristine condition at the end of each day.

MORE INFORMATION

Web www.dhcp.org

FAQ www.dhcp-handbook.com/dhcp_faq.html

HOWTO *DHCP Mini HOWTO*

HOW DHCP WORKS

The client daemon, **dhclient** (part of the **dhcp** package), contacts the server daemon, **dhcpd**, to obtain the IP address, netmask, broadcast address, nameserver address, and other networking parameters. The server provides a *lease* on the IP address to the client. The client can request the specific terms of the lease, including its duration; the server can, in turn, limit these terms. While connected to the network, a client typically requests extensions of its lease as necessary so its IP address remains the same. The lease can expire once the client is disconnected from the network, with the server giving the client a new IP address when it requests a new lease. You can also set up a DHCP server to provide static IP addresses for specific clients (refer to "Static Versus Dynamic IP Addresses" on page 368).

DHCP is broadcast based, so both client and server must be on the same subnet (page 371).

DHCP CLIENT

A DHCP client requests network configuration parameters from the DHCP server and uses those parameters to configure its network interface.

PREREQUISITES

Install the following package:

- **dhclient**

dhclient: THE DHCP CLIENT

When a DHCP client system connects to the network, dhclient requests a lease from the DHCP server and configures the client's network interface(s). Once a DHCP client has requested and established a lease, it stores information about the lease in a file named **dhclient.leases**, which is stored in the **/var/lib/dhclient** directory. This information is used to reestablish a lease when either the server or the client needs to reboot. The DHCP client configuration file, **/etc/dhclient.conf**, is required only for custom configurations. The following **dhclient.conf** file specifies a single interface, **eth0**:

```
$ cat /etc/dhclient.conf
interface "eth0"
{
send dhcp-client-identifier 1:xx:xx:xx:xx:xx:xx;
send dhcp-lease-time 86400;
}
```

In the preceding file, the 1 in the **dhcp-client-identifier** specifies an Ethernet network and ***xx:xx:xx:xx:xx:xx*** is the *MAC address* (page 1092) of the device controlling that interface. See page 454 for instructions on how to display a MAC address. The **dhcp-lease-time** is the duration, in seconds, of the lease on the IP address. While the client is connected to the network, dhclient automatically renews the lease each time half of the lease is up. A lease time of 86,400 seconds (or one day) is a reasonable choice for a workstation.

DHCP SERVER

The DHCP server maintains a list of IP addresses and other configuration parameters. When requested to do so, the DHCP server provides configuration parameters to a client.

PREREQUISITES

Install the following package:

- **dhcp**

Run chkconfig to cause **dhcpd** to start when the system enters multiuser mode:

```
# /sbin/chkconfig dhcpd on
```

Start **dhcpd**:

```
# /sbin/service dhcpd start
```

dhcpd: THE DHCP DAEMON

A simple DHCP server allows you to add clients to a network without maintaining a list of assigned IP addresses. A simple network, such as a home LAN sharing an Internet connection, can use DHCP to assign a dynamic IP address to almost all nodes. The exceptions are servers and routers, which must be at known network locations to be able to receive connections. If servers and routers are configured without DHCP, you can specify a simple DHCP server configuration in **/etc/dhcp/dhcpd.conf** (*FEDORA*) or **/etc/dhcpd.conf** (*RHEL*):

```
$ cat /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 86400;

option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.1;
option domain-name-servers 192.168.1.1;

subnet 192.168.1.0 netmask 255.255.255.0 {
   range 192.168.1.2 192.168.1.200;
}
```

The preceding configuration file specifies a LAN where the router and DNS are both located on **192.168.1.1**. The **default-lease-time** specifies the number of seconds the dynamic IP lease will remain valid if the client does not specify a duration. The **max-lease-time** is the maximum time allowed for a lease.

The information in the **option** lines is sent to each client when it connects. The names following the word **option** specify what the following argument represents. For example, the **option broadcast-address** line specifies the broadcast address of the network. The **routers** and **domain-name-servers** options can be followed by multiple values separated by commas.

The **subnet** section includes a **range** line that specifies the range of IP addresses that the DHCP server can assign. If you define multiple subnets, you can define options, such as **subnet-mask**, inside the **subnet** section. Options defined outside all **subnet** sections are global and apply to all subnets.

The preceding configuration file assigns addresses in the range between 192.168.1.2 and 192.168.1.200. The DHCP server starts at the bottom (*FEDORA*) or top (*RHEL*) of this range and attempts to assign a new IP address to each new client. Once the DHCP server reaches the top/bottom of the range, it starts reassigning IP addresses that have been used in the past, but are not currently in use. If you have fewer systems than IP addresses, the IP address of each system should remain fairly constant. You cannot use the same IP address for more than one system at a time.

Once you have configured a DHCP server, you can start (or restart) it by using the **dhcpd** init script:

```
# /sbin/service dhcpd restart
```

Once the server is running, clients configured to obtain an IP address from the server using DHCP should be able to do so.

STATIC IP ADDRESSES

As mentioned earlier, routers and servers typically require static IP addresses. While you can manually configure IP addresses for these systems, it may be more convenient to have the DHCP server provide them with static IP addresses.

When a system that requires a specific static IP address connects to the network and contacts the DHCP server, the server needs a way to identify the system so the server can assign the proper IP address to the system. The DHCP server uses the *MAC address* (page 1092) of the system's Ethernet card (NIC) as an identifier. When you set up the server, you must know the MAC address of each system that requires a static IP address.

Displaying a MAC address

You can use ifconfig to display the MAC addresses of the Ethernet cards (NICs) in a system. In the following example, the MAC addresses are the colon-separated series of hexadecimal number pairs following **HWaddr**:

```
$ /sbin/ifconfig | grep -i hwaddr
eth0      Link encap:Ethernet  HWaddr BA:DF:00:DF:C0:FF
eth1      Link encap:Ethernet  HWaddr 00:02:B3:41:35:98
```

Run ifconfig on each system that requires a static IP address. Once you have determined the MAC address of each of these systems, you can add a **host** section to the **/etc/dhcp/dhcpd.conf** file for each system, instructing the DHCP server to assign a specific address to the system. The following **host** section assigns the address **192.168.1.1** to the system with the MAC address of **BA:DF:00:DF:C0:FF**:

```
$ cat /etc/dhcp/dhcpd.conf
...
host router {
   hardware ethernet BA:DF:00:DF:C0:FF;
   fixed-address 192.168.1.1;
   option host-name router;
}
```

The name following **host** is used internally by **dhcpd**. The name specified after **option host-name** is passed to the client and can be a hostname or an FQDN.

After making changes to **dhcpd.conf**, restart **dhcpd** using the service utility and the **dhcpd** init script (page 453).

nsswitch.conf: WHICH SERVICE TO LOOK AT FIRST

With the advent of NIS and DNS, finding user and system information was no longer a simple matter of searching a local file. Where once you looked in **/etc/passwd** to get user information and in **/etc/hosts** to find system address information, you can now use several methods to find this type of information. The **/etc/nsswitch.conf** (name service switch configuration) file specifies the methods to use and the order in which to use them when looking for a certain type of information. You can also specify what action the system takes based on whether a method works or fails.

Format Each line in **nsswitch.conf** specifies how to search for a piece of information, such as a user's password. A line in **nsswitch.conf** has the following format:

info: ***method [[action]] [method [[action]]...]***

where ***info*** is the type of information that the line describes, ***method*** is the method used to find the information, and ***action*** is the response to the return status of the preceding ***method***. The action is enclosed within square brackets.

How nsswitch.conf Works

When called upon to supply information that **nsswitch.conf** describes, the system examines the line with the appropriate ***info*** field. It uses the methods specified on the line starting with the method on the left. By default, when it finds the desired information, the system stops searching. Without an ***action*** specification, when a method fails to return a result, the system tries the next method. It is possible for the search to end without finding the requested information.

INFORMATION

The **nsswitch.conf** file commonly controls searches for users (in **passwd**), passwords (in **shadow**), host IP addresses, and group information. The following list describes most of the types of information (***info*** in the format discussed earlier) that **nsswitch.conf** controls searches for.

automount	Automount (**/etc/auto.master** and **/etc/auto.misc**; page 744)
bootparams	Diskless and other booting options (See the **bootparam** man page.)
ethers	MAC address (page 1092)
group	Groups of users (**/etc/group**; page 472)
hosts	System information (**/etc/hosts**; page 472)
netgroup	Netgroup information (**/etc/netgroup**; page 474)
networks	Network information (**/etc/networks**)
passwd	User information (**/etc/passwd**; page 475)
protocols	Protocol information (**/etc/protocols**; page 476)
publickey	Used for NFS running in secure mode
rpc	RPC names and numbers (**/etc/rpc**; page 477)
services	Services information (**/etc/services**; page 477)
shadow	Shadow password information (**/etc/shadow**; page 477)

METHODS

Following is a list of the types of information that **nsswitch.conf** controls searches for (***method*** in the syntax shown on page 455). For each type of information, you can specify one or more of the following methods:[1]

files	Searches local files such as **/etc/passwd** and **/etc/hosts**
nis	Searches the NIS database; **yp** is an alias for **nis**
dns	Queries the DNS (**hosts** queries only)
compat	**±** syntax in **passwd, group**, and **shadow** files (page 458)

SEARCH ORDER

The information provided by two or more methods may overlap: For example, **files** and **nis** may each provide password information for the same user. With overlapping information, you need to consider which method you want to be authoritative (take precedence) and then put that method at the left of the list of methods.

The default **nsswitch.conf** file lists methods without actions, assuming no overlap (which is normal). In this case, the order is not critical: When one method fails, the system goes to the next one; all that is lost is a little time. Order becomes critical when you use actions between methods or when overlapping entries differ.

1. There are other, less commonly used methods. See the default **/etc/nsswitch.conf** file and the **nsswitch.conf** man page for more information. Although NIS+ belongs in this list, it is not implemented for Linux and is not discussed in this book.

The first of the following lines from **nsswitch.conf** causes the system to search for password information in **/etc/passwd** and, if that fails, to use NIS to find the information. If the user you are looking for is listed in both places, the information in the local file would be used—it would be authoritative. The second line uses NIS; if that fails, it searches **/etc/hosts;** if that fails, it checks with DNS to find host information.

```
passwd          files nis
hosts           nis files dns
```

ACTION ITEMS

Each method can optionally be followed by an action item that specifies what to do if the method succeeds or fails for any of a number of reasons. An action item has the following format:

[[!]STATUS=action]

where the opening and closing square brackets are part of the format and do not indicate that the contents are optional; ***STATUS*** (by convention uppercase although it is not case sensitive) is the status being tested for; and ***action*** is the action to be taken if ***STATUS*** matches the status returned by the preceding method. The leading exclamation point (**!**) is optional and negates the status.

STATUS ***STATUS*** may have the following values:

NOTFOUND—The method worked but the value being searched for was not found. Default action is **continue**.

SUCCESS—The method worked and the value being searched for was found; no error was returned. Default action is **return**.

UNAVAIL—The method failed because it is permanently unavailable. For example, the required file may not be accessible or the required server may be down. Default action is **continue**.

TRYAGAIN—The method failed because it was temporarily unavailable. For example, a file may be locked or a server overloaded. Default action is **continue**.

action Values for ***action*** are as follows:

return—Returns to the calling routine with or without a value.

continue—Continues with the next method. Any returned value is overwritten by a value found by the next method.

Example The following line from **nsswitch.conf** causes the system first to use DNS to search for the IP address of a given host. The action item following the DNS method tests whether the status returned by the method is not (**!**) UNAVAIL.

```
hosts           dns [!UNAVAIL=return] files
```

The system takes the action associated with the ***STATUS*** (**return**) if the DNS method does not return UNAVAIL (**!**UNAVAIL)—that is, if DNS returns SUCCESS, NOTFOUND, or TRYAGAIN. As a consequence, the following method (**files**) is

used only when the DNS server is unavailable: If the DNS server is *not un*available (read the two negatives as "is available"), the search returns the domain name or reports that the domain name was not found. The search uses the **files** method (check the local **/etc/hosts** file) only if the server is not available.

COMPAT METHOD: ± IN passwd, group, AND shadow FILES

You can put special codes in the **/etc/passwd**, **/etc/group**, and **/etc/shadow** files that cause the system, when you specify the **compat** method in **nsswitch.conf**, to combine and modify entries in the local files and the NIS maps. That is, a plus sign (**+**) at the beginning of a line in one of these files adds NIS information; a minus sign (**–**) removes information.

For example, to use these codes in the **passwd** file, specify **passwd: compat** in **nsswitch.conf**. The system then goes through the **passwd** file in order, adding or removing the appropriate NIS entries when it reaches each line that starts with a **+** or **–**.

Although you can put a plus sign at the end of the **passwd** file, specify **passwd: compat** in **nsswitch.conf** to search the local **passwd** file, and then go through the NIS map, it is more efficient to put **passwd: file nis** in **nsswitch.conf** and not modify the **passwd** file.

PAM

PAM (actually Linux-PAM, or Linux Pluggable Authentication Modules) allows a system administrator to determine how applications use *authentication* (page 1070) to verify the identity of a user. PAM provides shared libraries of modules (located in **/lib/security**) that, when called by an application, authenticate a user. The term "Pluggable" in PAM's name refers to the ease with which you can add and remove modules from the authentication stack. The configuration files kept in the **/etc/pam.d** directory determine the method of authentication and contain a list (i.e., stack) of calls to the modules. PAM may also use other files, such as **/etc/passwd**, when necessary.

Instead of building the authentication code into each application, PAM provides shared libraries that keep the authentication code separate from the application code. The techniques of authenticating users stay the same from application to application. In this way, PAM enables a system administrator to change the authentication mechanism for a given application without ever touching the application.

PAM provides authentication for a variety of system-entry services (login, ftp, and so on). You can take advantage of PAM's ability to stack authentication modules to integrate system-entry services with different authentication mechanisms, such as RSA, DCE, Kerberos, and smartcards.

20

sendmail: Setting Up Mail Clients, Servers, and More

In This Chapter

Sending and receiving email require three pieces of software. At each end, there is a client, called an MUA (Mail User Agent), which is a bridge between a user and the mail system. Common MUAs are mutt, KMail, Thunderbird, and Outlook. When you send an email, the MUA hands it to an MTA (a Mail Transfer Agent such as **sendmail**), which transfers it to the destination server. At the destination, an MDA (a Mail Delivery Agent such as **procmail**) puts the mail in the recipient's mailbox file. On Linux systems, the MUA on the receiving system either reads the mailbox file or retrieves mail from a remote MUA or MTA, such as an ISP's SMTP (mail) server, using POP (Post Office Protocol) or IMAP (Internet Message Access Protocol).

Most Linux MUAs expect a local copy of **sendmail** to deliver outgoing email. On some systems, including those with a dial-up connection to the Internet, **sendmail** relays email to an ISP's mail server. Because **sendmail** uses SMTP (Simple Mail Transfer Protocol) to deliver email, **sendmail** is often referred to as an SMTP server.

In the default Fedora/RHEL setup, the **sendmail** MTA uses **procmail** as the local MDA. In turn, **procmail** writes email to the end of the recipient's mailbox file. You can also use **procmail** to sort email according to a set of rules, either on a per-user basis or globally. The global filtering function is useful for systemwide filtering to detect spam and for other tasks, but the per-user feature is largely superfluous on a modern system. Traditional UNIX MUAs were simple programs that could not filter mail and thus delegated this function to MDAs such as **procmail**. Modern MUAs, by contrast, incorporate this functionality.

You do not need to set up sendmail to send and receive email

tip Most MUAs can use POP or IMAP for receiving email. These protocols do not require an MTA such as **sendmail**. As a consequence, you do not need to install or configure **sendmail** (or another MTA) to receive email. You still need SMTP to send email. However, the SMTP server can be at a remote location, such as your ISP, so you do not need to concern yourself with it.

INTRODUCTION

When the network that was to evolve into the Internet was first set up, it connected a few computers, each serving a large number of users and running several services. Each computer was capable of sending and receiving email and had a unique hostname, which was used as a destination for email.

Today the Internet has a large number of transient clients. Because these clients do not have fixed IP addresses or hostnames, they cannot receive email directly. Users on these systems usually maintain an account on an email server run by their employer or an ISP, and they collect email from this account using POP or IMAP. Unless you own a domain that you want to receive email at, you will not need to set up **sendmail** as an incoming SMTP server.

You can set up **sendmail** on a client system so that it simply relays outbound mail to an SMTP server. This configuration is required by organizations that use firewalls to prevent email from being sent out on the Internet from any system other than the company's official mail servers. As a partial defense against spreading viruses, some ISPs block outbound port 25 to prevent their customers from sending email directly to a remote computer. This configuration is required by these ISPs.

You can also set up **sendmail** as an outbound server that does not use an ISP as a relay. In this configuration, **sendmail** connects directly to the SMTP servers for the domains receiving the email. An ISP set up as a relay is configured this way.

You can set up **sendmail** to accept email for a registered domain name as specified in the domain's DNS MX record (page 780). However, most mail clients (MUAs) do not interact directly with **sendmail** to receive email. Instead, they use POP or IMAP—protocols that include features for managing mail folders, leaving messages on the server, and reading only the subject of an email without downloading the entire message. If you want to collect your email from a system other than the one running the incoming mail server, you may need to set up a POP or IMAP server, as discussed on page 689.

PREREQUISITES

Install the following packages:

- **sendmail** (required)
- **sendmail-cf** (required to configure **sendmail**)
- **squirrelmail** (optional; provides Webmail, page 686)
- **spamassassin** (optional; provides spam filtering, page 682)
- **mailman** (optional; provides mailing list support, page 688)
- **dovecot** (optional; provides IMAP and POP incoming mail server daemons)

Run chkconfig to cause **sendmail** to start when the system goes multiuser (by default, **sendmail** does not run in single-user mode):

```
# /sbin/chkconfig sendmail on
```

Start **sendmail**. Because **sendmail** is normally running, you need to restart it to cause **sendmail** to reread its configuration files. The following restart command works even when **sendmail** is not running—it just fails to shut down **sendmail**:

```
# /sbin/service sendmail restart
Shutting down sm-client:                                   [  OK  ]
Shutting down sendmail:                                    [  OK  ]
Starting sendmail:                                         [  OK  ]
Starting sm-client:                                        [  OK  ]
```

Run chkconfig to cause the SpamAssassin daemon, **spamd**, to start when the system enters multiuser mode (SpamAssassin is normally installed in this configuration):

```
# /sbin/chkconfig spamassassin on
```

As with **sendmail**, SpamAssassin is normally running. Restart it to cause **spamd** to reread its configuration files:

```
# /sbin/service spamassassin restart
Stopping spamd:                                            [  OK  ]
Starting spamd:                                            [  OK  ]
```

The IMAP and POP protocols are implemented as several daemons. See page 689 for information on these daemons and how to start them.

NOTES

Firewall An SMTP server normally uses TCP port 25. If the SMTP server system is running a firewall, you need to open this port. Using the Firewall Configuration window Trusted Services tab (page 824), put a check in the box labeled **Mail (SMTP)** to open this port. For more general information see Chapter 25, which details iptables.

cyrus This chapter covers the IMAP and POP3 servers included in the **dovecot** package. Fedora/RHEL also provides IMAP and POP3 servers in the **cyrus-imapd** package.

MORE INFORMATION

Web
sendmail: www.sendmail.org
IMAP: www.imap.org
IMAP and POP3: www.dovecot.org
IMAP and POP3: cyrusimap.web.cmu.edu
SquirrelMail: www.squirrelmail.org
Postfix: www.postfix.org/docs.html (alternative MTA, page 691)
Qmail: qmail.area.com
Mailman: www.list.org
procmail: www.procmail.org
SpamAssassin: spamassassin.org
Spam database: razor.sourceforge.net

JUMPSTART I: CONFIGURING sendmail ON A CLIENT

You may not need to configure sendmail to send email

tip With **sendmail** running, give the command described under "Test" on page 673. As long as **sendmail** can connect to port 25 outbound, you should not need to set up **sendmail** to use an SMTP relay as described in this section. If you receive the mail sent by the test, you can skip this section.

This JumpStart configures an outbound **sendmail** server. This server

- Uses a remote SMTP server—typically an ISP—to relay outbound email to its destination (an SMTP relay).
- Sends to the SMTP server email originating from the local system only. It does not forward email originating from other systems.
- Does not handle inbound email. As is frequently the case, you need to use POP or IMAP to receive email.

To set up this server, you must edit **/etc/mail/sendmail.mc** and restart **sendmail**.

Change **sendmail.mc**
The **dnl** at the start of the following line in **sendmail.mc** indicates that this line is a comment:

```
dnl define(‘SMART_HOST',‘smtp.your.provider')dnl
```

You can ignore the **dnl** at the end of the line. To specify a remote SMTP server, you must open **sendmail.mc** in an editor and change the preceding line, deleting **dnl** from the beginning of the line and replacing **smtp.your.provider** with the FQDN of your ISP's SMTP server (obtain this name from your ISP). Be careful not to alter the back ticks (‘) and the single quotation marks (') in this line. If your ISP's SMTP server is at **smtp.myisp.com**, you would change the line to

```
define(‘SMART_HOST',‘smtp.myisp.com')
```

Do not alter the back ticks (‘) or the single quotation marks (')

tip Be careful not to alter the back ticks (‘) or the single quotation marks (') in any line in **sendmail.mc**. These symbols control the way the m4 preprocessor converts **sendmail.mc** to **sendmail.cf**; **sendmail** will not work properly if you do not preserve these symbols.

Restart sendmail When you restart it, **sendmail** regenerates the **sendmail.cf** file from the **sendmail.mc** file you edited:

```
# /sbin/service sendmail restart
```

Test Test **sendmail** with the following command:

```
$ echo "my sendmail test" | /usr/sbin/sendmail user@remote.host
```

Replace *user@remote.host* with an email address on *another system* where you receive email. You need to send email to a remote system to make sure that **sendmail** is relaying your email.

JUMPSTART II: CONFIGURING sendmail ON A SERVER

If you want to receive inbound email sent to a registered domain that you own, you need to set up **sendmail** as an incoming mail server. This JumpStart describes how to set up such a server. This server

- Accepts outbound email from the local system only.
- Delivers outbound email directly to the recipient's system, without using a relay.
- Accepts inbound email from any system.

This server does not relay outbound email originating on other systems. Refer to "access: Sets Up a Relay Host" on page 680 if you want the local system to act as a relay. For this configuration to work, you must be able to make outbound connections from and receive inbound connections to port 25.

The line in **sendmail.mc** that limits **sendmail** to accepting inbound email from the local system only is

```
DAEMON_OPTIONS(‘Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

To allow **sendmail** to accept inbound email from other systems, remove the parameter **Addr=127.0.0.1,** from the preceding line:

```
DAEMON_OPTIONS(‘Port=smtp, Name=MTA')dnl
```

By default, **sendmail** does not use a remote SMTP server to relay email, so there is nothing to change to cause **sendmail** to send email directly to recipients' systems. (JumpStart I set up a SMART_HOST to relay email.)

Once you have restarted **sendmail**, it will accept mail addressed to the local system, as long as a DNS MX record (page 780) points at the local system. If you are not running a DNS server, you must ask your ISP to set up an MX record.

How sendmail Works

Outbound email

When you send email, the MUA passes the email to **sendmail**, which creates in the **/var/spool/mqueue** (mail queue) directory two files that hold the message while **sendmail** processes it. To create a unique filename for a particular piece of email, **sendmail** generates a random string and uses that string in filenames pertaining to the email. The **sendmail** daemon stores the body of the message in a file named **df** (data file) followed by the generated string. It stores the headers and other information in a file named **qf** (queue file) followed by the generated string.

If a delivery error occurs, **sendmail** creates a temporary copy of the message that it stores in a file whose name starts with **tf** (temporary file) and logs errors in a file whose name starts with **xf**. Once an email has been sent successfully, **sendmail** removes all files pertaining to that email from **/var/spool/mqueue**.

Incoming email

By default, the MDA stores incoming messages in users' files in the mail spool directory, **/var/spool/mail**, in **mbox** format. Within this directory, each user has a mail file named with the user's username. Mail remains in these files until it is collected, typically by an MUA. Once an MUA collects the mail from the mail spool, the MUA stores the mail as directed by the user, usually in the user's home directory hierarchy.

mbox versus **maildir**

The **mbox** format stores all messages for a user in a single file. To prevent corruption, the file must be locked while a process is adding messages to or deleting messages from the file; you cannot delete a message at the same time the MTA is adding messages. A competing format, **maildir**, stores each message in a separate file. This format does not use locks, allowing an MUA to read and delete messages at the same time as new mail is delivered. In addition, the **maildir** format is better able to handle larger mailboxes. The downside is that the **maildir** format adds overhead when you are using a protocol such as IMAP to check messages. The **dovecot** package supports both **mbox** and **maildir** formats. Qmail (page 691), a **sendmail** alternative, uses **maildir**-format mailboxes.

Mail Logs

The **sendmail** daemon stores log messages in **/var/log/maillog**. Other mail servers, such as the dovecot **imap-login** and **pop3-login** daemons, may also log information to this file. Following is a sample log entry:

/var/log/maillog

```
# cat /var/log/maillog
...
Mar  3 16:25:33 MACHINENAME sendmail[7225]: i23GPXvm007224:
to=<user@localhost.localdomain>, ctladdr=<root@localhost.localdomain>
(0/0), delay=00:00:00, xdelay=00:00:00, mailer=local, pri=30514,
dsn=2.0.0, stat=Sent
```

Each log entry starts with a timestamp, the name of the system sending the email, the name of the mail server (**sendmail**), and a unique identification number. The address of the recipient follows the **to=** label and the address of the sender follows **ctladdr=**. Additional fields provide the name of the mailer and the time it took to send the message. If a message is sent correctly, the **stat=** label is followed by **Sent**.

A message is marked **Sent** when **sendmail** sends it; **Sent** does not indicate that the message has been delivered. If a message is not delivered because an error occurred farther down the line, the sender usually receives an email saying that it was not delivered and giving a reason why.

If you send and receive a lot of email, the **maillog** file can grow quite large. The **rsyslog** logrotate (page 579) entry is set up to archive and rotate the **maillog** files regularly.

ALIASES AND FORWARDING

Three files can forward email: **.forward** (page 676), **aliases** (discussed next), and **virtusertable** (page 682). Table 20-1 on page 682 compares the three files.

/etc/aliases Most of the time when you send email, it goes to a specific person; the recipient, **user@system**, maps to a specific, real user on the specified system. Sometimes you may want email to go to a class of users and not to a specific recipient. Examples of classes of users include **postmaster**, **webmaster**, **root**, and **tech_support**. Different users may receive this email at different times or the email may be answered by a group of users. You can use the **/etc/aliases** file to map inbound addresses to local users, files, commands, and remote addresses.

Each line in **/etc/aliases** contains the name of a local pseudouser, followed by a colon, whitespace, and a comma-separated list of destinations. The default installation includes a number of aliases that redirect messages for certain pseudousers to **root**. These have the form

```
system:         root
```

Sending messages to the **root** account is a good way of making them easy to review. However, because **root**'s email is rarely checked, you may want to send copies to a real user. The following line forwards mail sent to **abuse** on the local system to **root** and **alex**:

```
abuse:          root, alex
```

You can create simple mailing lists with this type of alias. For example, the following alias sends copies of all email sent to **admin** on the local system to several users, including Zach, who is on a different system:

```
admin:          sam, helen, mark, zach@tcorp.com
```

You can direct email to a file by specifying an absolute pathname in place of a destination address. The following alias, which is quite popular among less conscientious system administrators, redirects email sent to **complaints** to **/dev/null** (page 469), where they disappear:

```
complaints:     /dev/null
```

You can also send email to standard input of a command by preceding the command with a pipe character (|). This technique is commonly used with mailing list software such as Mailman (page 688). For each list it maintains, Mailman has entries, such as the following entry for **mylist**, in the **aliases** file:

```
mylist:             "|/usr/lib/mailman/mail/mailman post mylist"
```

newaliases After you edit **/etc/aliases**, you must either run newaliases as **root** or restart **sendmail** to re-create the **aliases.db** file that **sendmail** reads.

praliases You can use praliases to list aliases currently loaded by **sendmail**:

```
# /usr/sbin/praliases | tail -5
vcsa:root
webalizer:root
wnn:root
www:webmaster
xfs:root
```

~/.forward Systemwide aliases are useful in many cases, but non**root** users cannot make or change them. Sometimes you may want to forward your own mail: Maybe you want mail from several systems to go to one address or perhaps you just want to forward your mail while you are working at another office for a week. The **~/.forward** file allows ordinary users to forward their email.

Lines in a **.forward** file are the same as the right column of the **aliases** file explained previously: Destinations are listed one per line and can be a local user, a remote email address, a filename, or a command preceded by a pipe character (|).

Mail that you forward does not go to your local mailbox. If you want to forward mail and keep a copy in your local mailbox, you must specify your local username preceded by a backslash to prevent an infinite loop. The following example sends Sam's email to himself on the local system and on the system at **tcorp.com**:

```
$cat ~sam/.forward
sams@tcorp.com
\sam
```

RELATED PROGRAMS

sendmail The **sendmail** package includes several programs. The primary program, **sendmail**, reads from standard input and sends an email to the recipient specified by its argument. You can use **sendmail** from the command line to check that the mail delivery system is working and to email the output of scripts. See page 673 for an example.

mailq or **sendmail –bp** The mailq utility (*RHEL*) displays the status of the outgoing mail queue and normally reports there are no messages in the queue. From *FEDORA*, **sendmail -bp** performs the same function. Messages in the queue usually indicate a problem with the local or remote **sendmail** configuration or a network problem.

```
# /usr/bin/mailq
/var/spool/mqueue is empty
                    Total requests: 0
```

mailstats The mailstats utility reports on the number and sizes of messages **sendmail** has sent and received since the date it displays on the first line:

```
# /usr/sbin/mailstats
Statistics from Fri Sep 11 12:01:04 2009
 M   msgsfr  bytes_from   msgsto    bytes_to  msgsrej msgsdis msgsqur  Mailer
 4        0          0K        1          1K        0       0       0  esmtp
 9        5          5K        2          2K        0       0       0  local
=============================================================
 T        5          5K        3          3K        0       0       0
 C        5                    3                    0
```

In the preceding output, each mailer is identified by the first column, which displays the mailer number, and by the last column, which displays the name of the mailer. The second through fifth columns display the number and total sizes of messages sent and received by the mailer. The sixth, seventh, and eighth columns display the number of messages rejected, discarded, and quarantined respectively. The row that starts with **T** lists the column totals, and the row that starts with **C** lists the number of TCP connections.

CONFIGURING sendmail

The **sendmail** configuration files reside in **/etc/mail**, where the primary configuration file is **sendmail.cf**. This directory contains other text configuration files, such as **access**, **mailertable**, and **virtusertable**. The **sendmail** daemon does not read these files but instead reads the corresponding ***.db** files in the same directory.

makemap You can use makemap or give the command **make** from the **/etc/mail** directory to generate the ***.db** files, although this step is not usually necessary. The **sendmail** init script automatically generates these files when you start or restart **sendmail**:

```
# /sbin/service sendmail restart
```

THE sendmail.mc AND sendmail.cf FILES

This **sendmail.cf** file is not intended to be edited by hand and contains a large warning to this effect:

```
$ cat /etc/mail/sendmail.cf
...
######################################################################
#####
#####   DO NOT EDIT THIS FILE!  Only edit the source .mc file.
#####
######################################################################
...
```

EDITING sendmail.mc AND GENERATING sendmail.cf

The **sendmail.cf** file is generated from **sendmail.mc** using the m4 macro processor. It can be helpful to use a text editor that supports syntax highlighting, such as vim, to edit **sendmail.mc**.

dnl Many of the lines in **sendmail.mc** start with **dnl**, which stands for **delete to new line**; this token causes m4 to delete from the **dnl** to the end of the line (the next NEWLINE character). Because m4 ignores anything on a line after a **dnl** instruction, you can use **dnl** to introduce comments; it works the same way as **#** does in a shell script.

Many of the lines in **sendmail.mc** *end* with **dnl**. Because NEWLINEs immediately follow these **dnl**s, these **dnl**s are superfluous; you can remove them if you like.

After you edit **sendmail.mc**, you need to regenerate **sendmail.cf** to make your changes take effect. When you restart **sendmail**, the **sendmail** init script regenerates **sendmail.cf**.

ABOUT sendmail.mc

Lines near the beginning of **sendmail.mc** provide basic configuration information:

```
divert(-1)dnl
include(`/usr/share/sendmail-cf/m4/cf.m4')dnl
VERSIONID(`setup for linux')dnl
OSTYPE(`linux')dnl
```

The line that starts with **divert** tells m4 to discard extraneous output it may generate when processing this file.

The **include** statement tells m4 where to find the macro definition file that it will use to process the rest of this file; it points to the file named **cf.m4**. The **cf.m4** file contains other **include** statements that include parts of the **sendmail** configuration rule sets.

The **VERSIONID** statement defines a string that indicates the version of this configuration. You can change this string to include a brief comment about changes you have made to this file or other information. The value of this string is not significant to **sendmail**.

Do not change the **OSTYPE** statement unless you are migrating a **sendmail.mc** file from another operating system.

Other statements you may want to change are explained in the following sections and in the **sendmail** documentation.

Quoting m4 strings

tip The m4 macro processor, which converts **sendmail.mc** to **sendmail.cf**, requires strings to be preceded by a back tick (`) and closed with a single quotation mark (').

MASQUERADING

Typically you want your email to appear to come from the user and the domain where you receive email; sometimes the outbound server is in a different domain than the inbound server. You can cause **sendmail** to alter outbound messages so that they appear to come from a user and/or domain other than the one they are sent from: In other words, you *masquerade* (page 1093) the message.

Several lines in **sendmail.mc** pertain to this type of masquerading. Each is commented out in the file that Fedora/RHEL distributes:

```
dnl MASQUERADE_AS(`mydomain.com')dnl
dnl MASQUERADE_DOMAIN(localhost)dnl
dnl FEATURE(masquerade_entire_domain)dnl
```

The MASQUERADE_AS statement causes email that you send from the local system to appear to come from the specified domain (**mydomain.com** in the commented-out line in the distributed file). Remove the leading **dnl** and change **mydomain.com** to the domain name that you want mail to appear to come from.

The MASQUERADE_DOMAIN statement causes email from the specified system or domain to be masqueraded, just as local email is. That is, email from the system specified in this statement is treated as though it came from the local system: It is changed so that it appears to come from the domain specified in the MASQUERADE_AS statement. Remove the leading **dnl** and change **localhost** to the name of the system or domain that sends the email that you want to masquerade. If the name you specify has a leading period, it specifies a domain. If there is no leading period, the name specifies a system or host. The **sendmail.mc** file can include as many MASQUERADE_DOMAIN statements as necessary.

The **masquerade_entire_domain** feature statement causes **sendmail** also to masquerade subdomains of the domain specified in the MASQUERADE_DOMAIN statement. Remove the leading **dnl** to masquerade entire domains.

Accepting Email from Unknown Hosts

As configured by Fedora/RHEL, **sendmail** accepts email from domains that it cannot resolve (and that may not exist). To turn this feature off and cut down the amount of spam you receive, add **dnl** to the beginning of the following line:

```
FEATURE(`accept_unresolvable_domains')dnl
```

When this feature is off, **sendmail** uses DNS to look up the domains of all email it receives. If it cannot resolve the domain, it rejects the email.

Setting Up a Backup Server

You can set up a backup mail server to hold email when the primary mail server experiences problems. For maximum coverage, the backup server should be on a different connection to the Internet from the primary server.

Setting up a backup server is easy. Just remove the leading **dnl** from the following line in the *backup* mail server's **sendmail.mc** file:

```
dnl FEATURE(`relay_based_on_MX')dnl
```

DNS MX records (page 780) specify where email for a domain should be sent. You can have multiple MX records for a domain, each pointing to a different mail server. When a domain has multiple MX records, each record usually has a different

priority; the priority is specified by a two-digit number, where lower numbers specify higher priorities.

When attempting to deliver email, an MTA first tries to deliver email to the highest-priority server. If that delivery attempt fails, it tries to deliver to a lower-priority server. If you activate the **relay_based_on_MX** feature and point a low-priority MX record at a secondary mail server, the mail server will accept email for the domain. The mail server will then forward email to the server identified by the highest-priority MX record for the domain when that server becomes available.

27

Programming the Bourne Again Shell

In This Chapter

Chapter 7 introduced the shells and Chapter 9 went into detail about the Bourne Again Shell. This chapter introduces additional Bourne Again Shell commands, builtins, and concepts that carry shell programming to a point where it can be useful. The first part of this chapter covers programming control structures, which are also known as control flow constructs. These structures allow you to write scripts that can loop over command-line arguments, make decisions based on the value of a variable, set up menus, and more. The Bourne Again Shell uses the same constructs found in such high-level programming languages as C.

The next part of this chapter discusses parameters and variables, going into detail about array variables, local versus global variables, special parameters, and positional parameters. The exploration of builtin commands covers type, which displays information about a command, and read, which allows you to accept user input in a shell script. The section on the exec builtin demonstrates how exec provides an efficient way to execute a command by replacing a process and explains how

you can use it to redirect input and output from within a script. The next section covers the trap builtin, which provides a way to detect and respond to operating system signals (such as that which is generated when you press CONTROL-C). The discussion of builtins concludes with a discussion of kill, which can abort a process, and getopts, which makes it easy to parse options for a shell script. (Table 27-6 on page 949 lists some of the more commonly used builtins.)

Next the chapter examines arithmetic and logical expressions and the operators that work with them. The final section walks through the design and implementation of two major shell scripts.

This chapter contains many examples of shell programs. Although they illustrate certain concepts, most use information from earlier examples as well. This overlap not only reinforces your overall knowledge of shell programming but also demonstrates how you can combine commands to solve complex tasks. Running, modifying, and experimenting with the examples in this book is a good way to become comfortable with the underlying concepts.

Do not name a shell script test

tip You can unwittingly create a problem if you give a shell script the name **test** because a Linux utility has the same name. Depending on how the **PATH** variable is set up and how you call the program, you may run your script or the utility, leading to confusing results.

This chapter illustrates concepts with simple examples, which are followed by more complex ones in sections marked "Optional." The more complex scripts illustrate traditional shell programming practices and introduce some Linux utilities often used in scripts. You can skip these sections without loss of continuity the first time you read the chapter. Return to them later when you feel comfortable with the basic concepts.

CONTROL STRUCTURES

The *control flow* commands alter the order of execution of commands within a shell script. Control structures include the **if...then**, **for...in**, **while**, **until**, and **case** statements. In addition, the **break** and **continue** statements work in conjunction with the control structures to alter the order of execution of commands within a script.

if...then

The **if...then** control structure has the following syntax:

```
if test-command
    then
        commands
fi
```

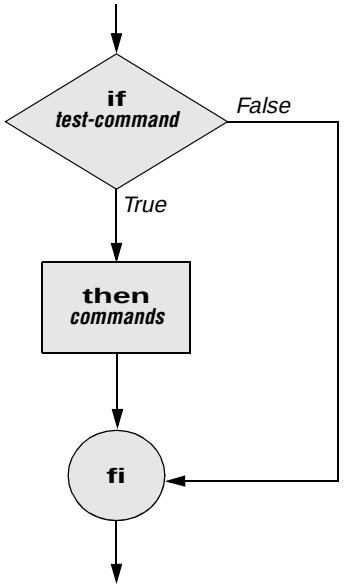

Figure 27-1 An **if...then** flowchart

The ***bold*** words in the syntax description are the items you supply to cause the structure to have the desired effect. The *nonbold* words are the keywords the shell uses to identify the control structure.

test builtin

Figure 27-1 shows that the **if** statement tests the status returned by the ***test-command*** and transfers control based on this status. The end of the **if** structure is marked by a **fi** statement (*if* spelled backward). The following script prompts for two words, reads them, and then uses an **if** structure to execute commands based on the result returned by the test builtin when it compares the two words. (See the test info page for information on the test utility, which is similar to the test builtin.) The test builtin returns a status of *true* if the two words are the same and *false* if they are not. Double quotation marks around **$word1** and **$word2** make sure that test works properly if you enter a string that contains a SPACE or other special character:

```
$ cat if1
echo -n "word 1: "
read word1
echo -n "word 2: "
read word2

if test "$word1" = "$word2"
    then
        echo "Match"
fi
echo "End of program."
```

```
$ if1
word 1: peach
word 2: peach
Match
End of program.
```

In the preceding example the *test-command* is **test "$word1" = "$word2"**. The test builtin returns a *true* status if its first and third arguments have the relationship specified by its second argument. If this command returns a *true* status (= 0), the shell executes the commands between the **then** and **fi** statements. If the command returns a *false* status (not = 0), the shell passes control to the statement following **fi** without executing the statements between **then** and **fi**. The effect of this **if** statement is to display **Match** if the two words are the same. The script always displays **End of program.**

Builtins In the Bourne Again Shell, test is a builtin—part of the shell. It is also a stand-alone utility kept in **/usr/bin/test**. This chapter discusses and demonstrates many Bourne Again Shell builtins. You usually use the builtin version if it is available and the utility if it is not. Each version of a command may vary slightly from one shell to the next and from the utility to any of the shell builtins. See page 936 for more information on shell builtins.

Checking arguments The next program uses an **if** structure at the beginning of a script to check that you have supplied at least one argument on the command line. The **–eq** test operator compares two integers, where the **$#** special parameter (page 931) takes on the value of the number of command-line arguments. This structure displays a message and exits from the script with an exit status of 1 if you do not supply at least one argument:

```
$ cat chkargs
if test $# -eq 0
    then
        echo "You must supply at least one argument."
        exit 1
fi
echo "Program running."
$ chkargs
You must supply at least one argument.
$ chkargs abc
Program running.
```

A test like the one shown in **chkargs** is a key component of any script that requires arguments. To prevent the user from receiving meaningless or confusing information from the script, the script needs to check whether the user has supplied the appropriate arguments. Sometimes the script simply tests whether arguments exist (as in **chkargs**). Other scripts test for a specific number or specific kinds of arguments.

You can use test to ask a question about the status of a file argument or the relationship between two file arguments. After verifying that at least one argument has been given on the command line, the following script tests whether the argument is the

name of an ordinary file (not a directory or other type of file) in the working directory. The test builtin with the **–f** option and the first command-line argument (**$1**) check the file:

```
$ cat is_ordfile
if test $# -eq 0
    then
        echo "You must supply at least one argument."
        exit 1
fi
if test -f "$1"
    then
        echo "$1 is an ordinary file in the working directory"
    else
        echo "$1 is NOT an ordinary file in the working directory"
fi
```

You can test many other characteristics of a file with test and various options. Table 27-1 lists some of these options.

Table 27-1 Options to the test builtin

Option	Tests file to see if it
–d	Exists and is a directory file
–e	Exists
–f	Exists and is an ordinary file (not a directory)
–r	Exists and is readable
–s	Exists and has a size greater than 0 bytes
–w	Exists and is writable
–x	Exists and is executable

Other test options provide ways to test relationships between two files, such as whether one file is newer than another. Refer to later examples in this chapter for more detailed information.

Always test the arguments

tip To keep the examples in this book short and focused on specific concepts, the code to verify arguments is often omitted or abbreviated. It is a good practice to test arguments in shell programs that other people will use. Doing so results in scripts that are easier to run and debug.

[] is a synonym for test

The following example—another version of **chkargs**—checks for arguments in a way that is more traditional for Linux shell scripts. The example uses the bracket ([]) synonym for test. Rather than using the word test in scripts, you can surround the arguments to test with brackets. The brackets must be surrounded by white-space (SPACEs or TABs).

```
$ cat chkargs2
if [ $# -eq 0 ]
    then
        echo "Usage: chkargs2 argument..." 1>&2
        exit 1
fi
echo "Program running."
exit 0
$ chkargs2
Usage: chkargs2 arguments
$ chkargs2 abc
Program running.
```

Usage message

The error message that **chkargs2** displays is called a *usage message* and uses the **1>&2** notation to redirect its output to standard error (page 284). After issuing the usage message, **chkargs2** exits with an exit status of 1, indicating that an error has occurred. The **exit 0** command at the end of the script causes **chkargs2** to exit with a 0 status after the program runs without an error. The Bourne Again Shell returns a 0 status if you omit the status code.

The usage message is commonly employed to specify the type and number of arguments the script takes. Many Linux utilities provide usage messages similar to the one in **chkargs2**. If you call a utility or other program with the wrong number or kind of arguments, you will often see a usage message. Following is the usage message that cp displays when you call it without any arguments:

```
$ cp
cp: missing file argument
Try 'cp --help' for more information.
```

if...then...else

The introduction of an **else** statement turns the **if** structure into the two-way branch shown in Figure 27-2. The **if...then...else** control structure has the following syntax:

```
if test-command
    then
        commands
    else
        commands
fi
```

Because a semicolon (**;**) ends a command just as a NEWLINE does, you can place **then** on the same line as **if** by preceding it with a semicolon. (Because **if** and **then** are separate builtins, they require a command separator between them; a semicolon and NEWLINE work equally well.) Some people prefer this notation for aesthetic reasons, while others like it because it saves space:

```
if test-command; then
        commands
    else
        commands
fi
```

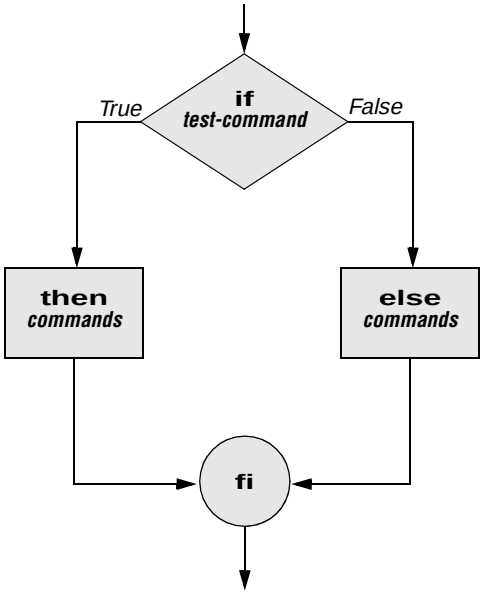

Figure 27-2 An **if...then...else** flowchart

If the ***test-command*** returns a *true* status, the **if** structure executes the commands between the **then** and **else** statements and then diverts control to the statement following **fi**. If the ***test-command*** returns a *false* status, the **if** structure executes the commands following the **else** statement.

When you run the next script, named **out**, with arguments that are filenames, it displays the files on the terminal. If the first argument is **–v** (called an option in this case), **out** uses less (page 150) to display the files one page at a time. After determining that it was called with at least one argument, **out** tests its first argument to see whether it is **–v**. If the result of the test is *true* (if the first argument is **–v**), **out** uses the shift builtin to shift the arguments to get rid of the **–v** and displays the files using less. If the result of the test is *false* (if the first argument is *not* **–v**), the script uses cat to display the files:

```
$ cat out
if [ $# -eq 0 ]
    then
        echo "Usage: out [-v] filenames..." 1>&2
        exit 1
fi
if [ "$1" = "-v" ]
    then
        shift
        less -- "$@"
    else
        cat -- "$@"
fi
```

optional In **out** the **––** argument to cat and less tells these utilities that no more options follow on the command line and not to consider leading hyphens (**–**) in the following list as indicating options. Thus **––** allows you to view a file with a name that starts with a hyphen. Although not common, filenames beginning with a hyphen do occasionally occur. (You can create such a file by using the command **cat > –fname**.) The **––** argument works with all Linux utilities that use the getopts builtin (page 946) to parse their options; it does not work with more and a few other utilities. This argument is particularly useful when used in conjunction with rm to remove a file whose name starts with a hyphen (**rm –– –fname**), including any that you create while experimenting with the **––** argument.

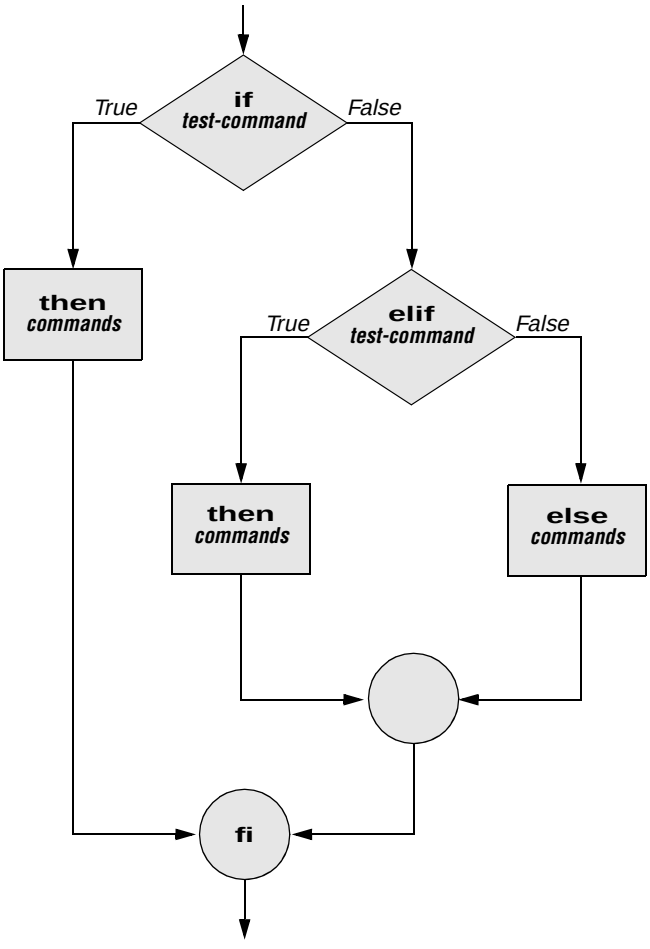

Figure 27-3 An **if...then...elif** flowchart

if...then...elif

The **if...then...elif** control structure (Figure 27-3) has the following syntax:

```
if test-command
    then
        commands
    elif test-command
        then
            commands
...
    else
        commands
fi
```

The **elif** statement combines the **else** statement and the **if** statement and allows you to construct a nested set of **if...then...else** structures (Figure 27-3). The difference between the **else** statement and the **elif** statement is that each **else** statement must be paired with a **fi** statement, whereas multiple nested **elif** statements require only a single closing **fi** statement.

The following example shows an **if...then...elif** control structure. This shell script compares three words that the user enters. The first **if** statement uses the Boolean operator AND (**–a**) as an argument to test. The test builtin returns a *true* status only if the first and second logical comparisons are *true* (that is, if **word1** matches **word2** and **word2** matches **word3**). If test returns a *true* status, the script executes the command following the next **then** statement, passes control to the statement following **fi**, and terminates:

```
$ cat if3
echo -n "word 1: "
read word1
echo -n "word 2: "
read word2
echo -n "word 3: "
read word3

if [ "$word1" = "$word2" -a "$word2" = "$word3" ]
    then
        echo "Match: words 1, 2, & 3"
    elif [ "$word1" = "$word2" ]
    then
        echo "Match: words 1 & 2"
    elif [ "$word1" = "$word3" ]
    then
        echo "Match: words 1 & 3"
    elif [ "$word2" = "$word3" ]
    then
        echo "Match: words 2 & 3"
    else
        echo "No match"
fi
```

```
$ if3
word 1: apple
word 2: orange
word 3: pear
No match
$ if3
word 1: apple
word 2: orange
word 3: apple
Match: words 1 & 3
$ if3
word 1: apple
word 2: apple
word 3: apple
Match: words 1, 2, & 3
```

If the three words are not the same, the structure passes control to the first **elif**, which begins a series of tests to see if any pair of words is the same. As the nesting continues, if any one of the **if** statements is satisfied, the structure passes control to the next **then** statement and subsequently to the statement following **fi**. Each time an **elif** statement is not satisfied, the structure passes control to the next **elif** statement. The double quotation marks around the arguments to `echo` that contain ampersands (&) prevent the shell from interpreting the ampersands as special characters.

optional THE lnks SCRIPT

The following script, named **lnks**, demonstrates the **if...then** and **if...then...elif** control structures. This script finds hard links to its first argument, a filename. If you provide the name of a directory as the second argument, **lnks** searches for links in that directory and all subdirectories. If you do not specify a directory, **lnks** searches the working directory and its subdirectories. This script does not locate symbolic links.

```
$ cat lnks
#!/bin/bash
# Identify links to a file
# Usage: lnks file [directory]

if [ $# -eq 0 -o $# -gt 2 ]; then
    echo "Usage: lnks file [directory]" 1>&2
    exit 1
fi
if [ -d "$1" ]; then
    echo "First argument cannot be a directory." 1>&2
    echo "Usage: lnks file [directory]" 1>&2
    exit 1
else
    file="$1"
fi
```

```
if [ $# -eq 1 ]; then
        directory="."
    elif [ -d "$2" ]; then
        directory="$2"
    else
        echo "Optional second argument must be a directory." 1>&2
        echo "Usage: lnks file [directory]" 1>&2
        exit 1
fi

# Check that file exists and is an ordinary file:
if [ ! -f "$file" ]; then
    echo "lnks: $file not found or special file" 1>&2
    exit 1
fi
# Check link count on file
set -- $(ls -l "$file")
linkcnt=$2
if [ "$linkcnt" -eq 1 ]; then
    echo "lnks: no other hard links to $file" 1>&2
    exit 0
fi

# Get the inode of the given file
set $(ls -i "$file")

inode=$1

# Find and print the files with that inode number
echo "lnks: using find to search for links..." 1>&2
find "$directory" -xdev -inum $inode -print
```

Alex has a file named **letter** in his home directory. He wants to find links to this file in his and other users' home directory file trees. In the following example, Alex calls **lnks** from his home directory to perform the search. The second argument to **lnks**, **/home**, is the pathname of the directory he wants to start the search in. The **lnks** script reports that **/home/alex/letter** and **/home/jenny/draft** are links to the same file:

```
$ lnks letter /home
lnks: using find to search for links...
/home/alex/letter
/home/jenny/draft
```

In addition to the **if...then...elif** control structure, **lnks** introduces other features that are commonly used in shell programs. The following discussion describes **lnks** section by section.

Specify the shell

The first line of the **lnks** script uses **#!** (page 288) to specify the shell that will execute the script:

```
#!/bin/bash
```

In this chapter the **#!** notation appears only in more complex examples. It ensures that the proper shell executes the script, even when the user is running a different shell or the script is called from another shell script.

Comments

The second and third lines of **lnks** are comments; the shell ignores the text that follows a pound sign up to the next NEWLINE character. These comments in **lnks** briefly identify what the file does and how to use it:

```
# Identify links to a file
# Usage: lnks file [directory]
```

Usage messages

The first **if** statement tests whether **lnks** was called with zero arguments or more than two arguments:

```
if [ $# -eq 0 -o $# -gt 2 ]; then
    echo "Usage: lnks file [directory]" 1>&2
    exit 1
fi
```

If either of these conditions is *true,* **lnks** sends a usage message to standard error and exits with a status of 1. The double quotation marks around the usage message prevent the shell from interpreting the brackets as special characters. The brackets in the usage message indicate that the **directory** argument is optional.

The second **if** statement tests whether the first command-line argument (**$1**) is a directory (the **–d** argument to test returns a *true* value if the file exists and is a directory):

```
if [ -d "$1" ]; then
    echo "First argument cannot be a directory." 1>&2
    echo "Usage: lnks file [directory]" 1>&2
    exit 1
else
    file="$1"
fi
```

If the first argument is a directory, **lnks** displays a usage message and exits. If it is not a directory, **lnks** saves the value of **$1** in the **file** variable because later in the script set resets the command-line arguments. If the value of **$1** is not saved before the **set** command is issued, its value will be lost.

Test the arguments

The next section of **lnks** is an **if...then...elif** statement:

```
if [ $# -eq 1 ]; then
        directory="."
    elif [ -d "$2" ]; then
        directory="$2"
    else
        echo "Optional second argument must be a directory." 1>&2
        echo "Usage: lnks file [directory]" 1>&2
        exit 1
fi
```

The first ***test-command*** determines whether the user specified a single argument on the command line. If the ***test-command*** returns 0 (*true*), the user-created variable named **directory** is assigned the value of the working directory (**.**). If the ***test-command*** returns *false,* the **elif** statement tests whether the second argument is a directory. If it is a directory, the **directory** variable is set equal to the second command-line argument, **$2**. If **$2** is not a directory, **lnks** sends a usage message to standard error and exits with a status of 1.

The next **if** statement in **lnks** tests whether **$file** does not exist. This test keeps **lnks** from wasting time looking for links to a nonexistent file.

The test builtin with the three arguments **!**, **–f**, and **$file** evaluates to *true* if the file **$file** does *not* exist:

```
[ ! -f "$file" ]
```

The **!** operator preceding the **–f** argument to test negates its result, yielding *false* if the file **$file** *does* exist and is an ordinary file.

Next **lnks** uses set and **ls –l** to check the number of links **$file** has:

```
# Check link count on file
set -- $(ls -l "$file")
linkcnt=$2
if [ "$linkcnt" -eq 1 ]; then
    echo "lnks: no other hard links to $file" 1>&2
    exit 0
fi
```

The set builtin uses command substitution (page 348) to set the positional parameters to the output of **ls –l**. The second field in this output is the link count, so the user-created variable **linkcnt** is set equal to **$2**. The **––** used with set prevents set from interpreting as an option the first argument produced by **ls –l** (the first argument is the access permissions for the file and typically begins with **–**). The **if** statement checks whether **$linkcnt** is equal to 1; if it is, **lnks** displays a message and exits. Although this message is not truly an error message, it is redirected to standard error. The way **lnks** has been written, all informational messages are sent to standard error. Only the final product of **lnks**—the pathnames of links to the specified file—is sent to standard output, so you can redirect the output as you please.

If the link count is greater than one, **lnks** goes on to identify the *inode* (page 1087) for **$file**. As explained on page 215, comparing the inodes associated with filenames is a good way to determine whether the filenames are links to the same file. The **lnks** script uses set to set the positional parameters to the output of **ls –i**. The first argument to set is the inode number for the file, so the user-created variable named **inode** is assigned the value of **$1**:

```
# Get the inode of the given file
set $(ls -i "$file")

inode=$1
```

Finally **lnks** uses the find utility to search for files having inode numbers that match **$inode**:

```
# Find and print the files with that inode number
echo "lnks: using find to search for links..." 1>&2
find "$directory" -xdev -inum $inode -print
```

The find utility searches for files that meet the criteria specified by its arguments, beginning its search with the directory specified by its first argument (**$directory**) and searching all subdirectories. The remaining arguments specify that the filenames of files having inodes matching **$inode** should be sent to standard output. Because files in different filesystems can have the same inode number and not be linked, find must search only directories in the same filesystem as **$directory**. The **–xdev** argument prevents find from searching directories on other filesystems. Refer to page 212 for more information about filesystems and links.

The echo command preceding the find command in **lnks**, which tells the user that find is running, is included because find frequently takes a long time to run. Because **lnks** does not include a final exit statement, the exit status of **lnks** is that of the last command it runs, find.

DEBUGGING SHELL SCRIPTS

When you are writing a script such as **lnks**, it is easy to make mistakes. You can use the shell's **–x** option to help debug a script. This option causes the shell to display each command before it runs the command. Tracing a script's execution in this way can give you information about where a problem lies.

You can run **lnks** as in the previous example and cause the shell to display each command before it is executed. Either set the **–x** option for the current shell (**set –x**) so that all scripts display commands as they are run or use the **–x** option to affect only the shell that is running the script called by the command line.

```
$ bash -x lnks letter /home
+ '[' 2 -eq 0 -o 2 -gt 2 ']'
+ '[' -d letter ']'
+ file=letter
+ '[' 2 -eq 1 ']'
+ '[' -d /home ']'
+ directory=/home
+ '[' '!' -f letter ']'
...
```

PS4 Each command that the script executes is preceded by the value of the **PS4** variable—a plus sign (**+**) by default, so you can distinguish debugging output from script-produced output. You must export **PS4** if you set it in the shell that calls the script. The next command sets **PS4** to **>>>>** followed by a SPACE and exports it:

```
$ export PS4='>>>> '
```

You can also set the **–x** option of the shell running the script by putting the following set command at the top of the script:

```
set -x
```

Put **set –x** anywhere in the script you want to turn debugging on. Turn the debugging option off with a plus sign.

```
set +x
```

The **set –o xtrace** and **set +o xtrace** commands do the same things as **set –x** and **set +x**, respectively.

for...in

The **for...in** control structure has the following syntax:

for ***loop-index*** *in* ***argument-list***
do
    commands
done

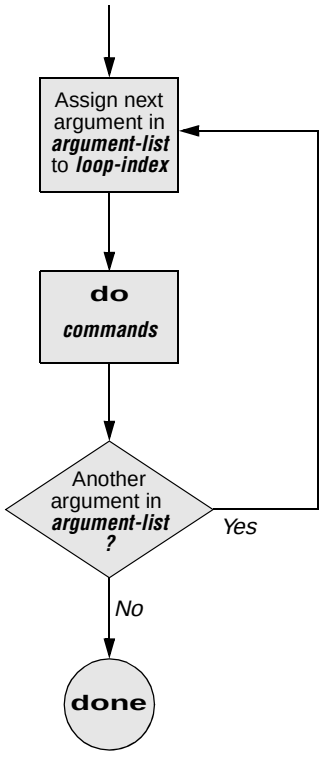

Figure 27-4 A **for...in** flowchart

The **for...in** structure (Figure 27-4, preceding page) assigns the value of the first argument in the ***argument-list*** to the ***loop-index*** and executes the ***commands*** between the **do** and **done** statements. The **do** and **done** statements mark the beginning and end of the **for** loop.

After it passes control to the **done** statement, the structure assigns the value of the second argument in the ***argument-list*** to the ***loop-index*** and repeats the ***commands***. The structure repeats the ***commands*** between the **do** and **done** statements one time for each argument in the ***argument-list***. When the structure exhausts the ***argument-list***, it passes control to the statement following **done**.

The following **for...in** structure assigns **apples** to the user-created variable **fruit** and then displays the value of **fruit**, which is **apples**. Next the structure assigns **oranges** to **fruit** and repeats the process. When it exhausts the argument list, the structure transfers control to the statement following **done**, which displays a message.

```
$ cat fruit
for fruit in apples oranges pears bananas
do
    echo "$fruit"
done
echo "Task complete."

$ fruit
apples
oranges
pears
bananas
Task complete.
```

The next script lists the names of the directory files in the working directory by looping over all the files, using test to determine which files are directories:

```
$ cat dirfiles
for i in *
do
    if [ -d "$i" ]
        then
            echo "$i"
    fi
done
```

The ambiguous file reference character * matches the names of all files (except hidden files) in the working directory. Prior to executing the **for** loop, the shell expands the * and uses the resulting list to assign successive values to the index variable **i**.