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®

SIXTH EDITION

Mark G. Sobell

ISBN-13: 978-0-13-275727-0

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

Objectives

After reading this chapter you should be able to:

- Run a live session and use palimpsest to view and change disk partitioning
- Install Fedora from a Live session
- Install or upgrade Fedora/RHEL using the Fedora/RHEL install DVD
- Modify system behavior with boot time parameters
- Modify partitions during installation
- Select software during installation
- Perform initial configuration with Firstboot
- List the requirement and considerations for a dual-boot configuration

Figure 3-1 Live session, automatic boot screen

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 might 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.

RUNNING A FEDORA LIVE SESSION

As discussed in Chapter 2, a live session is a Linux session 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 a live session.

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.

Figure 3-2 The Fedora Live Welcome menu

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).

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 29 if the system does not boot from the CD. Refer to "Modifying Boot Parameters (Options)" on page 67 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 shaded blue 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 shaded blue 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 booting from Fedora Desktop Live Media (what this book refers to as the *live CD*), the system will run the GNOME desktop manager. When you boot from

this CD, Fedora automatically logs in as the user named **liveuser** and displays the GNOME desktop (Figure 3-3).

KDE If you are booting from Fedora KDE Live Media, the system will run 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 LEFT 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 from a live session (preceding) or install Fedora/RHEL from the install DVD. Installing from a live session is simpler but does not give you the flexibility 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 the palimpsest GNOME Disk Utility (page 77) 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 to 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 Net Install CD, the install DVD, or a USB flash drive.

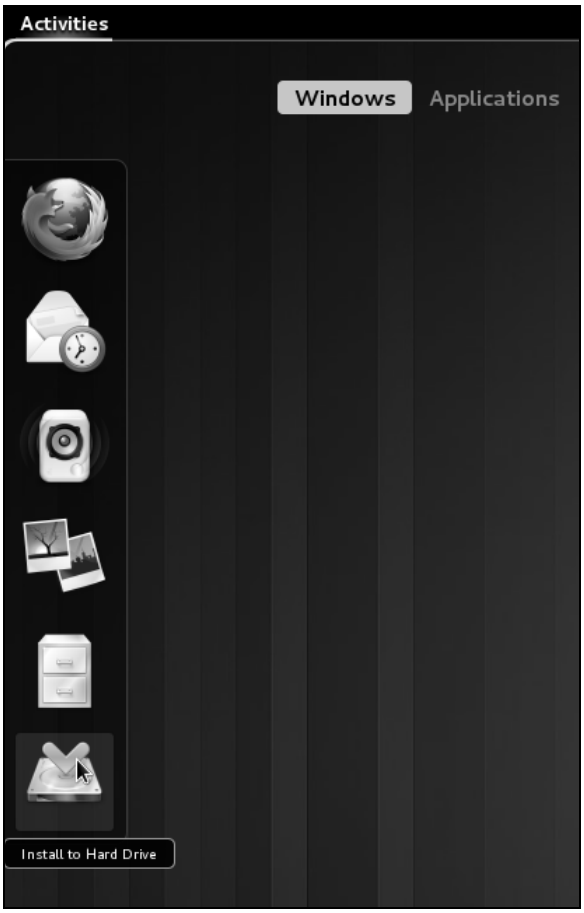

Figure 3-3 A full GNOME 3 Live desktop; install to hard drive

INSTALLING FROM A LIVE SESSION (FEDORA)

Bring up a live GNOME session as explained on page 52. GNOME will display either

- A full GNOME 3 desktop that has the word **Activities** in the upper-left corner of the screen or
- A window with the words **GNOME 3 Failed to Load**. When you click **Close** on that window, GNOME displays a desktop running in Fallback mode with the words **Applications** and **Places** in the upper-left corner of the screen.

GNOME 3 desktop

From a full GNOME 3 desktop, click **Activities**; GNOME displays buttons labeled **Windows** and **Applications** with the button labeled **Windows** highlighted (Figure 3-3). At the bottom of the icons on the left side of the window is an icon depicting a hard drive with a green tick above it. Left-click this icon to begin installing Fedora.

Figure 3-4 A GNOME Fallback Live desktop; install to hard drive

Fallback mode desktop

To begin installing Fedora from a desktop running in Fallback mode, select **Main menu: Applications⇨System Tools⇨Install to Hard Drive** (Figure 3-4) by (left-) clicking **Applications** at the upper-left corner of the screen, clicking **System Tools**, and finally clicking **Install to Hard Drive**.

Continue reading at "Using Anaconda" on page 59.

INSTALLING/UPGRADING FROM THE INSTALL DVD

To install/upgrade Fedora/RHEL from the install DVD, insert the DVD into the DVD drive and turn on or reset the system. After a few moments, the system displays the Welcome to Fedora/RHEL menu (Figure 3-5) 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 3-5. If you do not press a key, after 60 seconds Fedora/RHEL begins a graphical install/upgrade. Refer to "BIOS setup" on page 29 if the system does not boot from the DVD. Refer to "Modifying Boot Parameters (Options)" on page 67 if Fedora/RHEL does not boot or displays an error message.

The Welcome menu has the following selections:

Install a new system 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

Figure 3-5 The install DVD Welcome menu

basic video driver that works with most displays. Choose this selection if the previous selection fails just after the Disc Found screen (below).

Rescue installed system — Brings up a minimal Fedora/RHEL system but does not install it. After detecting the system's disks and partitions, the system enters single-user/rescue mode and allows you to mount an existing Linux filesystem. For more information refer to "Rescue Installed System" on page 457.

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 29] is set up).

Memory test — Runs the memory test described on page 53 (Fedora).

STARTING THE INSTALLATION

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

THE DISC FOUND SCREEN

The first screen the 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 caution box on the next page.

Test the install DVD

caution 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 might 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 67 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 can 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 81).

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 RHEL, 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 138) 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.

Figure 3-6 The language screen

At any time during the installation, you can switch to virtual console 2 (press 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 Anaconda displays during a default installation and explains the choices you can make on each of them.

Language — Anaconda displays the language screen (Figure 3-6) after it obtains enough information to start the X Window System. Installing from a live session does not display this screen. RHEL displays a welcome screen before it displays the language screen. Select the language you want to use for the installation. This language is not necessarily the same language the installed system will display. Click **Next**.

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 might 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.

Figure 3-7 The Install or Upgrade screen

Devices This screen asks you to specify the type of devices you are installing Linux on. In most cases click the radio button labeled **Basic Storage Devices**. If you are installing on an enterprise device, such as a SAN (Storage Area Network), click the radio button labeled **Specialized Storage Devices**.

Warning If Anaconda does not detect information on the hard drive you are installing Linux on, it displays the Storage Device Warning window. If you are sure there is no data you want to keep on the hard drive you are installing Linux on, put a tick in the check box labeled **Apply my choice to all devices with undetected partitions or filesystems** and click **Yes, discard my data** (Fedora) or click **Re-initialize** all (RHEL).

Hostname Specify the name of the system on this screen. If the system will not use DHCP to configure its network connection, click Configure Network at the lower-left corner of the screen to display the Network Connections window; see page 651 for more information. Click **Next**.

Time zone The time zone screen allows you to specify the time zone where the system is located (Figure 2-1, page 32). 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 1193) 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 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 1195). Click **Next**.

Root password Enter and confirm the password for the **root** user (Superuser). See page 409 for more information on **root** privileges. If you enter a password that is not very secure, Anaconda displays a dialog box with the words **Weak password**; click **Cancel** or **Use Anyway**, as appropriate. Click **Next**.

Install or Upgrade (This choice is not available from the live CD.) Anaconda displays the Install or Upgrade screen (Figure 3-7) only if it detects a version of Fedora/RHEL it can upgrade on the hard disk. Anaconda gives you the choice of upgrading the existing installation or overwriting the existing installation with a new one. Refer to

Figure 3-8 The Type of Installation screen

"Installing a Fresh Copy or Upgrading an Existing Fedora/RHEL System?" on page 34 for help in making this selection. Select one of the entries and click **Next**.

Type of Installation

The Type of Installation screen (Figure 3-8) allows you to specify partition information, encrypt the filesystem, and review partition information. Anaconda presents the following choices in the upper part of the screen:

- **Use All Space**—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(s)**—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 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 34) 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.

Click the radio button adjacent to the choice you want and click **Next**.

Encrypt system

To encrypt the filesystems you are creating, put a tick in the check box labeled **Encrypt system**. If you choose to encrypt the filesystems, Anaconda will ask for a passphrase. You will need this passphrase to log in on the system.

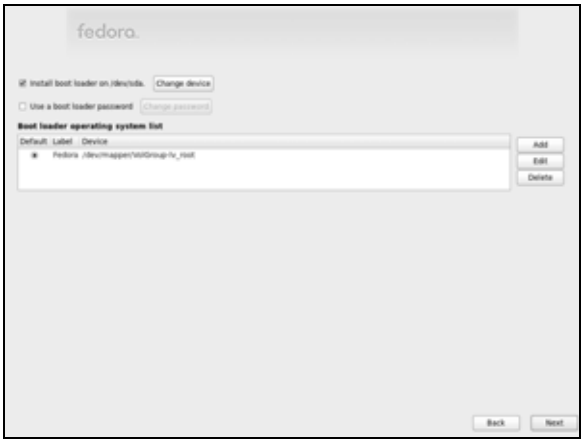

Figure 3-9 The Boot Loader Configuration screen

Default layout — The default layout the first four choices create includes two or three logical volumes (swap, root [/], and if the hard disk is big enough, **/home**) and one standard partition (**/boot**). With this setup, most of the space on the disk is assigned to the **/home** partition, or if there is no **/home** partition, to the root partition. For information on the Logical Volume Manager, see page 42.

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 list 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.

Two hard drives — If the machine you are installing Fedora/RHEL on has two or more hard drives, Anaconda displays a screen that allows you to specify which drives are data storage devices and which are install target devices. Disk Druid sets up data storage devices to be mounted on the installed system; they are not formatted. Install target devices are the devices the system is installed on. Some or all of the partitions on these devices will be formatted, destroying any data on them. This screen also allows you to specify which of the install target devices is to hold the boot loader.

Boot Loader Configuration — Anaconda displays the Boot Loader Configuration screen (Figure 3-9) only when you put a tick in the check box labeled **Review and modify partitioning layout** or select **Create custom layout** from the list in the Type of Installation screen. By default, Anaconda installs the grub boot loader (page 595). 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

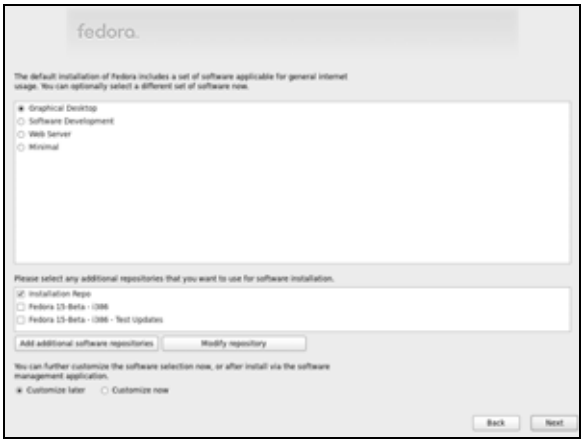

Figure 3-10 The Software Selection screen

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.

Select Network Interface

This window is displayed at this time by the Net Install CD only. See "Select Network Interface" on page 64.

Installing from a live CD

If you are installing from a live CD you are done with the first part of the installation. Click **Close** and then reboot the system by first clicking **Live System User** at the upper-right corner of the screen to display a drop-down menu. If **Suspend** appears at the bottom of this menu, you are running full GNOME 3; press and hold the ALT key to cause **Suspend** to change to **Power Off**. Click **Power Off** and then click **Restart** from the resulting window. If **Shut Down** appears at the bottom of the drop-down menu you are running in Fallback mode (page 92); click **Shut Down** and then click **Restart** from the resulting window. Anaconda ejects the CD as the system shuts down. Continue with "Firstboot: When You Reboot" on page 65.

Software Selection

As the Software Selection screen explains, by default Anaconda installs a basic system, including software that allows you to use the Internet. See Figure 3-10. Near the top of the screen are four check boxes that you can put ticks in to select categories of software to install: **Graphical Desktop** (selected by default), **Software Development**, **Web Server**, and **Minimal** (Fedora; RHEL selections differ slightly).

Fedora/RHEL software is kept in repositories (page 533). When you install Fedora, middle of the screen holds check boxes 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 15 –** *xxx*—Indicates Anaconda is to use the online Fedora 15 repository. The *xxx* indicates the system architecture (e.g., i386).
- **Fedora 15 –** *xxx* **– Updates**—Indicates Anaconda is to use the online Fedora 15 Updates repository. The *xxx* indicates the system architecture (e.g., i386).

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.

Select Network Interface

When you put a tick in either of the last two check boxes, Anaconda displays the Select Network Interface window. Select the interface you want to use from this window. Click **OK**; Anaconda opens the Network Connections window. If the system is configured using DHCP click **Close**. Otherwise configure the network connection as explained on page 651 and then click **Close**. Anaconda takes a moment to configure the network and then retrieves information from the repository you specified.

Below the repository selection frame in the Software Selection screen are buttons labeled **Add additional software repositories** and **Modify repository** (Fedora and RHEL).

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 the screen.
- **Customize now**—Displays the package selection screen (next) 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 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 side-by-side frames near the top of the screen (Figure 3-11). If you added repositories in addition to the Installation repository or if you are installing RHEL, 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 Software Compilation**; 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

Figure 3-11 The package selection screen

packages you want to install in addition to the base packages. If you install KDE and you do not want to install GNOME too, you must remove the tick from the check box labeled **GNOME Desktop Environment**. 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 Kickstart file (page 81) 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.

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 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 or other installation medium. 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.

Figure 3-12 The Welcome screen

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**.

Software Updates (RHEL) When installing RHEL, the next screen asks you to set up software updates by registering with RHN (Red Hat Network; page 554).

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 602.

Putting a tick in check box labeled **Add to Administrators group** (Fedora) adds the user to the **wheel** group. The user can then gain **root** privileges by running sudo and providing her password (not the **root** password). See page 415 for more information on using sudo and page 422 for more information on the **wheel** group. Click the button labeled **Use Network Login** to set up a network login using Kerberos, LDAP, or NIS.

Click **Advanced** to display the User Manager window (page 602).

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**.

Kdump (RHEL) Kdump captures and preserves kernel dump information. When installing RHEL, put a tick in the box labeled **Enable kdump** to enable Kdump.

Hardware Profile (Fedora) When you are installing Fedora, the next screen allows the system to share its profile with the Fedora Project. The information is shared anonymously and helps build an up-to-date Linux hardware database that includes distribution information. You can use this database to help you choose components when you buy or build a Linux system.

Click the radio button labeled **Send Profile** to cause the smolt hardware profiler to send monthly updates of the system's hardware profile to smolts.org. Select the radio button labeled **Do not send profile** if you do not want to share the system's profile. Click **Finish**.

When the Hardware Profile (Fedora) or Kdump (RHEL) screen closes, the installation is complete. You can now use the system and set it up as you desire. For example, you might 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 **mandb** (Fedora) or **makewhatis** (RHEL) database ensures the whatis (page 128) and apropos (page 127) 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 **crond** (page 611) 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 (page 409), give the following commands:

```
# /etc/cron.daily/man-db.cron          (Fedora)
# /etc/cron.daily/makewhatis.cron      (RHEL)
# /etc/cron.daily/mlocate.cron         (Fedora/RHEL)
```

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

INSTALLATION TASKS

This section details some common tasks you might 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)

To modify boot parameters, you must interrupt the automatic boot process by pressing a key such as the SPACE bar while Fedora/RHEL is counting down when you first boot from a live CD (page 53) or install DVD (page 56). When you press a key, Fedora displays the Welcome menu (Figure 3-2 on page 53 or Figure 3-5 on page 57). Use the ARROW keys to highlight the selection you want

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

before proceeding. With the desired selection highlighted, press the TAB key to display the boot command-line parameters (Figure 3-13).

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 for **linux boot parameters**.

What to do if the installation does not work

tip On some hardware, the installation might 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).

```
text console=ttyS0,115200n8
```

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
```

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. Specifying **acpi=off** has the same effect.

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. Specifying **apm=off** has the same effect.

askmethod Displays the Installation Method screen, which presents a choice of installation sources: Local CD/DVD, Hard drive, NFS directory, and URL (first installation CD, Net Install 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/FC15** 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/FC15** (no leading slash).
- The next two selections attempt to use NetworkManager (page 651) to set up a DHCP connection automatically. 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 allows you to enter the NFS server name, the name of the directory that contains the installation tree or the ISO image of the install DVD, and optionally NFS mount options (page 797). Enter the server's IP address and the *name* of the exported directory, not its device name. The remote (server) system must export (page 805) the directory hierarchy that holds the installation tree or the ISO image of the install DVD.
  - **URL**—Displays the URL Setup screen, which allows you to enter the URL of the directory that contains the installation tree or the ISO image of the install DVD, and optionally the URL of a proxy server, a username, and a password.

nodma Turns off direct memory access (DMA) for all disk controllers. This parameter might 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 can 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 81) 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** (below).

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

repo=***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:

```
repo=ftp://fedora.mirrors.pair.com/fedora/linux/releases/15/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 Sets the system up to rescue an installed system; see page 457 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 31) 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 free VNC client that runs on several platforms from www.realvnc.com. Use yum (page 534) 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 you also use the **vnc** option.

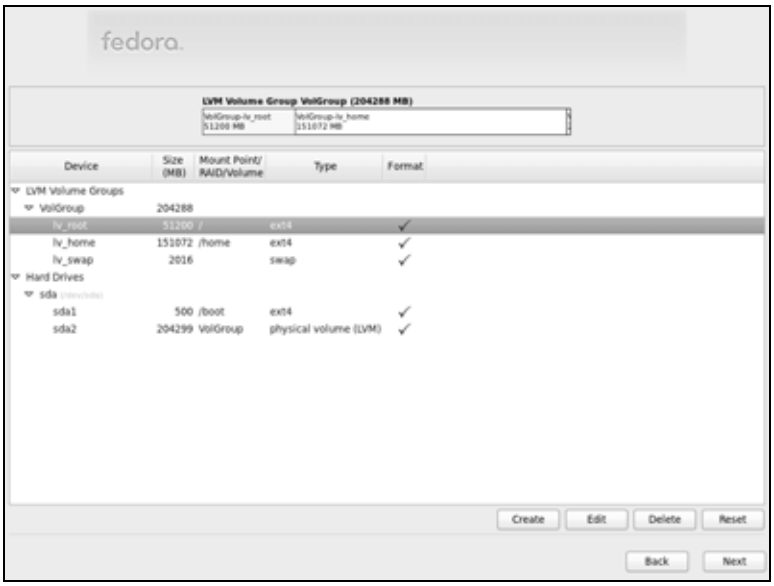

Figure 3-14 Disk Druid: main screen, default layout

USING DISK DRUID TO PARTITION THE DISK

See page 34 for a discussion of the setup of the hard disk and partitions.

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 77), parted (page 617), 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. See page 42 for a discussion of LVM (Logical Volume Manager) including PVs, VGs, and LVs.

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 Type of Installation screen (Figure 3-8, page 61).

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 "Type of Installation" on page 61). The middle of the screen holds a table listing hard drives and LVM Volume Groups and the partitions or LVs each holds, one per line.

Fedora names the Volume Group after the hostname you specified earlier in the installation. If you specified **tiger** as the hostname, the Volume Group name would be **vg_tiger**. If you accept the default hostname of **localhost.localdomain**, Fedora names the Volume Group **VolGroup** as in the examples in this section.

The highlighted logical volume, **lv_root**, is in the Volume Group named **VolGroup**, which is depicted graphically at the top of the screen. The **lv_swap** logical volume is very small; it is the sliver at the right end of the graphical representation.

The following buttons appear near the bottom of the screen:

- **Create**—Displays the Create Storage window (next) that allows you to create a partition or set up software RAID or LVM LVs
- **Edit**—Edits the highlighted device (next page)
- **Delete**—Deletes the highlighted device
- **Reset**—Cancels the changes you have made and causes the Disk Druid table to revert so it matches the layout of the disk

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
- **Size (MB)**—The size of the partition or LV in megabytes
- **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/LV 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

THE CREATE STORAGE WINDOW

Clicking Create on the Disk Druid main screen displays the Create Storage window (Figure 3-15). This window has three sections, each of which has one or more selections:

- Create Partition (pages 34 and 74)
  - Standard Partition—Create a partition
- Create Software RAID (pages 41 and 77)
  - RAID Partition—Create a software RAID partition
  - RAID Device—Join two or more RAID partitions into a RAID device
- Create LVM (pages 42 and 73)
  - LVM Volume Group—(VG) Specify PVs that make up a VG; also allows you to specify LVs that are in the VG

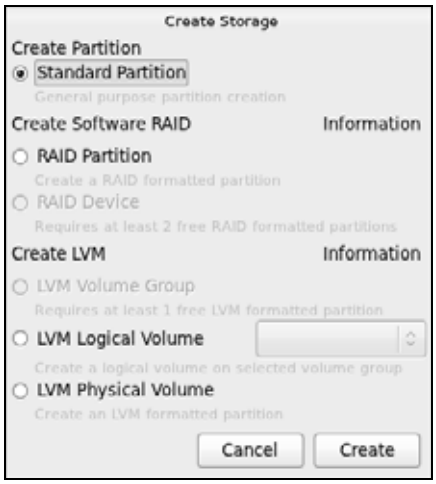

Figure 3-15 The Create Storage window

- LVM Logical Volume—(LV) Specify LVs that are in a VG
- LVM Physical Volume—(PV) Specify PVs that make up a VG

Make a selection by clicking the adjacent radio button. Click **Information** to display information about the adjacent section.

blank

8

Linux GUIs: X and GNOME

In This Chapter

Objectives

After reading this chapter you should be able to:

- Describe the history of the X Window System
- Start X from a character-based display
- Use X remotely across a network
- Customize the mouse buttons in the X Window System using the command line
- Explain the similarities, differences, and history of GNOME and KDE desktops
- Use the Nautilus File Browser
- Start a terminal emulator and run a graphical program from the emulator
- Search for files using the Search for Files window

THE NAUTILUS FILE BROWSER WINDOW

"Using Nautilus to Work with Files" on page 102 presented an introduction to using Nautilus. This section discusses the Nautilus File Browser window in more depth.

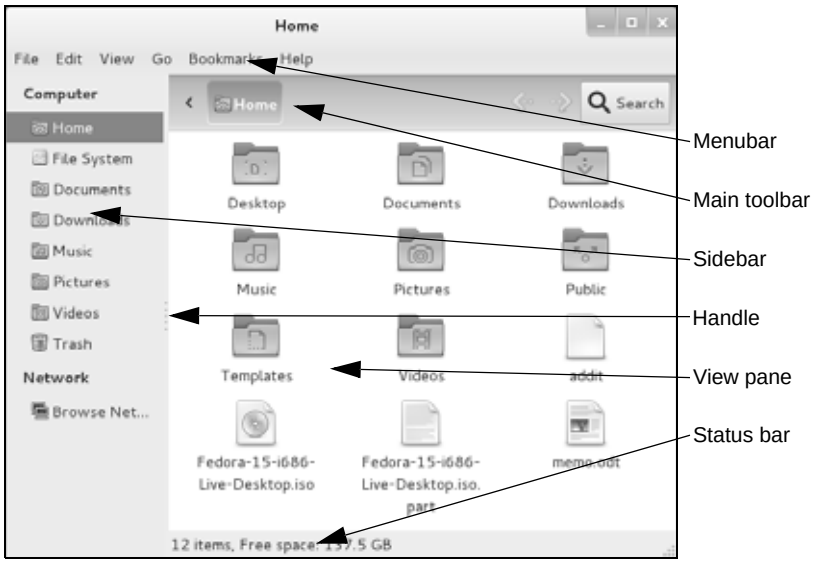

Figure 8-2 A Nautilus File Browser window displaying icons

RHEL: turn off Spatial view; turn on File Browser windows

tip Under RHEL, 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 272) and turn on File Browser windows. For more information refer to "The Two Faces of Nautilus" on page 103.

Figure 8-2 shows a File Browser window with a Sidebar, View pane, menubar, Main toolbar, 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 bottom of the **File Browser menubar: View** menu. Figure 8-2 shows an Icon view and Figure 8-3 on the next page 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 95 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 SIDEBAR

The Sidebar augments the information Nautilus displays in the View pane. You can close or display the Sidebar by pressing F9 or by selecting **File Browser menubar: View⇨Sidebar⇨Show Sidebar**. To change the horizontal size of the Sidebar, drag the handle (Figure 8-2) on its right side. The **File Browser menubar:**

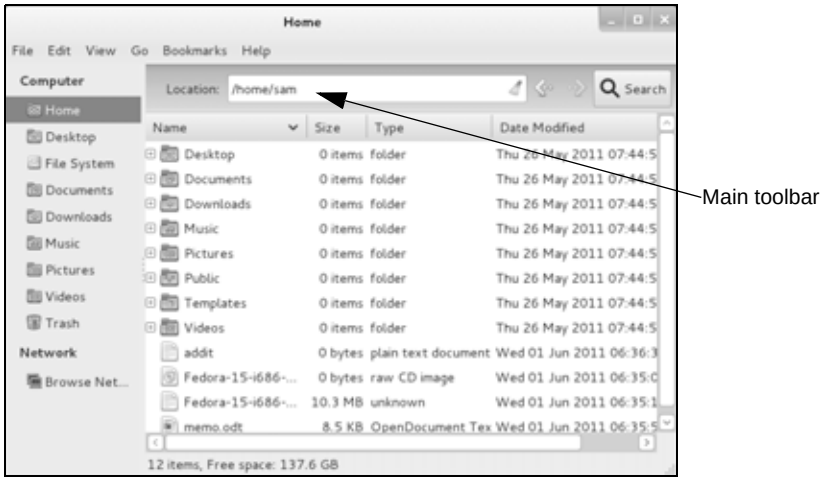

Figure 8-3 A Nautilus File Browser window displaying a List view and a textual location in the Main toolbar

View⇨Sidebar menu controls whether the Sidebar displays places or a file tree as described following.

Places Places lists folders, bookmarks, devices, and network locations. Double-click one of these places to display that place in the View pane. You can open a place in a new File Browser tab or window by right-clicking the directory in the Sidebar and selecting **Open in New Tab** or **Open in New Window**, respectively.

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 in the Sidebar and selecting **Remove**.

Tree Tree presents an expandable tree view of your home folder and each mounted filesystem. Each directory in the tree has a plus (+) or minus (–) sign to its left. Click a plus sign to expand a directory; click a minus sign to close a directory. Click a directory in the tree to display that directory in the View pane. Double-click a directory to expand or close it in the Sidebar and display it in the View pane.

Nautilus can open a terminal emulator

tip When you install the **nautilus-open-terminal** package (see page 534 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 190).

CONTROL BARS

This section discusses the three of the control bars that can appear in a File Browser window: the status bar, menubar, and Main toolbar (Figure 8-2, page 267). 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 depends 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: Location, Back, Forward, and Search. The Location buttons display the name of the directory that appears in the View pane. By default, Nautilus displays Location in iconic format. Press CONTROL-L to change the Location to textual format. 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. Display or remove the Main toolbar by selecting **File Browser menubar: View⇨Main toolbar**.

In iconic format, each button represents a directory in a pathname (page 191). 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. Nautilus displays another directory when you enter the pathname of the directory and press RETURN.

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. Display or remove the status bar by selecting **File Browser menubar: View⇨Statusbar**.

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

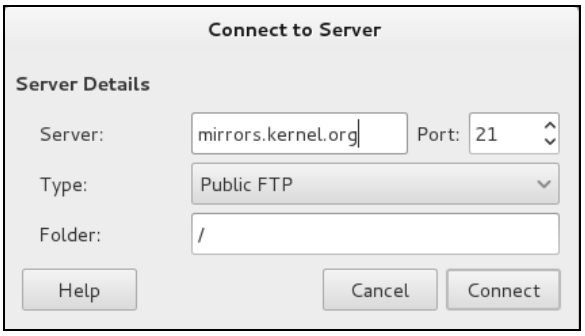

Figure 8-4 The Connect to Server window

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 **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 two selections from Edit: Compress 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 162) file compressed by gzip (page 161) and **.tar.bz2** creates a tar file compressed by bzip2 (page 160). Click the plus sign 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).

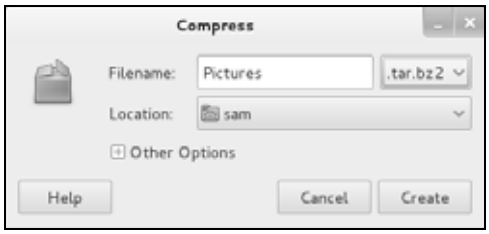

Figure 8-5 The Compress window

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

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

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.

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.

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 (by size and Always, Local Files Only, Never).

View Click the **Sidebar**, **Main toolbar**, 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 190).

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

Bookmarks Bookmarks appear at the bottom of this menu and in the Sidebar under Bookmarks. The Bookmarks selections are explained under "Places" on page 268.

Help The Help selections display local information about Nautilus.

optional

THE NAUTILUS SPATIAL VIEW (RHEL)

Under RHEL, 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, RHEL displays 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 103 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 might take some getting used to. It always provides one window per folder. By default, when you open a folder, Nautilus displays a new window.

To open a Spatial view of your home directory, Select **Main menu: Home Folder** 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**.

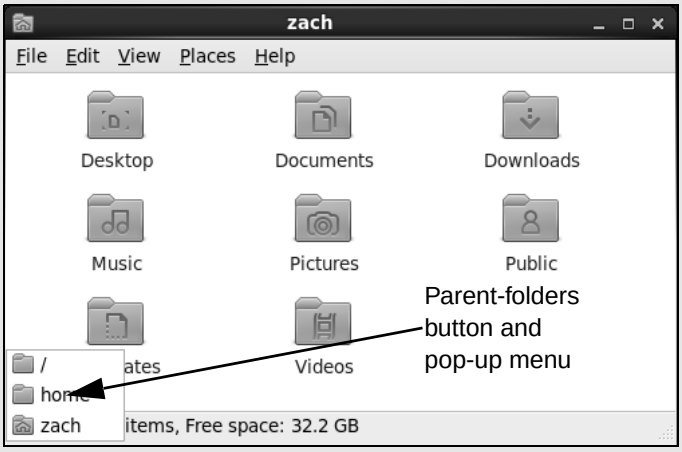

Figure 8-7 The Nautilus Spatial view

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 might be more familiar and can help keep the desktop from becoming 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**.

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.

PICK A FONT WINDOW

The Pick a Font window (Figure 8-8) appears when you select **Fonts** from the Tweak Tool window (page 94) and click one of the font buttons on the right side of the window. From the Pick a Font 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.

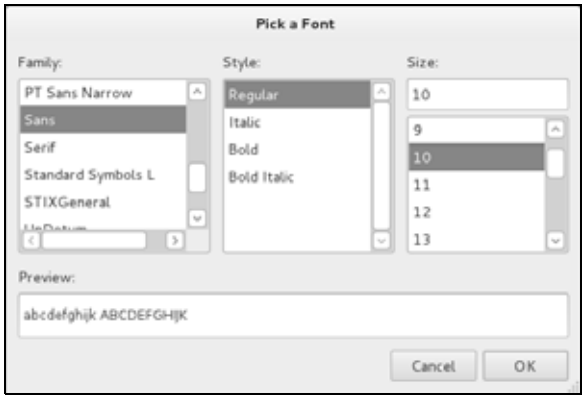

Figure 8-8 The Pick a Font window

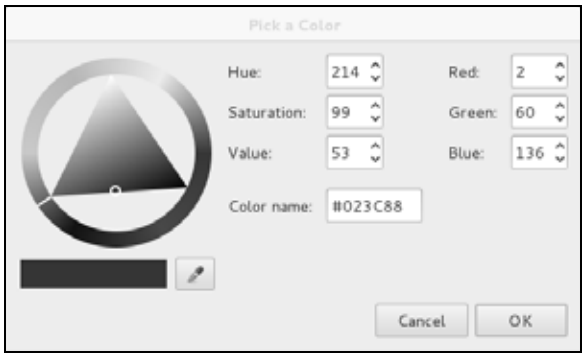

Figure 8-9 The Pick a Color window

PICK A COLOR WINDOW

The Pick a Color window (Figure 8-9) appears when you need to specify a color, such as when you click the colored button near the lower-right corner of the Background window (Figure 4-8, page 100).

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.

11

System Administration: Core Concepts

In This Chapter

Objectives

After reading this chapter you should be able to:

- Explain the need for and the responsibility of the privileged user (**root**)
- Gain privilege using su and sudo
- Describe the startup sequence using systemd (Fedora) and Upstart (RHEL)
- Explain the history and current role of SysVinit scripts
- Manage which services start at boot time
- Start and stop services on a running system
- Boot into single-user mode for system maintenance
- Shut down a running system
- Secure a system by applying updates, monitoring logs, and controlling access to files using SELinux, setuid permission, and PAM
- Use system administration tools to monitor and maintain the system
- List common steps for installing, configuring, and securing a server
- Configure a system using a static IP address or using DHCP

SECURING A SERVER

Two ways you can secure a server are by using TCP wrappers and by setting up a chroot jail. This section describes both techniques. Alternatively, you can use system-config-firewall (page 893) to secure a system. It covers protocols such as **httpd** that are not protected by **libwrap**. For more information refer to "Securing a System" on page 458.

TCP WRAPPERS: SECURE A SERVER (hosts.allow AND hosts.deny)

Follow these guidelines when you open a local system to access from remote systems:

- 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).

libwrap

As part of the client/server model, TCP wrappers, which can be used for any daemon that is linked against **libwrap**, rely on the **/etc/hosts.allow** and **/etc/hosts.deny** files as the basis of a simple access control list (ACL). 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. The output of ldd shows that one of the shared library dependencies of **sshd** is **libwrap**:

```
$ ldd /usr/sbin/sshd | grep libwrap
        libwrap.so.0 => /lib/libwrap.so.0 (0x00e7c000)
```

hosts.allow and **hosts.deny**

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 (e.g., **rpcbind**, **vsftpd**, **sshd**), *client_list* is a comma-separated list of one or more clients (see Table 11-5 on page 478), 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 to a server, the **hosts.allow** and **hosts.deny** files on the server system are consulted in the following order 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 **hosts.allow** or **hosts.deny**, 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 were empty. Although 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 to do so 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 any client 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. Thus a process cannot access files above the root directory because none exists. 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, executing, 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 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 with **root** privileges, 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, the following command sets up a chroot jail in the (existing) **/home/sam/jail** directory:

```
# /usr/sbin/chroot /home/sam/jail
/usr/sbin/chroot: failed to run command '/bin/bash': No such file or directory
```

This example sets up a chroot jail, but when the system 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, /. As a consequence, chroot cannot find the file identified by the pathname **/bin/bash**. In this situation the chroot jail works correctly but is not useful.

Getting a chroot jail to work the way you want is more complicated. To have the preceding example run bash in a chroot jail, 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 to **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.

```
$ pwd
/home/sam/jail
$ mkdir bin lib
$ cp /bin/bash bin
$ ldd bin/bash
        linux-gate.so.1 =>  (0x00988000)
        libtinfo.so.5 => /lib/libtinfo.so.5 (0x0076b000)
        libdl.so.2 => /lib/libdl.so.2 (0x00afb000)
        libc.so.6 => /lib/libc.so.6 (0x00110000)
        /lib/ld-linux.so.2 (0x00923000)

$ cp /lib/{libtinfo.so.5,libdl.so.2,libc.so.6,ld-linux.so.2} lib
```

Now start the chroot jail again. Although all the setup can be done by an ordinary user, you must be working with **root** privileges to run chroot:

```
$ su
Password:
#/usr/sbin/chroot .
bash-4.1# pwd
/
bash-4.1# ls
bash: ls: command not found
bash-4.1#
```

This time chroot finds and starts **bash**, which displays its default prompt (**bash-4.1#**). The pwd command works because it is a shell builtin (page 249). However, bash cannot find the ls utility because 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. An **exit** command allows you to escape from the jail.

If you provide chroot with a second argument, it takes that argument as the name of the program to run inside the jail. The following command is equivalent to the preceding one:

```
# /usr/sbin/chroot /home/sam/jail /bin/bash
```

To set up a useful chroot jail, first determine which utilities the users of the chroot jail 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 size of the base system with bash and the core utilities exceeds 50 megabytes.) You can find the source code for most common utilities in the **bash** and **coreutils** SRPMS (source rpm) packages.

The chroot utility fails unless you run it with **root** privileges. The result of running chroot with **root** privileges is a **root** shell (a shell with **root** privileges) running inside a chroot jail. Because a user with **root** privileges can break out of a chroot jail, it is imperative that you run a program in the chroot jail with reduced privileges (i.e., privileges other than those of **root**).

There are several ways to reduce the privileges of a user. For example, you can put su or sudo in the jail and then start a shell or a daemon inside the jail, using one of these programs to reduce the privileges of the user working in the jail. A command such as the following starts a shell with reduced privileges inside the jail:

```
# /usr/sbin/chroot jailpath /bin/su user -c /bin/bash
```

where *jailpath* is the pathname of the jail directory, and *user* is the username under whose privileges the shell runs. The problem with this scenario is that sudo and su, as compiled for Fedora/RHEL, call PAM. To run one of these utilities you need to put all of PAM, including its libraries and configuration files, in the jail, along with sudo (or su) and the **/etc/passwd** file. Alternatively, you can recompile su or sudo. The source code calls PAM, however, so you would need to modify the source so it does not call PAM. Either one of these techniques is time-consuming and introduces complexities that can lead to an unsecure jail.

The following C program[2] runs a program with reduced privileges in a chroot jail. Because this program obtains the UID and GID of the user you specify on the command line before calling **chroot**(), you do not need to put **/etc/passwd** in the jail. The program reduces the privileges of the specified program to those of the specified user. This program is presented as a simple solution to the preceding issues so you can experiment with a chroot jail and better understand how it works.

2. Thanks to David Chisnall and the Étoilé Project (etoileos.com) for the **uchroot.c** program.

```
$ cat uchroot.c

/* See svn.gna.org/viewcvs/etoile/trunk/Etoile/LiveCD/uchroot.c for terms of use. */

#include <stdio.h>
#include <stdlib.h>
#include <pwd.h>

int main(int argc, char * argv[])
{
        if(argc < 4)
        {
                printf("Usage: %s {username} {directory} {program} [arguments]\n",
argv[0]);
                return 1;
        }
        /* Parse arguments */
        struct passwd * pass = getpwnam(argv[1]);
        if(pass == NULL)
        {
                printf("Unknown user %s\n", argv[1]);
                return 2;
        }
        /* Set the required UID */
        chdir(argv[2]);
        if(chroot(argv[2])
                ||
                setgid(pass->pw_gid)
                ||
                setuid(pass->pw_uid))
        {
                printf("%s must be run as root.  Current uid=%d, euid=%d\n",
                                argv[0],
                                (int)getuid(),
                                (int)geteuid()
                                );
                return 3;
        }
        return execv(argv[3], argv + 3);
}
```

The first of the following commands compiles **uchroot.c** using cc (**gcc** package), creating an executable file named uchroot. Subsequent commands move uchroot to **/usr/local/bin** and give it appropriate ownership.

```
$ cc -o uchroot uchroot.c
$ su
password:
# mv uchroot /usr/local/bin
# chown root:root /usr/local/bin/uchroot
# exit
$ ls -l /usr/local/bin/uchroot
-rwxrwxr-x. 1 root root 5704 12-31 15:00 /usr/local/bin/uchroot
```

Using the setup from earlier in this section, give the following command to run a shell with the privileges of the user **sam** inside a chroot jail:

```
# /usr/local/bin/uchroot sam /home/sam/jail /bin/bash
```

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** directories somewhere other than 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 want to run a specific service inside the jail. To run a service inside a jail, make sure all files needed by that service are inside the jail. Using uchroot, the format of a command to start a service in a chroot jail is

```
# /usr/local/bin/uchroot user jailpath daemonname
```

where *jailpath* is the pathname of the jail directory, *user* is the username that runs the daemon, and *daemonname* is the pathname (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 877), and the **vsftpd** FTP server can automatically start chroot jails for clients (page 717).

SECURITY CONSIDERATIONS

Some services need to be run by a user or process with **root** privileges but release their **root** privileges once started (Apache, Procmail, and **vsftpd** are examples). If you are running such a service, you do not need to use uchroot or put su or sudo inside the jail.

A process run with **root** privileges can potentially escape from a chroot jail. For this reason, you should reduce privileges 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 to the jail.

DHCP: CONFIGURES NETWORK INTERFACES

Instead of storing network configuration information in local files on each system, DHCP (Dynamic Host Configuration Protocol) enables client systems to retrieve the necessary network configuration information from a DHCP server 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.
- DHCP facilitates the use of IP addresses 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 has lengthened the life of the IPv4 protocol.

DHCP is particularly useful for an administrator who is responsible for maintaining a large number of systems because new systems no longer need to be set up with unique configuration information.

MORE INFORMATION

Web www.dhcp.org
DHCP FAQ: www.dhcp-handbook.com/dhcp_faq.html

Local DHCP client: **/usr/share/doc/dhclient-***
DHCP server: **/usr/share/doc/dhcp-***

HOWTO *DHCP Mini HOWTO*

HOW DHCP WORKS

Using dhclient, the client contacts the server daemon, **dhcpd**, to obtain the IP address, netmask, broadcast address, nameserver address, and other networking parameters. In turn, 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 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. This lease might 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 378).

When you install Fedora/RHEL, the system runs a DHCP client, connects to a DHCP server if it can find one, and configures its network interface.

DHCP CLIENT

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

PREREQUISITES

Make sure the following package is installed:

- **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 the lease information in a file named **dhclient-*-*interface*.lease**, which resides in the **/var/lib/dhclient** directory. The *interface* is the name of the interface the client uses, such as **eth0**. The system uses this information to reestablish a lease when either the server or the client needs to reboot. The DHCP client configuration file, **/etc/dhcp/dhclient.conf**, is required only for custom configurations.

The following **dhclient.conf** file specifies a single interface, **eth0**:

```
$ cat /etc/dhcp3/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 1174) of the device controlling that interface. See page 493 for instructions on how to determine the MAC address of a device. 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 time is up. A lease time of 86,400 seconds (one day) is a reasonable choice for a workstation.

DHCP SERVER

A DHCP server maintains a list of IP addresses and other configuration parameters. Clients request network configuration parameters from the server.

PREREQUISITES

Install the following package:

- **dhcp**

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

```
# chkconfig dhcpd on
```

After configuring the DHCP server, start or restart **dhcpd**:

```
# service dhcpd start
```

dhcpd: THE DHCP DAEMON

A simple DCHP server (**dhcpd**) allows you to add clients to a network without maintaining a list of assigned IP addresses. A simple network, such as a home-based 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 if clients are to find them. If servers and routers are configured without DHCP, you can specify a simple DHCP server configuration in **/etc/dhcp/dhcpd.conf**:

```
$ 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;
option domain-name "example.com";

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

By default, **dhcpd** serves requests on all nonbroadcast network interfaces.

The preceding configuration file specifies a LAN where both the router and the DNS server are 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 the DHCP server can assign. In the case of 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 from 192.168.1.2 to 192.168.1.200. The DHCP server starts at the bottom of this range and attempts to assign a new IP address to each new client. Once the DHCP server reaches the top

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. Two systems cannot use the same IP address at the same time.

Once you have configured a DHCP server, restart it using the **dhcpd** init script (page 492). When the server is running, clients configured to obtain an IP address from the server using DHCP should be able to do so. See the **/usr/share/doc/dhcp*/*sample** files for sample **dhcpd.conf** files.

STATIC IP ADDRESSES

As mentioned earlier, routers and servers typically require static IP addresses. Although you can manually configure IP addresses for these systems, it might be more convenient to have the DHCP server provide them with static IP addresses. See page 652 if you want to configure a system to use a static IP address without using DHCP.

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 it can assign the proper IP address to that system. The DHCP server uses the *MAC address* (page 1174) of the system's network interface 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.

Determining a MAC address

The ip utility displays the MAC addresses of the Ethernet cards in a system. In the following example, the MAC address is the colon-separated series of hexadecimal number pairs following **link/ether**:

```
$ ip link show eth1
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 00:0c:29:12:35:6e brd ff:ff:ff:ff:ff:ff
```

Run ip on each system that requires a static IP address. Once you have determined the MAC addresses of these systems, you can add a **host** section to the **/etc/dhcp/dhcpd.conf** file for each one, instructing the DHCP server to assign a specific address to that 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 **dhcpd** init script (page 492).

nsswitch.conf: WHICH SERVICE TO LOOK AT FIRST

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

Syntax

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 syntax:

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

where *info* is the type of information 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.

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 this 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 action. It is possible for the search to end without finding the requested information.

INFORMATION

The **nsswitch.conf** file commonly controls searches for usernames, passwords, host IP addresses, and group information. The following list describes most of the types of information (*info* in the syntax given earlier) that **nsswitch.conf** controls searches for:

automount	Automount (**/etc/auto.master** and **/etc/auto.misc**; page 811)
bootparam	Diskless and other booting options (**bootparam** man page)
ethers	MAC address (page 1174)
group	Groups of users (**/etc/group**; page 506)
hosts	System information (**/etc/hosts**; page 507)
networks	Network information (**/etc/networks**)
passwd	User information (**/etc/passwd**; page 508)
protocols	Protocol information (**/etc/protocols**; page 510)
publickey	Used for NFS running in secure mode
rpc	RPC names and numbers (**/etc/rpc**; page 511)
services	Services information (**/etc/services**; page 511)
shadow	Shadow password information (**/etc/shadow**; page 511)

METHODS

Following is a list of the types of information that **nsswitch.conf** controls searches for (*method* in the syntax shown on the previous page). For each type of information, you can specify one or more of the following methods:[3]

compat	**+** syntax in **passwd**, **group**, and **shadow** files (page 496)
dns	Queries the DNS (**hosts** queries only)
files	Searches local files such as **/etc/passwd** and **/etc/hosts**
ldap	Queries an LDAP server (page 776)
nis	Searches the NIS database; **yp** is an alias for **nis**

SEARCH ORDER

The information provided by two or more methods might overlap: For example, both **files** and **nis** might 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 place 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 and all that is lost is a little time. Order becomes critical when you use actions between methods or when overlapping entries differ.

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 is used and is considered authoritative. The second line uses NIS to find an IP address given a hostname; if that fails, it searches **/etc/hosts;** if that fails, it checks with DNS to find the 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. 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* (uppercase by convention) 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.

3. Other, less commonly used methods also exist. 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 as a Linux server and is not discussed in this book.

STATUS *STATUS* might have any of the following values:

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

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

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

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

action There are two possible values for *action:*

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 a subsequent 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 (checks 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 the **nsswitch.conf** file. 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.

20

sendmail: Setting Up Mail Servers, Clients, and More

In This Chapter

Objectives

After reading this chapter you should be able to:

- Explain what **sendmail** is
- Explain the purpose and give examples of an MUA, an MTA, and an MDA
- Describe the role of SMTP, IMAP, and POP
- Configure the local SMTP service to use a smarthost for outgoing mail
- Configure the local SMTP server to accept incoming mail, relay outgoing mail for specific hosts, and deliver mail directly to the remote systems
- Configure mail aliases and mail forwarding
- Install and configure SpamAssassin on a mail client and a mail server
- Install and configure a Webmail service (SquirrelMail)
- Setup a mailing list (Mailman)
- Install and configure an IMAP server (Dovecot)
- Describe the process and purpose of authenticated relaying

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 Evolution, KMail, Thunderbird, mutt, and Outlook. When you send an email, the MUA hands it to an MTA (mail transfer agent, such as **exim4** or **sendmail**), which transfers it to the destination server. At the destination, an MDA (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 (Simple Mail Transfer Protocol) server, using POP (Post Office Protocol) or IMAP (Internet Message Access Protocol).

SMTP

Most Linux MUAs expect a local MTA such as **sendmail** to deliver outgoing email. On some systems, including those with a dial-up connection to the Internet, the MTA sends email to an ISP's mail server. Because most MTAs use SMTP (Simple Mail Transfer Protocol) to deliver email, they are often referred to as SMTP servers.

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

tip

Most MUAs can use POP or IMAP to receive email from an ISP's server. 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. Although you still need SMTP to send email, the SMTP server can be at a remote location, such as your ISP. Thus you might not need to concern yourself with it either.

In the default Fedora 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. Although by default RHEL uses **postfix** as the MTA, this chapter explains how to set up **sendmail** as the MTA.

INTRODUCTION TO sendmail

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 where you want to receive email, you will not need to set up **sendmail** to receive mail from nonlocal systems.

OUTBOUND EMAIL

When used as a server, **sendmail** accepts outbound email and directs it to the system it is addressed to (the destination system). This section describes the different ways you can set up **sendmail** to perform this task.

ACCEPTING EMAIL FOR DELIVERY

You can set up a **sendmail** server so it accepts outbound email from the local system only, from specified systems (such as a LAN), or from all systems. Accepting email from unknown systems makes it likely the server will propagate spam.

DELIVERING EMAIL

Direct connection

You can set up a **sendmail** server so it connects directly to the SMTP server on nonlocal destination systems. This SMTP server then delivers the email. Typically, **sendmail** delivers local email directly.

Smarthost

Alternatively, you can set up a **sendmail** server so it sends email bound for nonlocal systems to an SMTP server that relays the mail to its destination. This type of server is called a *smarthost* or *SMTP relay*.

Port 25

By default, SMTP uses port 25. Some Windows viruses use a simple, self-contained SMTP server to propagate themselves. As a partial defense against these types of viruses, some ISPs and larger organizations block all outgoing connections that originate or terminate on port 25, with the exception of connections to their own SMTP server (smarthost). Blocking this port prevents local systems from making a direct connection to send email. These organizations require you to use a smarthost for email bound for nonlocal systems.

INBOUND EMAIL

Although not typical, you can set up **sendmail** to accept email for a registered domain name as specified in the domain's DNS MX record (page 853). 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 email from a system other than the one running the incoming mail server, you might need to set up a POP or IMAP server, as discussed on page 754.

ALTERNATIVES TO sendmail

Over the years, **sendmail** has grown to be enormously complex. Its complexity makes it challenging to configure if you want to set up something more than a simple mail server. Its size and complexity also add to its vulnerability. For optimal security, make sure you run the latest version of **sendmail** and always keep **sendmail** up-to-date. Or you might want to consider using one of the following alternatives.

exim4 The default MTA under Ubuntu, **exim4** (www.exim.org; **exim** package), was introduced in 1995 by the University of Cambridge. It can be configured in several different ways and includes many features other MTAs lack.

Postfix Postfix (www.postfix.org; **postfix** package) is an alternative MTA. Postfix is fast and easy to administer but is compatible enough with **sendmail** to not upset **sendmail** users. Postfix has a good reputation for ease of use and security and is a drop-in replacement for **sendmail**.

Qmail Qmail (www.qmail.org) is a direct competitor of Postfix and has the same objectives. By default, Qmail stores email using the **maildir** format as opposed to the **mbox** format that other MTAs use (page 735).

MORE INFORMATION

Web **sendmail**: www.sendmail.org
exim4: www.exim.org, www.exim-new-users.co.uk, wiki.debian.org/PkgExim4
procmail: www.procmail.org
IMAP and POP3: www.dovecot.org, cyrusimap.org
SpamAssassin: spamassassin.apache.org, wiki.apache.org/spamassassin
Spam database: razor.sourceforge.net
Mailman: www.list.org
SquirrelMail: www.squirrelmail.org
Dovecot: www.dovecot.org
Postfix: www.postfix.org
Qmail: www.qmail.org

Local **exim4**: **/usr/share/doc/exim4*/***
Dovecot: **/usr/share/doc/dovecot***
man pages: **sendmail**, aliases, makemap, **exim4 exim4_files** update-exim4.conf update-exim4defaults spamassassin spamc **spamd**
SpamAssassin: **/usr/share/doc/spamassassin***, install the **perl** (page 1059) and **spamassassin** packages and give the following command:

```
$ perldoc Mail::SpamAssassin::Conf
```

SETTING UP A sendmail MAIL SERVER

This section explains how to set up a **sendmail** mail server. Under RHEL, before you start **sendmail**, you must stop the Postfix **master** daemon and run chkconfig so the **master** daemon does not start when the system enters multiuser mode:

```
# service postfix stop
Shutting down postfix:                                     [  OK  ]
# chkconfig postfix off
```

PREREQUISITES

Install the following packages:

- **sendmail**
- **sendmail-cf** (required to configure **sendmail**)

sendmail init script

As installed, **sendmail** is set up to run when the system enters multiuser mode. Give the following command to cause **sendmail** to reread its configuration files:

```
# service sendmail restart
Restarting sendmail (via systemctl):                       [  OK  ]
```

NOTES

Firewall

An SMTP server normally uses TCP port 25. If an SMTP server system that receives nonlocal mail is running a firewall, you need to open this port. Using the Fedora/RHEL graphical firewall tool (page 893), select **Mail (SMTP)** from the Trusted Services frame 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.

JUMPSTART I: CONFIGURING sendmail ON A CLIENT

You might not need to configure sendmail to send email

tip

With **sendmail** running, give the command described under "Test" on page 734. 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 (a smarthost or 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.

Change **sendmail.mc**

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

The **dnl** (page 739) 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')dnl
```

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:

```
# 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 an SMTP relay (smarthost).
- 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 742 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 853) points at the local system. If you are not running a DNS server, you must ask your ISP to set up an MX record.

WORKING WITH sendmail MESSAGES

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 envelope 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 **xf**. Once an email has been sent successfully, **sendmail** removes all files pertaining to that email from the **mqueue** directory.

Mail addressed to the local system

By default, **sendmail** delivers email addressed to the local system to 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.

Mail addressed to nonlocal systems

The scheme that **sendmail** uses to process email addressed to a nonlocal system depends on how it is configured: It can send the email to a smarthost, it can send the email to the system pointed to by the DNS MX record of the domain the email is addressed to, or it can refuse to send the email.

mbox versus **maildir**

The **mbox** format holds all messages for a user in a single file. To prevent corruption, a process must lock this file while it is adding messages to or deleting messages from the file; thus the MUA cannot delete a message at the same time the MTA is adding messages. A competing format, **maildir**, holds each message in a separate file. This format does not use locks, allowing an MUA to delete messages from a user at the same time as mail is delivered to the same user. 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 732), an alternative to **sendmail**, supports the **maildir** format only.

MAIL LOGS

The **sendmail** daemon stores log messages in **/var/log/maillog**. Other mail servers, such as Dovecot, might also log information to this file. Following is a sample log entry:

/var/log/maillog

```
# cat /var/log/maillog
...
Dec 14 15:37:27 plum sendmail[4466]: oBENbP2Q004464:
to=<mgs@sobell.com>, ctladdr=<sams@example.com> (500/500),
delay=00:00:02, xdelay=00:00:02, mailer=relay, pri=120289,
relay=smtp.gmail.com [72.14.213.109], dsn=2.0.0, stat=Sent (OK
id=1PSeQx-0001uM-3e)
```

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 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.

See the **sendmail** documentation for more information on log files. If you send and receive a lot of email, the **maillog** file can grow quite large. The logrotate (page 621) **syslog** entry archives and rotates these files regularly.

ALIASES AND FORWARDING

You can use the **aliases**, **~/.forward** (page 737), and **virtusertable** (page 743) files to forward email. Table 20-1 on page 743 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 real user on the specified system. Sometimes, however, you might 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 might receive this email at different times or the email might go to a group of users. You can use the **/etc/aliases** file to map local addresses and classes to local users, files, commands, and local as well as to nonlocal addresses.

Each line in **/etc/aliases** contains the name of a local (pseudo)user, 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

```
adm:        root
```

Sending messages to the **root** account is a good way to make them easy to review. However, because **root**'s email is rarely checked, you might want to send copies to

a real user. You can set up an alias to forward email to more than one user. The following line forwards mail sent to **abuse** on the local system to **sam** and **max** in addition to **root**:

```
abuse:      root, sam, max
```

It is common to forward mail sent to many special accounts to **root**. If Zach is to read mail sent and forwarded to **root**, you could fix each alias individually. Alternatively, you can simply add the following line *to the end* of the **aliases** file (order does matter in this case). This line forwards to Zach all mail sent and forwarded to **root**:

```
...
root:       zach
```

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, max, zach@example.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 503), where it disappears:

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

You can also send email to standard input of a command by preceding the command with the pipe character (|). This technique is commonly used by mailing list software such as Mailman (page 752). For each list it maintains, Mailman has entries, such as the following one for **painting_class**, in the **aliases** file:

```
painting_class:   "|/var/lib/mailman/mail/mailman post painting_class"
```

newaliases After you edit **/etc/aliases**, you must either run newaliases while you are working with **root** privileges or restart **sendmail** to recreate the **/etc/aliases.db** file that **sendmail** reads.

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

```
# praliases | head -5
@:@
abuse:root
adm:root
amanda:root
daemon:root
```

~/.forward Systemwide aliases are useful in many cases, but non**root** users cannot make or change them. Sometimes you might want to forward your own mail: Maybe you want mail from several systems to go to one address, or perhaps you want to forward your mail while you are working at another office. 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 earlier in this section: Destinations are listed one per line and can be a local user, a remote email address, a filename, or a command preceded by the 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 **example.com**:

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

RELATED PROGRAMS

sendmail The **sendmail** packages include 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 734 for an example. The command **apropos sendmail** displays a list of **sendmail**-related files and utilities.

mailq The mailq utility displays the status of the outgoing mail queue. When there are no messages in the queue, it reports the queue is empty. Unless they are transient, messages in the queue usually indicate a problem with the local or remote MTA configuration or a network problem.

```
# 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:

```
# mailstats
Statistics from Fri Dec 24 16:02:34 2010
M   msgsfr  bytes_from   msgsto    bytes_to  msgsrej msgsdis  Mailer
0        0          0K    17181     103904K        0       0  prog
4   368386    4216614K   136456    1568314K    20616       0  esmtp
9   226151   26101362K   479025   12776528K     4590       0  local
=============================================================
T   594537   30317976K   632662   14448746K    25206       0
C   694638               499700               146185
```

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 and seventh columns display the number of messages rejected and discarded 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: When you run the **sendmail** init script (page 733), it automatically generates these files.

THE sendmail.mc AND sendmail.cf FILES

The **sendmail.cf** file is not intended to be edited 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 might generate when processing this file.

The **include** statement tells m4 where to find the macro definition file 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 might 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 ('). Make sure not to change these characters when you edit the file or **sendmail** will not work properly.

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 1175) the message.

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

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

The MASQUERADE_AS statement causes email you send from the local system to appear to come from the specified domain (**mydomain.com** in the commented-out line). Remove the leading **dnl** and change **mydomain.com** to the domain name 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 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 it cannot resolve (and that might 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 853) 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.

blank

27

Programming the Bourne Again Shell

In This Chapter

Objectives

After reading this chapter you should be able to:

- Use control structures to add decision making and repetition in scripts
- Handle input and output of scripts
- Use local and environment variables
- Evaluate the value of variables
- Use bash builtin commands to call other scripts inline, trap signals, and kill processes
- Use arithmetic and logical expressions
- List standard programming practices that result in well written scripts

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. Although you might make use of shell programming as a system administrator, you do not have to read this chapter to perform system administration tasks. Feel free to skip this chapter and come back to it if and when you like.

The first part of this chapter covers programming control structures, also called 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 a shell script to accept user input. The section on the exec builtin demonstrates how to use exec to execute a command efficiently by replacing a process and explains how to use exec 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 the signal 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 1031 lists some of the more commonly used builtins.

Next the chapter examines arithmetic and logical expressions as well as 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 might run either 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. Return to them when you feel comfortable with the basic concepts.

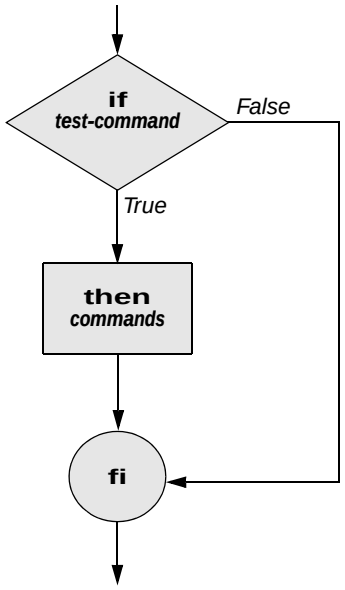

Figure 27-1 An **if...then** flowchart

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
```

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 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 typically use the builtin version if it is available and the utility if it is not. Each version of a command might vary slightly from one shell to the next and from the utility to any of the shell builtins. See page 1018 for more information on shell builtins.

Checking arguments

The next program uses an **if** structure at the beginning of a script to confirm that you have supplied at least one argument on the command line. The test **–eq** operator compares two integers; the **$#** special parameter (page 1013) 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. Some scripts simply test whether arguments exist (as in **chkargs**). Other scripts test for a specific number or specific kinds of arguments.

You can use test to verify 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**) checks 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 using test options; see Table 27-1.

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 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 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. This 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 whitespace (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 argument...
$ ./chkargs2 abc
Program running.
```

Usage messages

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 285). After issuing the usage message, **chkargs2** exits with an exit status of 1, indicating 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 wrong kind of arguments, it will often display a usage message. Following is the usage message that cp displays when you call it without any arguments:

```
$ cp
cp: missing file operand
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
```

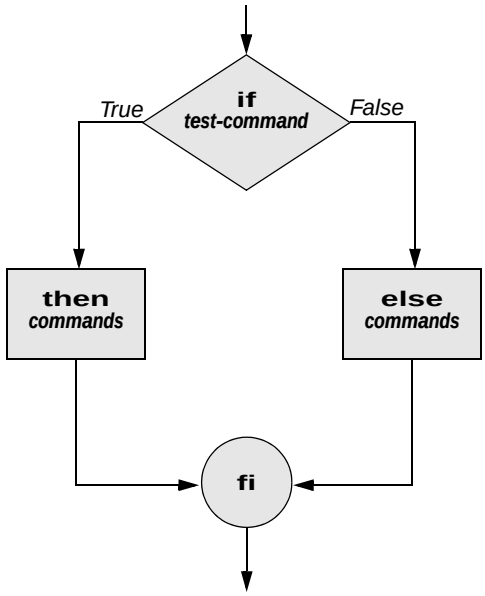

Figure 27-2 An **if...then...else** flowchart

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 [page 292].) Some people prefer this notation for aesthetic reasons; others like it because it saves space.

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

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 **out** script 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 149) to display the files one screen 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* (the first argument is **–v**), **out** uses the shift builtin (page 1014) to shift the arguments to get rid of the **–v** and displays the files using less. If the result of the test is *false* (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 1028) 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 you create while experimenting with the **--** argument.

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 enables 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

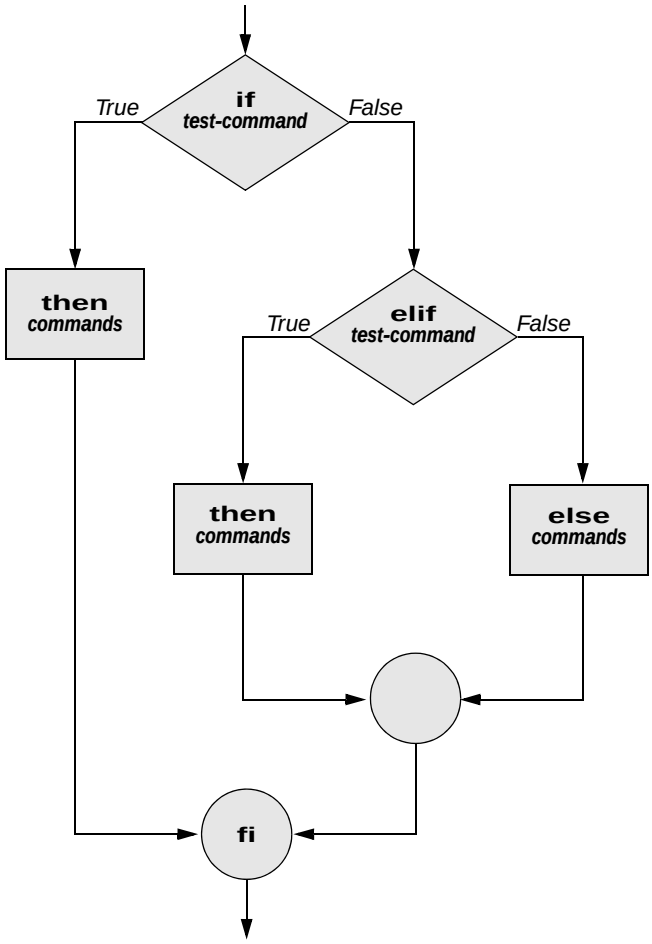

Figure 27-3 An **if…then…elif** flowchart

AND operator (**–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, **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 the directory hierarchy rooted at that directory. 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
```

Max 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, Max calls **lnks** from his home directory to perform the search. The second argument to **lnks**, **/home**, is the pathname of the directory where he wants to start the search. The **lnks** script reports that **/home/max/letter** and **/home/zach/draft** are links to the same file:

```
$ ./lnks letter /home
lnks: using find to search for links...
/home/max/letter
/home/zach/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 290) 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 a script running a different shell.

Comments

The second and third lines of **lnks** are comments; the shell ignores text that follows a hashmark (**#**) up to the next NEWLINE character. These comments in **lnks** briefly identify what the file does and explain 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 *true* 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 is 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 **directory** variable 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 **$file** does not exist. This test keeps **lnks** from wasting time looking for links to a non-existent file. The test builtin, when called with the three arguments **!**, **–f**, and **$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 351) 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 1, **lnks** goes on to identify the *inode* (page 1169) for **$file**. As explained on page 216, 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 the directory hierarchy rooted at the directory specified by its first argument (**$directory**) for files that meet the criteria specified by the remaining arguments. In this example, the remaining arguments send the names of files having inodes matching **$inode** to standard output. Because files in different filesystems can have the same inode number yet not be linked, find must search only directories in the same filesystem as **$directory**. The **–xdev** (cross-device) argument prevents find from searching directories on other filesystems. Refer to page 213 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 can take 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 all scripts display commands as they are run or use the **–x** option to affect only the shell 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 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='>>>> '
```

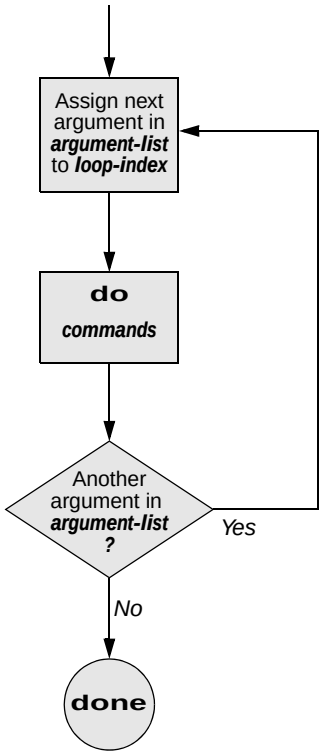

Figure 27-4 A **for...in** flowchart

You can also set the **–x** option of the shell running the script by putting the following set command near the beginning 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
```

The **for...in** structure (Figure 27-4) 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*. It then 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 through the files in the working directory and using test to determine which are directory files:

```
$ 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**.

f