Praise for Previous Editions of *A Practical Guide to Ubuntu Linux®*

"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

"This entire book is a real boon to any neophyte who does not have a solid handle on getting their own answers. That group is the one that I think will benefit the most from *A Practical Guide to Ubuntu Linux®*. Random access is easy, but reading cover to cover would also give one a nice foundational understanding of getting the most out of their machine and even enough guidance to get their feet wet in the sysadmin world. Anyone thrown into owning an Ubuntu server may find this to be a handy lifeline."

—*JR Peck*
Editor
GeekBook.org

"Very well thought out and simplified. [I] would buy another book from this author (Mark Sobell)."

—*Greg Dye*
Electronic Tech

"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 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

"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

"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

"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

"Mark G. Sobell's freshly revised reference work on Ubuntu Linux may be the most impressive computer book I've seen in the last 10 years. If you are currently stranded with a pile of abandoned computers on a desert isle, I'm telling you, this is the book."

—*From a review at DesktopLinux.com*
http://www.desktoplinux.com/news/NS8801274918.html

Praise for Other Books by Mark G. Sobell

"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 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

Excerpts of Chapters From

A Practical Guide to Ubuntu Linux®

THIRD EDITION

Mark G. Sobell

ISBN 978-0-13-254248-7

Copyright © 2010 Mark G. Sobell

PRENTICE
HALL

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

EXCERPT

Chapter 2 covered planning the installation of Ubuntu Linux: 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 downloading and burning CD/DVD ISO images; and collecting information about the system. This chapter focuses on installing Ubuntu. Frequently the installation is quite simple, especially if you have done a good job of planning. Sometimes you may run into a problem or have a special circumstance; this chapter gives you tools to use in these cases. Read as much of this chapter as you need to; once you have installed Ubuntu Linux, continue with Chapter 4, which covers getting started using the Ubuntu desktop. If you install a textual (command-line) system, continue with Chapter 5.

In This Chapter

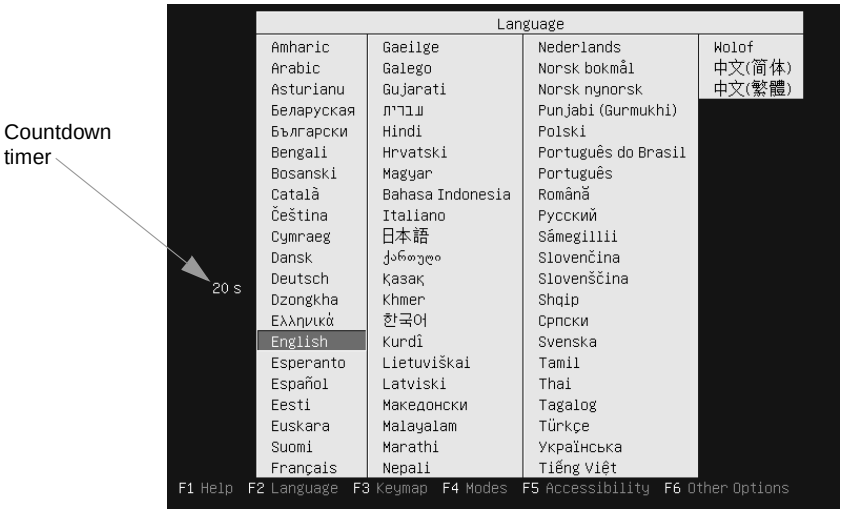

Figure 3-1 The language overlay with the countdown timer at the left

BOOTING FROM A LIVE/INSTALL DESKTOP CD OR A LIVE/INSTALL DVD

Ubuntu is commonly installed from a live/install Desktop CD or a live/install DVD. You can also start a live session from these disks. This section describes how to perform both tasks using each of these disks. Follow the "Basic Instructions" (on the next page) if you want to get going quickly. See "Detailed Instructions" on page 53 if you run into problems using the basic instructions, if you want to customize your installation, or if you simply want to know more about what is going on when you install Ubuntu.

LIVE SESSION

The live/install Desktop CD and the live/install DVD give you a chance to preview Ubuntu without installing it. Boot from either disk as explained in this section to begin a live session and work with Ubuntu as explained in Chapter 4. When you are finished, remove the CD/DVD and reboot the system; the system boots as it did before the live session.

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

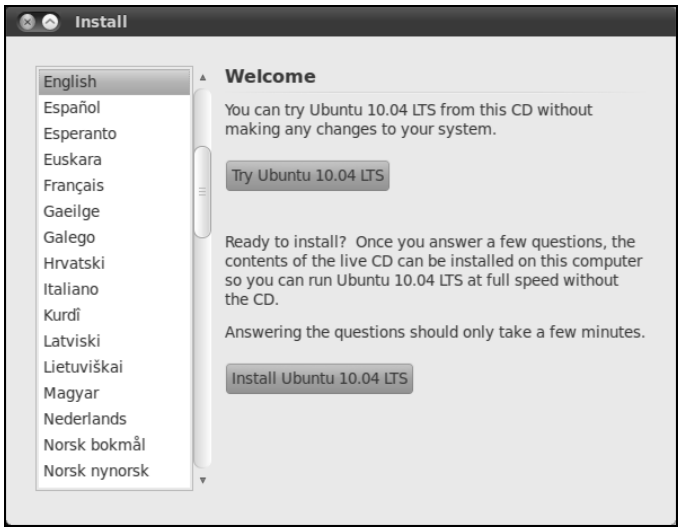

Figure 3-2 The Welcome screen of the Install window

BASIC INSTRUCTIONS

DVD Boot the system from the live/install DVD and do not press any keys. A short time after the countdown timer at the left of the screen (under the language overlay in Figure 3-1) goes from 30 to 0, Ubuntu displays the Welcome screen (Figure 3-2). Use the mouse to highlight the language you want to use (from the list on the left) and then click **Try Ubuntu 10.04** to start a live session or click **Install Ubuntu 10.04** to install Ubuntu.

CD Boot the system from the live/install Desktop CD and do not press any keys to bring up a live system running in English.

DETAILED INSTRUCTIONS

Booting the system Before Ubuntu can display a desktop or install itself on a hard disk from a live/install Desktop CD or a live/install DVD, the Ubuntu 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).

To boot from a live/install Desktop CD or a live/install DVD, insert the disk in the computer and turn on or reset the system. What Ubuntu displays depends on which of the two disks you use. The ubiquity installer handles all installation tasks. Much of this chapter describes how to use ubiquity.

Installation media This book covers four disks: the live/install DVD, the live/install Desktop CD, the Server CD, and the Alternate CD. This section covers basic operations with the first two of these disks. See "Advanced Installation" on page 77 for more advanced operations and descriptions of the Server CD and the Alternate CD.

Figure 3-3 The live/install DVD menu screen

Language overlay

The language overlay, shown in Figure 3-1 on page 52, allows you to choose a language to work with (the default language). The default language is the one a live system displays, the one the installer uses when you install Ubuntu, and the language Ubuntu displays by default once it is installed. As you install Ubuntu, you can change the default language from the Welcome screen as explained on page 57.

All four disks can display the language overlay. However, the live/install Desktop CD displays the language overlay only when you interrupt the boot process by pressing a key and the live/install DVD removes the language overlay unless you interrupt the boot process by pressing a key. The following sections go into detail about these differences.

CD and DVD menus

Each of the installation media displays a unique menu. Most of the menu selections are a subset of the selections available on the live/install DVD menu (Figure 3-3). The following sections describe three selections available on the live/install Desktop CD and the live/install DVD—those that allow you to bring up a live Ubuntu session, install Ubuntu on the hard disk, and test the installation medium for defects. See "Advanced Installation" on page 77 for information about more advanced operations and descriptions of the Server CD and the Alternate CD.

The Ubuntu logo and progress dots

As the system boots, it displays an Ubuntu logo and progress dots that turn on and off in sequence to indicate the system is working. Refer to "Seeing What Is Going On" on page 57 if you want to display system messages in place of the logo and progress dots.

Problems

Refer to "BIOS setup" on page 28 if the system does not boot from the CD/DVD. If you encounter problems with the display while you are booting a live session or

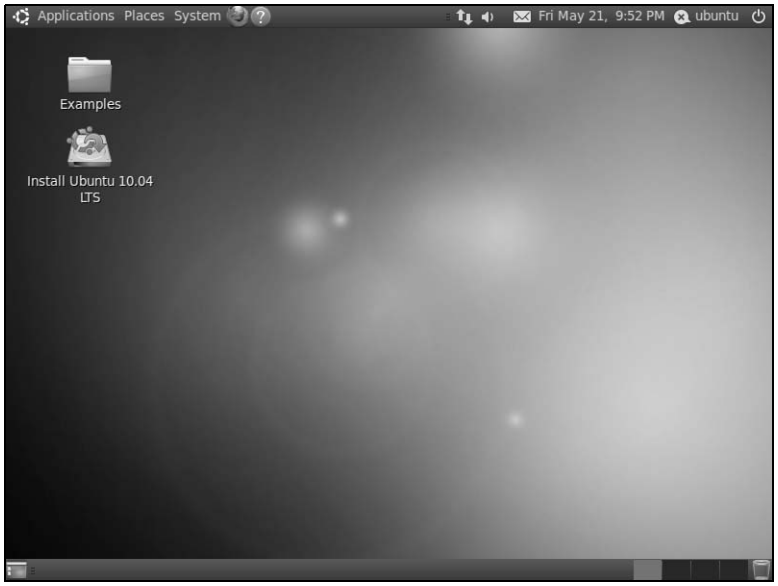

Figure 3-4 The GNOME desktop displayed by a live session

installing Ubuntu, reboot the system, perform an advanced installation (page 77), and use the F4 Modes menu to set the **nomodeset** parameter (Figure 3-22, page 81). If that tactic does not work, install Ubuntu using the textual installer on the DVD.

THE LIVE/INSTALL DVD

When you boot from a live/install DVD (and not from a CD), Ubuntu displays a language overlay over the DVD menu screen (Figure 3-1, page 52). A timer that counts down from 30 seconds appears to the left of the overlay (**20 s** appears in the figure). As the timer counts down, whether you press a key determines what happens next.

Bring up a live session

If you do not press a key, after the timer counts down to zero, Ubuntu displays a logo and progress dots and then brings up a live session running a GNOME desktop (Figure 3-4) running in English.

Install Ubuntu

If you press any key before the timer reaches zero, the system stops its countdown with the language overlay still displayed. You can then use the ARROW keys to highlight a language for the installer or live system to use and press RETURN to select the highlighted language and expose the DVD menu screen (Figure 3-3). On this screen you can use the ARROW keys to highlight a selection and press RETURN to make the selection. The first three selections bring up a live system, install Ubuntu on the hard disk, and check the DVD for defects. For more information refer to "Advanced Installation" on page 77.

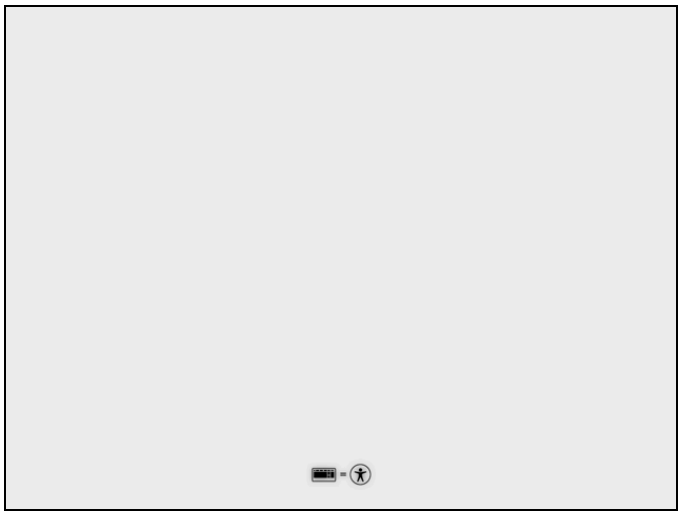

Figure 3-5 The symbols on the initial boot screen

THE LIVE/INSTALL DESKTOP CD

When you boot from a live/install desktop CD, Ubuntu displays the initial boot screen, a mostly blank screen with keyboard layout and accessibility symbols at the bottom (Figure 3-5). While the initial boot screen is displayed, whether you press a key determines what happens next. If you do not press a key, after a few seconds Ubuntu displays a logo and progress dots and then displays the Welcome screen of the Install window (Figure 3-2). To select a default language other than English from this screen, see "Changing the default language" on page 57.

Bring up a live session

From the Welcome screen, click **Try Ubuntu 10.04** to bring up a live session running a GNOME desktop (Figure 3-4).

Check the CD/DVD for defects

tip

Testing the CD/DVD takes a few minutes but can save you much aggravation if the installation fails or you run into problems after installing Ubuntu due to bad media. Whether you burned your own CD/DVD, purchased it, or are using the disk included with this book, it is a good idea to verify that the contents of the CD/DVD is correct.

With the DVD menu screen or one of the CD menu screens displayed, use the ARROW keys to highlight **Check disc for Defects** and press RETURN. Checking the CD/DVD takes a few minutes—Ubuntu keeps you apprised of its progress. When Ubuntu finishes checking the CD/DVD, it displays the result of its testing. Press RETURN to reboot the system.

Install Ubuntu

From the Welcome screen, click **Install Ubuntu 10.04** to install Ubuntu on the hard disk; continue with the Where are you? screen as described on page 59.

If you press a key while the initial boot screen is displayed, Ubuntu displays the language overlay covering the Desktop CD menu. This screen looks similar to Figure 3-1 on page 52 except that no countdown timer is visible because the system is not counting down; instead it is waiting for your input. You can use the ARROW

keys to highlight a language for the installer to use and press RETURN to select the language and expose the CD menu screen (similar to Figure 3-3 on page 54). On this screen you can use the ARROW keys to highlight a selection and press RETURN to make the selection. For more information refer to "Advanced Installation" on page 77.

THE WELCOME SCREEN

Two varieties of the Welcome screen exist. One screen, shown in Figure 3-2, allows you to choose between bringing up a live Ubuntu system and installing Ubuntu on the hard disk. It has two buttons: **Try Ubuntu 10.04** and **Install Ubuntu 10.04**. The other screen, which is similar to the one shown in Figure 3-2, simply marks the start of the installation process. It has three buttons: **Quit**, **Back** (grayed out and nonfunctional because you cannot go back from this screen), and **Forward**.

Quit button

When you click **Quit**, Ubuntu displays a GNOME desktop running under a live session (Figure 3-4, page 55).

Changing the default language

Along the left side of both Welcome screens is a *list box* (page 1157) that holds a list of languages. The highlighted language is the language the live session or the installer/installed system will use. If the highlighted language is not the language you want, use the ARROW keys or the mouse to highlight your desired language before proceeding. See "The Function Keys" on page 79 for information about changing the language, keyboard layout, and accessibility features used by a live session and the installer/installed system.

optional

SEEING WHAT IS GOING ON

If you are curious and want to see what Ubuntu is doing as it boots, perform an advanced installation (page 77) and remove **quiet** and **splash** from the boot command line (Figure 3-22, page 81): With the DVD menu screen or one of the CD menu screens displayed, press F6 to display the boot command line and a drop-down list. Next press ESCAPE to close the drop-down list. Then press BACKSPACE or DEL to back up and erase **quiet** and **splash** from the boot command line. If you have not added anything to this line, you can remove the two hyphens at the end of the line. If you have added to this line, use the LEFT ARROW key to back up over—but not remove—whatever you added, the hyphens, and the SPACE on each side of them. Then remove **quiet** and **splash**. Press RETURN. Now, as Ubuntu boots, it displays information about what it is doing. Text scrolls on the screen, although sometimes too rapidly to read.

ubiquity: INSTALLING UBUNTU GRAPHICALLY

This section covers the ubiquity graphical installer, written mostly in Python, that installs Ubuntu. You can also install Ubuntu using the textual installer (debian-installer; page 85).

Before you start, see what is on the hard disk

tip

Unless you are certain you are working with a new disk, or you are sure the data on the disk is of no value, it is a good idea to see what is on the hard disk before you start installing Ubuntu. You can use the palimpsest disk utility to mount partitions on a hard disk. You can then examine the files in these partitions and see what is on the disk. See page 66 for more information on palimpsest.

USING THE MOUSE TO WORK WITH THE INSTALL WINDOW SCREENS

You can use either the mouse or the keyboard to make selections from the Install window screens. To select a language from the Welcome screen using the mouse, left-click the language you want to use in the list box at the left. If the language you want does not appear on the displayed portion of the list, click or drag the scrollbar (Figure 3-2 on page 53 and Figure 4-16 on page 123) to display more languages; then click the language you want to use. Ubuntu highlights the language you click. Once you select a language, you are finished working with the Welcome screen. Click the button labeled **Forward** or **Install Ubuntu 10.04** to display the next screen.

USING THE KEYBOARD TO WORK WITH THE INSTALL WINDOW SCREENS

To use the keyboard to make selections, first use the TAB key to move the highlight to the object you want to work with. On the Welcome screen, the objects are the selected item in the list box and the buttons labeled or **Install Ubuntu 10.04** or **Quit**, **Back**, and **Forward**.

List box With a language in the list box highlighted, use the UP ARROW and DOWN ARROW keys to move the highlight to the language you want to use. The list scrolls automatically when you move the highlight to the next, undisplayed entry in the list.

Button Once you select a language, you are finished working with the Welcome screen. Use the TAB key to highlight the button labeled **Forward** or the button labeled **Install Ubuntu 10.04**. The button turns orange with an orange border when it is highlighted. Press RETURN to display the next screen.

Drop-down list To make a selection from a drop-down list, such as the one in the box labeled **Region** shown in Figure 2-1 on page 31, use the TAB key to highlight the box and then use the ARROW keys to move the highlight from one item to the next. With the selection you want to choose highlighted, press RETURN.

STARTING THE INSTALLATION

This book describes using the mouse to make selections from a graphical interface; you can use the keyboard if you prefer.

WELCOME SCREEN

The Welcome screen of the Install window (Figure 3-2) contains a welcome message and a list of languages for you to choose from. The language you choose will be the one ubiquity uses as you install the system and the default language for the installed system; you can change this default once the system is installed (page 145). Click **Forward**.

Ubuntu displays the Setting up the clock window and, if it can connect to a network time server, sets the clock. You can click **Skip** to bypass this step.

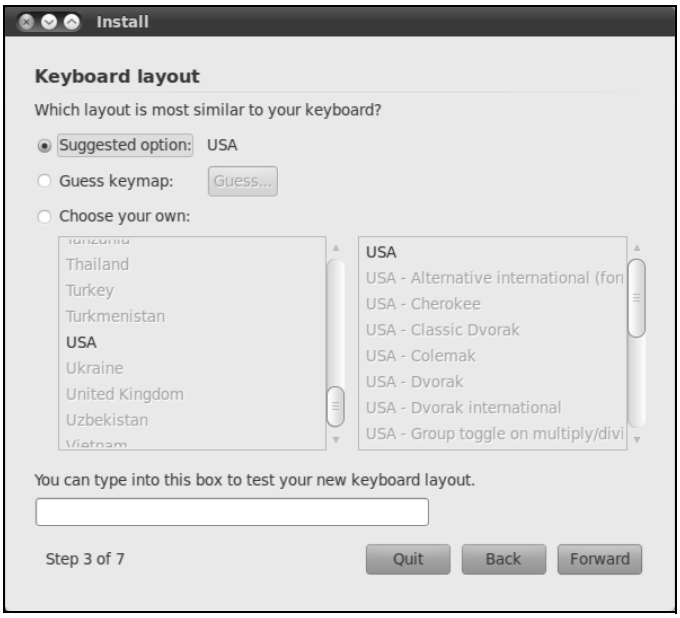

Figure 3-6 The Keyboard layout screen

WHERE ARE YOU?

As the first step in installing Ubuntu, ubiquity displays the Where are you? screen. This screen allows you to specify the time zone where the computer is located. You can use the map or the drop-down lists labeled **Region** and **Time Zone** to specify the time zone. When you click the name of a city on the map, the appropriate region appears in the box labeled **Region** and the name of the time zone or a city within the time zone appears in the box labeled **Time Zone**.

To use the **Region** drop-down list, click the down arrow at the right end of the box labeled **Region**; ubiquity expands the box into a list of parts of the world. Click the region you want to select. Now, repeat this process with the box labeled **Time Zone**. Click **Forward**.

KEYBOARD LAYOUT

The Keyboard layout screen (Figure 3-6) allows you to specify the type of keyboard to be used by the installed system. (See "F3 Keymap" on page 79 to change the layout of the keyboard ubiquity uses during installation.) When ubiquity displays the Keyboard layout screen, the *radio button* (page 1167) labeled **Suggested option** is selected and the name of a keyboard layout appears to the right of these words. If the suggested option is acceptable, click **Forward**.

Anytime the Keyboard layout screen is displayed, you can highlight the text box at the bottom of the screen and type some letters to see if the selected option is correct for the keyboard you are using.

When you select the radio button labeled **Guess keymap** and click **Guess**, ubiquity leads you through a series of questions and, based on your answers, tries to determine which type of keyboard you are using. Click **Forward** when you are satisfied with the result.

When you select the radio button labeled **Choose your own**, ubiquity activates the two list boxes below these words. Select a country and keyboard type from these list boxes and click **Forward**.

PREPARE DISK SPACE

The Prepare disk space screen controls how ubiquity partitions the hard disk. See page 36 for a discussion of the issues involved in partitioning a hard disk.

GUIDED PARTITIONING

With a single, clean hard disk—a hard disk with nothing installed on it, as it comes from the factory (i.e., no partition table)—the ubiquity partition editor displays a Prepare disk space screen similar to the one shown in Figure 3-7. In this case, the simplest way to partition the disk is to allow the ubiquity partitioner to do it for you. This technique is called *guided* partitioning. By default, the radio button labeled **Erase and use the entire disk** is selected and the name of the only hard disk in the system is displayed in the drop-down list below these words. If the system has two or more hard disks, you must select from this list the disk where you want to install Ubuntu. Click **Forward**. The ubiquity partition editor creates two partitions on the hard disk: a small swap partition (page 37) and a root partition (**/**, page 37) that occupies the rest of the disk.

The ubiquity partition editor does not partition the disk at this time. At any time before you click **Install** on the Ready to install screen, you can change your mind about how you want to partition the disk. Click the button labeled **Back**. You may have to back up through several screens to display the Prepare disk space screen again, but you can then set up the disk the way you want it.

See "Advanced Guided Partitioning" on page 70 for information on using the other selections in the Prepare disk space screen.

MIGRATE DOCUMENTS AND SETTINGS

If you are installing Ubuntu on a system that already has one or more operating systems installed on it, and you are not overwriting those operating systems, the Migrate documents and settings screen displays a list of accounts and settings from the existing operating systems. For example, if you are creating a dual-boot system on a system that already has Windows installed on it, this screen shows the accounts from the Windows system and a list of programs and settings. It might show your name from the Windows system and, under that, Internet Explorer and My Documents. Put ticks in the check boxes adjacent to those items you want to migrate to the Ubuntu system. On the lower portion of the screen, enter the information necessary to create an Ubuntu user to receive the migrated information. Click **Forward**.

Figure 3-7 The ubiquity partition editor showing one empty hard disk

WHO ARE YOU?

The Who are you? screen (Figure 3-8, next page) sets up the first Ubuntu user. This user can use sudo (page 98) to administer the system, including setting up additional users (page 594). Enter the full name of the user in the text box labeled **What is your name?**. As you type, ubiquity enters the first name from the name you are entering in the box labeled **What name do you want to use to log in?**. Press TAB to move the cursor to this box. If you want to use a different username, press BACKSPACE (page 151) to erase the username and enter a new one. Press TAB.

Enter the same password in the two (adjacent) boxes labeled **Choose a password to keep your account safe**. The strength of the password is displayed to the right of the password boxes. Although ubiquity accepts any password, it is a good idea to choose a stronger (more secure) password if the system is connected to the Internet. See "Changing Your Password" on page 148 for a discussion of password security.

The final text box specifies the name of the computer. For use on a local network and to connect to the Internet with a Web browser or other client, you can use a simple name such as **fox8**. If you are setting up a server system, see "FQDN" on page 823 for information on names that are valid on the Internet.

The three radio buttons at the bottom of the window configure the login process for the user you are specifying. Select **Require my password to log in** to cause Ubuntu to require a password for you log in on the system.

Select **Require my password to log in and to decrypt my home folder** if you are setting up an encrypted home folder.

Figure 3-8 The Install window, Who are you? screen

Select **Log in automatically** if you want Ubuntu to log you in automatically when the system boots—select this option only if you trust everyone who has physical access to the system. Click **Forward**.

READY TO INSTALL

The final screen ubiquity displays is the Ready to install screen (Figure 3-9). At this point, the ubiquity partition editor has not yet written to the disk. Thus, if you click **Quit** at this point, the hard disk will remain untouched. This screen summarizes your answers to the questions ubiquity asked in the previous screens. Click **Advanced** to display the Advanced Options window, which allows you to choose whether to install a boot loader (normally you want to) and whether to set up a network proxy (page 405). Click **OK** to close the Advanced Options window. If everything looks right in the summary, click **Install**. The installer begins installing Ubuntu on the hard disk.

When ubiquity writes to the hard disk

caution You can abort the installation by clicking the **Quit** button at any point, up to and including when the Ready to install screen (Figure 3-9) is displayed, without making any changes to the hard disk. Once you click **Install** in this screen, ubiquity writes to the hard disk.

The ubiquity installer displays messages to keep you informed of its progress. When the new system is installed, Ubuntu displays the Installation Complete window, which gives you the choice of continuing the live session (**Continue Testing**) or

Figure 3-9 The Install window, Ready to install screen

rebooting the system so you can use the newly installed copy of Ubuntu. Click **Restart Now** to reboot the system.

The installer displays the Ubuntu logo and progress dots. When it has finished shutting down the system, it asks you to remove the disk (so you do not reboot from the CD/DVD) and press RETURN. After you complete these steps, Ubuntu reboots the system and displays the Ubuntu GNOME login screen (Figure 4-1, page 100).

Log in as the user you specified on the Who are you? screen and continue with Chapter 4.

GRAPHICAL PARTITION EDITORS

A partition editor displays and can add, delete, and modify partitions on a hard disk. This section describes three graphical partition editors you can use to configure a hard disk in the process of installing Ubuntu. The gparted and palimpsest partition editors are available from a live session. The other partition editor is part of the ubiquity installer and is not available by itself. See page 87 for information on using the textual partition editor, which is available when you use the textual installer. After you install Ubuntu Linux, you can use parted (page 611) or palimpsest (page 66) to view and manipulate partitions. The gparted partition editor is not

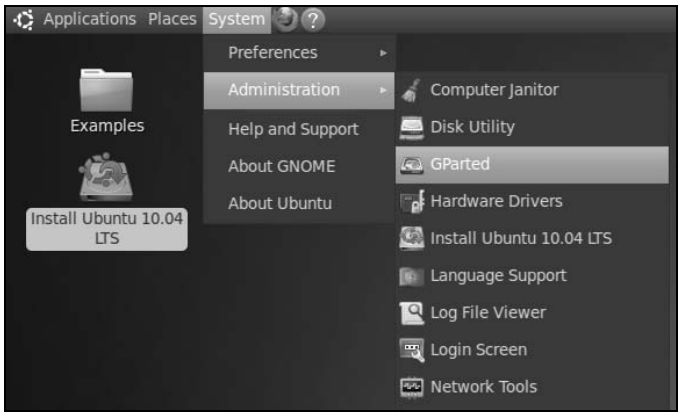

Figure 3-10 Selecting gparted from the Main menu

available from an installed system unless you install the **gparted** package (page 519). If you want a basic set of partitions, you can allow ubiquity to partition the hard disk automatically using guided partitioning.

See "Setting Up the Hard Disk" on page 33 for a discussion of free space, partitions, partition tables, and filesystems. "Manual Partitioning: Planning Partitions" on page 37 discusses some of the filesystems for which you may want to set up partitions if you choose to partition the hard disk manually.

Unless you are certain the hard disk you are installing Ubuntu Linux 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 disk before you start the installation. The gparted and palimpsest partition editors, which are available from a live session, are good tools for this job.

gparted: THE GNOME PARTITION EDITOR

Open a GParted window by selecting **Main menu: System⇨Administration⇨ GParted** as shown in Figure 3-10.

The gparted utility displays the layout of the hard disk and can be used to resize partitions, such as when you are setting up a dual-boot system by adding Ubuntu to a Windows system (page 76). Although you can create partitions using gparted, you cannot specify the mount point (page 35) for a partition—this step must wait until you are installing Ubuntu and using the ubiquity partition editor.

AN EMPTY HARD DISK

The gparted utility shows one large unallocated space for a new hard disk (empty, with no partition table). An exclamation point in a triangle is a warning; on a new disk it indicates an unrecognized file system (there is no partition table). If you have more than one hard disk, use the list box in the upper-right corner of the window to select which disk gparted displays information about. Figure 3-11 shows an empty

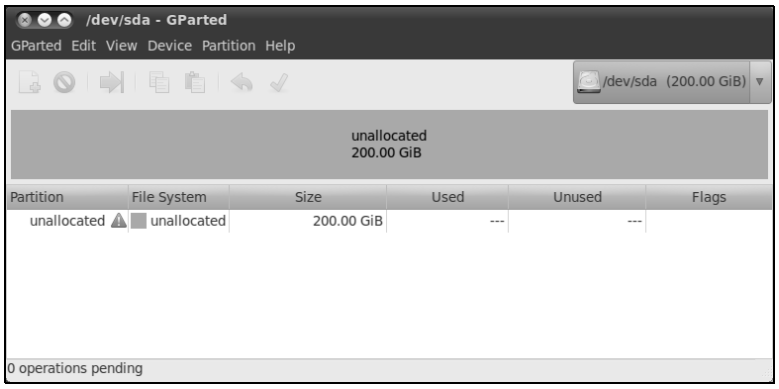

Figure 3-11 The gparted utility displaying an empty disk drive

200-*gibibyte* (page 1150) hard disk on the device named **/dev/sda**. Figure 3-7 on page 61 shows the ubiquity partition editor ready to partition an empty drive similar to the one shown in Figure 3-11.

RESIZING A PARTITION

Although you can resize a partition using the ubiquity partition editor while you are installing Ubuntu, you may find it easier to see what you are doing when you use the gparted partition editor from a live session for this task. This section explains how to use gparted to resize a partition.

Always back up the data on a hard disk

caution If you are installing Ubuntu on a disk that holds important data, back up the data before you start the installation. Things can and do go wrong. The power may go out in the middle of an installation, corrupting the data on the hard disk. There may be a bug in the partitioning software that destroys a filesystem. Although it is unlikely, you might make a mistake and format a partition holding data you want to keep.

Figure 3-12 (next page) shows gparted displaying information about a hard disk with a single partition that occupies the entire disk. This partition holds a single 200-gibibyte NTFS filesystem. The process of resizing a partition is the same regardless of the type of partition, so you can use the following technique to resize Windows, Linux, or other types of partitions.

To install Ubuntu on this system, you must resize (shrink) the partition to make room for Ubuntu. To resize the partition, right-click to highlight the line that describes the partition and click the arrow pointing to a line on the toolbar at the top of the window. The partition editor opens a small Resize/Move window, as shown in Figure 3-12.

At the top of the Resize/Move window is a graphical representation of the partition. Initially the partition occupies the whole disk. The spin box labeled **New Size (MiB)**

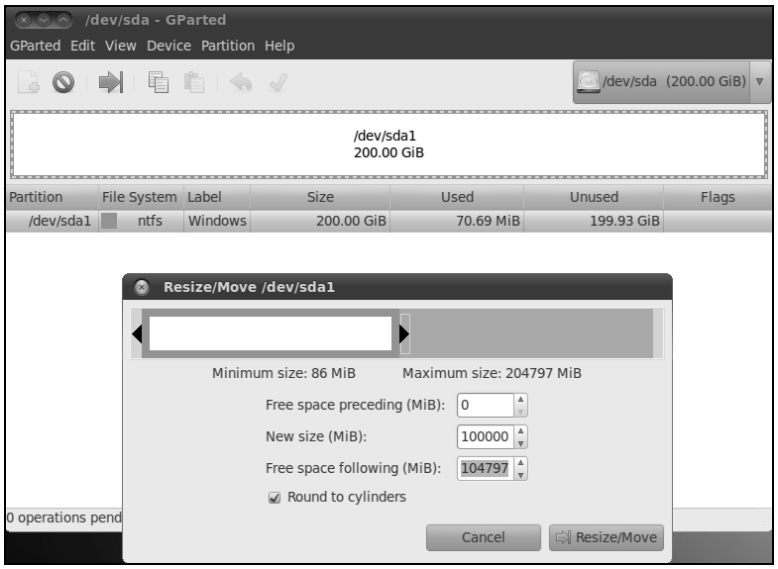

Figure 3-12 The gparted partition editor displaying a disk drive holding a Windows system

shows the number of mebibytes occupied by the partition—in this case, the whole disk. The two spin boxes labeled **Free Space** show no free space.

You can specify how the partition should be resized by (right-clicking and) dragging one of the triangles at the ends of the graphical representation of the partition or by entering the number of mebibytes you want to shrink the Windows partition to in the spin box labeled **New Size (MiB)**. The value in one of the spin boxes labeled **Free Space** increases when you make this change (as shown in Figure 3-12). Click **Resize/Move** to add the resize operation to the list of pending operations at the bottom of the window. Click the green check mark on the toolbar to resize the partition.

DELETING A PARTITION

Before you delete a partition, make sure it does not contain any data you need. To use gparted to delete a partition, highlight the partition you want to delete, click the circle with a line through it, and then click the green check mark on the toolbar.

palimpsest: THE GNOME DISK UTILITY

The palimpsest graphical disk utility can create, remove, and modify partitions and filesystems on many types of media, including internal and external hard disks, CD/DVDs, and USB flash drives. It can encrypt partitions and change passwords on already encrypted partitions.

Open the Palimpsest Disk Utility window by selecting **Main menu: System⇨ Administration⇨Disk Utility** (just above **GParted** in Figure 3-10 on page 64). To

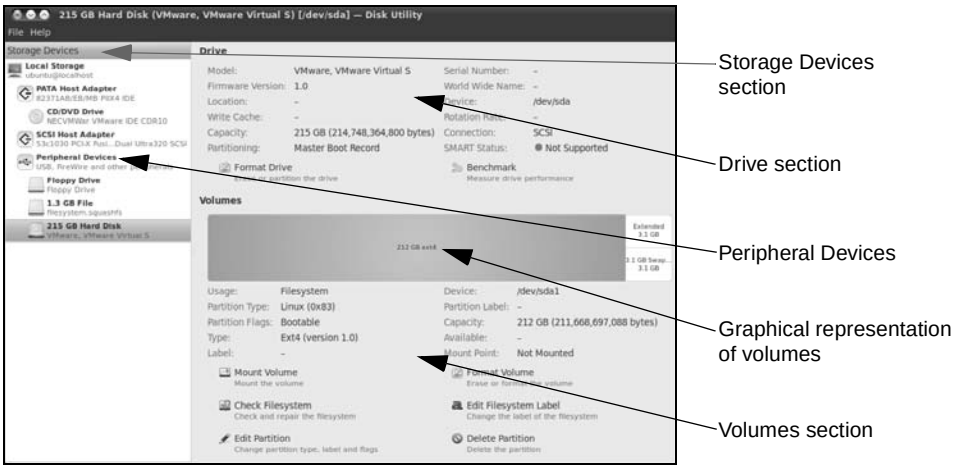

Figure 3-13 The palimpsest Disk Utility window

display information about a hard disk, click a hard disk under Storage Devices/Peripheral Devices on the left side of the window.

With a hard disk selected, the palimpsest Disk Utility window is divided into three sections (Figure 3-13): **Storage Devices** holds a list of CD/DVD drives, hard disks, and other devices; **Drive** holds information about the hard disk that is highlighted in the list of storage devices; and **Volumes** displays information about the partition that is highlighted in the graphical representation of the hard drive.

When you select a hard disk in the Storage Devices section, palimpsest displays information about that disk in the Drive section of the window. Click one of the partitions in the graphical representation of the hard disk and palimpsest displays information about that partition in the Volumes section.

From this window you can view, create, and delete partitions. Although you can create partitions using palimpsest, you cannot specify the mount point (page 35) for a partition—this step must wait until you are installing Ubuntu and using the ubiquity partition editor. You can save time if you use palimpsest to examine a hard disk and ubiquity to set up the partitions you install Ubuntu on.

DISPLAYING THE CONTENTS OF A FILESYSTEM

To display the contents of a filesystem, select the partition holding the filesystem as described above and click **Mount Volume** in the Volumes section of the Disk Utility window. Figure 3-13 shows **Unmount Volume** because the partition is already mounted. When palimpsest mounts the highlighted filesystem, the mounted filesystem appears as a directory (folder) on the desktop. When you click the mount point (the link following **Mount Point: mounted at**) in the Volumes section or double-click the directory icon on the desktop, Nautilus displays the filesystem in a file browser window (page 107). When you have finished examining the contents of the filesystem, click **Unmount Volume** to unmount the filesystem.

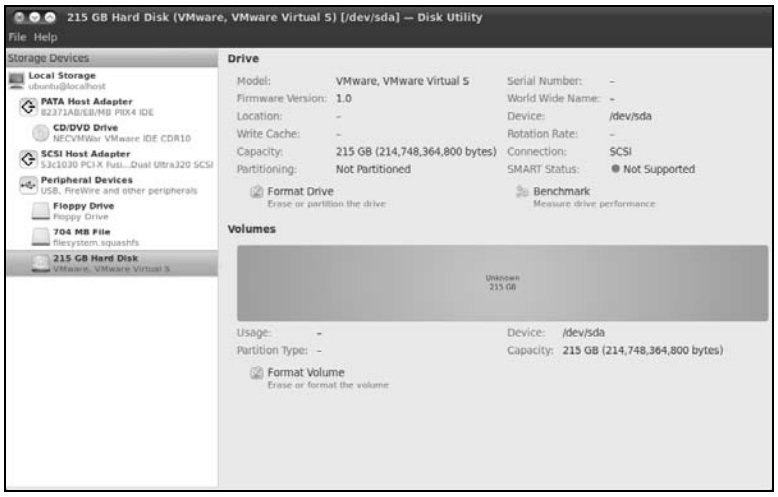

Figure 3-14 The palimpsest Disk Utility showing a disk without a partition table

WRITING A PARTITION TABLE

A new disk does not have a partition table (page 33) and looks similar to the disk highlighted in Figure 3-14. In the Drive section of a Disk Utility window, **Not Partitioned** follows the **Partitioning** label, the graphical representation of the disk is marked **Unknown**, and **Usage** is blank. If the disk you are working with already has a partition table, skip to the next section.

To partition a hard disk, click **Format Drive** in the Drive section of the Disk Utility window: palimpsest opens a Format window holding a drop-down list labeled **Scheme**. Select a scheme. In most cases you will want to accept the default scheme of **Master Boot Record**. Click **Format**. After checking that you really want to format the drive, palimpsest creates the partition table. Now **Master Boot Record** follows the **Partitioning** label, the graphical representation of the disk is marked **Free** (free space; page 33), and **Unallocated Space** follows the **Usage** label.

If you want to create a single filesystem that occupies the entire disk drive, instead of following the instructions in the preceding paragraph, click **Format Volume** in the Volumes section of the Disk Utility window: palimpsest opens a Format whole-disk volume window. To create a filesystem, follow the instructions for the Create partition window in the next section.

CREATING A PARTITION AND A FILESYSTEM

Once you have created a partition table, you will be able to create a partition that holds a filesystem in the free space. When you click **Create Partition**, palimpsest opens a Create partition window (Figure 3-15).

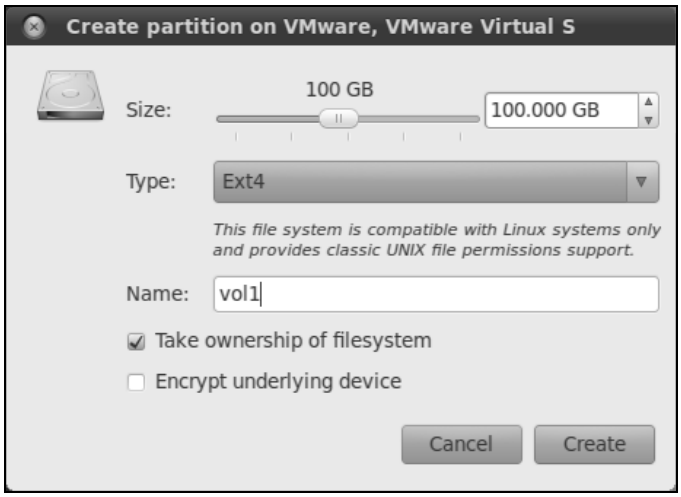

Figure 3-15 The palimpsest Create partition window

In this window, use the slider labeled **Size**, or the adjacent spin box, to specify the size of the new partition. Next specify a filesystem type; **ext4** filesystems are the most common. You can optionally enter a disk label in the text box labeled **Name**. This name is not the mount point for the disk. Typically you will want to own the filesystem, so allow the tick to remain in the check box labeled **Take ownership of file system**. If you want the filesystem to be encrypted, put a tick in the check box labeled **Encrypt underlying device**. Click **Create**. After checking with you, palimpsest creates the filesystem. Now the graphical representation of the disk is divided to represent the division of the hard disk and **Usage** corresponds to the highlighted section of the graphical representation (Filesystem or Unallocated Space). If you did not use all the free space, you can create additional partitions and filesystems in the same manner.

DELETING A PARTITION

Before deleting a partition, make sure it does not contain any data you need. To use the palimpsest utility to delete a partition, highlight the partition you want to delete in the graphical representation of the hard disk and click **Delete Partition**. After checking with you, palimpsest deletes the partition.

USING SMART TO DISPLAY DISK PERFORMANCE INFORMATION

SMART (Self-Monitoring, Analysis, and Reporting Technology) monitors hard disks and attempts to predict hard disk failures. To see a SMART report for a disk on the system, highlight the disk in the Storage Devices section and click **Smart Data** in the Drive section; palimpsest displays a window similar to the one shown in

Figure 3-16 SMART data as displayed by palimpsest

Figure 3-16. From this window you can run various self-tests and scroll through the information at the bottom of the window.

8

Linux GUIs: X and GNOME

EXCERPT

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

In This Chapter

THE NAUTILUS FILE BROWSER WINDOW

"Using Nautilus to Work with Files" on page 107 presented an introduction to using Nautilus. This section discusses the Nautilus File Browser window in more depth.

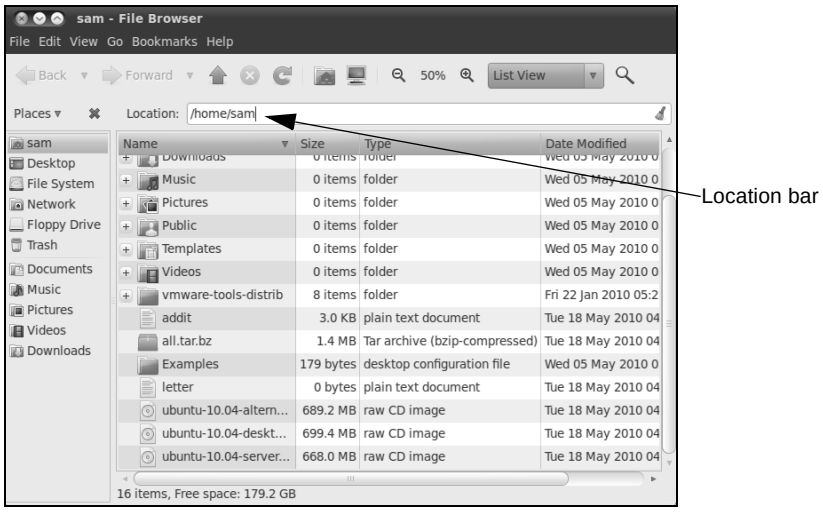

Figure 8-3 A Nautilus File Browser window displaying a List view and a textual location bar

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

THE VIEW PANE

The View pane displays icons or a list of filenames. Select the view you prefer from the drop-down list at the right end of the location bar. Figure 8-2 shows an Icon view and Figure 8-3 shows a List view. A Compact view is also available. Objects in the View pane behave exactly as objects on the desktop do. See the sections starting on page 101 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 124) to cut/copy and paste objects.

Nautilus can open a terminal emulator

tip When you install the **nautilus-open-terminal** package (see page 519 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 204).

THE SIDE PANE

The Side pane augments the information Nautilus displays in the View pane. Press F9 or click the **X** at the top of the Side pane to close it. You can display the Side pane by

pressing F9 or by selecting **File Browser menubar: View⇨Side Pane**. To change the horizontal size of the Side pane, drag the handle (Figure 8-2, page 276) on its right side.

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

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

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

Information Information presents information about the folder displayed by or highlighted in the View pane.

Tree Tree presents an expandable tree view of your home folder and each mounted filesystem. Each directory in the tree has a 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 it in the Side pane and display it in the View pane.

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

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

Emblems Similar to the Emblems tab in the Object Properties window (page 129), Emblems allows you to drag emblems from the Side pane and drop them on objects in the View pane. Drag and drop the **Erase** emblem to erase emblems associated with an object. You cannot erase emblems that Ubuntu places on objects, such as locked and link emblems.

CONTROL BARS

This section discusses the four control bars that initially appear in a File Browser window: the status bar, menubar, Main toolbar, and location bar (Figure 8-2, page 276). From **File Browser menubar: View**, you can choose which of these bars to display—except for the menubar, which Nautilus always displays.

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

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

To change the magnification of the display in the View pane, click the plus or minus sign in a magnifying glass on either side of the magnification percentage. Right-click the magnification percentage itself to return to the default magnification. Left-click the magnification percentage to display a drop-down list of magnifications. Click the button to the right of the right-hand magnifying glass to choose whether to view files as icons, as a list, or in compact format. Click the magnifying glass at the right end of the toolbar to change the Location bar into a search text box.

Location bar Below the Main toolbar is the location bar, which displays the name of the directory that appears in the View pane. It can display this name in two formats: iconic (using buttons) and textual (using a text box). Press CONTROL-L to switch to textual format. When you display a different directory in the View pane, Nautilus changes the Location bar back to iconic format.

In iconic format, each button represents a directory in a pathname (page 205). The View pane displays the directory of the depressed (darker) button. Click one of these buttons to display that directory. If the leftmost button holds a triangle that points to the left, Nautilus is not displaying buttons for all the directories in the absolute (full) pathname; click the button with a triangle in it to display more directory buttons.

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

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

MENUBAR

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

Figure 8-4 The Connect to Server window

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

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

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

The **Edit⇨Compress** selection creates a single archive file comprising the selected objects. This selection opens a Compress window (Figure 8-5) that allows you to specify the name and location of the archive. The drop-down list to the right of the text box labeled **Filename** allows you to specify a filename extension that determines the type of archive this tool creates. For example, **.tar.gz** creates a tar (page 176) file compressed by gzip (page 175) and **.tar.bz2** creates a tar file compressed by bzip2 (page 174). 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).

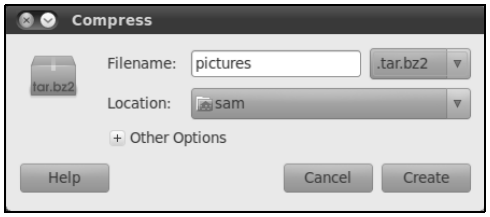

Figure 8-5 The Compress window

The **Edit⇨Backgrounds and Emblems** selection has three buttons on the left: Patterns, Colors, and Emblems. Click **Patterns** to display many pattern objects on the right side of the window. Drag and drop one of these objects on the View pane of a File Browser window to change the background of all File Browser View panes. Drag and drop the Reset object to reset the background to its default color and pattern (usually white). The Colors button works the same way as the Patterns button. The Emblems button works the same way as the Emblems tab in the Side pane (page 278).

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

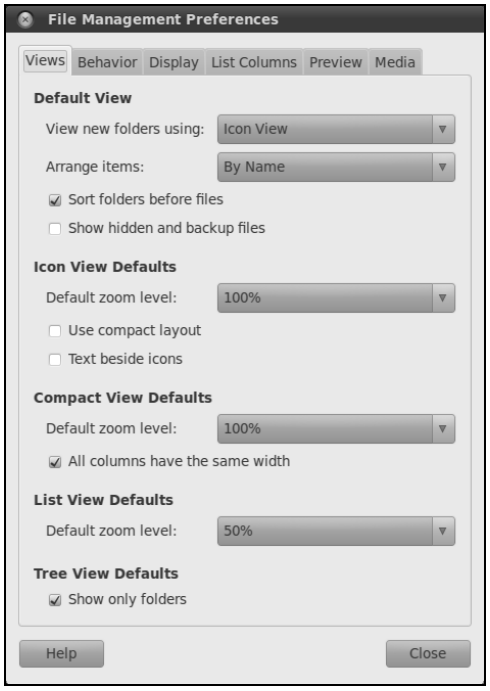

Figure 8-6 The File Management Preferences window, Views tab

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

Delete versus Move to Trash

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

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

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

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

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

View

Click the **Main Toolbar**, **Side Pane**, **Location Bar**, and **Statusbar** selections in the View submenu to display or remove these elements from the window. The **Show Hidden Files** selection displays in the View pane those files with hidden filenames (page 204).

Go

The Go selections display various folders in the View pane.

Bookmarks

Bookmarks appear at the bottom of this menu and in the Side pane under Places. The Bookmarks selections are explained under "Places" on page 278.

Help

The Help selections display local and online information about Nautilus.

optional

THE NAUTILUS SPATIAL VIEW

Nautilus gives you two ways to work with files: the traditional File Browser view described in the previous section and the innovative Spatial view shown in Figure 8-7. By default, Ubuntu displays the Browser view.

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

Turn on the Spatial view by selecting **File Browser menubar: Edit⇨Preferences**. Then click the **Behavior** tab in the File Management Preferences window and put a tick in the check box labeled **Open each folder in its own window**, click **Close**, and close the File Browser window. Next time you open a File Browser window, it will display a Spatial view.

Figure 8-7 The Nautilus Spatial view

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.

You can turn off the Nautilus Spatial view

tip To turn off the Nautilus Spatial view, open a File Browser window. From the menubar, open the File Management Preferences window by selecting **Edit⇨Preferences**. Click the **Behavior** tab in this window and remove the tick from the check box labeled **Open each folder its own window**.

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

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

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

Window memory

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

Parent-folders button

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

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

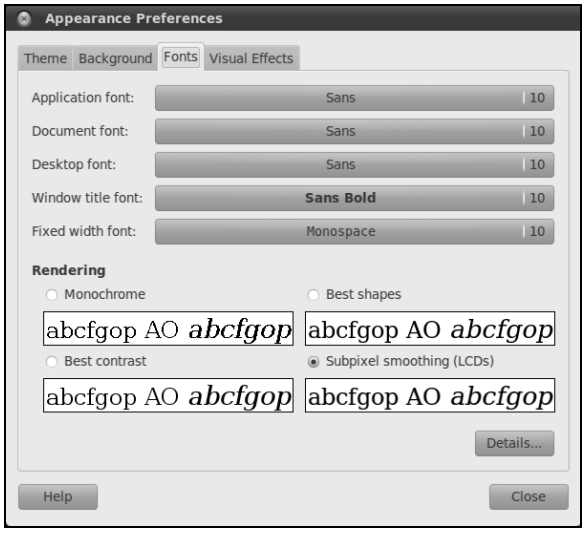

Figure 8-8 The Appearance Preferences window, Fonts tab

GNOME UTILITIES

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

FONT PREFERENCES

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

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

PICK A FONT WINDOW

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

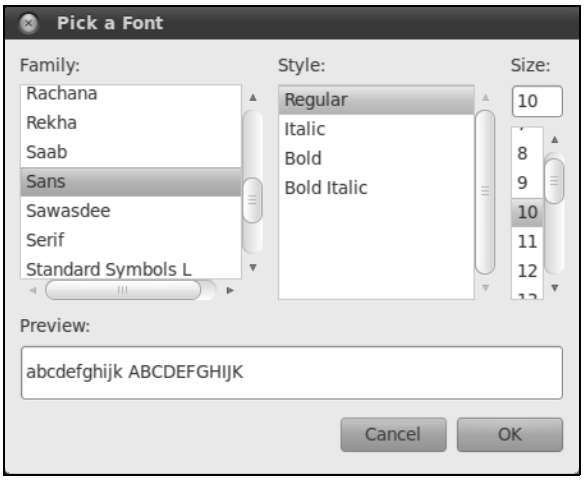

Figure 8-9 The Pick a Font window

PICK A COLOR WINDOW

The Pick a Color window (Figure 8-10) appears when you need to specify a color, such as when you specify a solid color for the desktop background (page 114) or a panel. To specify a color for a panel, right-click the panel to display its context menu, select **Properties**, click the **Background** tab, click the radio button labeled **Solid color**, and click within the box labeled **Color**. GNOME displays the Pick a Color window.

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

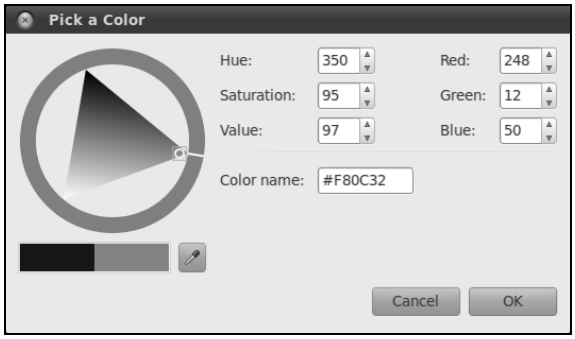

Figure 8-10 The Pick a Color window

blank

11

System Administration: Core Concepts

EXCERPT

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

In This Chapter

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.

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 language (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 (0xb7ec7000)
```

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 (such as **portmap**, **vsftpd**, and **sshd**), ***client_list*** is a comma-separated list of one or more clients (see Table 11-2 on page 461), 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 the **hosts.allow** or **hosts.deny** file, access is granted.

The first match determines whether the client is allowed to access the server. When either **hosts.allow** or **hosts.deny** does not exist, it is as though that file was empty.

Although it is not recommended, you can allow access to all daemons for all clients by removing both files.

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

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

This line prevents any client from connecting to any service, unless specifically permitted 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 anyone to connect to the OpenSSH daemon (ssh, scp, sftp) but allows telnet connections only from the same network as the local system and users on the 192.168. subnet:

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

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

SETTING UP A chroot JAIL

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

The root directory appears at the top of the directory hierarchy and has no parent. 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 **/tmp/jail**, the program would have no concept of any files in **/tmp** 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. The following command sets up a chroot jail in the (existing) **/tmp/jail** directory:

```
$ sudo /usr/sbin/chroot /tmp/jail
/usr/sbin/chroot: cannot 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** (**/tmp/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
/tmp/jail
$ mkdir bin lib
$ cp /bin/bash bin
$ ldd bin/bash
        linux-gate.so.1 =>  (0x0032c000)
        libncurses.so.5 => /lib/libncurses.so.5 (0x00d4d000)
        libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0x0091d000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0x00110000)
        /lib/ld-linux.so.2 (0x0026a000)

$ cp /lib/{libncurses.so.5,ld-linux.so.2} lib
$ cp /lib/tls/i686/cmov/{libdl.so.2,libc.so.6} 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:

```
$ sudo /usr/sbin/chroot /tmp/jail
bash-4.1# pwd
/
bash-4.1# ls
bash: ls: command not found
bash-4.1# exit
exit
$
```

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 261). 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:

```
$ sudo /usr/sbin/chroot /tmp/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** source packages.

The chroot utility fails unless you run it with **root** privileges—the preceding examples used sudo to gain these 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:

```
$ sudo /usr/sbin/chroot jailpath /usr/bin/sudo -u user /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 Ubuntu, 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 insecure jail.

The following C program[1] 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.

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

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.

```
$ 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;
    }
    char buf[100];
    return execv(argv[3], argv + 3);
}
```

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

```
$ cc -o uchroot uchroot.c
$ sudo mv uchroot /usr/local/bin
$ sudo chown root:root /usr/local/bin/uchroot
$ ls -l /usr/local/bin/uchroot
-rwxr-xr-x 1 root root 7922 2010-07-17 08:26 /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:

```
$ sudo /usr/local/bin/uchroot sam /tmp/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

```
$ sudo /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 847), and the **vsftpd** FTP server can automatically start chroot jails for clients (page 703).

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 218) 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 applies to IPv4 only, as IPv6 (page 387) forces systems to configure their IP addresses automatically (called autoconfiguration) when they connect to a network.

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

MORE INFORMATION

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

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 may 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 382). DHCP is broadcast based, so both client and server must be on the same subnet (page 385).

When you install Ubuntu, the system runs a DHCP client, connects to a DHCP server if it can find one, and configures its network interface. You can use firestarter (page 866) to configure and run a DHCP server.

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:

- **dhcp3-client**

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*.leases**, which resides in the **/var/lib/dhcp3** directory. The ***interface*** is the name of the interface that 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. You need to change the default DHCP client configuration file, **/etc/dhcp3/dhclient.conf**, only for custom configurations.

The following **/etc/dhcp3/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 1158) of the device controlling that interface. See page 474 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 (or 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:

- **dhcp3-server**

dhcp3-server init script

When you install the **dhcpd3-server** package, the **dpkg postinst** script attempts to start the **dhcpd3** daemon and fails because **dhcpd3** is not configured—see **/var/log/syslog** for details. After you configure **dhcpd3**, call the **dhcp3-server** init script to restart the **dhcpd3** daemon:

```
$ sudo service dhcp3-server restart
```

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/dhcp3/dhcpd.conf**:

```
$ cat /etc/dhcp3/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;
}
```

The **/etc/default/dhcp3-server** file specifies the interfaces that **dhcpd** serves requests on. By default, **dhcpd** uses **eth0**. To use another interface or to use more than one interface, set the **INTERFACES** variable in this file to a SPACE-separated list of the interfaces you want to use; enclose the list within quotation marks.

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

STATIC IP ADDRESSES

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

When a system that requires a specific static IP address connects to the network and contacts the DHCP server, the server needs a way to identify the system so it can assign the proper IP address to that system. The DHCP server uses the *MAC address* (page 1158) of the system's Ethernet card (NIC) as an identifier. When you set up the server, you must know the MAC address of each system that requires a static IP address.

Determining a MAC address

The ifconfig utility displays the MAC addresses of the Ethernet cards in a system. In the following example, the MAC addresses are the colon-separated series of hexadecimal number pairs following **HWaddr**:

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

Run ifconfig on each system that requires a static IP address. Once you have determined the MAC addresses of these systems, you can add a **host** section to the **/etc/dhcp3/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/dhcp3/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 473).

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. Where 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.

How nsswitch.conf WORKS

When called upon to supply information that **nsswitch.conf** describes, the system examines the line with the appropriate ***info*** field. It uses the methods specified on 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 792)
bootparam	Diskless and other booting options (See the **bootparam** man page.)
ethers	MAC address (page 1158)
group	Groups of users (**/etc/group**; page 492)
hosts	System information (**/etc/hosts**; page 493)
networks	Network information (**/etc/networks**)
passwd	User information (**/etc/passwd**; page 494)
protocols	Protocol information (**/etc/protocols**; page 495)
publickey	Used for NFS running in secure mode
rpc	RPC names and numbers (**/etc/rpc**; page 496)
services	Services information (**/etc/services**; page 497)
shadow	Shadow password information (**/etc/shadow**; page 497)

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:[2]

files	Searches local files such as **/etc/passwd** and **/etc/hosts**
nis	Searches the NIS database; **yp** is an alias for **nis**
dns	Queries the DNS (**hosts** queries only)
compat	**±** syntax in **passwd**, **group**, and **shadow** files (page 477)

SEARCH ORDER

The information provided by two or more methods may overlap: For example, both **files** and **nis** may 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.

2. 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* may 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 may 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 may not be accessible or the required server may 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

exim4: Setting Up Mail Servers, Clients, and More

EXCERPT

Sending and receiving email require three pieces of software. At each end, there is a client, called an MUA (mail user agent), which is a bridge between a user and the mail system. Common MUAs are mutt, Evolution, KMail, Thunderbird, and Outlook. When you send an email, the MUA hands it to an MTA (a mail transfer agent, such as **exim4** or **sendmail**), which transfers it to the destination server. At the destination, an MDA (a mail delivery agent, such as **procmail**) puts the mail in the recipient's mailbox file. On Linux systems, the MUA on the receiving system either reads the mailbox file or retrieves mail from a remote MUA or MTA, such as an ISP's SMTP (Simple Mail Transfer Protocol) server, using POP (Post Office Protocol) or IMAP (Internet Message Access Protocol).

In This Chapter

SMTP Most Linux MUAs expect a local MTA such as **exim4** 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 to deliver email, they are often referred to as SMTP servers. By default, when you install **exim4** on an Ubuntu system, **exim4** uses its own builtin MDA to deliver email to the recipient's mailbox file.

You do not need to set up exim4 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 **exim4**. As a consequence, you do not need to install or configure **exim4** (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 may not need to concern yourself with it, either.

INTRODUCTION TO exim4

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 **exim4** to receive mail from nonlocal systems.

Smarthost You can set up **exim4** on a client system so it sends mail bound for nonlocal systems to an SMTP server that relays the mail to its destination. This type of server is called a *smarthost*. Such a configuration is required by organizations that use firewalls to prevent email from being sent out on the Internet from any system other than the company's official mail servers. As a partial defense against spreading viruses, some ISPs block outbound port 25 to prevent their customers from sending email directly to a remote computer. This configuration is required by these ISPs.

You can also set up **exim4** as a server that sends mail to nonlocal systems and does not use an ISP as a relay. In this configuration, **exim4** connects directly to the SMTP servers for the domains receiving the email. An ISP set up as a smarthost is configured this way.

You can set up **exim4** to accept email for a registered domain name as specified in the domain's DNS MX record (page 828). However, most mail clients (MUAs) do not interact directly with **exim4** to receive email. Instead, they use POP or IMAP—protocols that include features for managing mail folders, leaving messages on the server, and reading only the subject of an email without downloading the entire message. If you want to collect your email from a system other than the one running the incoming mail server, you may need to set up a POP or IMAP server, as discussed on page 735.

ALTERNATIVES TO exim4

sendmail The most popular MTA today, **sendmail** (**sendmail** package) first appeared in 4.1BSD. The **sendmail** system is complex, but its complexity allows **sendmail** to be flexible and to scale well. On the downside, because of its complexity, configuring **sendmail** can be a daunting task. See www.sendmail.org for more information.

Postfix Postfix (**postfix** package) is an alternative MTA. Postfix is fast and easy to administer, but is compatible enough with **sendmail/exim4** to not upset **sendmail/exim4** users. Postfix has a good reputation for ease of use and security and is a drop-in replacement for **sendmail**. Point a browser at www.postfix.org/docs.html for Postfix documentation.

Qmail Qmail 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 720). The Qmail Web site is www.qmail.org.

MORE INFORMATION

Web **exim4**: www.exim.org (includes the complete **exim4** specification), www.exim-new-users.co.uk, wiki.debian.org/PkgExim4
SpamAssassin: spamassassin.apache.org, wiki.apache.org/spamassassin
Spam database: razor.sourceforge.net
Mailman: www.list.org
procmail: www.procmail.org
SquirrelMail: www.squirrelmail.org
IMAP: www.imap.org
Dovecot: www.dovecot.org
Postfix: www.postfix.org/docs.html (alternative MTA)
Qmail: www.qmail.org/top.html

Local **exim4**: **/usr/share/doc/exim4*/***
SpamAssassin: **/usr/share/doc/spam***
Dovecot: **/usr/share/doc/dovecot***
man pages: **exim4** **exim4_files** update-exim4.conf update-exim4defaults spamassassin spamc **spamd**
SpamAssassin: Install the **perl-doc** and **spamassassin** packages and give the following command:

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

SETTING UP A MAIL SERVER (exim4)

This section explains how to set up an **exim4** mail server.

PREREQUISITES

Install the following packages:

- **exim4** (a virtual package)
- **eximon4** (optional; monitors **exim4**)

- **mailutils** (optional; installs mail, which is handy for testing **exim4** from the command line)
- **exim4-doc-html** (optional; **exim4** documentation in HTML format)
- **exim4-doc-info** (optional; **exim4** documentation in **info** format)

exim4 init script

When you install the **exim4** package, the **dpkg postinst** script minimally configures **exim4** and starts the **exim4** daemon. After you configure **exim4**, call the **exim4** init script to restart **exim4**:

```
$ sudo service exim4 restart
```

After changing the **exim4** configuration on an active server, use **reload** in place of **restart** to reload **exim4** configuration files without interrupting the work **exim4** is doing. The **exim4** init script accepts several nonstandard arguments:

```
$ service exim4
Usage: /etc/init.d/exim4 {start|stop|restart|reload|status|what|force-stop}
```

The **status** and **what** arguments display information about **exim4**. The **force-stop** argument immediately kills all **exim4** processes.

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. To do so, use gufw (page 876) to set a policy that allows SMTP service.

Log files

You must be a member of the **adm** group or work with **root** privileges to view the log files in **/var/log/exim4**.

sendmail and **exim4**

Although it does not work the same way **sendmail** does, Ubuntu configures **exim4** as a drop-in replacement for **sendmail**. The **exim4-daemon-light** package, which is part of the **exim4** virtual package, includes **/usr/sbin/sendmail**, which is a link to **exim4**. Because the **exim4** daemon accepts many of **sendmail**'s options, programs that depend on **sendmail** will work with **exim4** installed in place of **sendmail**.

Local and nonlocal systems

The **exim4** daemon sends and receives email. A piece of email that **exim4** receives can originate on a local system or on a nonlocal system. Similarly, email that **exim4** sends can be destined for a local or a nonlocal system. The **exim4** daemon processes each piece of email based on its origin and destination.

The local system versus local systems

The local system is the one **exim4** is running on. *Local systems* are systems that are on the same LAN as the local system.

As it is installed, **exim4** delivers mail to the local system only.

JUMPSTART I: CONFIGURING exim4 TO USE A SMARTHOST

This JumpStart configures an **exim4** server that sends mail from users on local systems to local and nonlocal destinations and does not accept mail from nonlocal systems. This server

- Accepts email originating on local systems for delivery to local systems.
- Accepts email originating on local systems for delivery to nonlocal systems, delivering it using an SMTP server (a smarthost)—typically an ISP—to relay email to its destination.
- Does not deliver email originating on nonlocal systems. As is frequently the case, you need to use POP or IMAP to receive email.
- Does not forward email originating on nonlocal systems to other nonlocal systems (does not relay email).

To set up this server, you need to change the values of a few configuration variables in **/etc/exim4/update-exim4.conf.conf** (page 724) and restart **exim4**. The dpkg-reconfigure utility (page 726) guides you in editing this file; this JumpStart uses a text editor. Working with **root** privileges, use a text editor to make the following changes to **update-exim4.conf.conf**:

```
dc_eximconfig_configtype='smarthost'
smarthost='mail.example.net'
```

Configuration type Set the **dc_eximconfig_configtype** configuration variable to **smarthost** to cause **exim4** to send mail bound for nonlocal systems to the system that the **smarthost** configuration variable specifies. This line should appear exactly as shown in the preceding example.

Smarthost With **dc_eximconfig_configtype** set to **smarthost**, set **smarthost** to the FQDN or IP address (preferred) of the remote SMTP server (the smarthost) that **exim4** uses to relay email to nonlocal systems. Replace *mail.example.net* with this FQDN or IP address. For Boolean variables in **update-exim4.conf.conf**, **exim4** interprets the null value (specified by '') as a value of *false*. With these changes, the file should look similar to this:

```
$ cat /etc/exim4/update-exim4.conf.conf
...
dc_eximconfig_configtype='smarthost'
dc_other_hostnames='example.com'
dc_local_interfaces='127.0.0.1 ; ::1'
dc_readhost=''
dc_relay_domains=''
dc_minimaldns='false'
dc_relay_nets=''
dc_smarthost='mail.example.net'
CFILEMODE='644'
dc_use_split_config='false'
dc_hide_mailname=''
dc_mailname_in_oh='true'
dc_localdelivery='mail_spool'
```

The **exim4** server does not use the value of the **dc_local_interfaces** variable in a smarthost configuration, so you can leave it blank. However, in other configurations, the value of **127.0.0.1 ; ::1** prevents **exim4** from accepting email from nonlo-

cal systems. It is a good idea to configure **exim4** this way and change this variable only when you are ready to accept mail from other systems.

To minimize network accesses for DNS lookups, which can be helpful if you are using a dial-up line, change the value of the **dc_minimaldns** configuration variable to **true**.

/etc/mailname The **/etc/mailname** file initially holds the node name (**uname –n**) of the server. The string stored in **/etc/mailname** appears as the name of the sending system on the envelope-from and From lines of email that originates on the local system. If you want email to appear to come from a different system, change the contents of this file. You can modify this file using a text editor; the dpkg-reconfigure utility can also change it.

The following file causes mail sent from the local system to appear to come from ***username*@example.com**, where ***username*** is the username of the user who is sending the email:

```
$ cat /etc/mailname
example.com
```

See page 724 for more information on **exim4** configuration variables. After making these changes, restart **exim4** (page 716).

Test Test **exim4** with the following command:

```
$ echo "my exim4 test" | exim4 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 **exim4** is sending email to the remote SMTP server (the smarthost). If the mail is not delivered, check the email of the user who sent the email (on the local system) for errors. Also check the log file(s) in the **/var/log/exim4** directory.

JUMPSTART II: CONFIGURING exim4 TO SEND AND RECEIVE MAIL

To receive email sent from a nonlocal system to a registered domain (that you control), you need to configure **exim4** to accept email from nonlocal systems. This JumpStart describes how to set up a server that

- Accepts email from local and nonlocal systems.
- Delivers email that originates on local systems to a local system or directly to a nonlocal system, without using a relay.
- Delivers email that originates on nonlocal systems to a local system only.
- Does not forward email originating on nonlocal systems to other nonlocal systems (does not relay email).

This server does not relay email originating on nonlocal systems. (You must set the **dc_relay_domains** variable [page 726] for the local system to act as a relay.) For this

configuration to work, you must be able to make outbound connections and receive inbound connections on port 25 (see "Firewall" on page 716).

Working with **root** privileges, use a text editor to set the following configuration variables in **/etc/exim4/update-exim4.conf.conf**:

```
dc_eximconfig_configtype='internet'
dc_other_hostnames='mydom.example.com'
dc_local_interfaces=''
```

Configuration type

Set **dc_eximconfig_configtype** to **internet** to cause **exim4** to send mail directly to nonlocal systems as specified by the DNS MX record (page 828) for the domain the mail is addressed to and to accept email on the interfaces specified by **dc_local_interfaces** (next page). This line should appear exactly as shown above.

Other hostnames

The **dc_other_hostnames** configuration variable specifies the FQDNs or IP addresses that the local server receives mail addressed to. Replace *mydom.example.com* with these FQDN or IP addresses. You must separate multiple entries with semicolons. These values do not necessarily include the FQDN or the IP address of the local server.

Local interfaces

Set **dc_local_interfaces** to the interface you want **exim4** to listen on. Set it to the null value ('') to listen on all interfaces.

As in JumpStart I, you may need to change the value of **/etc/mailname** (page 718). For Boolean variables in this file, **exim4** interprets the null value (specified by '') as *false*. The file should look similar to this:

```
$ cat /etc/exim4/update-exim4.conf.conf
...
dc_eximconfig_configtype='internet'
dc_other_hostnames='mydom.example.com'
dc_local_interfaces=''
dc_readhost=''
dc_relay_domains=''
dc_minimaldns='false'
dc_relay_nets=''
dc_smarthost=''
CFILEMODE='644'
dc_use_split_config='false'
dc_hide_mailname=''
dc_mailname_in_oh='true'
dc_localdelivery='mail_spool'
```

See page 724 for more information on **exim4** configuration variables. Once you have restarted **exim4**, it will accept mail addressed to the local system. To receive email addressed to a domain, the DNS MX record (page 828) for that domain must point to the IP address of the local system. If you are not running a DNS server, you must ask your ISP to set up an MX record or else receive mail at the IP address of the server. If you receive email addressed to an IP address, set **dc_other_hostnames** to that IP address.

WORKING WITH exim4 MESSAGES

When **exim4** receives email, from both local and nonlocal systems, it creates in the **/var/spool/exim4/input** directory two files that hold the message while **exim4** processes it. To identify a particular message, **exim4** generates a 16-character message ID and uses that string in filenames pertaining to the email. The **exim4** daemon stores the body of the message in a file named by the message ID followed by **–D** (data). It stores the headers and envelope information in a file named by the message ID followed by **–H** (header).

Frozen messages

If **exim4** cannot deliver a message, it marks the message as *frozen* and makes no further attempt to deliver it. Once it has successfully delivered an email, **exim4** removes all files pertaining to that email from **/var/spool/exim4/input**.

Mail addressed to the local system

By default, **exim4** delivers email addressed to the local system to users' files in the mail spool directory, **/var/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 **exim4** 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 **exim4** daemon supports both **mbox** and **maildir** formats (see **dc_localdelivery** on page 725). Qmail (page 715), an alternative to **sendmail** and **exim4**, uses **maildir**-format mailboxes.

MAIL LOGS

By default, **exim4** sends normal log messages to **/var/exim4/mainlog**, with other messages going to other files in the same directory. The following lines in a **mainlog** file describe an email message sent directly to a remote system's SMTP server. The **exim4** daemon writes one line each time it receives a message and one line each time it attempts to deliver a message. The **Completed** line indicates that **exim4** has completed its part in delivering the message. Each line starts with the date and time of the entry followed by the message ID.

```
$ tail -3 /var/log/exim4/mainlog
2010-07-19 23:13:12 1IBljk-0000t8-1Z <= zachs@example.com U=sam P=local S=304
2010-07-19 23:13:17 1IBljk-0000t8-1Z => zachs@example.com R=dnslookup T=remote_smtp
   H=filter.mx.meer.net   [64.13.141.12]
2010-07-19 23:13:17 1IBljk-0000t8-1Z Completed
```

The next entry on each line except the **Completed** line is a two-character status flag that tells you which kind of event the line describes:

<=	Received a message
=>	Delivered a message normally
–>	Delivered a message normally to an additional address (same delivery)
***>**	Did not deliver because of a **–N** command-line option
******	Did not deliver because the address bounced
==	Did not deliver because of a temporary problem

Information following the flag is preceded by one of the following letters, which indicates the type of the information, and an equal sign:

H	Name of remote system (host)
U	Username of the user who sent the message
P	Protocol used to receive the message
R	Router used to process the message
T	Transport used to process the message
S	Size of the message in bytes

The first line in the preceding example indicates that **exim4** received a 304-byte message to be delivered to **zachs@example.com** from **sam** on the **local** system. The next line indicates that **exim4** looked up the address using DNS (**dnslookup**) and delivered it to the remote SMTP server (**remote_smtp**) at **filter.mx.meer.net**, which has an IP address of 64.13.141.12.

The following log entries describe a message that **exim4** received from a remote system and delivered to the local system:

```
2010-07-19 23:13:32 1IBlk4-0000tL-8L <= zachs@gmail.com H=wx-out-0506.google.com
   [66.249.82.229]   P=esmtp S=1913 id=7154255d0707192313y304a1b27t39f...@mail.gmail.com
2010-07-19 23:13:32 1IBlk4-0000tL-8L => sam <sams@example.com> R=local_user T=mail_spool
2010-07-19 23:13:32 1IBlk4-0000tL-8L Completed
```

See the **exim4** specification for more information on log files. If you send and receive a lot of email, the mail logs can grow quite large. The logrotate (page 622) **exim4-base** file archives and rotates these files regularly.

WORKING WITH MESSAGES

You can call **exim4** with many different options to work with mail that is on the system and to generate records of mail that has passed through the system. Most of these options begin with **–M** and require the message ID (see the preceding section)

of the piece of email you want to work with. The following command removes a message from the queue:

```
$ sudo exim4 -Mrm 1IEKKj-0006CQ-LM
Message 1IEKKj-0006CQ-LM has been removed
```

Following are some of the **exim4** options you can use to work with a message. Each of these options must be followed by a message ID. See the **exim4** man page for a complete list.

–Mf Mark message as frozen
–Mrm Remove message
–Mt Thaw message
–Mvb Display message body
–Mvh Display message header

ALIASES AND FORWARDING

You can use the **aliases** and **.forward** (page 723) files to forward email.

/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 may want email to go to a class of users and not to a specific recipient. Examples of classes of users include **postmaster**, **webmaster**, **root**, and **tech_support**. Different users may receive this email at different times or the email may 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. Because email sent to the **root** account is rarely checked, the default installation includes an entry similar to the following that redirects email sent to **root** to the initial user:

```
root:  sam
```

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**:

```
abuse: sam, max
```

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 489), 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 734). 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"
```

See the **exim4_files** man page for information on **exim4** files, including **aliases**.

newaliases After you edit **/etc/aliases**, you must run newaliases while you are working with **root** privileges. The **/usr/bin/newaliases** file is a symbolic link to **exim4**. Running newaliases calls **exim4**, which rebuilds the **exim4** alias database.

~/.forward Systemwide aliases are useful in many cases, but non**root** users cannot make or change them. Sometimes you may want to forward your own mail: Maybe you want mail from several systems to go to one address or perhaps you 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

exim4 The **exim4** packages include several programs. The primary program, **exim4**, reads from standard input and sends an email to the recipient specified by its argument. You can use **exim4** from the command line to check that the mail delivery system is working and to email the output of scripts. See "Test" on page 718 for an example. The command **apropos exim4** displays a list of **exim4**-related files and utilities. In addition, you can call **exim4** with options (page 721) or through links to cause it to perform various tasks.

exim4 –bp When you call **exim4** with the **–bp** option, or when you call the mailq utility (which is a symbolic link to **exim4**), it displays the status of the outgoing mail queue. When there are no messages in the queue, it displays nothing. Unless they are transient, messages in the queue usually indicate a problem with the local or remote MTA configuration or a network problem.

```
$ sudo exim4 -bp
24h   262 1IBhYI-0006iT-7Q <sam@> *** frozen ***
          zachs@example.com
```

eximstats The eximstats utility displays statistics based on **exim4** log files. Call this utility with an argument of the name of a log file, such as **/var/log/mainlog** or **/var/log/mainlog2.gz**. Without any options, eximstats sends information based on the log file in text format to standard output. When you include the **–html** option, eximstats generates output in HTML format, suitable for viewing with a browser:

```
$ eximstats -html /var/log/exim4/mainlog.2.gz  > exim.0720.html
```

If you are not a member of the **adm** group, you must run the preceding command with **root** privileges. See the eximstats man page for more information.

eximon Part of the **eximon4** package, eximon displays a simple graphical representation of the **exim4** queue and log files.

CONFIGURING AN exim4 MAIL SERVER

The **exim4** daemon is a complex and capable MTA that is configured by **/etc/default/exim4** and the files in the **/etc/exim4** directory hierarchy. The former allows you to specify how the daemon is to be run; the latter configures all other aspects of **exim4**. You can configure **exim4** by editing its configuration files with a text editor (discussed in the next section) or by using dpkg-reconfigure (page 726).

/etc/default/exim4 The default **/etc/default/exim4** file sets QUEUERUNNER to **combined**, which starts one daemon that both runs the queue and listens for incoming email. It sets QUEUEINTERVAL to **30m**, which causes the daemon to run the queue (that is, check whether the queue contains mail to be delivered) every 30 minutes. See the comments in the file for more information.

USING A TEXT EDITOR TO CONFIGURE exim4

The files in the **/etc/exim4** directory hierarchy control how **exim4** works—which interfaces it listens on, whether it uses a smarthost or sends email directly to its destination, whether and for which systems it relays email, and so on. You can also create an **exim4.conf.localmacros** file to turn on/off **exim4** functions (see page 737 for an example). Because of its flexibility, **exim4** uses many configuration variables. You can establish the values of these variables in one of two ways: You can edit a single file, as the JumpStart sections of this chapter explain, or you can work with the approximately 40 files in the **/etc/exim4/conf.d** directory hierarchy. For many configurations, working with the single file **update-exim4.conf.conf** is sufficient. This section describes the variables in that file but does not discuss working with the files in **conf.d**. Refer to the **exim4** specification if you need to set up a more complex mail server.

THE update-exim4.conf.conf CONFIGURATION FILE

update-exim4.conf The update-exim4.conf utility reads the **exim4** configuration files in **/etc/exim4**, including **update-exim4.conf.conf**, and generates the **/var/lib/exim4/config.autogenerated**

file. When **exim4** starts, it reads this file for configuration information. Typically you do not need to run update-exim4.conf manually because the **exim4** init script (page 716) runs this utility before it starts, restarts, or reloads **exim4**.

Split configuration

Setting the **dc_use_split_config** variable in **update-exim4.conf.conf** to **false** specifies an unsplit configuration, wherein update-exim4.conf merges the data from **exim4.conf.localmacros**, **update-exim4.conf.conf**, and **exim4.conf.template** to create **config.autogenerated**. Setting this variable to **true** specifies a split configuration, wherein update-exim4.conf merges the data from **exim4.conf.localmacros**, **update-exim4.conf.conf**, and all the files in the **conf.d** directory hierarchy to create **config.autogenerated**.

Following is the list of configuration variables you can set in **update-exim4.conf.conf**. Enclose all values within single quotation marks. For Boolean variables, **exim4** interprets the null value (specified by '') as **false**.

CFILEMODE='*perms***'**
Sets the permissions of **config.autogenerated** to the octal value ***perms***, typically 644.

dc_eximconfig_configtype='*type***'**
Specifies the type of configuration that **exim4** will run, where ***type*** is one of the following:

internet Sends and receives email locally and remotely. See "JumpStart II" on page 718 for an example.

smarthost Sends and receives email locally and remotely, using a smarthost to relay messages to nonlocal systems. See "JumpStart I" on page 716 for an example.

satellite Sends email remotely, using a smarthost to relay messages; does not receive mail locally.

local Sends and receives local messages only.

none No configuration; **exim4** will not work.

dc_hide_mailname='*bool***'**
Controls whether **exim4** displays the local mailname (from **/etc/mailname**, page 718) in the headers of email originating on local systems. Set ***bool*** to **true** to hide (not display) the local mailname or **false** to display it. When you set this variable to **true**, **exim4** uses the value of **dc_readhost** in headers.

dc_local_interfaces='*interface-list***'**
The ***interface-list*** is a semicolon-separated list of interfaces that **exim4** listens on. Set ***interface-list*** to the null value ('') to cause **exim4** to listen on all interfaces. Set it to 127.0.0.1 to prevent **exim4** from accepting email from other systems.

dc_localdelivery='*lcl-transport***'**
Set ***lcl-transport*** to **mail_spool** to cause **exim4** to store email in **mbox** format; set it to **maildir_home** for **maildir** format. See page 720 for more information.

dc_mailname_in_oh='*bool***'**
Used internally by **exim4**. Do not change this value.

dc_minimaldns='*bool*'
Set ***bool*** to **true** to minimize DNS lookups (useful for dial-up connections) or to **false** to perform DNS lookups as needed.

dc_other_hostnames='*host-list*'
The ***host-list*** is a semicolon-separated list of IP addresses and/or FQDNs the local system accepts (but does not relay) email for; **localhost** (127.0.0.1) is assumed to be in this list.

dc_readhost='*hostname*'
The ***hostname*** replaces the local mailname in the headers of email originating on local systems. This setting is effective only if **dc_hide_mailname** is set to **true** and **dc_eximconfig_configtype** is set to **smarthost** or **satellite**.

dc_relay_domains='*host-list*'
The ***host-list*** is a semicolon-separated list of IP addresses and/or FQDNs the local system accepts mail for, but does not deliver to local systems. The local system relays mail to these systems. For example, the local system may be a secondary server for these systems.

dc_relay_nets='*host-list*'
The ***host-list*** is a semicolon-separated list of IP addresses and/or FQDNs of systems that the local system relays mail for. The local system is a smarthost (page 717) for these systems.

dc_smarthost='*host-list*'
The ***host-list*** is a semicolon-separated list of IP addresses (preferred) and/or FQDNs the local system sends email to for relaying to nonlocal systems (a smarthost; page 717). See "JumpStart I" on page 716 for an example.

dc_use_split_config='*bool*'
Controls which files update-exim4.conf uses to generate the configuration file for **exim4**. See "Split configuration" (page 725) for more information.

27.bash3.fm Page 953 Wednesday, July 28, 2010 4:38 PM

27

Programming the Bourne Again Shell

EXCERPT

In This Chapter

Chapter 7 introduced the shells and Chapter 9 went into detail about the Bourne Again Shell. This chapter introduces additional Bourne Again Shell commands, builtins, and concepts that carry shell programming to a point where it can be useful. Although you may 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

27.bash3.fm Page 954 Wednesday, July 28, 2010 4:38 PM

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 1015 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 may 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.

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
```

27.bash3.fm Page 955 Wednesday, July 28, 2010 4:38 PM

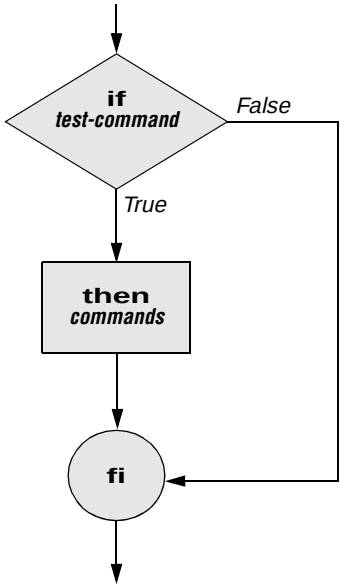

Figure 27-1 An **if...then** flowchart

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

test builtin

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

27.bash3.fm Page 956 Wednesday, July 28, 2010 4:38 PM


```
$ ./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 may vary slightly from one shell to the next and from the utility to any of the shell builtins. See page 1002 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 997) 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

27.bash3.fm Page 957 Wednesday, July 28, 2010 4:38 PM

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 white-space (SPACEs or TABs).

27.bash3.fm Page 958 Wednesday, July 28, 2010 4:38 PM


```
$ 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 297). 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

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

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

27.bash3.fm Page 959 Wednesday, July 28, 2010 4:38 PM

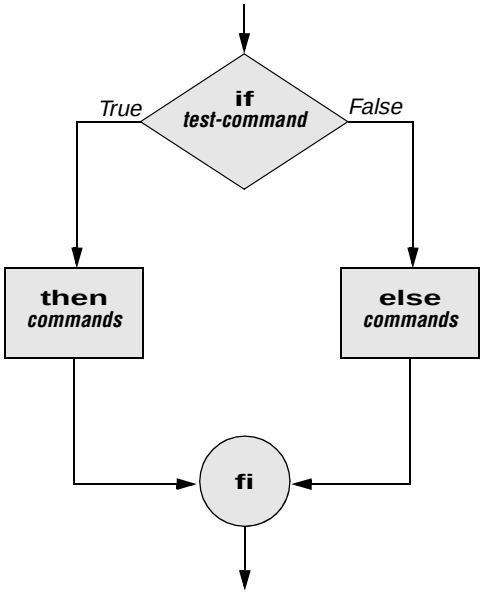

Figure 27-2 An **if...then...else** flowchart

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

When you run the **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 162) 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 998) 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
```

27.bash3.fm Page 960 Wednesday, July 28, 2010 4:38 PM

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 1012) 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.

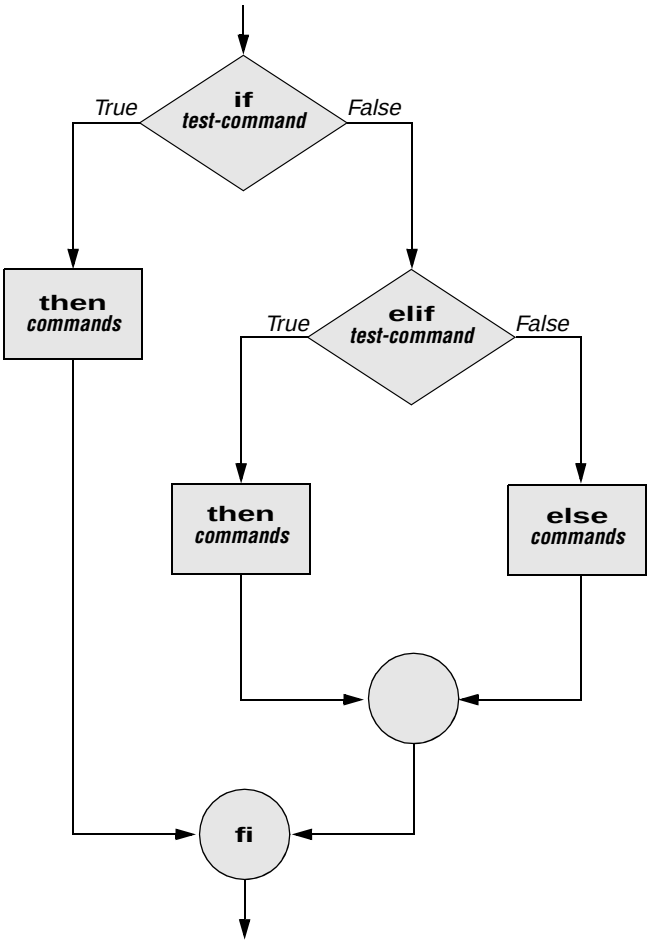

Figure 27-3 An **if...then...elif** flowchart

27.bash3.fm Page 961 Wednesday, July 28, 2010 4:38 PM

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 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
```

27.bash3.fm Page 962 Wednesday, July 28, 2010 4:38 PM


```
$ ./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
```

27.bash3.fm Page 963 Wednesday, July 28, 2010 4:38 PM


```
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 302) to specify the shell that will execute the script:

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

27.bash3.fm Page 964 Wednesday, July 28, 2010 4:38 PM

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
```

27.bash3.fm Page 965 Wednesday, July 28, 2010 4:38 PM

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 **$file** does not exist. This test keeps **lnks** from wasting time looking for links to a nonexistent file. The test builtin, when called with the three arguments **!**, **–f**, and **$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 362) 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 1153) for **$file**. As explained on page 229, 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
```

27.bash3.fm Page 966 Wednesday, July 28, 2010 4:38 PM

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 226 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='>>>> '
```

27.bash3.fm Page 967 Wednesday, July 28, 2010 4:38 PM

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

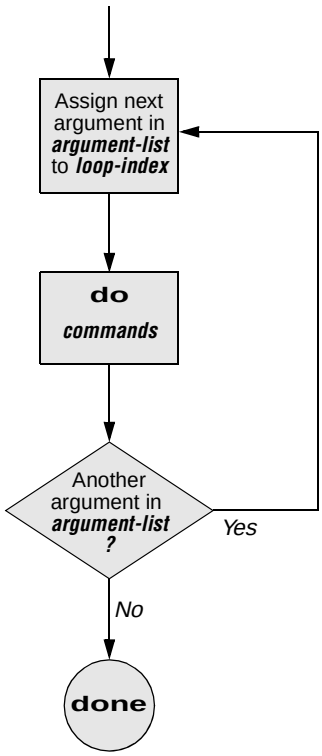

Figure 27-4 A **for...in** flowchart

27.bash3.fm Page 968 Wednesday, July 28, 2010 4:38 PM

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

After it passes control to the **done** statement, the structure assigns the value of the second argument in the *argument-list* to the *loop-index* and repeats the *commands*. 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**.