Do what you never thought possible with LINUX!

BESTSELLING SERIES

OVER **2.5 MILLION** SECRETS BOOKS IN PRINT

unleash the power
900 plus pages packed with the hottest insider tips for tapping the full potential of LINUX
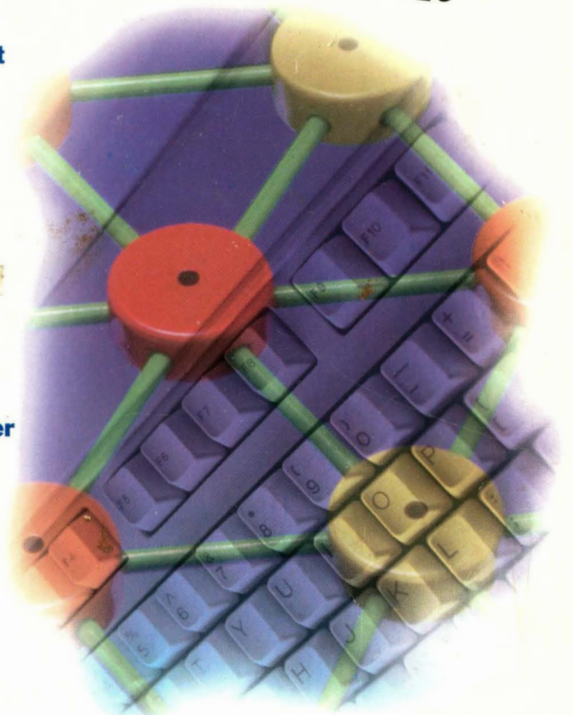
expose the mysteries
Complete overview of the latest programming techniques — from Tcl/Tk scripting to X programming

reveal the secrets
Expert advice on using LINUX as an Internet host — WWW server and anonymous FTP server

Red Hat LINUX 5.1 on CD-ROM!

Red Hat® LINUX® secrets®

2nd EDITION

IDG BOOKS WORLDWIDE

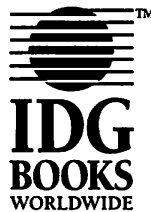INFO WORLD

Naba Barkakati

Red Hat® LINUX® Secrets®, 2nd Edition

Red Hat® LINUX® Secrets®,
2nd Edition

Naba Barkakati

IDG
BOOKS
WORLDWIDE

is a trademark under exclusive
license to IDG Books Worldwide, Inc.,
from International Data Group, Inc.

IDG
BOOKS
WORLDWIDE

ABOUT IDG BOOKS WORLDWIDE

Welcome to the world of IDG Books Worldwide.

IDG Books Worldwide, Inc., is a subsidiary of International Data Group, the world's largest publisher of computer-related information and the leading global provider of information services on information technology. IDG was founded more than 25 years ago and now employs more than 8,500 people worldwide. IDG publishes more than 275 computer publications in over 75 countries (see listing below). More than 90 million people read one or more IDG publications each month.

Launched in 1990, IDG Books Worldwide is today the #1 publisher of best-selling computer books in the United States. We are proud to have received eight awards from the Computer Press Association in recognition of editorial excellence and three from *Computer Currents*' First Annual Readers' Choice Awards. Our best-selling *...For Dummies*® series has more than 50 million copies in print with translations in 38 languages. IDG Books Worldwide, through a joint venture with IDG's Hi-Tech Beijing, became the first U.S. publisher to publish a computer book in the People's Republic of China. In record time, IDG Books Worldwide has become the first choice for millions of readers around the world who want to learn how to better manage their businesses.

Our mission is simple: Every one of our books is designed to bring extra value and skill-building instructions to the reader. Our books are written by experts who understand and care about our readers. The knowledge base of our editorial staff comes from years of experience in publishing, education, and journalism — experience we use to produce books for the '90s. In short, we care about books, so we attract the best people. We devote special attention to details such as audience, interior design, use of icons, and illustrations. And because we use an efficient process of authoring, editing, and desktop publishing our books electronically, we can spend more time ensuring superior content and spend less time on the technicalities of making books.

You can count on our commitment to deliver high-quality books at competitive prices on topics you want to read about. At IDG Books Worldwide, we continue in the IDG tradition of delivering quality for more than 25 years. You'll find no better book on a subject than one from IDG Books Worldwide.

John Kilcullen
CEO
IDG Books Worldwide, Inc.

Steven Berkowitz
President and Publisher
IDG Books Worldwide, Inc.

VIII WINNER
Eighth Annual
Computer Press
Awards ≥1992

IX WINNER
Ninth Annual
Computer Press
Awards ≥1993

X WINNER
Tenth Annual
Computer Press
Awards ≥1994

XI WINNER
Eleventh Annual
Computer Press
Awards ≥1995

Credits

Acquisitions Editor
Laura Lewin

Development Editors
Laura E. Brown
Luann Rouff

Technical Editor
Phil Hughes

Copy Editors
Luann Rouff
Anne Friedman
Colleen Dowling

Project Coordinator
Tom Debolski

**Graphics and
Production Specialists**
Hector Mendoza
Linda Marousek
Christopher Pimentel

Quality Control Specialists
Mick Arellano
Mark Schumann

Proofreader
Arielle Carole Mennelle

Indexer
Liz Cunningham

Packaging Coordinator
Cyndra Robbins

About the Author

Naba Barkakati is an expert programmer and successful computer-book author who has experience in a wide variety of systems, ranging from MS-DOS and Windows to UNIX and the X Window System. He bought his first personal computer — an IBM PC-AT — in 1984 after graduating with a Ph.D. in electrical engineering from the University of Maryland at College Park. While pursuing a full-time career in engineering, Naba dreamed of writing software for the emerging PC software market. As luck would have it, instead of building a software empire like Microsoft, he ended up writing successful computer books.

Over the past ten years, Naba has written 22 computer books on a number of topics ranging from Windows programming with Visual C++ to Linux. He had authored several best-selling titles such as *The Waite Group's Turbo C++ Bible*, *Object-Oriented Programming in C++*, *X Window System Programming*, *Visual C++ Developer's Guide*, and *Borland C++ 4 Developer's Guide*. Naba's books have been translated into many languages, including Spanish, French, Polish, Greek, Italian, Chinese, Japanese, and Korean. His most recent books are *UNIX Webmaster Bible* and *Discover Perl 5*, both published by IDG Books Worldwide.

Naba lives in North Potomac, Maryland, with his wife Leha and their children, Ivy, Emily, and Ashley.

To my wife Leha and our daughters, Ivy, Emily, and Ashley

Preface

If you are beginning to use Linux you need a practical guide. One that not only gets you going with the installation and setup of Linux, but also shows you how to use Linux for specific tasks, such as an Internet host or a software development platform.

The first edition of this book was a technical guide designed to address these needs. That book included Version 3.0 of the popular Slackware distribution of Linux on a CD-ROM with the standard installation and setup information.

A great deal has happened in the two years since the first edition was published. The Linux kernel, the core operating system, is now at release 2.0. New kernel features, as well as support, for new hardware continue to grow. A huge upsurge in user interest has occurred, as evidenced by the growing number of Linux books, especially those for beginners. In addition, newer Linux distributions — combinations of the operating system with applications and installation tools — have been developed to simplify the installation process.

In particular, the Red Hat Linux distribution has emerged as the choice of users who want a simpler installation process than that offered by Slackware Linux. Red Hat includes a number of graphical tools that run under the X Window System to help users with set up and configuration tasks. In addition to installation differences, Red Hat Linux also differs from Slackware Linux in the location and format of some system configuration files. Although experts argue over the relative merits of one arrangement of configuration files over another, the simpler installation steps afforded by the Red Hat Linux distribution certainly better serve the novice user. In any case, the core Linux operating system and the collection of applications are nearly identical in all Linux distributions.

Red Hat LINUX Secrets, 2nd Edition follows the successful model of the first edition with one significant change: it includes the latest release of Red Hat Linux (version 5.1, with the 2.0.34 kernel) on the companion CD-ROM. This book provides detailed technical information on installing and customizing Linux, including coverage of various types of computers and peripherals.

This book then goes a step beyond existing books and shows you how to use Linux as a solution to specific problems. In addition, it provides information on freely available software packages — such as news and mail, graphics, and text utilities — on the book's companion CD-ROM.

The unique aspects of *Red Hat LINUX Secrets, 2nd Edition* are the tips, techniques, shortcuts, and little-known facts about using Linux in various real-world tasks, ranging from simply learning UNIX to setting up a WWW server for your business.

By reading this book you get the following benefits:

- Learn how to install and set up Linux from the Red Hat Linux CD-ROM included with the book

- Learn how to use various peripherals (video cards, hard disks, and network cards) in Linux

- Learn about dial-up networking (with SLIP and PPP) under Linux

- Get tips, techniques, and shortcuts for specific uses of Linux, such as:

  - Learn how to use Linux as an Internet host (WWW server and anonymous FTP server)

  - Understand how Linux and DOS can coexist

  - Learn UNIX on Linux

  - Learn C and C++ programming on Linux

  - Learn Motif programming on Linux

- Receive many Linux tools and utilities

- Learn about Linux resources that can serve as continuing sources of information in the ever-changing world of Linux

Organization of the Book

Red Hat LINUX Secrets, 2nd Edition has 24 chapters, organized into four parts and four appendixes:

Part I: Configuring Your Linux System includes two chapters that essentially guide you through the steps of installing Linux from the CD-ROM bundled with the book. The second chapter in this part describes how to apply patches and reconfigure the Linux kernel as new revisions become available.

Part II: Running Linux focuses on how to run Linux after it has been installed. Part II includes six chapters that explain how to set up the graphical interface with *X,* how to use Linux commands (which essentially are UNIX commands), how to access DOS from Linux, and how to use the Tcl/Tk scripting language for quick-and-dirty programming.

Part III: Exploiting Your Hardware in Linux provides all the information you need to use various types of hardware in Linux. The hardware descriptions cover everything from the computer processor, memory, and bus to serial ports and PC cards (that use the PCMCIA interface).

Part IV: Using Linux for Fun and Profit has seven chapters that show how to use Linux for specific purposes such as setting up an Internet host, running a World Wide Web server, developing software, and preparing documents.

Appendix A: The Best of Linux Applications describes several popular Linux applications on the book's companion CD-ROM.

Appendix B: Linux Commands presents alphabetically arranged reference entries for the most important Linux commands.

Appendix C: Linux Resources lists resources on the Internet where you can obtain the latest information about Linux.

Appendix D: CD-ROM Installation Instructions summarizes how to install Linux from the book's companion CD-ROM.

If you are a new user, you should start in Part I with the installation of Linux from the CD-ROM. If you have already installed Linux, you might begin with Chapter 3 and learn how to make the most of Linux in everyday use. If you have specific hardware questions, you should go directly to a relevant chapter in Part III. Part IV is meant to describe specific Linux uses. Read the relevant chapters in Part IV to get going with specific tasks. When you need information on a specific Linux command, turn to Appendix B and look for that command in the alphabetically arranged reference entries.

Conventions Used in This Book

Red Hat LINUX Secrets, 2nd Edition uses a simple notational style. All listings, filenames, function names, variable names, and keywords are typeset in a monospace font for ease of reading. The first occurrences of new terms and concepts are in *italic*. Text you are directed to type is in **boldface**.

Each chapter starts with a short list of all the neat things you learn in that chapter. The summary at the end of the chapter tells you a bit more about what the chapter covered.

Following the time-honored tradition of the IDG Books *Secrets* series, I use icons to help you pinpoint useful information quickly. Following is what I had in mind for the icons:

The Note icon marks an interesting fact — something I thought you'd like to know.

The Tip icon marks things you can do to make your job simpler — hints you can try.

Caution

The Caution icon highlights potential pitfalls. With this icon, I'm telling you: Watch out! This could hurt your system!

CD

The CD icon points to specific programs or documentation on this book's companion CD-ROM.

Cross-Reference

The Cross Reference icon points out paragraphs that lead you to other chapters in the book for a deeper discussion of a topic.

Secret

The Secret icon marks facts that are not well-documented but important to know. Once you know these facts, they may clear up many questions.

Wizard

The Wizard icon marks technical information of interest to an advanced user.

Sidebars

This is a sidebar. I use sidebars throughout the book to highlight interesting, but not critical, information. Sidebars explain concepts you may not have encountered before or give insight on a related topic. If you're in a hurry, you can safely skip the sidebars. On the other hand, if you find yourself flipping through the book looking for interesting information, searching for the sidebars is a good idea.

Red Hat Linux 5.1 CD-ROM

Red Hat LINUX Secrets, 2nd Edition addresses the needs of new users who want to put Linux to some productive use on their home or office PC. Because installation is one of the difficult phases in getting started with Linux, the editorial team and I decided to include a copy of Red Hat Linux 5.1 (with Linux kernel 2.0.34) on the companion CD-ROM. Red Hat Linux is easy to install and is well-supported by Red Hat (www.redhat.com).

Red Hat Linux 5.1 is a complete Linux distribution with the 2.0.34 kernel (operating system) plus a large selection of Linux software. In particular, the following software is on the Red Hat Linux 5.1 CD-ROM:

- Linux kernel 2.0.34 with driver modules for all major PC hardware configurations including IDE/EIDE and SCSI drives, PCMCIA devices, and CD-ROMs

- Complete set of installation and configuration tools for setting up devices (such as keyboard and mouse) and services (network)

- Graphical user interface based on the XFree86 3.3.2 package with fvwm2 and AfterStep window managers

- Full TCP/IP networking for Internet, LANs, and intranets

- Tools for connecting your PC to your Internet service provider using PPP, SLIP, or dialup serial communications programs

- Complete suite of Internet applications including electronic mail (sendmail, elm, pine, mailx), news (inn, tin, trn), Internet Relay Chat (ircii), telnet, FTP, and NFS

- Apache Web server 1.2.6 to turn your PC into a Web server and Netscape Communicator 4.05 to surf the Net

- Samba 1.9 LAN Manager software for Microsoft Windows connectivity

- Several text editors (GNU Emacs 20.2, JED, Joe, vim)

- Graphics and image manipulation software such as XV, XPaint, Xfig, Gnuplot, Ghostscript, Ghostview, GIMP, ImageMagick, and xanim

- Programming languages (GNU C and C++ 2.7.2.3, Perl 5.004, Tcl/Tk 8.0.2, Python 1.5.1, GNU AWK 3.0.3) and software development tools (GNU Debugger 4.17, RCS 5.7, GNU Bison 1.25, flex 2.5.4a, TIFF, and JPEG libraries)

- Support for industry standard Executable and Linking Format (ELF) and Intel Binary Compatibility Specification (iBCS)

- Complete suite of standard UNIX utilities

- Tools to access and use DOS files and applications (DOSEMU 0.66.7, mtools 3.8)

- Text formatting and typesetting software (groff, TeX, and LaTeX)

- Games such as GNU Chess, xtetris, acm, colour-yahtzee, flying, fortune-mod, mysterious, paradise, xchomp, xevil, xgalaga, xgammon, xpilot

This long list of software shouldn't overwhelm you. You only have to learn to use what you need. Besides, this book shows you both how to install Linux and use most of this software.

If you have enough space available on your PC's hard disk (or, better yet, a spare second hard disk), Red Hat Linux installation can be as simple as creating a boot disk, booting the PC, and filling up information in a series of dialog boxes. But don't take my word for it — you can see for yourself!

Now it's time to begin your Linux adventure. Take out the companion CD-ROM, turn to Chapter 1, and let the fun begin. Before you know it, you'll be a Linux expert!

I hope you enjoy this book as much as I enjoyed writing it!

Acknowledgments

I am grateful to Laura Lewin for getting me started with this book. Thanks to everyone else at IDG Books Worldwide who transformed my raw manuscript into this well-edited and beautifully packaged book, especially development editors Laura Brown and Luann Rouff, who guided me through the manuscript submission process and kept everything moving. I appreciate the guidance and support you both gave me during this project. Also, thanks to Lenora Chin Sell for making the necessary arrangements with Red Hat for the book's companion CD-ROM.

I also want to thank Phil Hughes, publisher of *Linux Journal* (www.linuxjournal.com), for reviewing the manuscript for technical accuracy. Phil provided many useful suggestions for improving the book's content.

Of course, if it were not for Linux, no reason would exist for this book. For this, we have Linus Torvalds and the legions of Linux developers around the world to thank.

Finally, and, as always, my greatest thanks go to my wife Leha for her patience and understanding and for taking care of everything while I stayed glued to my PCs the past few months. As I wrap up the book, my daughters, Ivy, Emily, and Ashley are tracking my progress and counting the days to the deadline. Thanks for being there!

Contents at a Glance

Contents

Introduction

L inux is truly amazing when you consider how it originated and how it continues to evolve. From its modest beginning as the hobby of one person — Linus Torvalds of Finland — Linux has grown into a full-fledged 32-bit operating system (64-bit on the DEC Alpha processor) with features that rival those of commercial x86 UNIX operating systems, such as Solaris and SCO UNIX. To top it off, Linux — with all its source code — is available free to anyone. All you must do is download it from an Internet site or get it on a CD-ROM for a nominal fee from one of many Linux CD vendors.

Linux certainly is an exception to the rule "you get what you pay for." Even though Linux is free, it is no slouch when it comes to performance, features, and reliability. The robustness of Linux has to do with the way it was developed. Many developers around the world collaborated over the Internet to add features. Incremental versions are continually being downloaded by users and tested in a variety of system configurations. Linux revisions go through much more rigorous beta testing than any commercial software does.

Since the release of Linux 1.0 on March 14, 1994, the number of Linux users around the world has grown exponentially. Based on data gathered by the Linux Counter (`http://counter.li.org/`), the estimated installed base is anywhere from 1 million to more than 5 million users worldwide. The same data provides some interesting statistics about Linux:

- Most Linux users are in United States, Canada, and Europe.
- The percentage of Linux users who use it at home is 87 percent; 35 percent use it at work; some use it both at work and home.
- The percentage of users who get Linux via FTP is 42 percent. About 40 percent of users buy Linux on a CD-ROM, typically from a vendor such as InfoMagic or Red Hat Linux.
- Slackware Linux continues to be the most popular Linux distribution: Red Hat and Debian are the other two prominent distributions.
- The percentage of users who run Linux on Intel 486 systems is 42 percent — 30 percent on Pentium processors. Systems typically have 8 to 32MB of memory and 500MB to 1.5GB disk.
- The percentage of Linux systems that use an Ethernet network is 50 percent, while 34 percent use dial-up networking with SLIP and PPP.
- Most Linux systems have anywhere from one to eight users.
- The percentage of Linux systems that are used as Internet servers (World Wide Web, FTP, Mail, firewall, and even as router) is 61 percent.

The Linux distributions — Red Hat, Slackware, and Debian — differ in the way each handles the installation process. Because installation is the most difficult step in getting started with Linux, it helps to select a Linux distribution that makes installation easy. Red Hat Linux excels when it comes to the installation process. Red Hat provides an installation program that guides new users through the crucial parts of the installation process, including disk partitioning. The ease of installation makes Red Hat Linux a good choice for all users, especially new users.

Red Hat Linux comes with extensive online information on topics such as installing and configuring the operating system for a wide variety of PCs and peripherals. Although expert users can manage to install and run Linux with the online documentation alone, new users find consulting a book (such as this one) helpful for detailed guidance on Linux installation. Typically, users also move to Linux with some specific purpose in mind (such as setting up a World Wide Web server or learning X programming).

Part I:

Configuring Your Linux System

Chapter 1: Installing Linux

Chapter 2: Upgrading Linux

Chapter 1

Installing Linux

In This Chapter

▷ Understanding the complete installation process for the Red Hat Linux CD-ROM that accompanies this book

▷ Checking what you need to know about your PC's hardware before installing Linux

▷ Performing initial installation steps under MS-DOS or Windows

▷ Installing Red Hat Linux from the companion CD-ROM

▷ Running Linux for the first time

Starting with the Intel 80386 processor, continuing with the 80486, and now with the Pentium and Pentium II, the computing power of PC processors continues to grow steadily. Processor clock speeds have increased from the 16MHz 80386 of a few years ago to the 450MHz Pentium II processors of today. The IBM PC-AT-based Industry Standard Architecture (ISA) bus is being supplanted by the high-performance Peripheral Component Interconnect (PCI) bus. The hardware performance of a modern, run-of-the-mill PC clearly is on a par with that of workstations, such as those from Hewlett-Packard and Sun.

When it comes to operating systems, however, PCs have not kept up with workstations. Many PCs still run 16-bit operating systems, such as MS-DOS or Windows 3.1, which do not fully exploit the 32-bit processing capabilities of the PC's processors. Workstations, on the other hand, run UNIX — a multitasking and multiuser operating system (this means the operating system can run several programs simultaneously and support more than one user at a time). Typically, workstations also use the X Window System for a graphical user interface.

Differences between 16-bit and 32-bit operating systems

Intel 80386, 80486, and Pentium processors have 32-bit registers and can process data items 32 bits at a time. But 16-bit operating systems, such as MS-DOS and Windows 3.1, use only the 16 low-order bits of the 32-bit registers (see the figure) and work with data items in 16-bit chunks.

```
                        A 32-bit register
|<------------------------------------------------------------>|
                                    |<------- 16 Low-order bits ------->|
 _____
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾
 31            24 23            16 15             8 7            0
                              bit positions
 └─ Most significant bit                  Least significant bit ─┘
```
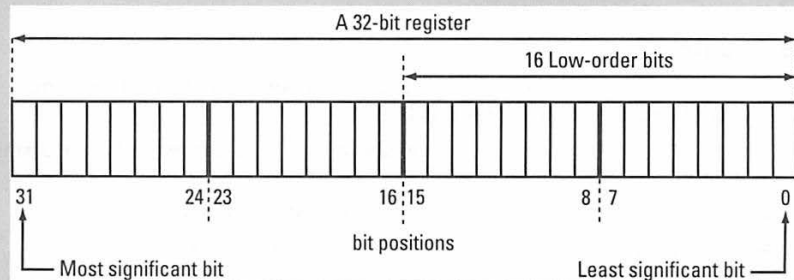
Thirty-two-bit operating systems, such as Linux and Windows NT, exploit the 32-bit registers and process data 32 bits at a time. You need a 32-bit operating system, such as Linux, to exploit the capabilities of your PC's 32-bit processor.

UNIX has been available for PCs for some time, but you had to pay nearly as much for a fully configured commercial PC UNIX operating system as you did for the PC itself. This situation changed when Linus Torvalds of the University of Helsinki in Finland decided to build a UNIX-like operating system for the PC. What started as a simple task-switching example, with two processes that printed AAAA... and BBBB... on a dumb terminal, has grown into a full-fledged multitasking and multiuser operating system that rivals commercially available UNIX systems for Intel 80×86 systems. Many programmers around the world contributed code and collaborated to bring Linux to its current state. With the release of Version 1.0 in March 1994, Linux became an operating system of choice for UNIX enthusiasts, as well as for people who are looking for a low-cost UNIX platform for a specific use, such as developing software or running an Internet host.

After you overcome your initial fear of the unknown and install Linux, you will see how you can use Linux to turn your PC into a UNIX workstation. The best part is you can get Linux free — just download it from one of several Internet sites. The best way for beginners and experts alike to get started, though, is to buy a book (such as this one) that comes with a Linux CD-ROM. This complete guide to Linux starts with installation and moves on to specific tasks (such as developing software or connecting to the Internet) that you may want to perform with your Linux PC.

Note

Installation can be one of the tricky steps in Linux, especially if you have a no-name IBM-compatible PC. You need some specific information about hardware, such as the type of disk controller, video card, and CD-ROM drive. Linux controls the hardware through drivers, so you need to make sure the current release of Linux includes drivers for your hardware. Because Linux is free, you cannot demand — or expect — support for some specific hardware. Linux is, however, continually growing through collaboration among programmers throughout the world. If your hardware is popular enough, a good chance exists that someone has developed a driver for it. In any case, the version of Red Hat Linux on the companion CD-ROM already supports such a wide variety of hardware that all your PC's peripherals probably are supported.

To get you started on your Linux experience, this chapter shows you how to install Linux from the companion CD-ROM. The chapter starts with an overview of the entire installation process; then it guides you step by step through the installation process.

Cross-Reference

If you have installed Linux already, Chapter 2 shows you how to configure and upgrade Linux to take advantage of fixes or enhancements for the Linux operating system.

Understanding the Linux Installation Process

Before starting a big job, I always find it helpful to visualize the entire sequence of tasks I must perform. The process is similar to studying a map before you drive to a place where you have never been. Linux installation can be a big job, especially if you run into snags. This section shows you the road map for the installation process. After reading this section, you should be mentally prepared to install Linux.

The exact steps for installing Linux may depend on the Linux *distribution* — the exact packaging of the operating system — you are using. This book shows you the installation steps for the companion Red Hat CD-ROM. Here are the general steps for installing Red Hat Linux:

1. Gather information about your PC's hardware before you install Linux. Linux accesses and uses various PC peripherals through software components called *drivers*. You have to make sure the version of Linux you are about to install has the necessary drivers for your system's hardware configuration. Conversely, if you do not have a system yet, look at the list of hardware supported by Linux and make sure you buy a PC with components that Linux supports.

2. You may have to perform a process known as *partitioning* to allocate parts of your hard disk for use by Linux. If you are lucky enough to have a spare hard disk, you may decide to keep MS-DOS and Windows on the first hard disk and install Linux on the second hard disk. With a spare

second disk you needn't worry about partitioning under DOS or Windows. If you have only one hard disk, however, you must partition that disk into several parts. Use a part for DOS and Windows, and leave the rest for Linux.

3. Under DOS or Windows, create a Linux boot disk (or just a boot disk). The *boot disk* is used to boot your PC and start an initial version of the Linux operating system. If you boot your PC under MS-DOS (it has to be MS-DOS only; not an MS-DOS window in Windows 95), you can skip this step — instead, you can boot Linux directly from the CD-ROM.

4. Boot your PC with the Linux boot disk (or start MS-DOS and run a command from the CD-ROM). This procedure automatically runs the Red Hat Linux installation program. From this point on, you will respond to a number of dialog boxes as the Red Hat installation program takes you through the steps.

5. Respond to a dialog box that asks you if you have a color monitor. From subsequent dialog boxes, you select the keyboard type and indicate where Red Hat Linux is located (in the CD-ROM drive).

6. Prepare the hard disk partitions where you plan to install Linux. If you have created space for Linux by reducing the size of an existing DOS partition, now you have to create the partitions for Linux. Typically, you need at least two partitions: one for the Linux files and the other for use as the *swap partition*, a form of virtual memory. To perform this step, the installation program gives you the option of using the Linux fdisk command or Red Hat's new disk management utility called *Disk Druid*.

7. Format the hard disk partitions, indicate which partition is the swap partition, and specify the partition where to install Linux (this is called the *root partition*). You also get a chance to mount other disk partitions. Mounting a partition associates a physical disk partition with a directory in the Linux file system.

8. Select various software components to install. Each component represents a part of Linux — from the base operating system to components such as the emacs editor, programming tools, the Perl scripting language, and the X Window System (a graphical windowing system). You simply select the components you need and let the Red Hat installation program do its job.

9. If you installed the X Window System, the installation program configures the mouse and then runs the Xconfigurator program that creates a configuration file (/etc/X11/XF86Config) used by the X Window System. In response to dialog boxes presented by the Xconfigurator program, you provide information about your video card and monitor.

10. If you installed the networking software, the installation program enables you to configure the network. You specify a number of parameters including an IP address, host name, and domain name.

11. Specify the local time zone.

12. Configure any printers you have. You can configure a printer connected directly to the PC or a remote printer on the network. Red Hat even enables you to configure a printer connected to a PC running Windows 95 or Windows NT.

13. Select a root password. The root user is the *super user*—a user who can do anything—in Linux.

14. Install the Linux Loader (LILO) program on your hard disk so you can boot Linux when you power up your PC after shutting it down.

15. If you find that Linux does not work properly with one or more of your system components (such as the CD-ROM drive or sound card), you may have to reconfigure the Linux operating system to add support for those system components.

The following sections guide you through the basic installation steps and the initial booting of Linux.

Your PC must have a CD-ROM drive—one supported by Linux—to install Linux from this book's companion CD-ROM. If your PC does not have a CD-ROM drive, but your PC is on a network, you may use another PC's CD-ROM drive and install Linux over the network using NFS or FTP.

Preparing Your PC for Linux Installation

Before you install Linux, you should prepare your PC for the installation. You can be in either of two situations:

■ You already have a PC that runs one of the popular PC operating systems, such as MS-DOS, Microsoft Windows 3.1, Windows 95, Windows NT, or OS/2.

■ You are about to buy a PC and you plan to run Linux on that PC at least some of the time.

If you are about to purchase a PC, you are lucky, because you can get a PC configured with peripherals that Linux supports. To pick a Linux-compatible configuration, all you must do is consult the current list of hardware Linux supports and then select a PC that includes only supported hardware components. You may have to ask the vendor explicitly for detailed information about peripherals such as the video card, CD-ROM drive, and networking card to ensure you can use the peripherals under Linux. Selecting a PC with Linux-supported hardware greatly minimizes the potential for problems installing Linux. The next few sections list hardware Linux supports.

You can also buy a PC with Linux already installed. You'll find advertisements for such Linux workstations in the *Linux Journal* (see Appendix C).

If you want to install Linux on an existing PC, verify the latest Linux distribution supports all the hardware on your PC. In other words, you have to take an inventory of your PC's hardware components and determine whether Linux currently supports any of them.

Taking stock of your PC's components

Like many other operating systems, Linux supports various types of hardware through device drivers. For each type of peripheral device, such as a networking card or a CD-ROM drive, Linux needs a driver. In fact, each kind of peripheral needs a separate driver. Because Linux is available free (or relatively inexpensively) and because many programmers scattered throughout the world cooperate to develop Linux, you cannot demand support for a specific kind of hardware. Your best bet is to hope someone who can write a Linux driver has the same hardware you do. In all likelihood, this person will write a driver, which eventually will find its way into a version of Linux; then you can use that hardware under Linux.

Check the list of hardware the version of Linux on the companion CD-ROM supports (I summarize this list in the next few sections). You can install and run Linux even if no Linux drivers are available for certain peripherals. At minimum, however, to install Linux from the companion CD-ROM, you must have a Linux-compatible processor, bus type, hard disk, keyboard, and CD-ROM drive. If you want to run the X Window System, you also must ensure that XFree86 (the *X* server for Linux) supports the mouse, the video card, and the monitor. (Chapter 4 tells you all about *X*.)

The following sections provide an overview of the hardware the version of Linux on the companion CD-ROM supports.
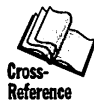
Cross-Reference

Chapters 9 through 17 cover all PC hardware in detail. Turn to those chapters for more information on whether Linux supports your system's unique hardware configuration. In Chapters 9 through 17, you also can find information on how to get the most from your PC's hardware under Linux.

Processor

The *processor* is the central processing unit (CPU) — the integrated circuit chip that performs all the processing in the PC. At minimum, you need an Intel 80386 processor to run Linux. Any Intel 80386-, 80486-, and Pentium-compatible processor can run the Linux operating system. Among the compatibles, Linux can run on AMD K5, K6, and Cyrix and IBM processors.

Note

You cannot run Linux on an 80286 PC; the 80286 is not a 32-bit processor.

Bus

The *bus* is the standard electrical connection between the processor and its peripherals. Several types of PC buses exist. The most popular bus is the Industry Standard Architecture (ISA) bus, formerly called the *AT bus* because IBM introduced it in the IBM PC-AT computer in 1984. Other buses include Extended Industry Standard Architecture (EISA); VESA Local Bus (VLB); Micro Channel Architecture (MCA); and, most recently, Peripheral Component Interconnect (PCI).

Note

Linux currently supports all common PC buses. Support for the MCA bus (used in IBM PS/2 computers) first appeared in Linux kernel Version 2.0.7 (Chapter 2 explains the kernel version numbers).

Memory

Commonly referred to as *random access memory* (RAM), memory is not a factor in compatibility. You need at least 8MB of RAM to get good performance, however. Although you may be able to install and run Linux on a PC that has 4MB RAM, you cannot run the X Window System on that PC. The X Window System manages the graphical interface through an *X server*, which is a large program that needs a great deal of memory to run efficiently.

Secret

If you are buying a new PC, you should get at least 16MB of RAM. If you have an old PC with less than 8MB of RAM, you may want to add more memory to bring the total up to 16MB. The more physical memory a system has, the more efficiently it runs multiple programs because the programs can all fit in memory. Although Linux can use a part of the hard disk as virtual memory, such disk-based memory is much slower than physical memory. The amount of physical memory required depends on the size of the Linux operating system and any other software you have to run all the time, such as the *X* Window System. Although Linux alone would run on 4MB of memory, you need at least 8MB to run *X*. Add to this any applications (such as editor or compiler) that you might run and you'll soon see why you need at least 16MB of RAM for adequate performance.

Video card and monitor

Most PCs have what is known as *Super VGA* (Video Graphics Array) video cards. Linux works fine with all video cards in text mode. But when it comes to XFree86 — the Linux version of the X Window System — the story is quite different. If XFree86 does not support your video card explicitly, you have to work hard to get XFree86 configured for your video card.

The kind of monitor you use is not particularly critical, but it must be capable of displaying at the screen resolutions the video card uses, which are expressed in terms of the number of picture elements, or *pixels*, horizontally and vertically (such as 1024×768).

Generally, XFree86's support for a video card depends on the *video chipset* — the integrated circuit that controls the monitor and causes the monitor to display output. A video-card manufacturer, however, may use a video chipset in a nonstandard manner. In such a case, you need a special version of XFree86 to support that video card. To help you select a video card, following is a list of video cards XFree86 Version 3.3.1 (the version included on the companion CD-ROM) supports:

- ATI Mach8, ATI Mach32, and ATI Mach64
- ATI VGA Wonder series
- Compaq AVGA
- DEC 21030
- Diamond Viper VLB and Viper PCI
- Enhanced Graphics Adapter (EGA)
- Genoa GVGA

- Hercules monochrome
- Hyundai HGC-1280 monochrome
- IBM 8514/A, XGA, and XGA-II
- Matrox MGA2064W (Millennium) and Matrox MGA1064SG (Mystique)
- Number Nine Imagine I128
- NVidia NV1
- Orchid P9000
- Sigma LaserView PLUS monochrome
- Video 7 (also known as Headland Technologies HT216-32)
- Video cards based on the Advance Logic AL2101, 2228, 2301, 2302, 2308, and 2401 chipsets
- Video cards based on Alliance AP6422 and AT24 chipsets
- Video cards based on the ARK Logic ARK1000PV, ARK1000VL, ARK2000PV, and ARK2000MT chipsets
- Video cards based on the Chips & Technologies 64200, 64300, 65520, 65525, 65530, 65535, 65540, 65545, 65546, 65548, 65550, and 65554 chipsets
- Video cards based on the Cirrus Logic CLGD5420, 542x, 5430, 5434, 5436, 544x, 546x, 5480, 62x5, 6420, 6440, 754x, and 7555 chipsets (x is any digit)
- Video cards based on the IIT AGX-010, 014, 015, and 016 chipsets
- Video cards based on the MX68000 and MX68010 chipsets
- Video cards based on the NCR 77C22, 77C22E, and 77C22E+ chipsets
- Video cards based on the OAK OTI-067, OTI-077, and OTI-087 chipsets
- Video cards based on the RealTek RTG3106 chipset
- Video cards based on the S3 732 (Trio32), 764 (Trio64), Trio64V+, 801, 805, 864, 866, 868, 86C325 (ViRGE), 86C375 (ViRGE/DX), 86C385 (ViRGE/GX), 86C988 (ViRGE/VX), 911, 924, 928, 964, and 968 chipsets (see Chapter 4 for a list of supported S3 cards)
- Video cards based on the SGS-Thomson STG2000 chipset
- Video cards based on the SiS 86c201, 86c202, and 86c205 chipsets
- Video cards based on the Weitek P9000 chipset
- Video cards based on the Trident 8800CS, 8200LX, 8900x, 9000, 9000i, 9100B, 9200CXr, 9320LCD, 9400CXi, 9420, 9420DGi, 9430DGi, 9440, 96xx, and Cyber938x chipsets (x represents any digit)
- Video cards based on the Tseng ET3000, ET4000, ET4000AX, ET6000, W32, W32i, and W32p chipsets
- Video cards based on the Western Digital WD90C00, WD90C10, WD90C11, WD90C24, WD90C31 and WD90C33 chipsets

- Video Graphics Array (VGA)

- Western Digital Paradise PVGA1

For the basic Linux installation, you needn't worry about the video card. Detailed information about the video card becomes important when you want to configure XFree86.

Cross-Reference

Although you go through an *X* configuration step during Red Hat Linux installation, you can find further details on configuring XFree86 in Chapter 4.

Hard drive

Linux supports any hard drive your PC's Basic Input and Output System (BIOS) supports. In many older 386 and 486 PCs, you had to use a separate driver to access large hard drives — the system BIOS could not handle these drives. You can't install Linux on such systems. In short, Linux supports your hard drive only if the system BIOS supports the hard drive without any additional drivers with one significant restriction. To be able to boot Linux from a large hard drive (that's any drive with more than 1,024 cylinders), the Linux Loader (LILO), the Linux kernel, and the LILO configuration files must be located in the first 1,024 cylinders of the drive. This is because the Linux Loader uses BIOS to load the kernel and the BIOS cannot access cylinders beyond the first 1,024.

For hard drives connected to your PC through a SCSI controller card, Linux must have a driver that enables the SCSI controller to access and use the hard drive. A summary of supported SCSI controllers appears in the "SCSI Controllers" section of this chapter.

Tip

The only remaining decision about the hard drive is its capacity. If you have an old PC, you may have a relatively low-capacity hard drive — perhaps as low as 500MB. Although you can install Linux and X Window System within 400MB, doing so does not leave much room for Windows 95, should you want to keep it. Therefore, if your old PC has an IDE interface and one small hard disk (500MB or smaller), you may want to add a second hard drive because most IDE controllers can support two hard drives. If you have a SCSI card, you can connect up to seven SCSI devices to it.

If you are buying a new PC, remember a complete Linux installation (with the all the Red Hat packages) takes nearly 400MB of disk space. On top of that, you need some disk space for your work. Luckily, most new PCs nowadays come with disk sizes ranging anywhere from 1.6GB to 6GB. Any of these disk sizes is large enough, so you can keep Windows on one of the partitions and have the option of booting either Windows or Linux.

Tip

Consider buying a second hard drive for Linux. A second drive makes the installation process considerably less risky, because you needn't partition the drive on which Windows 95 may be installed.

The PC BIOS

All IBM-compatible PCs come with a BIOS built into read-only memory (ROM). The BIOS contains a set of input/output (I/O) functions for accessing the PC's peripheral devices, such as the keyboard, display, printer, serial port, and floppy disk or hard drive.

The BIOS is essentially software stored on ROM, and as such, PC vendors revise and update the BIOS just like any software. Typically, the BIOS is revised to handle new devices, such as new hard drives with much larger capacity than originally envisioned, or even new bus types, such as PCI. BIOS revisions may also improve performance by doing various tasks more efficiently.

Because the BIOS is a crucial element in getting Linux to work with a PC's peripherals, you might consider a BIOS upgrade to get Linux going on an older 386 or 486 PC. The upgrade process involves replacing a pair of chips with new ones — you must contact your PC's manufacturer to get new revisions of the BIOS chips compatible with your PC.

Another reason to upgrade your PC's BIOS (and probably the CMOS Real-Time Clock as well) is the Year 2000 problem where the PC fails to maintain the correct year once 2000 rolls in.

Floppy disk drive

As they do for the hard drive, Linux drivers use the PC BIOS to access the floppy disk drive. Therefore, your floppy disk drive is compatible with Linux. You may, though, have to boot Linux from a floppy disk during the installation. For this purpose, you need a high-density 5.25" (1.2MB-capacity) or 3.5" (1.44MB-capacity) floppy disk drive. You can avoid booting from a floppy provided you can boot your PC under MS-DOS (not an MS-DOS window under Windows 95) and you can access the CD-ROM from the DOS command prompt.

Keyboard and mouse

Linux supports any keyboard that already works with your PC, but the mouse needs explicit support in Linux. You need a mouse if you want to configure and run XFree86, the X Window System for Linux.

Linux supports the following popular mice:

- Microsoft serial mouse
- Mouse Systems serial mouse
- Logitech Mouseman serial mouse
- Logitech serial mouse
- Logitech bus mouse
- PS/2 (auxiliary device) mouse
- Microsoft bus mouse
- ATI XL Inport bus mouse
- QuickPort (C&T 82C710) mouse (used on TI Travelmate and Toshiba laptop PCs)

SCSI controller

The Small Computer System Interface, commonly called *SCSI* (and pronounced *scuzzy*), is a standard way of connecting many types of peripheral devices to a computer. SCSI is in many kinds of computers — from high-end UNIX workstations to PCs. To use a SCSI device on your PC, you need a SCSI controller card that plugs into one of the connector slots on your PC's bus.

Typically, you connect hard drives and CD-ROM drives through a *SCSI controller.* A single controller enables you to connect up to six SCSI devices to your PC. If you want to access and use a SCSI device under Linux, you have to make sure Linux supports your SCSI controller card.

The Linux release on the companion CD-ROM already supports the following popular SCSI controllers:

- Acculogic ISApport based on the NCR 53c406a chipset
- Adaptec AHA-1510 and AHA-152x (based on AIC-6260 or AIC-6360 chips; x is any digit) controllers for the ISA bus
- Adaptec AHA-154x (x is any digit) controllers for the ISA bus
- Adaptec AHA-174x (x is any digit) controllers for the EISA (Extended Industry Standard Architecture) bus
- Adaptec AHA-274x controller for the EISA bus and AHA-284x controller for the VLB (VESA Local Bus); both cards use AIC-7770 chips
- Adaptec AHA 2920 controller
- Adaptec AHA-2940 and AHA-3940 controllers for the PCI bus; these cards use the AIC-7870 chip
- Adaptec AVA-1505 and AVA-1515 controllers for the ISA bus (supported by the Adaptec 152x driver in Linux)
- Always IN2000
- AMI Fast Disk VLB/EISA (supported by the BusLogic driver in Linux)
- BusLogic SCSI controllers for ISA, EISA, VLB, and PCI (all models)
- DPT PM2001 and PM2012A controllers
- DPT Smartcache controllers for ISA, EISA, and PCI buses, including models PM2011, PM2021, PM2041, PM3021, PM2012B, PM2022, PM2122, PM2322, PM2042, PM3122, PM3222, PM3332, PM2024, PM2124, PM2044, PM2144, PM3224, and PM3334
- DTC 329x controller for the EISA bus (supported by the Adaptec 154x driver in Linux)
- Future Domain TMC-16x0 and TMC-3260 (PCI) SCSI controllers
- Future Domain TMC-8xx and TMC-950
- ICP-Vortex PCI-SCSI Disk Array Controllers including models GDT6111RP, GDT6121RP, GDT6117RP, GDT6127RP, GDT6511RP, GDT6521RP, GDT6517RP, GDT6527RP, GDT6537RP, and GDT6557RP

- ICP-Vortex EISA-SCSI Controllers including models GDT3000B, GDT3000A, GDT3010A, GDT3020A and GDT3050A
- Media Vision Pro Audio Spectrum 16 SCSI controller for ISA bus
- Media Vision Premium 3D SCSI based on the NCR 53c406a chipset
- NCR 5380 generic SCSI cards
- Qlogic FAS408 controller
- Quantum ISA-200S and ISA-200MG
- SCSI controllers based on the NCR 53c7x0 and 53c8x0 (for PCI bus) chipsets
- Seagate ST-01 and ST-02 controllers for ISA bus
- Sound Blaster 16 SCSI controller for ISA bus (supported by the Adaptec 152x driver in Linux)
- Tekram DC-390, DC-390W/U/F
- Trantor T128, T128F, and T228 controllers for ISA bus
- Trantor T130B based on the NCR 53c400 chipset (supported by the NCR 5380 driver)
- UltraStor 14F (for ISA bus), UltraStor 24F (for EISA bus), and UltraStor 34F (for VLB bus)
- Western Digital WD7000 SCSI controllers

Tip This list keeps growing as Linux developers add support for new SCSI controllers. For a more complete list, check the online documentation included on the CD-ROM. The "Looking up the online documentation" section at the end of this chapter shows you how to find and use the online documentation.

CD-ROM drive

CD-ROM (Compact Disc Read-Only Memory) drives are popular because each CD-ROM can hold up to 650MB of data. This is a relatively large amount of storage compared to a floppy disk. CD-ROMs are reliable and inexpensive to manufacture. Vendors can use a CD-ROM to distribute a large amount of information at a reasonable cost.

This book provides Linux on a CD-ROM, so you need a CD-ROM drive to install the software. Most new PCs come with a CD-ROM drive. If the basic configuration does not include a CD-ROM drive, you can add one at a fraction of the cost of your PC — usually around a hundred U.S. dollars. If you have an older PC that doesn't have a CD-ROM drive, you need to buy one to install Linux from this book's companion CD-ROM.

CD-ROM drives became popular on PCs over the past few years as users went for the multimedia experience. As you may know, in the context of the PC,

multimedia refers to the use of multiple media — sound, images, animation, and video — in software applications. The sound card and CD-ROM drive were the two common elements of all multimedia software.

The combination of sound cards and CD-ROM drives has been so popular that many sound cards (such as Creative Labs' Sound Blaster Pro) were sold with a CD-ROM drive. You had to connect the CD-ROM drive with a cable to the sound card, which included the appropriate hardware connector. Linux supports CD-ROM drives (such as the Sound Blaster Pro CD) that connect to a sound card.

Nowadays, most CD-ROM drives use the Enhanced Integrated Drive Electronics (EIDE) interface that also connects hard disk drives to the PC. Linux supports all EIDE CD-ROM drives.

Some CD-ROM drives are SCSI devices that connect to a SCSI controller card. Linux supports a SCSI CD-ROM drive as long as it has a driver for the SCSI controller.

Following are some of the common CD-ROM drives Linux supports:

- Any EIDE CD-ROM drive (also referred to as AT Attachment Packet Interface, or ATAPI drives)
- Any SCSI CD-ROM drive that can transfer data in blocks of 512 or 2,048 bytes (this includes most of the CD-ROM drives on the market)
- Aztech CDA268, Orchid CDS-3110, and Okano/Wearnes CDD-110, Conrad TXC, CyCDROM CR520ie, CyCDROM CR540ie, CyCDROM CR940ie
- Creative Labs CD-200(F)
- Funai E2550UA/MK4015
- GoldStar R420
- IBM External ISA CD-ROM drive
- Lasermate CR328A
- LMS Philips CM 206
- Longshine LCS-7260
- Matsushita/Kotobuki/Panasonic CD-ROM drive models CR-521, CR-522, CR-523, CR-562, and CR-563 (the CD-ROM drives bundled with the Sound Blaster Pro sound card)
- MicroSolutions Backpack parallel port drive
- Mitsumi CD-ROM drive (CR DC LU05S, FX001D/F)
- Optics Storage Dolphin 8000AT
- Sanyo H94A
- Sony CDU31A, CDU33A, CDU-510, CDU-515, CDU-531, and CDU-535 drives
- Teac CD-55A SuperQuad

Tip

The Sound Blaster 16 sound card features one of two CD-ROM interfaces: a proprietary interface and a SCSI interface. Linux supports both interfaces, but you need to know which interface your Sound Blaster 16 board provides. You should find this information in the manual that accompanies the Sound Blaster 16 board.

Note, you can connect an ATAPI CD-ROM drive to an IDE controller and treat it as a second hard drive. If you want a simple way to attach a CD-ROM drive to your PC and you don't need an additional hard drive, you might consider installing an ATAPI CD-ROM drive as a slave of the boot hard drive.

Sound card

On PCs, sound cards and CD-ROM drives go hand in hand because most CD-ROM-based multimedia programs include sound effects you can enjoy only if you have a sound card. Under Linux, you also can play sound on the sound card. If you have a sound card, you can play audio CDs or play Doom (a popular game) with full sound effects.

The version of Linux on the companion CD-ROM supports the following sound cards:

- 6850 UART (Universal Asynchronous Receiver Transmitter) MIDI (Musical Instrument Digital Interface)
- Adlib (OPL2) sound card
- ATI Stereo F/X (Sound Blaster-compatible)
- Audio Excell DSP16
- Aztech Sound Galaxy NX Pro
- Crystal CS4232 (PnP — Plug and Play) based cards
- ECHO-PSS (Orchid SW32 and Cardinal DSP16)
- Ensoniq SoundScape (but you must start DOS to initialize card)
- Gravis Ultrasound, Ultrasound MAX, and Ultrasound 16-bit sampling daughterboard
- Logitech SoundMan 16 (PAS-16 compatible), SoundMan Games, and SoundMan Wave (Jazz16/OPL4)
- MediaTriX AudioTriX Pro
- Media Vision Premium 3D Jazz16 (Sound Blaster Pro-compatible), Pro Audio Spectrum 16 (PAS-16), and Pro Sonic 16 Jazz
- Microsoft Sound System (AD1848)
- MPU-401 MIDI
- OAK OTI-601D cards (Mozart)
- OPTi 82C928/82C929 cards (MAD16/MAD16 Pro/ISP16/Mozart)

- Sound Blaster, Sound Blaster 16, and Sound Blaster Pro
- Sound Galaxy NX Pro
- ThunderBoard (Sound Blaster-compatible)
- Turtle Beach Wavefront cards (Maui, Tropez)
- WaveBlaster and other Sound Blaster 16 daughterboards
- Cards based on the ESS Technologies AudioDrive chips such as 688, 1688, and 1868

Network adapter

A network adapter is necessary only if you are going to connect your Linux PC to an Ethernet network. Linux supports a variety of Ethernet network adapters. Arcnet and IBM's Token Ring network are also supported.

See Chapter 16 for more information on Linux's support for token ring and other network adapters.

Cross-Reference

Following is a list of Ethernet cards Linux supports:

- 3Com 3C503, 3C505, 3C507, 3C509, 3C509B (ISA bus), and 3C579 (for EISA bus)
- 3Com Etherlink III Vortex Ethercards (3C590, 3C592, 3C595, 3C597) for PCI bus, 3Com Etherlink XL Boomerang Ethercards (3C900, 3C905) for PCI bus and 3Com Fast EtherLink Ethercard (3C515) for ISA bus
- Ethernet cards based on AMD LANCE (79C960) chip for ISA and PCI
- Accton parallel port Ethernet adapter
- Allied Telesis AT1500, AT1700, and LA100PCI-T
- Ansel Communications AC3200 EISA
- Apricot Xen-II
- AT&T GIS WaveLAN
- AT-Lan-Tec and RealTek parallel-port Ethernet adapter
- Cabletron E21xx (x is any digit)
- Cogent EM110
- D-Link DE600 and DE620 parallel-port Ethernet adapter
- Danpex EN-9400
- Digital Equipment Corporation (DEC) DEPCA, EtherWORKS, EtherWORKS 3, DE425 (EISA), DE434 (PCI), DE435 (PCI), DE450, DE500, DE450-XA, DE500-XA, and DEC QSilver
- Fujitsu FMV-181, FMV-182, FMV-183, FMV-184

- Hewlett-Packard HP J2405A, PCLAN (27245 and 27xxx series), PCLAN PLUS (27247B and 27252A), and 10/100VG PCLAN (J2577, J2573, 27248B, J2585) (ISA, EISA, and PCI)

- ICL EtherTeam 16i / 32 (EISA)

- Intel EtherExpress and EtherExpress Pro

- KTI ET16/P-D2, ET16/P-DC ISA

- New Media Ethernet

- Novell Ethernet NE1000, NE1500, NE2000, and NE2100 (all clones may not work)

- PureData PDUC8028 and PDI8023

- Racal-Interlan NI5210 (based on the i82586 Ethernet chip), NI6510 (uses the AMD 7990 LANCE chip and does not work with more than 16MB of memory)

- SEEQ 8005

- Schneider & Koch G16

- SMC (Western Digital) WD8003, WD8013, SMC Elite, SMC Elite Plus, SMC Elite 16 Ultra, SMC EtherEZ (ISA), SMC 9000 Series, SMC PCI EtherPower 10/100

- Zenith Z-Note and IBM ThinkPad 300 built-in Ethernet adapter

- Znyx 312 Etherarray

Tip   If you plan to use Linux on a stand-alone PC at home, you can use Serial Li Internet Protocol (SLIP) or Point-to-Point Protocol (PPP) to connect to the Internet over a dial-up connection through an Internet service provider (IS See Chapter 18 for details.

Making a hardware checklist

Now that you have seen a summary of various hardware peripherals Linux supports, you should have a rough idea of whether you have the right PC hardware to use Linux. If you are buying a new PC to run Linux, the hardware list should help you decide the hardware configuration of the new PC.

To summarize, go through the following checklist to see whether you are ready to install Linux from this book's companion CD-ROM:

- Does your PC have an 80386 or better processor, with the ISA, EISA, VLB, MCA, or PCI bus; at least 8MB of RAM; a high-density floppy disk drive; and a large hard drive (at least 500MB)? Remember, if you plan to run both Windows 95 and Linux on your PC, you need at least 1GB of disk space.

■ Does your PC have a CD-ROM drive that Linux supports? (You need a CD-ROM drive to install Linux from this book's companion CD-ROM.)

Tip

If you don't know what kind of CD-ROM drive you have, you should watch the system boot up in DOS and watch the DOS driver load — the driver usually displays a message with the brand of the CD-ROM drive.

■ Can you get a second hard drive? (If so, you can install Linux on that hard drive. Installing Linux on a second drive prevents you from messing up your first hard drive, which usually has MS-DOS or Windows loaded on it.)

■ If you have a SCSI controller with any SCSI devices you want to use under Linux, is the SCSI controller supported by Linux?

■ Is your video card supported by Linux? (If not, you won't be able to set up and run the X Window System.)

■ Is your mouse supported by Linux? (If not, you won't be able to set up and run the X Window System.)

As the comments after the questions indicate, you do not necessarily have to answer each question Yes. You must answer the first two items Yes, however, because without that basic hardware configuration, Linux cannot run on your system.

Note

If you plan to install Linux on a second hard disk, you needn't go through the process of partitioning (dividing) your hard disk under MS-DOS. Skip the next few sections and proceed to create the Linux boot disk (see the section "Creating the boot disk"). Then you can boot Linux from the boot disk and partition the second hard drive for use under Linux.

Partitioning your hard drive under MS-DOS or Windows

If your PC has a single hard disk drive, chances are you have Microsoft Windows 95 (or Windows 3.1) installed on that drive. If your hard drive is at least 1GB, I recommend you keep Windows installed on your system even if you want to work mostly in Linux. After all, you have to perform some of the Linux installation steps under MS-DOS or Windows. Also, you can access the Windows files from Linux. You get the best of both worlds if you keep MS-DOS and Windows around when you install Linux.

Tip

If your new PC has one large hard disk (typically larger than 2GB) but there are two drives — C and D — this means the disk already has two partitions. In this case, you may simply want to use the extended partition for Linux.

Typically, your PC hard disk is set up as a single large drive, designated by the drive letter *C.* Unless you can scrounge up a second hard disk for your PC or if you already have a second disk you can spare, your first task is to divide your one and only hard disk to make room for Linux.

Working in MS-DOS or Windows before you install Linux

If you are a MS-DOS or Windows beginner, you may find it difficult to follow some of the Linux installation steps you must perform under MS-DOS. Following are some of the terms and concepts you should know to perform the necessary installation steps:

- *Boot floppy drive.* The first floppy drive (A) is the boot floppy drive. If you put a floppy disk in the A drive and turn on the power, your PC automatically tries to start from the floppy. (Many new PCs enable you to set the boot device; however, by default, nearly all PCs come configured with the first floppy drive as the boot device.) This feature is built into the computer and does not depend on what operating system is installed on your system's hard disk. In most new systems, the A drive is a 3.5-inch floppy drive. Many older systems, however, use a 5.25-inch floppy drive as the A drive. To install Linux, you need a high-density boot floppy drive.

- *Partitions.* A physical hard drive can be divided into several parts, each of which can be treated as a separate logical hard drive. Although most new PCs use the entire hard disk as a single drive, you can partition the disk into up to four sections, called the *four primary partitions*. To install Linux on your hard disk, you have to create at least two partitions for Linux — one partition for the Linux file system and the other for *swap space* (virtual memory in which the contents of the physical memory can be stored temporarily). If you want to keep Windows in one partition and also install Linux, you need three partitions.

- *Repartitioning a hard disk.* If your hard disk has only one partition, the process of creating more partitions is referred to as *repartitioning*. To repartition a disk, you

have to back up its contents. Then use the MS-DOS FDISK command to delete the old partition and create several new ones, and finally restore the old contents to one of the partitions. The Red Hat Linux distribution (on this book's CD-ROM) includes an MS-DOS program called *FIPS*, which you can use to repartition a hard disk without destroying the contents of the old partition. This utility is not guaranteed to work perfectly, however, so you still need to back up your important files before attempting to repartition the hard disk with FIPS.

- *Formatting a disk.* The MS-DOS FDISK program only defines a section of the physical disk to be used by an operating system such as MS-DOS. You still must prepare that section of disk before MS-DOS can use it to store files and directories. *Formatting* is the process of making a partition ready for a file system.

- *Directories and files.* In MS-DOS, a drive is divided into directories. Each directory, in turn, can contain other directories and files. The file is where the actual information is stored. The directories help you organize your documents and programs. All files for the Windows operating system, for example, usually are in the C:\WINDOWS directory. The directory C:\WINDOWS\SYSTEM contains some special files Windows needs. As you see, the backslash character is used as a separator between names of directories. The directory C:\ is known as the root directory; WINDOWS is a subdirectory of the root directory, and SYSTEM is a subdirectory of WINDOWS. Therefore, C:\WINDOWS\SYSTEM is two levels down from the root directory.

■ *Filenames.* An MS-DOS filename consists of a name of 1 to 8 characters, followed by a period and then an extension that can have 0 to 3 characters. README.TXT is an MS-DOS filename with the name README and the extension TXT. Executable programs have the COM or EXE extension, whereas DOC typically represents a document that can be opened by a word processing program.

■ *MS-DOS commands.* You use MS-DOS through commands you enter at a prompt.

The MS-DOS command interpreter displays a prompt (usually the current drive and directory name, followed by a greater than sign, such as C:\>). MS-DOS commands often have options that start with a slash. You can use the /S option with the FORMAT command, for example, to format and copy the system files to a disk. Note, you can use the MS-DOS commands even under Windows 95 in an MS-DOS window.

Steps to repartition hard disk

In the following sections, I assume your PC has a single hard disk on which MS-DOS is already installed (Windows may be installed as well, but this doesn't matter in the task you are about to perform). To repartition the disk, you must perform the following tasks:

1. Back up the contents of your hard disk.

   If you bought a new PC, you are lucky, because the hard disk should not have much data to back up.

2. Create an MS-DOS boot disk (in Windows 95, create a startup disk by using the Add/Remove Programs option in the Control Panel).

   You use this disk to start your PC and partition the hard disk.

3. Run FDISK from the disk and create the new partitions.

4. Format the MS-DOS partition and restore the files from the backup you created in Step 1.

The next four sections guide you through these steps.

Partitioning your hard disk under Windows 95

If your PC runs Windows 95, you still must follow the same steps to partition the hard disk as you would have under MS-DOS. The only difference is you first have to create a startup disk by using the Add/Remove Programs option in the Control Panel and then boot the PC with that startup disk. After this, you can run the FDISK program to create the new partitions.

Back up your hard disk

Backing up the hard disk takes a long time, but no safe way to repartition the hard disk exists without a backup. Red Hat Linux, however, comes with the FIPS utility program, which can partition a hard disk without destroying the data currently on the disk.

How you back up your hard disk depends on what you have on your hard disk. If you have a new PC, the hard disk probably contains Windows and any other software the vendor bundled with the system. If you have the original disks for Windows and the bundled software, you may decide to skip the backup and reinstall everything after you create the new partitions. If you have an old PC, you may decide to back up only the directories you cannot reinstall from original floppy disks or CD-ROM. If you have a word processing program, for example, you don't have to back up that program's directory, because you can always reinstall the program. All you need to backup are the directories that contain documents you had created with the word processing program.

To back up your hard disk, you can use the backup utility that comes with MS-DOS. In DOS Versions 5.0 and earlier, use the BACKUP utility to back up all or some of the directories of your hard disk. In DOS Versions 6.0 and later, use the MSBACKUP utility.

Create a bootable disk for MS-DOS

Usually, you turn on your PC's power and the PC automatically loads the operating system (MS-DOS or Windows 95, for example) from the first hard drive: the C drive. If you put a disk in the A drive and power up the PC, however, the PC loads the operating system from the disk. If the disk does not contain the operating system, you get this familiar message:

```
Non-System disk or disk error
Replace and strike any key when ready
```

This message tells you the PC tried to load, but could not find operating system files on the floppy disk. Usually, you remove the disk and press a key, and the PC boots from the hard drive.

To change the hard drive partitions, you actually want to boot the PC from the A drive. This way, you can be sure the hard drive is not in use when you change its partitions. To create a bootable disk, perform these steps:

1. Put a floppy disk in your A drive, and type the command **FORMAT A: /S**.

   MS-DOS prompts you with the following message:

```
Insert new diskette for drive A
and press ENTER when ready...
```

The contents of the disk are destroyed when you format the disk.

2. Press Enter, because you already have the floppy disk in the A drive.

   MS-DOS formats the disk and places the necessary operating-system files on the disk (that's what the /S option of the FORMAT command does). DOS also prompts you for a Volume label; you can press Enter in response. Finally, DOS asks whether you want to format another disk; press **N** to indicate you are done with formatting.

3. Copy other necessary files to the A drive.

   At minimum, you need to copy the FDISK.EXE program to the disk so you can use it to partition the hard disk and the FORMAT.COM program to format the new DOS partition. Use the following commands to copy the FDISK.EXE program to the disk in the A drive (text in parentheses is my comment):

```
C:
CD \WINDOWS\COMMAND  (DOS commands are in this directory)
COPY FDISK.EXE A:
COPY FORMAT.COM A:
```

You also may need other programs to restore the backup you created earlier. Those programs needn't be on the boot disk; just make sure you have a copy of them on disk somewhere.

4. Test the boot disk.

   Close all running programs, put the newly created boot disk in the A drive, and press Ctrl+Alt+Delete to reboot your PC. MS-DOS should start and you should see an A:\> prompt.

Repartition hard disk with FDISK

Now that you have successfully backed up your hard disk and prepared a bootable MS-DOS disk, you can repartition your hard disk. To begin this procedure, put the MS-DOS boot disk in drive A, and restart your PC (press Ctrl+Alt+Delete. Or, turn the power off and then on). When you see the A:\> prompt, type **FDISK** and press Enter. FDISK runs and displays a message about enabling support for large disks using a new file system called FAT32. Because FAT32 file system is not supported by many operating systems, including older versions of Windows 95, you should not enable this feature of FDISK. Press Enter to accept the default answer of No. Then, FDISK displays the screen shown in Figure 1-1.

```
                           MS-DOS Version 6
                        Fixed Disk Setup Program
                   (C)Copyright Microsoft Corp. 1983 - 1993

                             FDISK Options

      Current fixed disk drive: 1

      Choose one of the following:

      1. Create DOS partition or Logical DOS Drive
      2. Set active partition
      3. Delete partition or Logical DOS Drive
      4. Display partition information
      5. Change current fixed disk drive


      Enter choice: [1]




      Press Esc to exit FDISK
```

Figure 1-1: FDISK's opening screen

Caution

Back up your hard disk before you use FDISK to repartition it (refer to the previous section "Back up your hard disk"). When you alter the partitions, you cannot access the old data on the disk. You can run FDISK, view the disk partitions, and exit without damaging your hard disk, but enter the commands carefully — you don't want to wipe out your hard disk's contents accidentally.

Tip

FDISK is available in all versions of MS-DOS and in Windows 95. (In Windows 95, you can start your PC in MS-DOS mode and use the FDISK command to repartition the disk.) In this session with FDISK, you delete a partition and create a smaller DOS partition, leaving some disk space for the Linux partitions.

Use the following strategy to repartition your hard disk:

1. Select FDISK menu Option 4 to look at your hard disk's current partition information.

2. Select FDISK menu Option 3 to delete the DOS partition.

3. Select FDISK menu Option 1 to create a new, smaller DOS partition that leaves enough space for Linux (I discuss the size of partitions in the "Create New DOS Partition" section).

4. Select FDISK menu Option 2 and mark the newly created DOS partition active.

Later, you partition the rest of the disk under Linux as described in the section "Partitioning Your Hard Disk under Linux."

Check your hard disk's current partitioning information

Before you delete the partition, press 4 and then press Enter to view the current partition information. Figure 1-2 shows the FDISK screen for a typical disk that contains a single DOS partition for the entire disk.

```
                    Display Partition Information

Current fixed disk drive: 1

Partition Status   Type    Volume Label  Mbytes   System   Usage
  C: 1        A    PRI DOS   LNBPC-C       325     FAT16    100%




Total disk space is  325 Mbytes (1 Mbyte = 1048576 bytes)






  Press Esc to continue
```

Figure 1-2: FDISK screen showing typical partition information

Disk 1 refers to the first physical hard disk on your system. The PRI DOS entry in the Type field indicates the partition is a primary DOS partition — the partition that contains the files needed to boot MS-DOS. The entry in the Partition field tells you this is the C drive in MS-DOS. You probably have the same situation.

Delete the primary DOS partition

Press Esc to return to the FDISK main menu (refer to Figure 1-1). To delete the primary DOS partition, press **3** and then press Enter. FDISK displays another menu, shown in Figure 1-3.

```
                Delete DOS Partition or Logical DOS Drive

Current fixed disk drive: 1

Choose one of the following:

1.  Delete Primary DOS Partition
2.  Delete Extended DOS Partition
3.  Delete Logical DOS Drive(s) in the Extended DOS Partition
4.  Delete Non-DOS Partition


Enter choice: [ ]




  Press Esc to return to FDISK Options
```

Figure 1-3: FDISK screen showing the options for deleting a partition

You can delete several kinds of partitions. An extended partition, for example, is simply a partition that can be further subdivided into logical drives. Typically, however, when an entire disk is devoted to Windows, you have only a primary DOS partition to delete.

To delete the primary DOS partition, press 1 and then press Enter. FDISK displays the screen shown in Figure 1-4, requesting confirmation that you really want to delete the partition.

```
                    Delete Primary DOS Partition

Current fixed disk drive: 1

Partition  Status   Type    Volume Label  Mbytes   System   Usage
  C: 1        A    PRI DOS   LNBPC-C         325    FAT16     100%




Total disk space is  325 Mbytes (1 Mbyte = 1048576 bytes)




WARNING! Data in the deleted Primary DOS Partition will be lost.
What primary partition do you want to delete..? [1]


 Press Esc to return to FDISK Options
```

Figure 1-4: The FDISK screen is requesting confirmation before deleting the primary DOS partition.

Press Enter. FDISK deletes the partition and returns to the main screen shown in Figure 1-1.

Create a new DOS partition

In this step, you create the primary DOS partition again, but this time, you make it smaller. At this point you must decide how much disk space you want to leave aside for MS-DOS (and Windows) and how much you want to devote to Linux. The final choice depends on the total capacity of your hard disk and your planned use of Linux.

The following table shows a rough calculation of how much disk space you need:

Item	Disk Space (MB)
Complete Linux installation from companion CD-ROM	400
Linux swap space (for use in virtual memory)	16
User space for Linux (so you can work in Linux)	150
Windows	500
Total	1,066

If your PC has more than 16MB memory, make the swap space the same size as the amount of memory. Otherwise, set aside at least 16MB of disk space for swap space.

If you have a 1.2GB hard disk, allocate 500MB to the primary DOS partition and leave the rest for Linux. Unfortunately, with today's disk-hungry applications, you may find 500MB is inadequate for all your Windows applications. I assume you want to use Linux in earnest, so I recommend setting aside enough disk space for Linux.

If you have a bigger capacity disk (many new systems come with 2 or 4GB disks), remember these minimums and proportionately increase the sizes of the DOS and Linux partitions.

If you have an old PC with a smaller capacity disk — say, 500MB — you should either devote the entire disk to Linux or get a second disk, so you can keep Windows and still run Linux.

To create the new primary DOS partition, press 1 at the FDISK main screen (refer to Figure 1-1) and then press Enter. FDISK displays the menu shown in Figure 1-5.

```
                    Create DOS Partition or Logical DOS Drive

         Current fixed disk drive: 1

         Choose one of the following:

         1. Create Primary DOS Partition
         2. Create Extended DOS Partition
         3. Create Logical DOS Drive(s) in the Extended DOS Partition




         Enter choice: [1]




         Press Esc to return to FDISK Options
```

Figure 1-5: FDISK screen with options for creating a partition

Press Enter to indicate you want to create a primary DOS partition. FDISK asks you for the partition's size, in megabytes. Specify the amount of space for the DOS partition and press Enter. FDISK creates the primary DOS partition and returns to the main screen shown in Figure 1-1.

Make the DOS partition active

To boot from the primary DOS partition, you have to make it active. From FDISK's main screen (refer to Figure 1-1), press 2 and then press Enter. FDISK asks you for the partition you want to make active. This should be the primary DOS partition you just created. Press Enter to make this partition active.

Press Esc to quit FDISK. FDISK informs you it will restart the PC. Leave the DOS boot disk in the A drive and press Enter. The PC reboots and you again see the A:\> prompt.

Restore the MS-DOS partition

The new primary DOS partition will be your PC's C drive. Before you can use this drive, however, you have to format it. To format the C drive, type **FORMAT C: /S**. The FORMAT program displays a warning message. You can ignore it because you are formatting a newly created partition on your hard disk.

After the new C drive is formatted, you have to restore the contents of the hard disk. If you had backed up only your files and not the Windows files, you have to start by installing Windows on your newly formatted C drive. Typically, you have to boot the PC from the first Windows installation disk and follow the instructions.

After you have Windows installed on the C drive, you can use your backup program to restore all the files you previously backed up from the hard disk. To complete this step, you must follow the appropriate instructions for the backup program you used. If you used the MSBACKUP program in DOS Versions 6.0 or later, for example, you can use the same program to restore the files from the backup floppy disks.

Repartitioning with FIPS

If you understand the partitions as being sections of the hard disk, you may wonder whether a way exists to cordon off the unused part of a hard disk and make a new partition out of the unused part without destroying any existing data. This idea is behind a utility program called *FIPS* (The *First* Nondestructive *Interactive Partition Splitting* Program). FIPS can split an existing primary DOS partition into two partitions.

Tip

Although you have no guarantee FIPS will split a DOS partition successfully, you may consider using it to create room for Linux, especially if you have a brand-new PC with only DOS and Windows installed on the hard disk. In this case, even if something goes wrong with FIPS, you can reinstall Windows and the applications. I used FIPS to split the DOS partition on a new PC's hard disk and create room for Linux.

The FIPS.EXE program and related files are located in the \DOSUTIL subdirectory of the companion CD-ROM. To use FIPS, follow these steps:

1. For FIPS to work, all used areas of the disk must be contiguous or at least as tightly packed as possible. You can prepare the disk for FIPS by running a defragmenter. In MS-DOS 6.0 or later, use the program DEFRAG to defragment the disk. In Windows 95, click the right mouse button on the disk symbol in the Explorer window, select Properties from the pop-

up menu, then click the Properties tab, and click the Defragment Now button. You also should check the hard disk for errors by running a program such as Norton Disk Doctor (in Windows 95, use SCANDISK). If you happen to have it, use Norton Speed Disk to defragment the hard disk because it's significantly faster than the DEFRAG utility.

2. Create a bootable disk, using the command FORMAT A: /S.

3. Copy the following files from the CD-ROM to the formatted disk (the following example assumes D: is the CD-ROM drive):

```
COPY D:\DOSUTILS\FIPS.EXE A:
COPY D:\DOSUTILS\RESTORRB.EXE A:
COPY D:\DOSUTILS\FIPSDOCS\ERRORS.TXT A:
```

FIPS.EXE is the program that splits partitions. ERRORS.TXT is a list of FIPS error messages. You consult this list for an explanation of any error messages FIPS displays. RESTORRB.EXE is a program that enables you to restore certain important parts of your hard disk from a backup of those areas created by FIPS.

4. Leave the bootable disk in the A drive and press Ctrl+Alt+Delete to reboot the PC. The PC boots from A and displays the A\> prompt.

5. Type **FIPS**. The FIPS program runs and shows you information about your hard disk. FIPS gives you an opportunity to save backup copy of important disk areas before proceeding. After that, FIPS shows the first free cylinder where the new partition can start (as well as the size of the partition, in megabytes). Figure 1-6 shows the output of the FIPS program at this stage.

```
Sectors per FAT: 256
Sectors per track: 63
Drive heads: 255
Hidden sectors: 63
Number of sectors (long): 4192902
Physical drive number: 80h
Signature: 29h

Checking boot sector ... OK
Checking FAT ... OK
Searching for free space ... OK

Do you want to make a backup copy of your root and boot sector before
proceeding (y/n)? y
Do you have a bootable floppy disk in drive A: as described in the
documentation (y/n)? y

Writing file a:\rootboot.000

Enter start cylinder for new partition (17 - 260):

Use the cursor keys to choose the cylinder, <enter> to continue

Old partition      Cylinder      New Partition
 1059.0 MB           135            988.4 MB
```

Figure 1-6: The FIPS program is prompting the user for new partition size.

6. Use the left and right arrow keys to adjust the starting cylinder of the new partition (the one that results from splitting the existing partition) to change the partition size. Press the right arrow to increase the starting cylinder number (this leaves more room in the existing partition and reduces the size of the new partition you are creating).

7. When you are satisfied with the size of the new partition, press Enter. FIPS displays the modified partition table and prompts you to enter **C** to continue or **R** to reedit the partition table.

8. Press **C** to continue. FIPS displays some information about the disk and asks whether you want to write the new partition information to the disk.

9. Press **Y**. FIPS writes the new partition table to the hard disk and then exits.

Remove the disk from the A drive and reboot the PC. When the system comes up, everything in your hard disk should be intact, but the C drive will be smaller. You have created a new partition from the unused parts of the old C drive.

You needn't do anything with the newly created partition under DOS. Later, in the section "Partitioning Your Hard Disk under Linux," you learn how to use the new partition under Linux.

Creating the Red Hat boot disk

After you repartition the hard disk and make room on the hard disk for Linux, you can begin the next step of installing Linux from this book's CD-ROM: creating the Linux boot disk. (For this step, you should turn on your PC without any disk in the A drive and then run Windows as usual.)

Tip   You do not need a boot disk if you can start your PC under MS-DOS — not an MS-DOS window in Windows 95 — and access the CD-ROM from the DOS command prompt. If you run Windows 95, restarting the PC in MS-DOS mode is enough. However, the CD-ROM may not be accessible in MS-DOS mode because the startup files — AUTOEXEC.BAT and CONFIG.SYS — may not be configured correctly. Try restarting your PC in MS-DOS mode and see if the CD-ROM can be accessed. If you succeed, then skip this section and proceed to the section "Booting Linux for Installation."

Like the MS-DOS boot disk, the Linux boot disk is used to start your PC and start Linux. Once you have installed Linux, you no longer need the Linux boot disk except in an emergency (when you have to reinstall Linux from the CD-ROM).

The boot disk is the initial version of Linux you use to start Linux, prepare the hard disk, and load the rest of Linux. Creating the boot disk involves using a utility program called RAWRITE.EXE to copy a special file called the *Linux boot image* to a disk.

To create the Linux boot disk under Windows, follow these steps:

1. Open an MS-DOS window (select MS-DOS Prompt from the Programs area in the Start menu).

2. In the MS-DOS window, enter the following commands at the MS-DOS prompt (my comments are in parentheses and your input is in boldface):

```
D:   (use the drive letter for the CD-ROM drive)
CD \DOSUTILS
RAWRITE
Enter disk image source file name: \images\boot.img
Enter target diskette drive: A
Please insert a formatted diskette into drive A: and press -ENTER-
:
```

As instructed, you should put a formatted disk into your PC's A drive and then press **Enter**. RAWRITE copies the boot-image file to the disk.

After you see the DOS prompt, you can take the Linux boot disk out of the A drive and (if you haven't done so already) label it appropriately. A label such as *Linux Boot Disk* would be appropriate.

Creating the Red Hat supplementary install disk

If you have a device that uses the PCMCIA interface (commonly found on notebook PCs), you need a second disk besides the boot disk. The second disk is called a *supplementary install disk,* which has a Linux file system with a number of directories. The disk contains programs and files that may be needed under special circumstances, such as installing from a PCMCIA CD-ROM drive.

Tip You should go ahead and create a supplementary install disk because you also need this disk to fix any problems you encounter with your Linux system after you complete the installation.

To create the supplementary disk, insert the Red Hat Linux CD-ROM in the CD-ROM drive and type the following commands in an MS-DOS window:

```
d:   (use the drive letter for the CD-ROM drive)
cd \images
\dosutils\rawrite
Enter disk image source file name: supp.img
Enter target diskette drive: a
Please insert a formatted diskette into drive A: and press -ENTER- :
```

Insert a formatted floppy disk into A drive and press Enter. When the DOS prompt returns, the supplementary install disk is ready.

Installing Linux over the network using NFS

You can install Linux from another system on a network provided both systems are on the network and the other system has the Red Hat distribution on a directory that's exported through the Network File System (NFS). To perform such an installation, you must be knowledgeable about NFS or ask the help of someone who manages the network. You have to create the boot floppy disk as usual. After you use the boot floppy, designate NFS as the installation method and provide information about your network card and the network, including the IP addresses of your PC, gateway, and name server. Next you specify the NFS server's name or IP address and the server's directory containing the Red Hat distribution. Then, you can proceed with the usual installation steps.

Booting Linux for Installation

The Red Hat Linux installation program runs under Linux—this means you need to run Linux on your PC before you can go through the installation steps. This initial version of Linux can come from a boot floppy or the CD-ROM itself. This initial Linux operating system, in turn, runs the installation program, which prepares the disk partitions and copies all necessary files from the CD-ROM to the disk.

You can boot your PC with an initial version of the Linux operating system in one of the following ways:

- Load Linux by executing the AUTOBOOT.BAT command file while your PC is running MS-DOS.

- Boot your PC from the Linux boot floppy you had created earlier.

The following sections describe the two approaches of booting Linux and initiating the Red Hat installation.

Starting Linux from the Red Hat CD-ROM

You can start Linux directly from the CD-ROM while your PC is running MS-DOS. An MS-DOS program called LOADLIN.EXE can load a Linux kernel into memory and begin running Linux. The Linux kernel itself is in another file. You needn't understand all the details of how LOADLIN starts Linux. In fact, the Red Hat CD-ROM provides a DOS batch file—AUTOBOOT.BAT—in the \DOSUTILS directory that runs LOADLIN with appropriate arguments.

Note
You can use AUTOBOOT to start Linux directly from CD-ROM only if your PC is running MS-DOS alone (not an MS-DOS window under Windows 95). Also, you must be able to use the CD-ROM from MS-DOS. If your PC runs Windows 95, select Shutdown from the Start menu and then click the button labeled

`Restart the Computer in MS-DOS mode`. From the DOS prompt, if you can see the directory of the CD-ROM (with the command DIR D: where *D* is the drive letter of the CD-ROM drive), then you can start Linux directly from the CD-ROM. Otherwise, you have to use a boot floppy.

To start Linux, place the Red Hat CD_ROM in the CD-ROM drive and use the following commands from the DOS prompt:

```
D:
cd \dosutils
autoboot
```

After Linux starts, the Red Hat installation program begins to run. The section "Installing Linux from the Red Hat CD-ROM" describes the installation steps in detail.

Booting from the Linux floppy

To start Linux for installation, put the Linux boot floppy in your PC's A drive and restart your PC (either press the reset button or press Ctrl+Alt+Delete). Your PC goes through its normal startup sequence, such as checking memory and running the ROM BIOS code. Then the PC loads Linux from the floppy and begins running the Red Hat installation program.

Watching the boot process

A few moments after you start the boot process, an initial screen appears — the screen displays a welcome message and ends with a `boot:` prompt. The welcome message tells you help is available by pressing one of the function keys — F1 through F4.

If you do want to read the help screens, press the function key corresponding to the help you want. If you do nothing for a minute, the boot process proceeds with the loading of the Linux kernel into the PC's memory. To start booting Linux immediately, press Enter. After the Linux kernel loads, it automatically starts the Red Hat Linux installation program.

Tip As the Linux kernel begins to run, various messages appear onscreen. These messages tell you whether the Linux kernel has detected your hardware. The messages typically flash by too quickly for you to follow. Afterwards, the screen shows a message that asks you if you have a color monitor. At this point, press Shift+PgUp to scroll back and read the messages about your hardware. You should see messages corresponding to hardware in your PC. In particular, look for a message about the CD-ROM because the kernel has to detect the CD-ROM to proceed with the rest of the installation.

You can also view the messages by pressing Alt+F4 — this switches the display to another virtual screen where all kernel messages appear.

The following listing shows a typical sequence of messages displayed when I
boot one of my PCs using the Red Hat Linux boot floppy (this particular PC
has a 200MHz Pentium processor with 64MB memory, 4GB disk, an ATAPI
CD-ROM drive, and an Iomega ZIP drive):

```
boot:
Loading initrd.img.................
Loading vmlinuz.........
Uncompressing Linux...done.
Now booting kernel
Console: 16 point font, 400 scans
Console: colour VGA+ 80x25, 1 virtual console (max 63)
pcibios_init : BIOS32 Service Directory structure at 0x000f69f0
pcibios_init : BIOS32 Service Directory entry at 0xfd7e0
pcibios_init : PCI BIOS revision 2.10 entry at 0xfd9df
Probing PCI hardware.
Calibrating delay loop.. ok - 299.01 BogoMIPS
Memory: 62188/65536K available (736k kernel code, 384k reserved, 1256k
data)
Swansea University Computer Society NET3.035 for Linux 2.0
NET3: Unix domain sockets 0.13 for Linux NET3.035.
Swansea University Computer Society TCP/IP for NET3.034
IP Protocols: IGMP, ICMP, UDP, TCP
VFS: Diskquotas version dquot_5.6.0 initialized
Checking 386/387 coupling... Ok, fpu using exception 16 error
reporting.
Checking 'hlt' instruction... Ok.
Linux version 2.0.34 (root@porky.redhat.com) (gcc version 2.7.2.3) #1
Fri May 8 16:05:57 EDT
Starting kswapd v 1.4.2.2
Serial driver version 4.1.3 with no serial options enabled
tty00 at 0x3f8 (irq = 4) is a 16550A
PS/2 auxiliary pointing device detected -- driver installed.
Real Time Clock Driver v1.07
Ramdisk driver initialized : 16 ramdisks of 4096K size
ide: i82371 PIIX (Triton) on PCI bus 0 function 57
    ide0: BM-DMA at 0xfcf0-0xfcf7
    ide1: BM-DMA at 0xfcf8-0xfcff
hda: ST34342A, 4130MB w/0kB Cache, CHS=523/255/63
hdc: Pioneer CD-ROM ATAPI Model DR-A24X 0105, ATAPI CDROM drive
hdd: IOMEGA ZIP 100 ATAPI, ATAPI FLOPPY drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0x170-0x177,0x376 on irq 15
Floppy drive(s): fd0 is 1.44M
FDC 0 is a National Semiconductor PC87306
md driver 0.35 MAX_MD_DEV=4, MAX_REAL=8
scsi : 0 hosts.
scsi : detected total.
Partition check:
hda: hda1 hda2 hda3
ide-floppy: hdd: I/O error, pc = 5a, key = 5, asc = 24, ascq = 0
ide-floppy: Can't get drive capabilities
hdd: 98304kB, 96/64/32 CHS, 4096 kBps, 512 sector size, 2941 rpm
hdd: The drive reports both 100663296 and 0 bytes as its capacity
```

```
RAMDISK: Compressed image found at block 0
VFS: Mounted root (ext2 filesystem)
Welcome to Red Hat Linux
```

In this example, the message shows Linux has detected the ATAPI CD-ROM drive. The letters hdc refers to the device name Linux uses for the CD-ROM drive.

Installing Linux from the Red Hat CD-ROM

After you start the initial version of Linux following the procedures described in the section "Booting Linux for Installation," Linux runs the Red Hat Linux installation program from the CD-ROM. The rest of the installation occurs under the control of the installation program.

Interacting with the Red Hat installation program

The installation program uses a full-screen text-based interface. Figure 1-7 shows a typical Red Hat Linux installation screen.
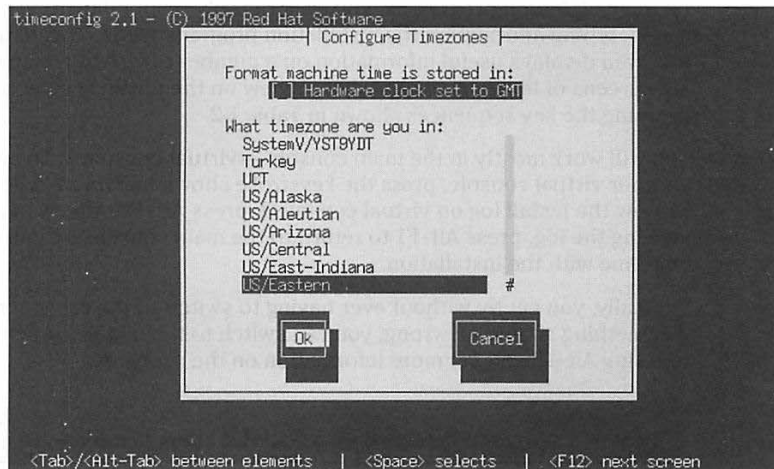
Figure 1-7: A typical installation screen with a dialog box

Each screen typically presents a dialog box with various elements such as lists of items from which you select items and buttons that you select to indicate action. Typically, the buttons are labeled OK and Cancel. The bottom of the screen displays a help message that tells you how to navigate around the text screen.

Instead of a mouse, you use specific keys to move the text cursor. Consult Table 1-1 for the keystrokes to use for specific actions.

Table 1-1	Interacting with the Red Hat installation program
Keystrokes	**Action**
Arrow keys	Moves from one item to the next in a dialog box.
Tab	Moves forward to the next item in a dialog box.
Alt+Tab	Moves backward to the previous item in a dialog box.
Spacebar	Selects the item on which the cursor rests. If the item is already selected, turns the selection off. Thus, the spacebar toggles the selection.
Enter	Presses the button on which the cursor rests. If the dialog box has a single button, pressing Enter is equivalent to pressing that button.
F12	Accepts current values and proceeds to the next dialog box. This is equivalent to pressing the OK button.

Monitoring the installation process

As the installation progresses, you are primarily responding to various dialog boxes, typing information the installation program needs. The installation program displays useful information on a number of virtual consoles — these are screens of text in memory you can view on the physical screen by pressing the key sequences shown in Table 1-2.

Tip

You will work mostly in the main console — virtual console 1. To switch to another virtual console, press the keystroke shown in Table 1-2. For example, to view the install log on virtual console 3, press Alt+F3. After you are done viewing the log, press Alt+F1 to return to the main console, so you can continue with the installation.

Typically, you get by without ever having to switch to the other screens but, if something should go wrong, you can switch to the install log screen by pressing Alt+F3 and get more information on the problem.

Table 1-2		Virtual consoles during Red Hat Linux installation
Virtual Console	**Keystroke**	**Description**
1	Alt+F1	This is the main console where the installation program displays the text-based user interface through which you install Linux.
2	Alt+F2	This console displays a shell prompt where you can use Linux commands to monitor the progress of installation. The shell prompt appears only after you insert the CD-ROM and press Enter in response to a dialog displayed by the installation program.

Virtual Console	Keystroke	Description
3	Alt+F3	This is the install log. Messages from the installation program appear here.
4	Alt+F4	The Linux kernel displays its messages on this console. After Linux initially boots, you may want to switch to this console to see the kernel messages because they include information about hardware Linux detects in your PC.
5	Alt+F5	This console shows the outputs of any programs run during the installation process.

Understanding the Red Hat installation phases

Linux installation is a lengthy process with the following major phases:

1. *Getting Ready to Install:* Indicate the type of keyboard and specify where Red Hat Linux is located (for example, the CD-ROM). If your PC has a PC Card device with PCMCIA interface, this phase detects such devices and asks you for the supplementary install disk that contains files needed for PCMCIA support.

2. *Partitioning and Using the Hard Disk:* Answer questions about any SCSI devices you have and proceed to partition the disk. You also specify the partition to be used as swap area and which partition will hold the Linux root directory (represented by /). The partitions are also formatted before use. Red Hat includes the Disk Druid program to perform all the steps in this phase of installation.

3. *Selecting the Components to Install:* Select which components of Red Hat Linux — such as X Window System, Emacs editor, and Web server — you want to install. After you select the components, the installation program takes over and installs the selected components on the hard disk.

4. *Configuring Linux:* In this phase you set up the X Window System, the TCP/IP network, the time zone, and any printer. You also set a password for the root — the super user. You conclude the configuration step and the installation by specifying where the Linux Loader (LILO) should be stored.

If you have all configuration information (such as IP addresses and host names for the TCP/IP network configuration) handy and all goes well, installing Linux from the companion CD-ROM on a fast (166MHz or better) Pentium PC should take approximately an hour. For example, on my 200MHz Pentium PC with 64MB RAM and a 1GB disk partition devoted to Linux, the entire installation took about 40 minutes. On older 486 PCs, the installation process may take somewhat longer.

The Red Hat installation program probes — attempts to determine the presence of — specific hardware and tailors the installation steps accordingly. For example, if the installation program detects a PC Card device that uses PCMCIA interface, the program automatically asks for the supplementary install disk. This means you may see some variation in the sequence of steps depending on your specific hardware configuration.

The following sections describe each of the Red Hat Linux installation phases in detail.

Getting ready to install

In this phase, you go through the following steps before moving on to disk setup and actual installation of Linux:

1. The installation program displays a black-and-white dialog box to ask if you have a color monitor (see Figure 1-8). Press Tab to move between Yes and No. Once the cursor is on the correct button, press Enter.
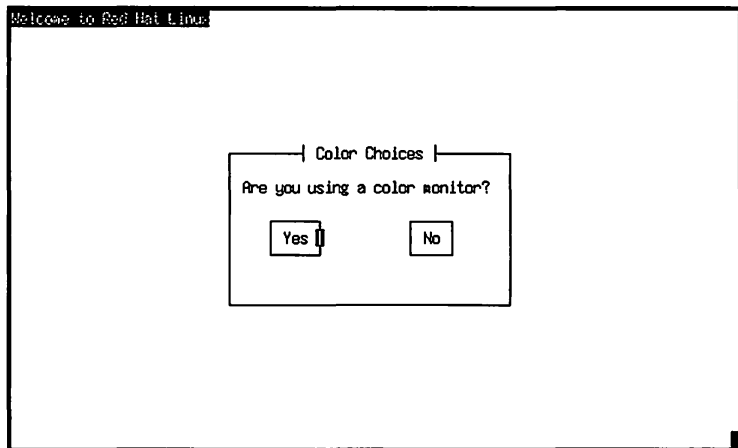
Figure 1-8: The dialog box asks you whether you have a color monitor.

2. A welcome screen appears. The message tells you to visit the Red Hat Web site at:

    `http://www.redhat.com`

    Press Enter to continue the installation.

3. The installation program checks if there are any PC Card devices that use the PCMCIA interface. If any PCMCIA device is found, the installation program displays a dialog box that asks you to insert the Red Hat supplementary install disk. If you have a PCMCIA device (such as an Ethernet card on a portable PC for installing over the network), insert the supplementary install disk and then select OK.

4. The installation program displays a list of keyboard types, as shown in Figure 1-9.


```
kbdconfig 1.5 - (C) 1997 Red Hat Software
                    ┤ Configure Keyboard ├

            What type of keyboard do you have?

                sg-latin1
                sg-latin1-lk450
                slovene
                sunkeymap
                sunt4-es
                sunt5-de-latin1
                sunt5-es
                sunt5-fi-latin1
                sunt5-ru
                uk
                us                             #


                  Ok              Cancel



   <Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```
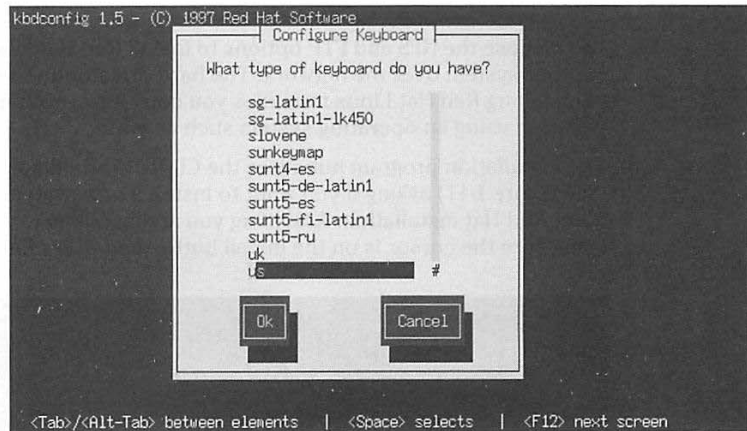
Figure 1-9: Selecting a keyboard type during Red Hat installation

Use the arrow keys to move up and down the list and press Spacebar. Select the keyboard type that matches what your system has. Then press F12 to proceed to the next step.

5. Specify where the Red Hat packages are located from the dialog box shown in Figure 1-10.


```
Welcome to Red Hat Linux


                    ┤ Installation Method ├

            What type of media contains the
            packages to be installed?

                Local CDROM
                NFS image
                hard drive
                FTP

                      Ok



   <Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```
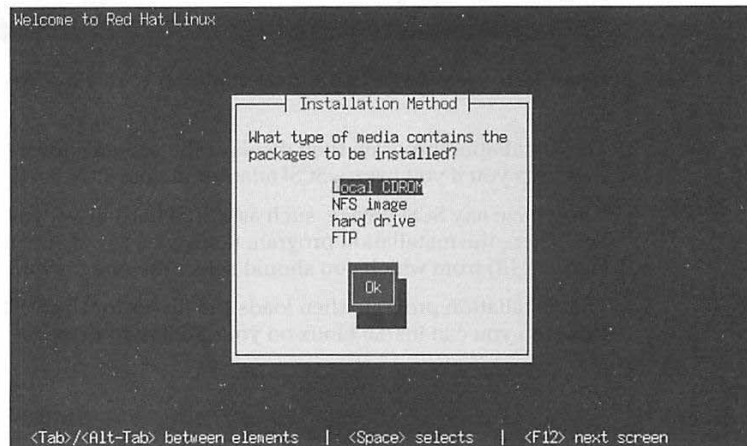
Figure 1-10: Specifying where the Red Hat Linux packages are located.

The Local CDROM option refers to the CD-ROM drive on your PC. Because you are installing from this book's companion CD-ROM, you should put the CD-ROM into the drive, select Local CDROM, and press Enter.

You can use the NFS and FTP options to install Red Hat Linux from another system over the network. The hard drive option refers to a way of installing Red Hat Linux from files you copy into another hard drive partition using an operating system such as Windows 95.

6. The installation program initializes the CD-ROM and displays a dialog box (see Figure 1-11) asking if you want to install a new system or upgrade an older Red Hat installation. Assuming you are installing for the first time, make sure the cursor is on the Install button and press Enter.
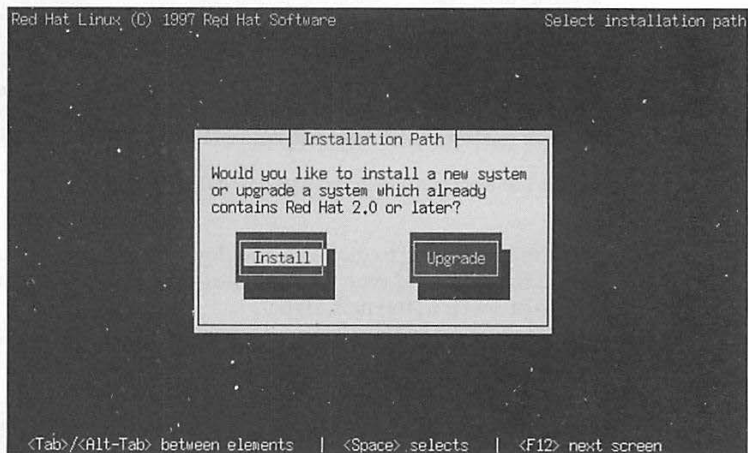
Figure 1-11: This is the dialog box to select whether you are installing or upgrading.

7. The installation program then displays the screen shown in Figure 1-12 that asks you if you have a SCSI adapter in your PC.

If you have any SCSI device, such as a SCSI hard drive, you must select Yes. Then, the installation program displays a list of SCSI adapters (see Figure 1-13) from which you should select the one on your system.

The installation program then loads the driver for the SCSI adapter. After this step you can install Linux on your SCSI hard drive.

Figure 1-12: Select Yes if you have a SCSI adapter in your PC .
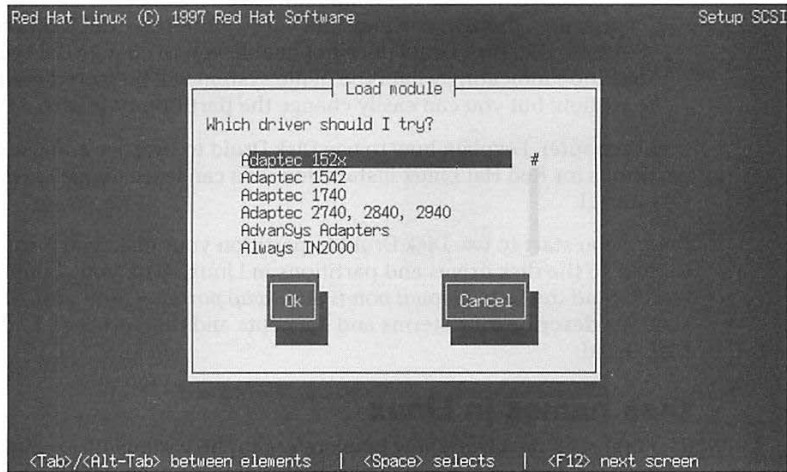
Figure 1-13: Selecting a driver for your PC's SCSI adapter, if any.

The next major phase of installation involves partitioning the hard disk for use in Linux.

Partitioning and using the hard disk

Like MS-DOS, Linux requires you to partition and prepare a hard disk before you can install Linux on the hard disk. You usually do not perform this step because, when you buy your PC from a vendor, the vendor takes care of preparing the hard disk and installing Windows and all other applications on the hard disk. Because you are installing Linux from scratch, however, you must perform this crucial step yourself. You have to prepare the hard disk partitions under Linux before you can install the rest of Linux. As you see in the following sections, this task is just a matter of following instructions.

When the Red Hat Linux installation program reaches the disk partitioning phase, it displays a text dialog box that gives you two options to partition and use the hard disk:

- **Disk Druid**: This is a Red Hat utility program that uses text dialogs to enable you to partition the disk and, at the same time, specify what parts of the Linux file system are to be loaded on which partition.

- fdisk: This is the Linux file partitioning program, similar in concept to the DOS FDISK command, but with many more capabilities than the DOS version. When you use the Linux fdisk program, you have to type cryptic one-letter commands to manipulate disk partitions. Once you learn the commands, though, you may find fdisk more powerful than Disk Druid. For example, Disk Druid does not enable you to change the type of a partition indicating what type of file system will be stored on the partition; but you can easily change the partition type with fdisk.

In this chapter, I explain how to use Disk Druid to prepare and use the disk partitions for Red Hat Linux installation. You can learn about fdisk in Chapter 11.

Before you start to use Disk Druid to partition your disk, you must know how to refer to the disk drives and partitions in Linux. Also, you should understand the terms *mount points* and *swap partition*. The next three sections describe these terms and concepts and then proceed to describe Disk Druid.

Disk names in Linux

The first step is to learn how Linux refers to the various disks. Linux treats all devices as files and has actual files that represent each device. In Linux, these *device files* are located in the /dev directory. If you are new to UNIX, you may not yet know about UNIX filenames, but you will learn more as you continue to use Linux. If you know how MS-DOS filenames work, you will find Linux filenames are similar, except it does not use drive letters (such as A and C) and substitutes the slash (/) for the MS-DOS backslash (\) as the separator between directory names.

Because Linux treats a device as a file in the /dev directory, the hard disk names start with /dev. Table 1-3 lists the hard disk and floppy drive names you may have to use.

Table 1-3	Hard disk and floppy drive names
Name	**Description**
/dev/hda	First Integrated Drive Electronics (IDE) drive (in DOS, usually the C drive)
/dev/hdb	Second IDE drive
/dev/sda	First Small Computer System Interface (SCSI) drive
/dev/sdb	Second SCSI drive
/dev/fd0	First floppy drive (the A drive in DOS)
/dev/fd1	Second floppy drive (the B drive in DOS)

When you use the Red Hat Disk Druid or Linux fdisk program to prepare the Linux partitions, you must identify the disk drive by its name such as /dev/hda.

Tip
When Disk Druid or fdisk displays the list of partitions, the partition names are of the form /dev/hda1, /dev/hda2, and so forth. Linux constructs each partition name by appending the partition number (1 through 4, for the four primary partitions on a hard disk) to the disk's name. Therefore, if your PC's single IDE hard drive has two partitions, you will notice the installation program uses /dev/hda1 and /dev/hda2 as the names of these partitions. Here are more examples of hard disk partition names in Linux:

Name	Partition
/dev/hda1	First primary partition of first IDE drive
/dev/hda2	Second primary partition of first IDE drive
/dev/hda3	Third primary partition of first IDE drive
/dev/hda4	Fourth primary partition of first IDE drive
/dev/hda5	First logical partition of first IDE drive
/dev/hda6	Second logical partition of first IDE drive
/dev/hdb1	First primary partition of second IDE drive
/dev/sda1	First primary partition of first SCSI drive
/dev/sda2	Second primary partition of first SCSI drive
/dev/sdb1	First primary partition of second SCSI drive
/dev/sdc1	First primary partition of third SCSI drive

Mount point

In Linux, you use a physical disk partition by associating with a specific part of the file system, which is a hierarchical arrangement of directories — a directory tree. If you have more than one disk partition (you may have a second disk with a Linux partition), you can use all of them in Linux. All you have to do is decide which part of the Linux directory tree should be located on each partition — a process known in Linux as *mounting a file system on a device* (the disk partition is a device).

Note

The term *mount point* refers to the directory you associate with a disk partition or any other device.

Suppose you have two disks on your PC and you have created Linux partitions on both disks. Figure 1-14 illustrates how you can mount different parts of the Linux directory tree (the file system) on these two partitions.
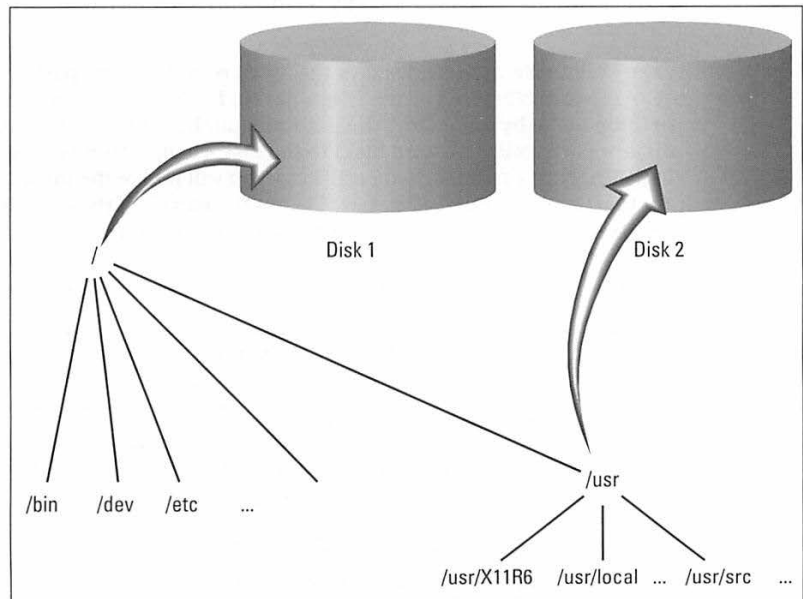
Figure 1-14: An example of mounting the Linux file system on two disk partitions

Swap partition

Most advanced operating systems support the concept of virtual memory, in which part of your system's hard disk is used as an extension of the physical memory (RAM). When the operating system runs out of physical memory, it

can move (or swap out) the contents of currently unneeded parts of RAM to make room for a program that needs more memory. As soon as the time comes to access anything in the swapped-out data, the operating system has to find something else to swap out and then swap in the required data from disk. This process of swapping data back and forth between the RAM and the disk also is known as *paging*.

Because the disk is much slower than RAM, the system's performance is slower when the operating system has to perform a lots of paging, but virtual memory enables you to run programs you otherwise would be unable to run.

Linux supports virtual memory and can make use of a swap partition. When you create the Linux partitions, you should create a swap partition for Linux. With the Disk Druid utility program, described in the next section, it is simple to create a swap partition. Simply mark a partition type as swap device and Disk Druid will perform the necessary tasks.

Disk preparation

As you get ready to prepare the disk for Linux installation, you should see the Disk Setup dialog box with the button marked Disk Druid already selected, as shown in Figure 1-15. To use Disk Druid, press Enter.
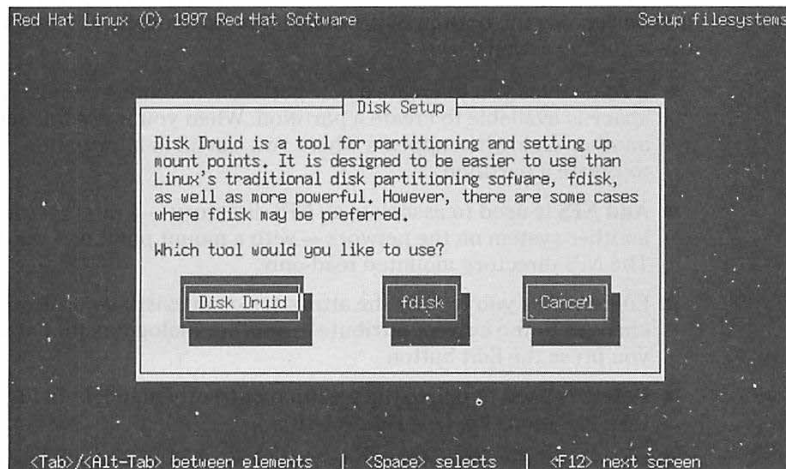
Figure 1-15: Disk Setup dialog box from Red Hat Linux installation program

Disk Druid gathers information about the hard drives on your system and displays a dialog box with the list of disk drives and the current partition information for one of the drives, as shown in Figure 1-16.
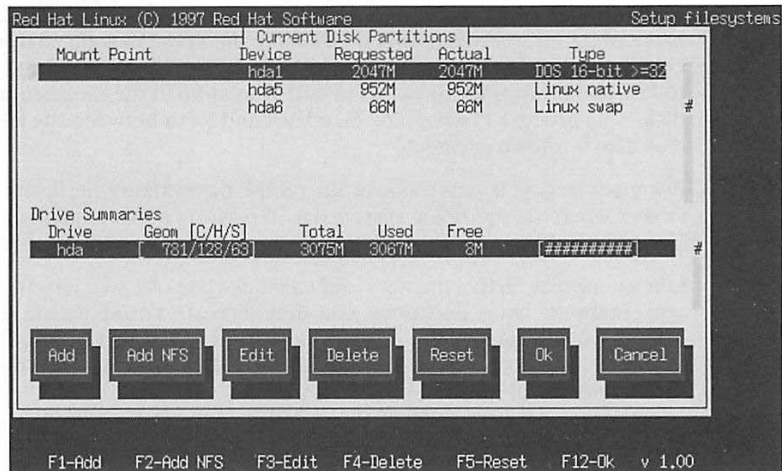
Figure 1-16: Main dialog box of the Disk Druid program

You perform specific disk setup tasks in Disk Druid through the seven buttons across the bottom of the dialog box. Specifically, the buttons perform the following actions:

- **Add** enables you to create a new partition, assuming enough free disk space is available to create a partition. When you press this button, another dialog box appears where you can fill in information necessary to create a partition.

- **Add NFS** is used to associate an NFS directory—a directory located on another system on the network—with a mount point on your system. The NFS directory mounted read-only.

- **Edit** enables you to alter the attributes of an existing partition. You make changes to the current attribute in another dialog box that appears when you press the Edit button.

- **Delete** is used to delete the partition currently highlighted in the Current Disk Partitions list (see Figure 1-16).

- **Reset** restores all partition information back to the way it was before you started running Disk Druid. This is possible because Disk Druid does not actually make changes to the partition table (which is stored on the hard drive) until after you exit by pressing the OK button.

- **OK** causes Disk Druid to ask you if you really want to update the partition tables on all the hard drives in your system. If you answer Yes, Disk Druid makes changes to the partition tables and exits.

- **Cancel** is used to exit Disk Druid without saving any changes you might have made.

Exactly what you do in Disk Druid depends on the hard drives in your PC and the partitions they already have. For my discussion, I assume you have created the necessary hard disk space for Linux by one of the following methods:

- You started with a single hard drive with a single partition (only C drive in Windows). Then you used FIPS to split that partition into two (see the section "Repartitioning with FIPS"). After partitioning, you end up with two DOS partitions.

- Your PC had a single large hard drive (greater than 2GB) that had two partitions (C and D drives in Windows). In this case, the second partition is an extended partition that contains the logical drive D. You want to install Linux on the extended partition that used to be the D drive in Windows.

Both of these situations call for the same sequence of steps:

1. Delete the DOS partition — the one with enough space for Linux installation, not the one where Windows is installed.

2. Create two Linux partitions out of the available disk space — one for swap space and the other for the Linux file system.

The next section shows the specific steps you will perform in Disk Druid to set up the minimum number of partitions Linux needs. You can prepare other disk partitions (if you have a second hard drive, for example) following the steps outlined in the next section.

After you set up the partitions, you have to initialize the swap partition and format the partition meant for the Linux file system. These steps are described in the sections "Initializing the swap space" and "Formatting the partitions."

Cross-Reference

Setting up the partitions

To prepare an existing DOS partition for Linux, you have to perform the following steps in Disk Druid:

1. Delete the DOS partition you want to use for Linux. To do this, select the partition (this will typically be the one with Device name hda2) from the Current Disk Partitions list (see Figure 1-16) and then press the Delete button. As a shortcut, you may simply select the partition and press **F4**. Before deleting the partition, remember to note the partition's size from the Current Disk Partitions list.

2. Create a new partition for the Linux file system. To do this, press the Add button (or press **F1**). You will see the Edit New Partition dialog box (see Figure 1-17) where you should fill in / as the mount point and enter the size in megabytes. To compute the size, simply subtract the size of the swap space (32MB or the amount of RAM in your PC, whichever is more) from the original size of the partition. Select the OK button and then press **Enter** to complete this step and return to the Disk Druid's main dialog box.
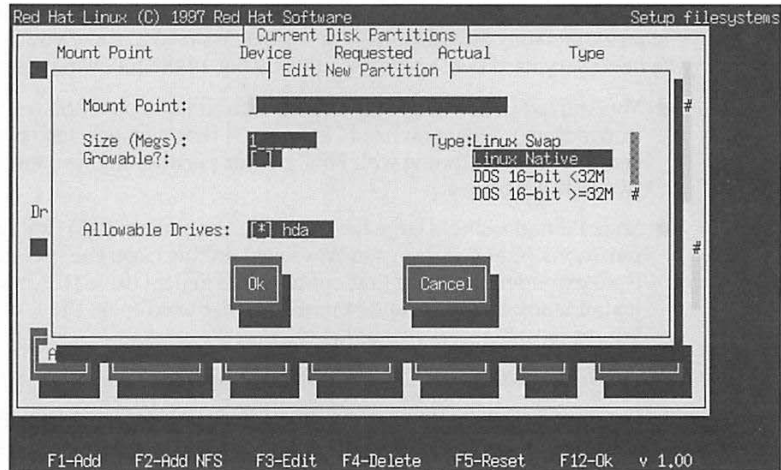
Figure 1-17: Edit New Partition dialog box where you fill in the attributes of a new partition

3. Create another new partition and set it as a Linux swap space. To do this, press F1 from the Disk Druid main dialog box (Figure 1-16). Then, in the Edit New Partition dialog box (Figure 1-17), enter the size of the partition. Press Tab to move to the list of partition types and use the arrow keys to select Linux Swap as the type—when you do so, the text Swap Partition appears in the Mount Point field. Next, press Tab to select the OK button and press Enter to define the new partition and return to the Disk Druid's main dialog box.

4. If you want to access the DOS partition under Linux, make sure you assign a mount point for this partition. Basically, you will assign a Linux directory name where the DOS partition will appear. For example, you can use /dosc as the mount point for the DOS partition; then you can access the DOS files in the /dosc directory in Linux. This mnemonic is good because the name dosc should remind you this drive is the C drive under DOS. To assign the mount point, select the DOS partition (typically, this is listed as Device hda1 in the Current Disk Partitions list) and press F3 to edit the attributes. The Edit Partition dialog box appears (see Figure 1-18). Type the mount point (for example, /dosc) and press Enter to return to the Disk Druid's main dialog box.

5. Make the changes permanent by pressing the OK button in the Disk Druid main dialog box (Figure 1-16).
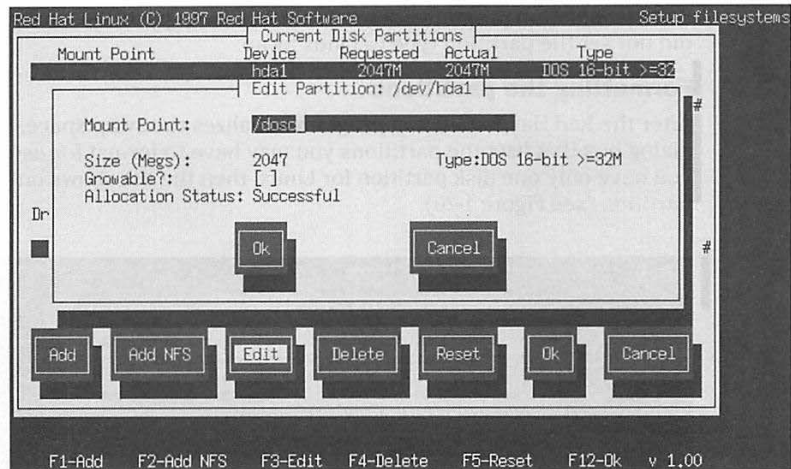
Figure 1-18: Edit Partition dialog box enables you to change the attributes of a partition.

Initializing the swap space

After you finish specifying the partitions in Disk Druid, the Red Hat installation program displays a dialog box (see Figure 1-19) that lists all partitions of type Linux Swap and asks you whether you want to initialize the swap spaces. If you created one swap partition, the list will include that single partition. You should go ahead and initialize the swap partition by pressing Tab until the OK button is selected and then pressing **Enter**.
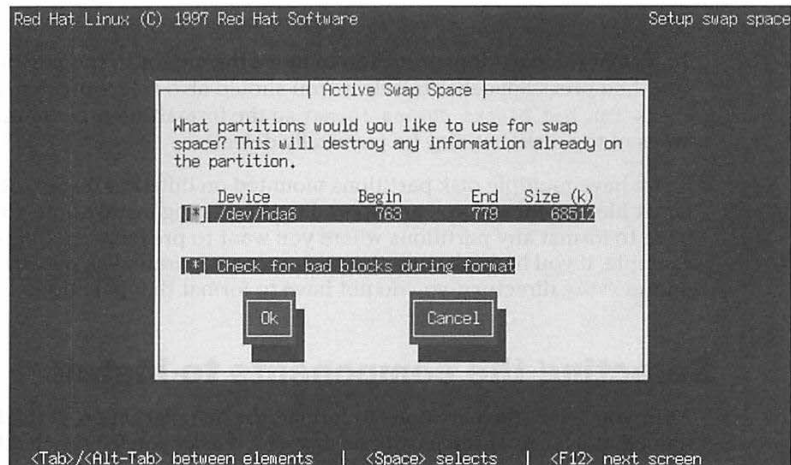
Figure 1-19: Dialog box that enables you to initialize the swap space

If the installation program does not find any swap partition, chances are you did not set the partition type to Linux Swap.

Tip

Formatting the partitions

After the Red Hat installation program initializes the swap space, it displays a dialog box that lists the partitions you may have to format for use in Linux. If you have only one disk partition for Linux, then the list shows only that partition (see Figure 1-20).

Red Hat Linux (C) 1997 Red Hat Software   Choose partitions to format
─────────────────────────── Format Partitions ───────────────────────────

What partitions would you like to format? We strongly
suggest formatting all of the system partitions,
including /, /usr, and /var. There is no need to
format /home or /usr/local if they have already been
configured during a previous install.

[*] /dev/hda5   /

[*] Check for bad blocks during format

Ok   Cancel

<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen

Figure 1-20: The Format Partitions dialog box enables you to format the partitions for use in Linux.

To format the partition, press Tab to move the cursor to the partition's name and then press Spacebar to select. You should also select the item marked Check for bad blocks during format so the formatting process marks any areas of the disk that may be physically defective.

Note

If you have multiple disk partitions mounted on different directories of the Linux file system and you are upgrading an existing installation, you do not have to format any partitions where you want to preserve existing data. For example, if you have all user directories on a separate disk partition mounted on the /home directory, you do not have to format that partition.

Selecting the components to install

After you select the partitions to format, the installation program does not immediately format the partitions. Instead, it asks you for the Red Hat Linux components you want to install. This way, after you have selected the

components, you can go away for a cup of coffee and the Red Hat installation program can format the disk partition and copy all necessary files to that partition.

Red Hat uses special files called *packages* to bundle a number of files that make up a specific software. For example, all configuration files, documentation, and binary files for the Perl programming language comes in a Red Hat package. You use a special program called Red Hat Package Manager (RPM) to install, uninstall, and find out information about packages. Chapter 2 describes how to use RPM. For now, just remember a component is made up of several Red Hat packages.

Figure 1-21 shows the dialog box with the list of components you can elect to install. Each component is identified by a descriptive label with a square bracket ([ ]) prefix.
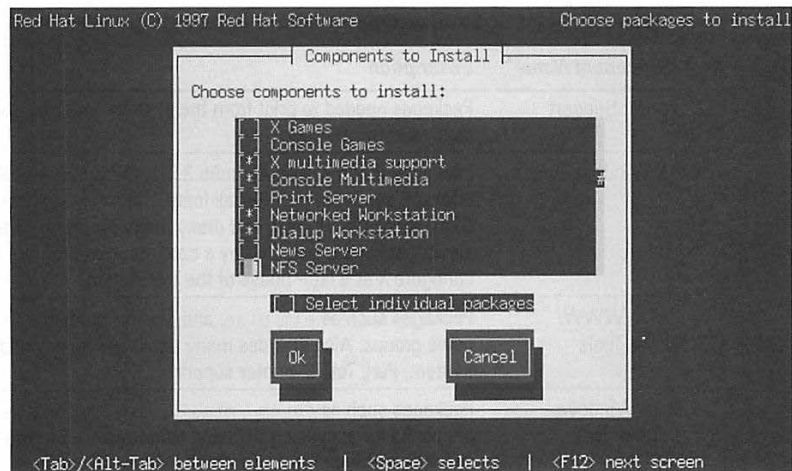
Figure 1-21: Dialog box from where you select the components to install

An asterisk (*) in the square bracket means the component is selected. When the dialog box first appears, many of the components are already selected, as indicated by the asterisks. You can think of the marked components as the minimal set of components recommended for installation by Red Hat. You can, however, choose to install any or all of the components. Use the arrow keys to move up and down in the scrolling list (Figure 1-21) and press Spacebar to select or deselect a component.

Table 1-4 shows the components included in the list. In the table, an asterisk prefix indicates the component is selected by default. In addition to these user-selectable components, the Red Hat installation program automatically installs a large number of packages needed to run Linux.

Tip

Each component requires specific packages to run. The Red Hat installation program automatically checks for any package dependencies and shows you a list of packages required, but that you have not selected. In this case, you should install the required packages.

Because each component is a collection of many different Red Hat packages, the installation program also gives you an option to select individual packages. If you select the item labeled "Select individual packages," that appears after the list in Figure 1-21 and then press the OK button, the installation program takes you to further dialog boxes to select individual packages. If you are installing Linux for the first time, however, you needn't go down to that level of detail to install specific packages. Simply pick from Table 1-4 the components you think you need. You can always install additional packages later on with the RPM utility program.

Table 1-4   List of components to install in Red Hat Linux

Component Name*	Description
Printer Support	Packages needed to print from the system. Includes utilities such as lpr and Ghostscript.
*X Window System	Packages that make up XFree86 3.3.1, the X Window System for Intel x86 systems. Includes all fonts, the fvwm2 window manager, Ghostscript, Tcl/Tk, Perl, and many utility programs. The actual *X* server package is selected by a configuration program when you configure *X* at a later phase of the installation.
*Mail/WWW/ News Tools	Packages such as elm, pine, and trn for reading e-mail and news groups. Also includes many packages for X Window System, Perl, Tcl/Tk, printer support.
DOS/Windows Connectivity	Packages such as dosemu, mtools, lha, zip, and unzip that are useful for accessing DOS and Windows files from Linux. Also includes many packages for the X Window System.
*File Managers	File managers such as mc (Midnight Commander) and xfm along with Perl, Tcl/Tk, and the X Window System.
Graphics Manipulation	Image manipulation packages such as ImageMagick, gimp, and Ghostscript along with all of X Window System.
X Games	The entire X Window System component plus the X game packages xbill, xboard, xboing, xchomp, xdemineur, xevil, xfishtank, xgalaga, xgammon, xgopher, xjewel, xlander, xpat2, xpilot, xpuzzles, xsnow, and xtrojka.
Console Games	Old UNIX games such as hangman and fish as well as new games such as yahtzee, Doom, gnuchess, trojka, and some card games.
*X Multimedia Support	Packages for sound and animation under *X*. Also includes all X Window System packages.
*Console Multimedia	Sound and animation support packages such as aumix, sndconfig, rhsound, cdp, maplay, and playmidi.

Component Name*	Description
Print Server	Packages needed to make the Linux system a print server.
*Networked Workstation	TCP/IP networking packages such as FTP (File Transfer Protocol), Telnet, and NFS (Network File System).
*Dialup Workstation	TCP/IP networking packages, as well as packages such as ppp and dip needed for dial-up networking. Also includes communication packages such as minicom and lrzsz (for sending and receiving files).
News Server	Perl and INN (Internet News server) packages needed to make the Linux system a server for news groups.
NFS Server	Packages such as portmap and nfs-server that make the Linux system a NFS server.
SMB (Samba) Connectivity	Packages such as samba and smbfs that are needed to use the Linux PC as a LAN Manager server. Includes many other packages, such as TCP/IP networking packages and printer support packages.
IPX/Netware (tm) Connectivity	Packages needed to use the Linux PC as a Novell Netware server. Includes much of the TCP/IP networking packages as well.
Anonymous FTP/ Gopher Server	The wu-ftpd and anonftp packages along with all TCP/IP networking packages needed to support anonymous FTP in Linux.
Web Server	The Apache Web server and other networking packages needed to use the Linux system as a Web server.
DNS Name Server	The bind, bind-utils, and caching-nameserver packages along with other support packages needed to provide Domain Name Service (DNS) on the Linux system.
Postgres (SQL) Server	The postgresql, postgresql-data, and postgresql-devel packages along with many support packages to implement the Postgres SQL server (a database) on the Linux system.
Network Management Workstation	Packages to support network management through SNMP (Simple Network Management Protocol) and IP firewall administration utility. Also includes many networking packages required by the network management tools.
TeX Document Formatting	Packages needed for the TeX (pronounced *tech,* as in *technology*) document-formatting system.
Emacs	Version of Emacs text editor that does not require *X.*
Emacs with X windows	GNU Emacs 20.2 that runs in *X.* Also includes the X Window System component and the version of Emacs that does not require *X.*
C Development	The GNU C compiler, the gdb debugger, Revision Control System (RCS), various header files, and other utilities for C programming.
Development Libraries	Various libraries needed for software development in Linux.

(continued)

Table 1-4 (Continued)	
Component Name*	**Description**
C++ Development	The GNU C++ compiler and other associated files needed for C++ programming.
X Development	*X* header files and libraries as well as the X Window System needed for developing applications that use *X*.
Extra Documentation	Various online manual pages (called *man* pages in UNIX) as well as Linux HOWTO files and Frequently Asked Questions (FAQs) for many Linux software. You should go ahead and install this component.
Everything	Select this item to install all the components.

*Components marked with an asterisk (*) prefix are selected by default.

After you have selected the components to select, press Tab to move to the OK button in Figure 1-21 and then press Enter. The installation program displays an informative dialog box, as shown in Figure 1-22.

Figure 1-22: Informative message about install log just before the selected components are installed

The dialog box informs you a log of the installation will be in the /tmp/install.log file. This file will essentially list all the Red Hat packages installed in your system. Here are a few lines from the beginning of the /tmp/install.log file after I finished installing Red Hat Linux on a PC:

```
Installing setup.
Installing filesystem.
```

```
Installing basesystem.
Installing anonftp.
Installing AnotherLevel.
Installing ldconfig.
Installing termcap.
Installing libtermcap.
Installing glibc.
Installing grep.
Installing fileutils.
Installing bash.
Installing aout-libs.
Installing slang.
Installing newt.
Installing chkconfig.
Installing apache.
```

You can review the install log later and keep the file for future reference.

Press Enter to proceed with the installation. The Red Hat installation program formats the disk partitions and then installs the packages. As it installs packages, the installation program displays a status screen that shows the progress of the installation with information such as total number of packages to install, number installed so far, estimated amount of disk space needed, and estimated time remaining to install.

Tip The formatting and installation takes a while—you can take a break and check back in ten minutes or so. When you come back, you should be able to get a sense for the time remaining from the status screen the installation program updates continually.

Configuring Linux

When the installation program finishes loading all the selected packages, it moves on to the configuration phase. Assuming you are installing the X Window System and the networking component, the configuration steps are as follows:

- Configure the mouse
- Configure X
- Configure the network
- Set the time zone
- Configure services
- Configure printers
- Set the root password
- Create a custom boot disk
- Install LILO

The following sections describe each of these configuration steps.

Configure the mouse

If you have installed the X Window System component (as described in the section "Selecting the Components to Install"), the Red Hat installation program runs the mouseconfig utility program to detect your PC's mouse. Figure 1-23 shows a typical outcome of the mouse detection.

Figure 1-23: Dialog box informing you of the type of mouse detected

In this case, the mouseconfig program found a mouse that uses a PS/2 style interface. Because this is simply meant as information for you, press Enter to proceed to the next step.

The mouse configuration program displays another dialog box (Figure 1-24) that gives you the option to emulate a 3-button mouse with a typical 2-button mouse on PCs.

If you select Yes, you can simulate a middle button click by pressing both buttons simultaneously. Because many X applications assume a 3-button mouse, you should go ahead and select the Yes button on the dialog box.
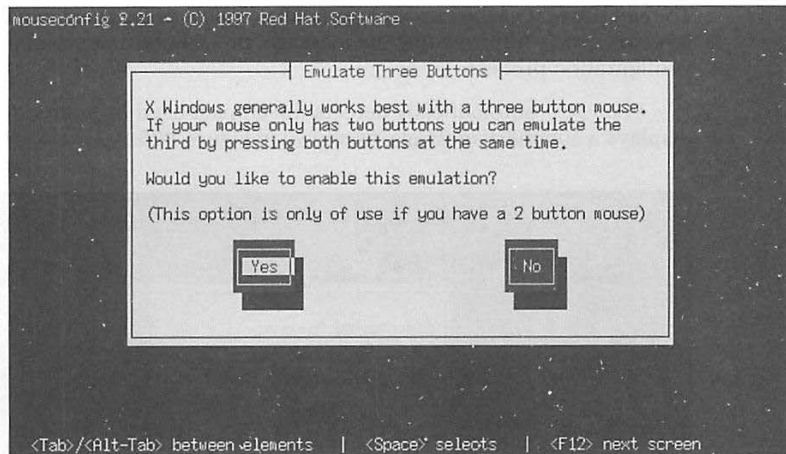
Figure 1-24: Dialog box asking you if you want to emulate a 3-button mouse

Configure *X*

If you selected the X Window System component (as described in the "Selecting the Components to Install" section), the Red Hat installation program now runs the Xconfigurator utility program to create a configuration file the *X* server needs.

The Xconfigurator program starts with a welcome dialog box, as shown in Figure 1-25.
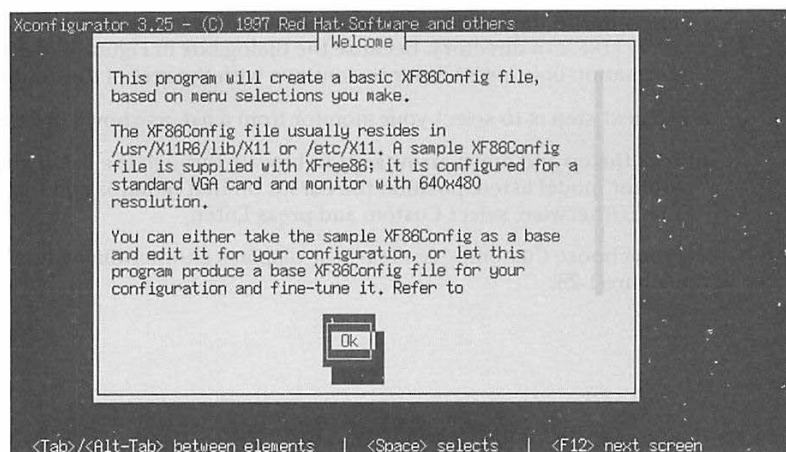
Figure 1-25: Welcome dialog box from the Xconfigurator program

To read the rest of the message in the dialog box, press the down arrow key to scroll down. After reading the message, press Enter to continue with the configuration process.

Xconfigurator automatically detects the chipset used by your video card and displays a summary message, as shown in Figure 1-26.
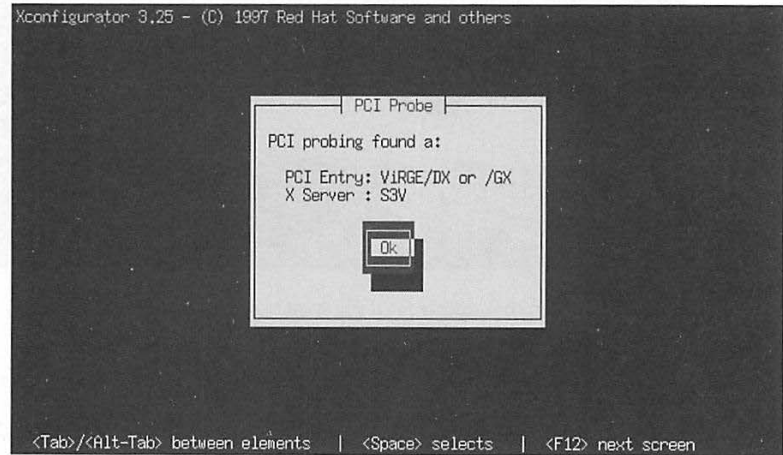
Figure 1-26: Result of probing the video card

In this case, Xconfigurator finds a video card based on the S3 ViRGE/DX (or /GX) chipset and recommends the S3V X server. In fact, the message from Xconfigurator is somewhat cryptic. The S3V X server refers to the X server program with the full name XF86_S3V that will be installed in the /usr/X11R6/bin directory. Because the dialog box in Figure 1-26 is for your information only, press Enter to continue with the rest of the configuration.

The next step is to select your monitor from a list, as shown in Figure 1-27.

Press the up and down arrow keys to browse through the list. If you find your monitor model listed, position the cursor on that monitor and then press Enter. Otherwise, select Custom and press Enter.

If you choose Custom, Xconfigurator displays another dialog box, as shown in Figure 1-28.
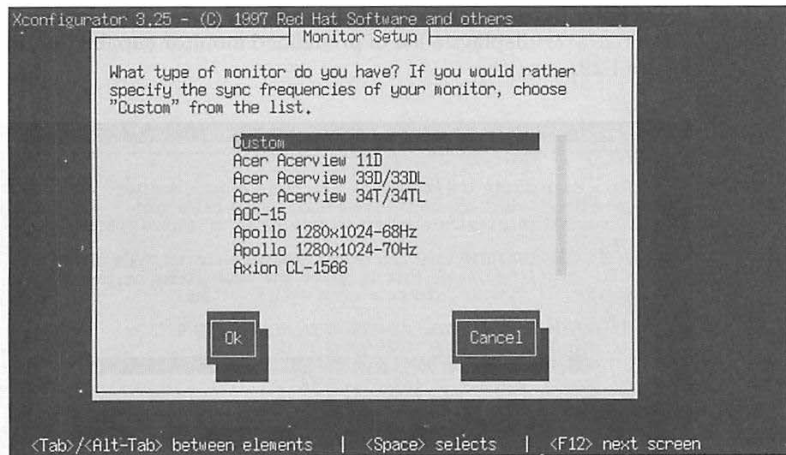
Figure 1-27: Selecting your monitor type
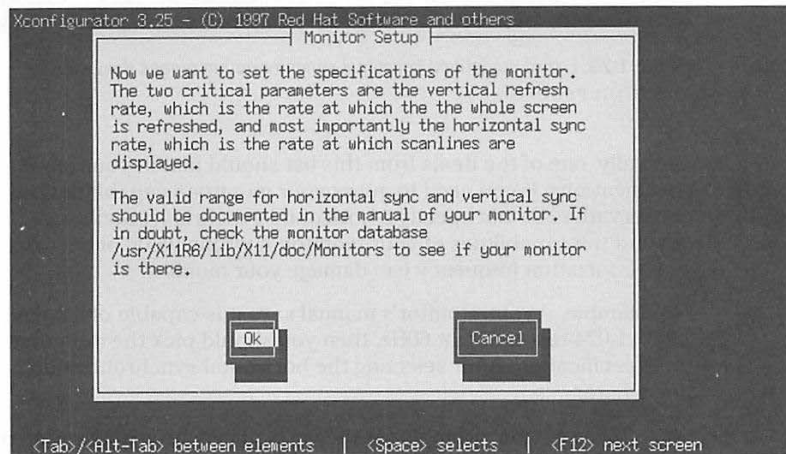
Figure 1-28: Xconfigurator informs you it needs certain information about your monitor.

As the dialog box (Figure 1-28) says, you have to provide two critical parameters of your monitor:

- Horizontal synchronization frequency—the number of times per second the monitor can display horizontal raster lines, in kilohertz (kHz)

- Vertical synchronization rate or vertical refresh rate—how many times a second the monitor can display the entire screen

You can find this information in your monitor's manual.

Press Enter in response to the dialog box of Figure 1-28 to continue. Xconfigurator displays a list of predefined monitor capabilities, as shown in Figure 1-29.

```
Xconfigurator 3.25 - (C) 1997 Red Hat Software and others
┤ Monitor Setup (Continued) ├

You must indicate the horizontal sync range of your monitor. You
can either select one of the predefined ranges below that
correspond to industry-standard monitor types, or give a specific
range.
It is VERY IMPORTANT that you do not specify a monitor type with
a horizontal sync range that is beyond the capabilities of your
monitor. If in doubt, choose a conservative setting.

   Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
   High Frequency SVGA, 1024x768 @ 70 Hz
   Monitor that can do 1280x1024 @ 60 Hz
   Monitor that can do 1280x1024 @ 74 Hz
   Monitor that can do 1280x1024 @ 76 Hz


          Ok                  Cancel



<Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```

Figure 1-29: List of predefined horizontal synchronization ranges displayed by Xconfigurator

Caution

Typically, one of the items from this list should match your monitor's specifications. If you need to guess your monitor's capabilities, be conservative. Do not specify a horizontal synchronization range that is beyond the capabilities of your monitor. A wrong value of horizontal synchronization frequency can damage your monitor.

For example, if your monitor's manual says it is capable of displaying 1,280×1,024 resolution at 60Hz, then you should pick the item that matches this specification. After selecting the horizontal synchronization capabilities, press Enter.

Xconfigurator now prompts for the vertical synchronization rate of your monitor, as shown in Figure 1-30.

Pick the vertical synchronization range nearest to your monitor's specifications and press Enter.

Xconfigurator displays a dialog box (see Figure 1-31) informing you the X server will be run to gather information about your video card.
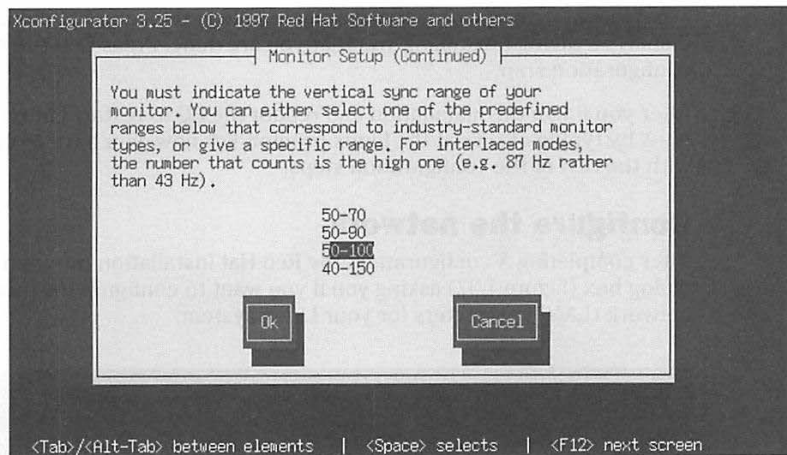
Figure 1-30: List of vertical synchronization ranges displayed by Xconfigurator
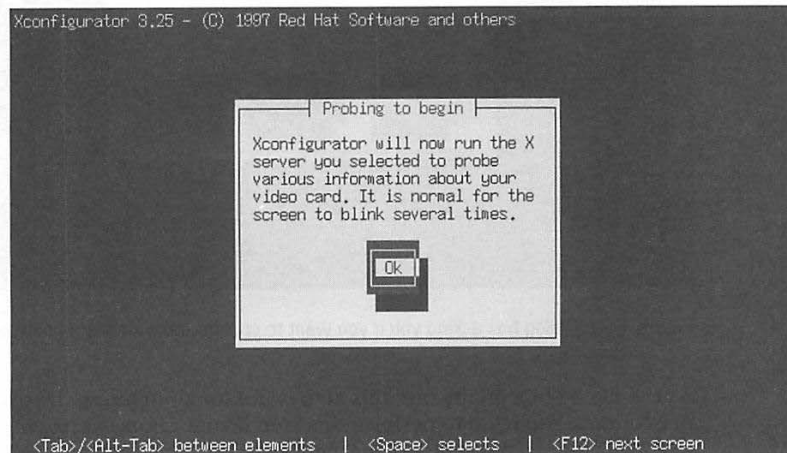
Figure 1-31: Message prior to video card probing by Xconfigurator

Because the message is meant as information to you, read the message and then press Enter to proceed.

The screen blinks a few times and then XConfigurator displays a default video mode—in terms of resolution such as 1,024×768 and color such as 8 bit. To accept the default setting, press Enter.

Xconfigurator then writes the XF86Config file in the /etc/X11 directory and displays a dialog box telling you that you are done. Press Enter to exit the *X* configuration step.

Tip After you finish the installation and reboot the PC to restart Linux, you can try *X* by typing startx at the Linux prompt. For now, you have to continue with the rest of the configuration steps.

Configure the network

After completing *X* configuration, the Red Hat installation program displays a dialog box (Figure 1-32) asking you if you want to configure the local area network (LAN) parameters for your Linux system.

Figure 1-32: Dialog box asking you if you want to configure the local area network

As the dialog box points out, this step is not for configuring the dial-up networking. You need to perform this step if your Linux system is connected to a TCP/IP LAN, through an Ethernet card.

If your system is on a LAN, even if it is a LAN that is not connected to the Internet, you should select the Yes button. The Red Hat installation program then displays a dialog box where you have to enter certain parameters for TCP/IP configuration (Figure 1-33).
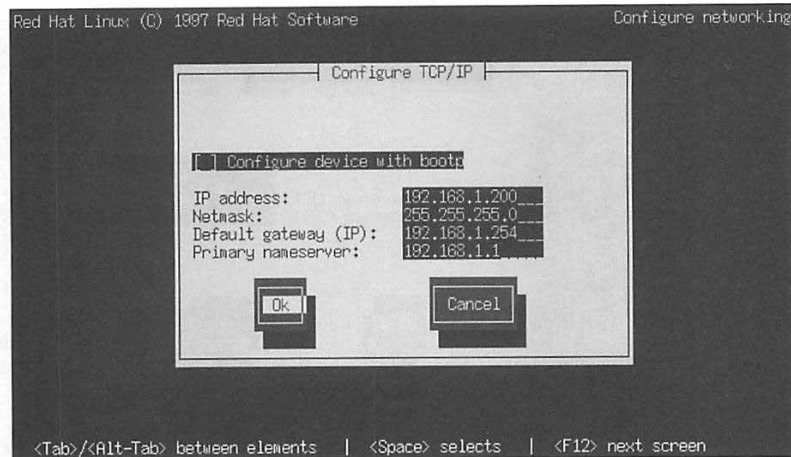
Figure 1-33: TCP/IP configuration dialog where you enter IP addresses for a TCP/IP LAN (sample values filled in)

The dialog box asks for four key parameters:

■ IP address of the Linux system (the one you are installing)

■ Network mask

■ IP address of the gateway (the system through which you might go to any outside network)

■ IP address of the name server

Cross-Reference

If you have a private LAN (one not directly connected to the Internet), you may use an IP address from a range that has been designated for private use. A common IP address for private LANs are the addresses in the range 192.168.1.1 through 192.168.1.254. You learn more about TCP/IP networking and IP addresses in Chapter 16.

After you enter the requested parameters, press Tab to select OK and then press Enter. The installation program then displays another dialog box (Figure 1-34) that asks for more information about the TCP/IP LAN.
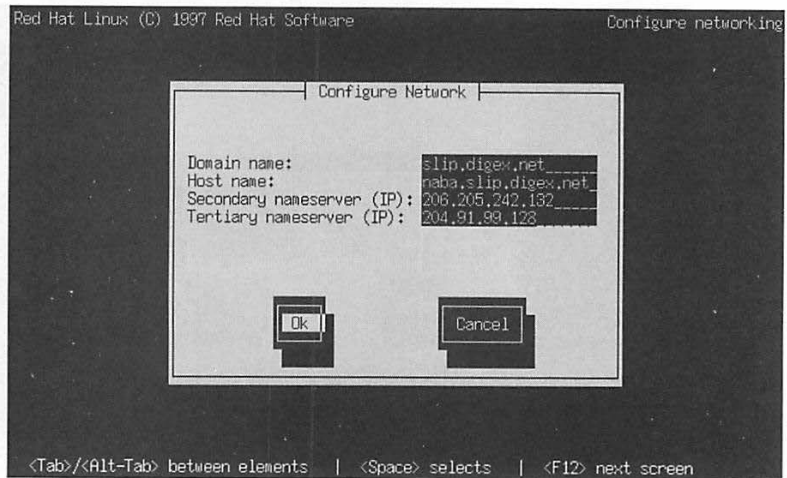
Figure 1-34: Dialog box for further network information (sample values filled in)

This time you have to enter the following information:

- Domain name of your LAN
- The host name for your Linux system (for a private LAN, you can assign your own host name)
- IP address of a secondary name server
- IP address of a tertiary name server

After you enter the requested information, select OK and press Enter.

Set the time zone

After completing the network configuration, you must select the *time zone* — the difference between the local time and the current time at Greenwich, England, which is the standard reference time (also known as Greenwich Mean Time, or GMT). The installation program shows you a list of time zones (Figure 1-35), in terms of country or regions.

You see time zones for many countries organized by the continents and for many regions of the United States. Press the up and down arrow keys to select your location. If you live on the East Coast of the United States, for example, you would select USA/Eastern. If your country is not listed, select one of the entries that designates the difference between your local time and GMT. After you select your time zone, press **Enter**.
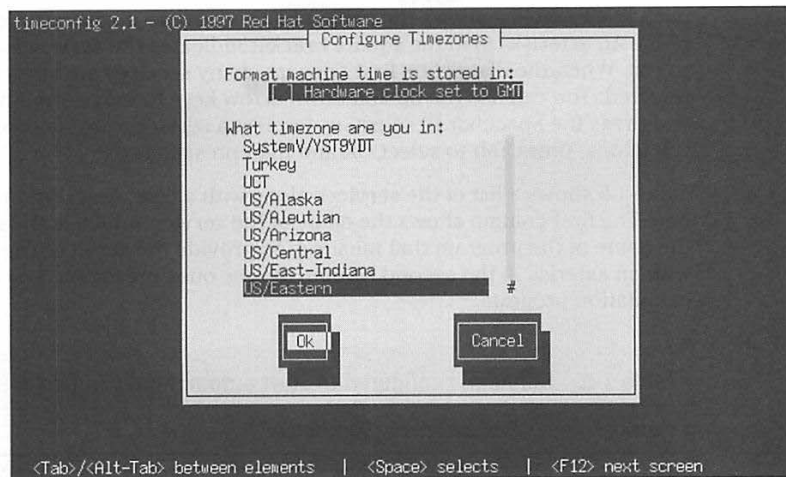
Figure 1-35: Dialog box where you select your time zone

Configure services

After time zone selection, the Red Hat installation program proceeds to set up the services that should run automatically every time you start your Linux system. You have to select the desired services from the list shown in the dialog box of Figure 1-36.
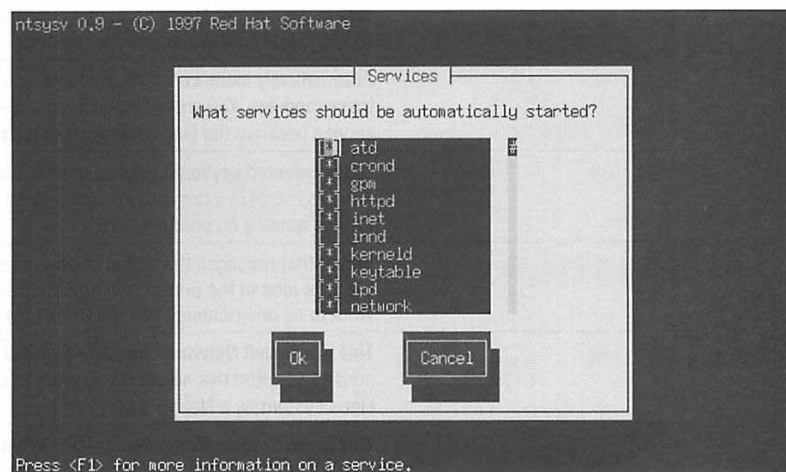
Figure 1-36: Dialog box where you select the services to start automatically

Each line in the list shows the name of a service with a square bracket prefix ([ ]). An asterisk (*) in the square bracket indicates the service is selected to run. When the dialog box first appears, many services are already selected. You can use the up and down arrow keys to move around in the list and press the Spacebar to select or deselect a service. After making your selections, press Tab to select OK and then press Enter.

Table 1-5 shows a list of the services along with a brief description of each one. The first column shows the name of the service, which is the same as the name of the program that must run to provide the service. The names with an asterisk in the second column are the ones preselected by the installation program.

Table 1-5   Services configured to start automatically in Linux

Service Name	Pre-selected	Description
atd	*	Runs commands scheduled by the at and batch commands.
crond	*	Runs user-specified programs according to a periodic schedule set by the crontab command.
gpm	*	Enables use of mouse in text-mode screens.
httpd	*	This is the Apache World Wide Web (WWW) server.
inet	*	This is the Internet super server, also known as inetd. It starts other Internet services, such as Telnet and FTP, whenever they are needed.
innd		This is the Internet News server you can use to support local new groups on your system.
kerneld	*	Automatically loads kernel modules as Linux needs these modules. You should always run the kerneld service because the Linux operating system needs it.
keytable	*	Loads selected keyboard map as specified in the file /etc/sysconfig/keyboard. You should leave this service running on your system.
lpd	*	Server that manages the queue of print jobs and sends the print jobs to the printer. You need this server if you want to do any printing from the Linux system.
mars-nwe		This is a Novell Netware compatible MARS file and print server. Run this server if you want to use your Linux system as a Netware server.
named		This is the Domain Name Server (DNS) that translates host names into IP addresses. You can run a copy on your system, if you want.

Service Name	Pre-selected	Description
network	*	This server enables you to activate or deactivate all network interfaces configured to start at system boot time.
nfs	*	Exports file systems using the Network File System (NFS) protocol so that other systems (running NFS) can share files from your system.
nfsfs	*	Server to mount or unmount NFS file systems.
portmap	*	Server used by any software that relies on Remote Procedure Calls (RPC). NFS requires the portmap service.
postgresql	*	Starts stops the PostgreSQL server that handles database requests (PostgreSQL is a free database that comes with Red Hat Linux).
random	*	Server needed to generate high-quality random numbers on the Linux system.
routed		Updates IP routing tables using the RIP protocol. You needn't enable this service unless you need to route IP packets between multiple networks in your organization.
rusersd		Enables users on any system on the network to find out who is logged in at a system.
rwhod		Enables remote users get a list of all users on the Linux system running the rwhod service.
sendmail	*	Moves mail messages from one machine to another. Start this service if you want to send mail from your Linux system.
smb	*	Starts and stops the Samba smbd and nmbd services used to support LAN Manager services on a Linux system.
snmpd	*	Simple Network Management Protocol (SNMP) service used for network management functions.
sound	*	Saves certain sound card settings at system shutdown and restores them at startup.
syslog	*	Service used by many other programs (including other services) to log various error and status messages in a log file (usually, the /var/log/messages file). You should always run this service.
ypbind		Service needed for Network Information System (NIS). You needn't start ypbind unless you are using NIS.

Configure printers

After setting up the services, the Red Hat installation program displays a dialog box (Figure 1-37) that asks whether you want to configure a printer. If you intend to do any printing from the Linux system, select Yes from this dialog box.

```
Red Hat Linux (C) 1997 Red Hat Software                    Configure printer




                        ┤ Configure Printer ├
                    Would you like to configure a printer?

                    ┌─────┐              ┌─────┐
                    │ Yes │              │ No  │
                    └─────┘              └─────┘




<Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```
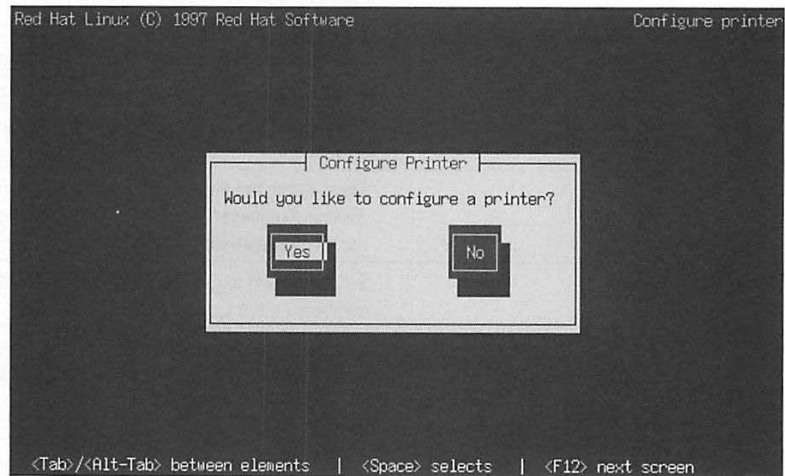
Figure 1-37: This dialog box asks if you want to configure a printer.

Tip

You should select Yes from the dialog box of Figure 1-37 even if you do not have any printer directly connected to the Linux system. You have to configure a printer even if you plan to print on a remote printer. This step enables you to select a shared printer connected to another Linux system or a Windows 95 system.

After you select Yes and press Enter, the installation program asks you about the type of printer connection, as shown in Figure 1-38.

You can select from three types of printer connections:

- **Local**: Refers to a printer connected directly to the PC where you are installing Linux.

- **Remote lpd**: Refers to a printer physically connected to another Linux system on the local network.

- **LAN Manager**: Refers to a printer connected to another PC on the local network and that runs the LAN Manager protocol (typically used to share resources such as disks and printers among PCs running Windows 95 or Windows NT).
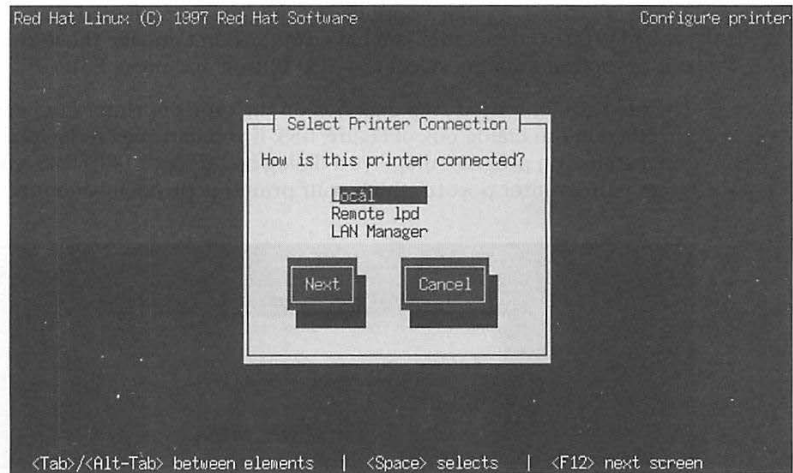
Figure 1-38: Selecting the type of printer connection

You should specify the printer connection that applies to your configuration. After you select a connection type, the installation program displays a dialog box showing two standard printer options, as shown in Figure 1-39.
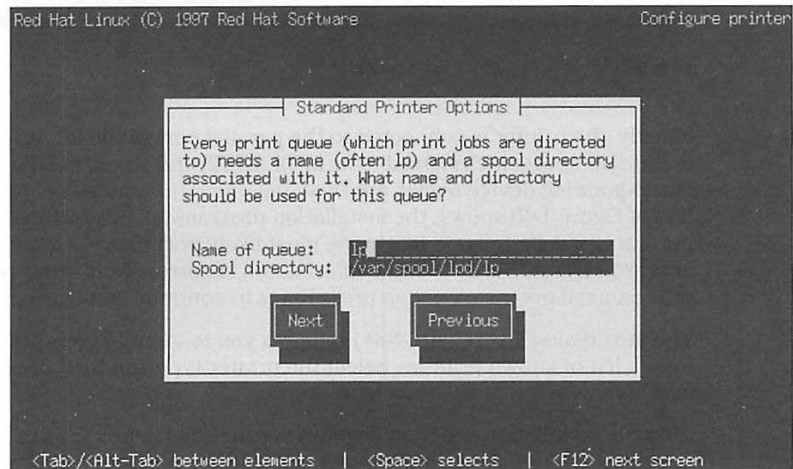
Figure 1-39: Selecting standard printer options

These are called standard options because they are needed no matter what type of printer connection you have. You needn't change these standard printer options; simply select the Next button and press Enter.

The next configuration step depends on the type of printer connection you specified in the dialog box of Figure 1-38. If you had selected a local printer, the installation program displays a dialog box (Figure 1-40) asks you to specify the printer port to which your printer is physically connected.
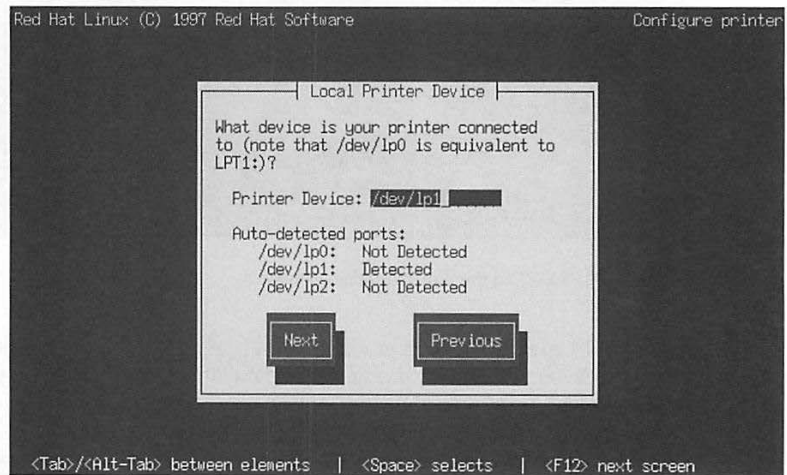
Figure 1-40: Printer port for local printer

Usually, the printer is connected to the parallel port of the PC. In Windows, the parallel ports are referred to as LPT1, LPT2, and so on. In Linux, the corresponding device names are /dev/lp0, /dev/lp1, and so on. As the dialog box of Figure 1-40 shows, the installation program tells you which parallel ports it has detected and fills in the most likely port name in a text-entry area. You should verify that the information is correct (and enter the correct port name, if necessary); then press Enter to continue configuring the printer.

The next dialog box (Figure 1-41) requires you to identify your printer type from a list of known printers. Select the printer type and then press **Enter** to continue with the printer configuration.

Next the installation program displays another dialog box (Figure 1-42) requesting information on the paper size and the resolution of the printer (typically 600×600 dots per inch in many modern laser printers). You should select the appropriate values.
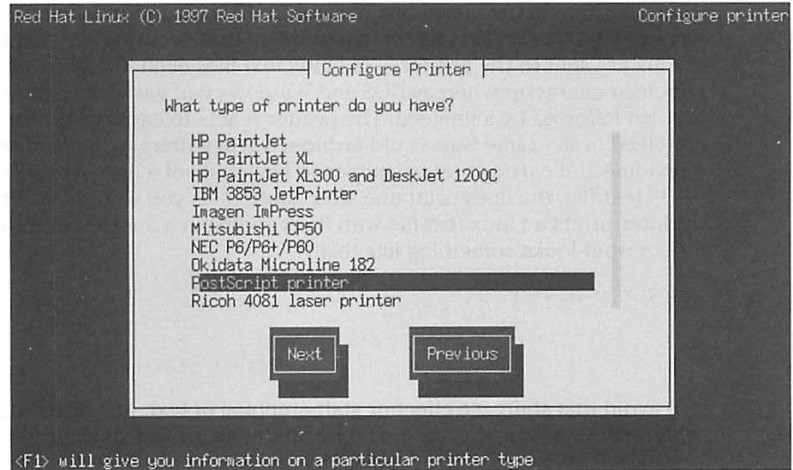
Figure 1-41: Selecting the printer type
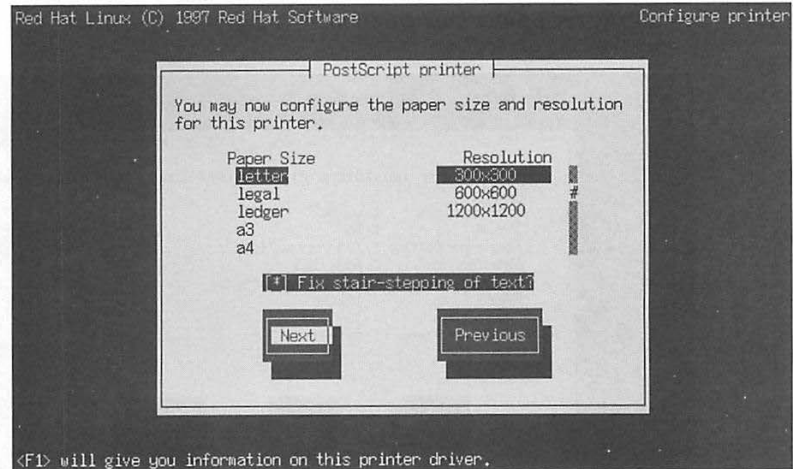
Figure 1-42: Dialog box to select paper size and resolution of selected printer

Another item in the dialog box of Figure 1-42 is an option labeled `Fix stair-stepping of text`. This refers to a problem that occurs when text files from Linux are sent to the printer — in Linux text files each line ends with a single linefeed character whereas DOS and Windows end each line with a carriage return followed by a linefeed. The printer reacts to carriage return and linefeed in the same way as old-fashioned typewriters — linefeed advances to next line and carriage return moves to beginning of a line. When printing a DOS text file, the lines print one after another, as you would expect. When the printer prints a Linux text file with lines ending in a linefeed only, however, the output looks something like this:

```
Line 1 ends here
                Line 2 here
                             and so on
                                        like a staircase.
```

To avoid this staircase effect or stair-stepping of text, just check the box marked `Fix stair-stepping of text` in the dialog box of Figure 1-42 and Linux will take care of the problem by sending a carriage return before each linefeed whenever you print a text file.

After you select the options in Figure 1-42 and press Enter, the installation program shows you all the options in another dialog box (Figure 1-43) so you can verify the printer configuration.

Figure 1-43: Verifying the printer settings

After confirming the settings are correct, press Enter to proceed to the next configuration step.

Set the root password

After completing the printer configuration, the installation program prompts you for a root password, as shown in Figure 1-44.
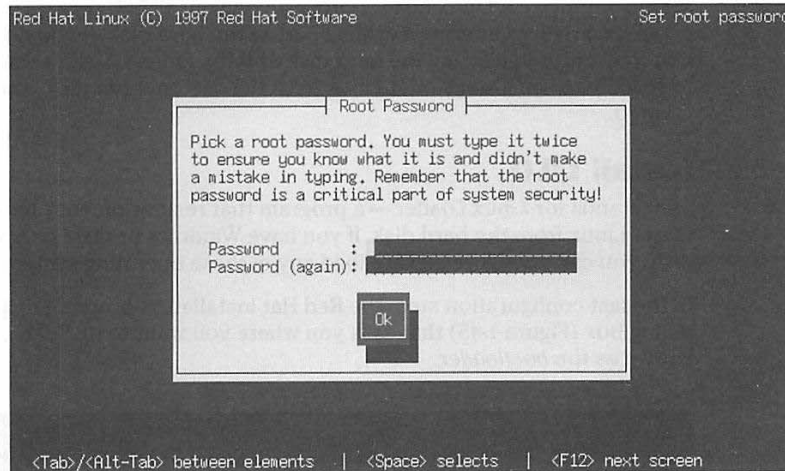
Figure 1-44: A dialog box enables you to set the root password.

The root user is the super user in Linux. Because the super user can do anything in the system, you should assign a password you can remember, but that others cannot guess easily. Typically, you would make the password at least eight characters long, include a mix of letters and numbers, and, for good measure, throw in some special characters such as + or *.

Type the password on the first line and then re-enter the password on the next line. The password does not appear onscreen. You have to type the password in twice and both entries must match before the installation program accepts the password. This ensures you did not make any mistakes in typing. After you type the password twice, select OK and press Enter to proceed to the next configuration step.

Create a custom boot disk

After you specify a root password, the installation program displays a dialog box that asks if you want to create a custom boot disk. You can use this disk to boot your Linux system if the Linux kernel on the hard disk is damaged or if the Linux Loader (described in the next section) does not work. Press Enter to answer Yes. The installation program then asks you to insert a formatted floppy into your PC's A drive. You should place a formatted floppy in the A drive and press Enter (note, all data on the floppy will be destroyed).

The installation program copies the Linux kernel and some other files to the floppy. After the setup program prepares the custom boot disk, remove the floppy from the A drive, label it appropriately, and save it for future use.

You can use this boot disk to start Linux on your PC. In the next section, you see how to place a Linux Loader program on your PC's C drive. That process, however, involves altering a critical part of the hard disk. If you do not feel comfortable with altering the hard disk or if the Linux Loader somehow fails to work, you can always boot Linux from the custom boot disk you just created.

Install LILO

LILO stands for *Linux Loader* — a program that resides on your hard disk and starts Linux from the hard disk. If you have Windows or OS/2 on your hard disk, you can configure LILO to load any of these operating systems as well.

In the last configuration step, the Red Hat installation program displays a dialog box (Figure 1-45) that asks you where you want to install LILO, also known as the *bootloader*.
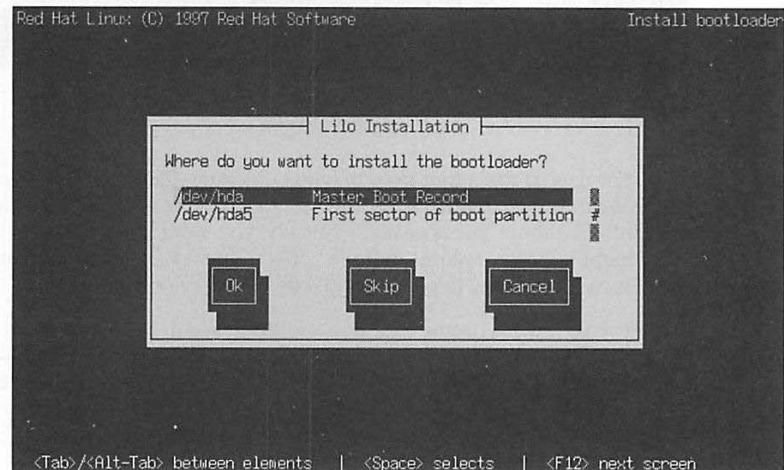
Figure 1-45: Specifying where to install LILO

The dialog box gives you the option to install LILO in one of two locations:

- Master Boot Record (MBR), which is located in the first sector of your PC's hard disk (the C drive)
- First sector of the partition where you loaded Linux

Select a location — preferably the Master Boot Record — press Tab to select OK, and then press **Enter**.

Caution

Although a Skip button enables you to skip LILO installation, you need to install LILO, or you will be unable to start Linux after you finish the installation. At the same time, though, you should know installing LILO in the Master Boot Record can be risky. If LILO is not configured properly, your PC may be unable to boot from the hard disk. Unfortunately, the Red Hat installation program does not offer any other option for booting Linux. You have to wait until you have started Linux for the first time before you can prepare a boot disk to start Linux from your PC's A drive.

After you select the location for LILO installation, the installation program displays another dialog box (Figure 1-46) that enables you to enter any special options that may be needed by Linux as it boots.
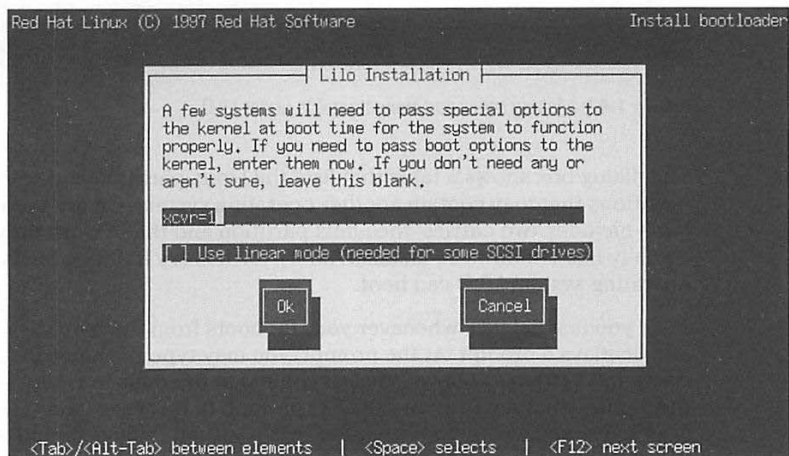
Figure 1-46: Entering boot-time options for Linux

Whether you need any special options depends on what hardware you have. For example, on my system, I have a 3Com 3C503 Ethernet card connected to the local Ethernet using an external transceiver. When Linux loads the driver software for the 3Com 3C503 card, I need to specify the option xcvr=1 to make sure the driver uses the external transceiver. I enter this option on the first line of the dialog box, as shown in Figure 1-46.

After you enter any necessary options, select the OK button, and press Enter to continue with LILO installation. The Red Hat installation program displays another dialog box (Figure 1-47) that gives you the option to make other disk partitions bootable.
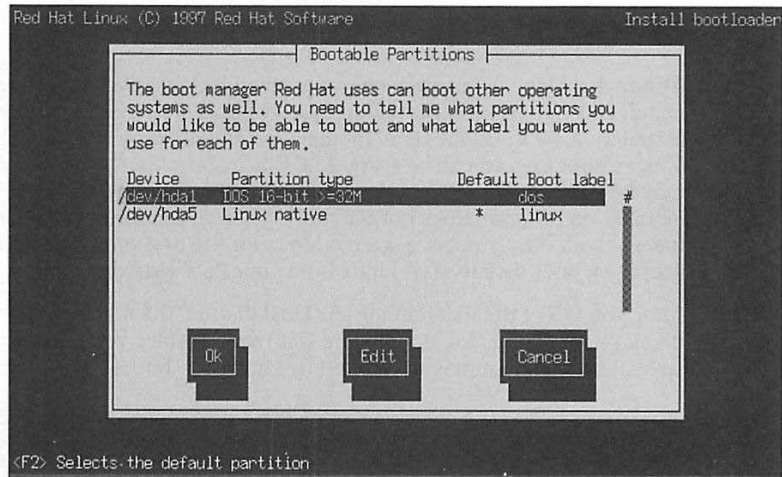
Figure 1-47: Making other partitions bootable under LILO

The dialog box shows a table that lists the Linux partition and any other partitions that may contain another operating system. On my system, the table includes two entries: the Linux partition and the DOS partition (that actually has Windows 95 installed on it). Each entry in this table is an operating system LILO can boot.

After you install LILO, whenever your PC boots from the hard disk, LILO runs and displays a prompt. At the prompt, you may type the name of an operating system to boot—the last column of the table in Figure 1-47 shows the names you may enter at the LILO prompt. In this case, you type linux to boot Linux and dos to boot from the DOS partition (which should start Windows 95, if that's what you have installed on that partition).

If you enter nothing at the LILO prompt, it waits for a few seconds and boots the default operating system. The default operating system is the one with an asterisk in the Default column in Figure 1-47 (in this case Linux is the default).

If you want to change any of the entries, select Edit and then press Enter. If you accept the choices as shown in the dialog box of Figure 1-47, press Tab to select OK and then press Enter. This installs LILO and completes the Red Hat Linux installation.

Starting Linux for the First Time

After you finish installing LILO, the Red Hat Linux installation program automatically reboots the system. The PC goes through its normal power-up sequence and loads LILO from the C drive and the following prompt appears:

```
LILO boot:
```

When you install LILO, if you specified the Linux partition as the default one, you can simply wait; after a few seconds, LILO boots Linux.

If you want to boot from another partition (such as DOS), press the Tab key. LILO displays the names of the available bootable partitions. For example, a typical display might be:

```
linux dos
```

You can then type the name of the partition you want to boot (or press Enter to boot from the first partition).

After LILO boots Linux, you should see a long list of opening messages, including the names of the devices Linux detects. One of the first few messages says Calibrating delay loop.. ok - 299.01 BogoMIPS; the number that precedes BogoMIPS depends on your system's processor type. *BogoMIPS* is a Linux jargon term that is the subject of countless discussions in various USENET newsgroups devoted to Linux. (*Usenet* is a loose collection of computers that exchange electronic mail and news. You learn more about Usenet in Chapters 18 and 19.) Linux uses the BogoMIPS measurement in situations in which the operating system has to wait for a specified period.

At the end of all the messages, you see the Linux login prompt, as follows:

```
Red Hat Linux release 5.1 (Manhattan)
Kernel 2.0.34 on an i586
hostname login:
```

where *hostname* is the name you assigned to your system when you configured the network. If you do not configure the network, localhost is used as the system name.

Because there are no other users at this point, type **root** and press Enter. Then type the root password (the one you set during installation) to log in as the super user. Now you can perform a few initial chores and learn how to shut down your Linux system.

What is BogoMIPS?

When your Linux system boots, you notice a message such as `Calibrating delay loop.. ok - 299.01 BogoMIPS`, with some number before `BogoMIPS`. BogoMIPS is one of those words that confound new Linux users. This sidebar explains what BogoMIPS means.

As you may know, *MIPS* stands for *millions of instructions per second* — a measure of how fast your computer runs programs. (As such, MIPS is not a good measure of performance, because comparing the MIPS of different types of computers is difficult.) *BogoMIPS* is *bogus MIPS*, which refers to an indication of the computer's speed. Linux uses the BogoMIPS number to calibrate a delay loop, in which the computer processes some instructions repeatedly until a specified amount of time has passed.

The BogoMIPS numbers can range anywhere from 1 to 300 or more, depending on the type of processor (386, 486, or Pentium). A typical 33MHz 80386DX system has a BogoMIPS of about 6, whereas a 66MHz 80486DX2/66 system shows a *BogoMIPS* of about 33. The BogoMIPS for older Pentium systems is on par with (or slightly less than) that of 80486 processors, because the *BogoMIPS* calculation does not take advantage of any advanced features (such as the capability to execute instructions in parallel) of the Pentium. On my 75MHz Pentium system, Linux reports a *BogoMIPS* of 30.22. On a more recent 200MHz Pentium MMX system, however, the *BogoMIPS* is 299.01.

Recovering from a forgotten root password

If you forget the root password, you can follow these steps to set a new root password:

1. Power up your PC as usual. At the LILO boot prompt, type the name of the Linux boot partition followed by the word `single`, as follows (you type the text shown in boldface):

   ```
   LILO boot: linux single
   ```

   This causes Linux to start up as usual, but run in a single-user mode that does not require you to log in. After Linux starts, you see the following command-line prompt:

   ```
   bash#
   ```

2. Use the `passwd` command to change the root password as follows:

   ```
   bash# passwd
   New UNIX password:
   ```

   Type the password you want to use (it won't appear onscreen) and then press Enter. Linux asks for the password again, as follows:

   ```
   Retype new UNIX password:
   ```

   Type the password again, and press Enter. If you enter the same password both times, the `passwd` command changes the password and displays a message:

   ```
   passwd: all authentication tokens updated successfully
   ```

3. Now you reboot the PC by pressing Ctrl+Alt+Delete. After Linux starts, it displays the familiar login prompt again. Now you should be able to log in as root with the new password.

Creating a boot floppy disk

After you successfully start Linux and log in as root, you should immediately create a Linux boot floppy disk. You can use this floppy disk to boot your Linux system if the Linux kernel on the hard disk is damaged. You should also have the boot floppy disk before you do anything to the Linux kernel, such as upgrade it to a new version following the procedures outlined in Chapter 2.

To create the boot floppy disk, follow these steps:

1. Insert a formatted floppy disk into your PC's A drive. If you have an unformatted floppy, you can format it with the following command:

   ```
   fdformat /dev/fd0H1440
   Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
   Formatting ... done
   Verifying ... done
   ```

2. Copy the file `vmlinuz` from the `/boot` directory to the floppy disk. (The `vmlinuz` file happens to be the Linux kernel.) Use the following command:

   ```
   cd /boot
   cp vmlinuz /dev/fd0
   cp: overwrite `/dev/fd0'? y
   ```

3. Determine the kernel's root device — the disk partition where the root file system is located — with the following command:

   ```
   rdev
   /dev/hda3 /
   ```

   In this case, the root device is `/dev/hda3`; the device name may be different on your system.

4. Set the kernel's root device. Use the device reported by the `rdev` command in the previous step. For example, to set the root device to `/dev/hda3`, I type:

   ```
   rdev /dev/fd0 /dev/hda3
   ```

5. Mark the root device as read-only. Use the following command:

   ```
   rdev -R /dev/fd0 1
   ```

   This causes Linux initially to mount the root file system as read-only. Then Linux checks the file system for any errors and, if no errors exist, the file system is mounted in read-write mode. By setting the root device as read-only, you avoid several warning and error messages.

After you prepare the boot floppy disk, you should try it. Go through the next four sections. Then, when you are ready to shutdown the system, put the boot floppy disk in the A drive and type the following command:

```
/sbin/shutdown -r now
```

After the PC reboots, Linux should start as usual and you should see a login prompt. If it does not work, remove the floppy disk from the A drive and press **Ctrl+Alt+Delete** to boot from the hard disk. Then, you can repeat the procedure for creating the boot floppy disk.

Starting X

In one of the Red Hat Linux installation steps, you configured the X Window System. The configuration step creates a file that contains information about your mouse, keyboard, video card, and monitor. (You learn all about the *X* configuration file in Chapter 4.) For now, you should quickly start *X* and see if the configuration file works. To start *X,* type the following command at the Linux prompt:

```
startx
```

You will see a number of messages flash across the screen as the *X* server starts.

Caution

If the screen goes crazy, press Ctrl+Alt+Backspace to kill *X*. If this does not stop *X,* press the reset button of your computer. Otherwise, your monitor may be damaged.

If the configuration file does not work at all, you will see a number of error messages: one saying the *X* server is aborting and then a message about being unable to connect to the *X* server that goes something like this:

```
_X11TransSocketUNIXConnect: Can't connect: errno = 111
giving up.
xinit:  Connection refused (errno 111): unable to connect to X server
xinit:  No such process (errno 3): Server error.
```

Cross-Reference

Although the error messages seem to refer to sockets and connections, the reason is usually problems with the configuration file — the *X* server could not start because it had problems with some of the settings in the *X* configuration file. One common problem is the *X* server could not find a suitable video mode (screen resolution) for your monitor and video card. Consult Chapter 10 to learn more about the *X* configuration file and what settings might help you get *X* running on your system.

If all goes well, the screen will turn gray for a time and you will see a cursor shaped like an *X.* After a little while, the screen background changes and a number of windows appear, as shown in Figure 1-48.

Cross-Reference

Depending on your video card and monitor, your screen layout may be different from Figure 1-48. In particular, your screen might show only a small part of a larger virtual screen; then you have to move the mouse down to see the toolbar. For now you can ignore these nuances. Chapter 5 explains the layout of various elements onscreen and how you may customize them.
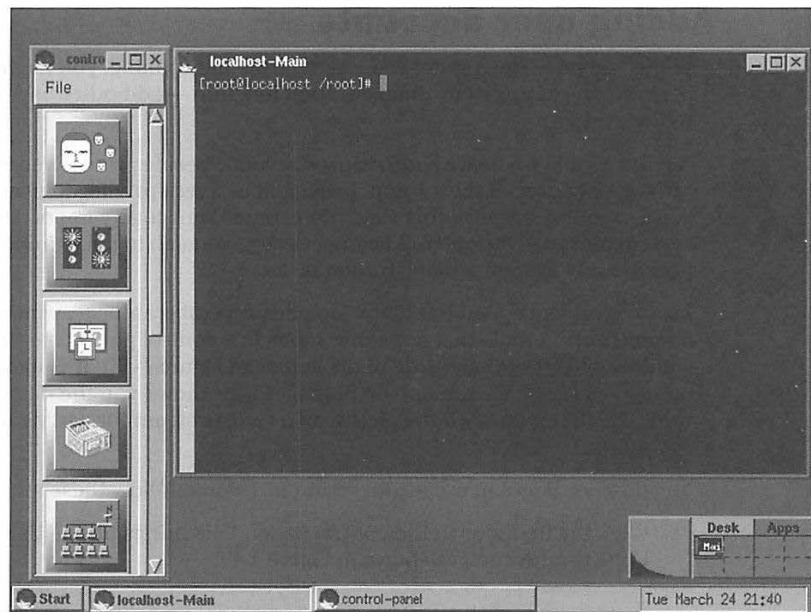
Figure 1-48: Successfully starting *X* for the first time

Look at the initial *X* window screen as it appears in Figure 1-48. You should note three key items on the screen:

- Along the left edge, you see a control panel — a window with a number of large buttons, each with a picture on it. These icons represent tools (computer programs) you can start by clicking each icon. Chapter 6 shows you how to use the control panel tools to perform some Linux system administration chores.

- To the right of the control panel — occupying much of the screen space — is a terminal window. You can type Linux commands in this window. Chapter 6 introduces you to some Linux commands and Appendix B lists many commonly used Linux commands.

- A task bar, similar to the one in Windows 95, appears along the bottom edge of the screen. As in Windows 95, the task bar has a Start button and a button for each of the open windows. The right edge of the task bar shows the current date and time.

Adding user accounts

While you have the *X* graphical interface up and running, you might as well perform one key system administration function — add other user accounts to the system.

Tip A good idea is to create another user account besides root. Even if you are the only user of the system, logging in as a less-privileged user is good practice, because this way, you cannot damage any important system files inadvertently. When necessary, you can log in as root and perform any system administration tasks.

As a convenience, Red Hat Linux includes several graphical system administration tools that appear as icons in a control panel (see the tall window on the left-hand side of the screen in Figure 1-48). If you move the mouse over each icon, a pop-up help message shows you the name of this tool. The first icon, with the picture of a face, is meant for adding user accounts.

To add myself as a new user, for example, I would follow these steps:

1. Click the first icon in the control panel. This brings up the User Configurator tool, as shown in Figure 1-49.
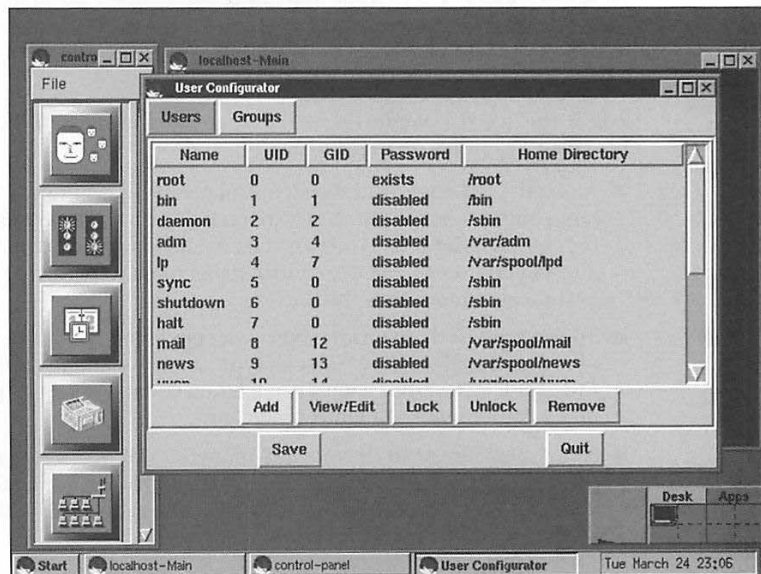
Figure 1-49: Main window of the User Configurator tool

2. Click the Add button in the main window of Figure 1-49. This brings up the Edit User Definition dialog box, as shown in Figure 1-50.
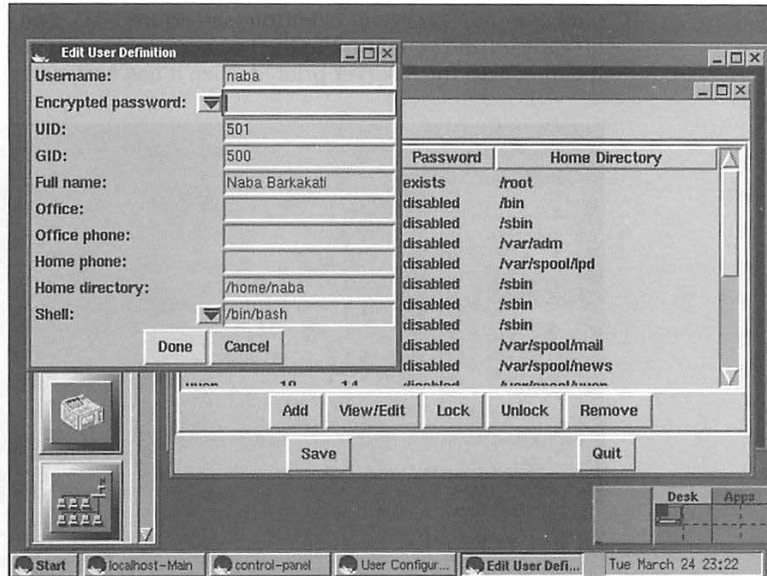
Figure 1-50: Entering information for a new user account

3. Fill in the requested information. In particular, you must enter the user name, password and the home directory. For my account, I use /home/naba as the home directory. You can use a similar home directory for your account. After you fill in all the fields, click the Done button. This returns you to the main window of the User Configurator tool (Figure 1-49).

4. Click the Quit button. A dialog box asks for confirmation. Click Save and Quit button to save the new user information and exit the tool.

Quitting X

Now that you have tried *X* and used one of the graphical tools under *X*, you should learn how to quit *X*. To shut down the X Window System, follow these steps:

1. Click the Start button on the task bar (at the bottom-left corner of the screen). The Start menu pops up (similar to Windows 95).

2. Move the mouse to the `Exit Fvwm` item (*Fvwm* refers to the window manager — you have to exit the window manager to shut down *X*). Another menu appears.

3. Click the `Yes, Really Quit` button (see Figure 1-51). The *X* screen disappears and you return to the text screen. You can now see some of the messages the *X* server printed when it had first started.
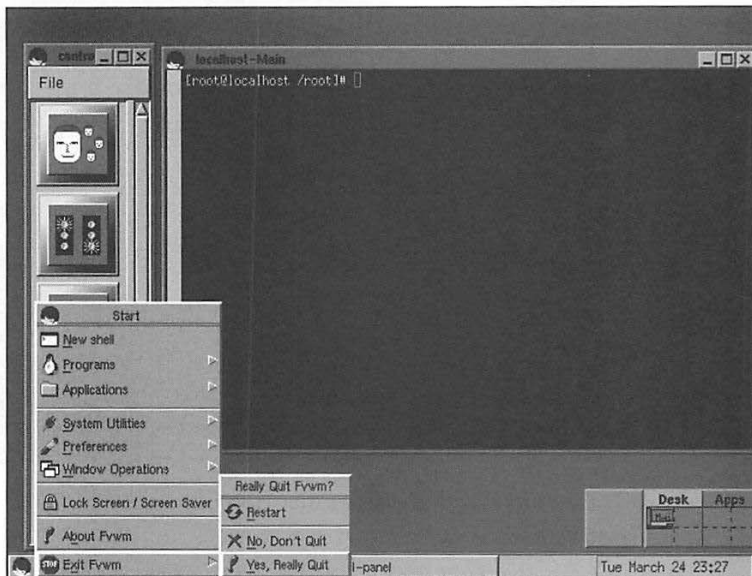
Figure 1-51: Shutting down the X Window System

Looking up the online documentation

You should know an important source of information in Linux. Every so often, you see instructions asking you to enter a Linux command. After a while, you are bound to get to the point at which you vaguely recall a command's name, but you cannot remember the exact syntax of what you are supposed to type. This situation is where the Linux online manual pages can come to your rescue.

You can view the manual page — commonly referred to as the *man page* — for a command by using the man command. (You do have to remember this command to be able to look up online help.) To view the man page for the `passwd` command, for example, type the following command:

```
man passwd
```

Linux displays the information page by page. Press the Spacebar to move to the next page. When you finish, press **q** to return to the Linux command prompt.

Having touted the usefulness of the online help pages, I must point out the term *Linux command* refers to any executable file, ranging from a script file that contains other Linux commands to standard Linux executable programs. Although man pages exist for most standard programs, many programs do not have any online help. For example, if you type `help` at a Linux command prompt, you get a list of commands that are defined internally to the command processor (called a *shell*). When you type **man help**, however, Linux responds as follows:

```
No manual entry for help
```

Nevertheless, whenever you are at a loss about some command, using the man command is worthwhile to see whether any online help for that command exists.

Another form of online documentation is the HOWTO files in the /usr/doc/HOWTO directory of your system (these files are installed if you select the Extra Documentation component during installation). The HOWTO files all have a `-HOWTO.gz` suffix in the file name. Each file is a text file that has been compressed using the `gzip` command.

Each HOWTO file contains information about some area of Linux, such as the hardware it supports or how to create a boot disk. Table 1-6 lists the HOWTO files that should be in the `/usr/doc/HOWTO` directory of your Linux system provided you had installed the Extra Documentation component.

To view any of these files, go to the `/usr/doc/HOWTO` directory and use the `zcat` command as follows:

```
cd /usr/doc/HOWTO
zcat topic-name-HOWTO.gz | more
```

where `topic-name` is the first part of the filename excluding the `-HOWTO.gz` suffix. Thus, to view the `Bootdisk-HOWTO.gz` file, type:

```
zcat Bootdisk-HOWTO.gz | more
```

You needn't type the long filename; just type `zcat Bootd` and then press Tab. The Linux command processor — the shell — automatically completes the filename that matches what you typed. Then you can type `| more` to complete the command line.

Table 1-6   Linux HOWTO files in `/usr/doc/HOWTO` **directory**

HOWTO file name	*Contents*
`3Dfx-HOWTO.gz`	How to configure Linux to support the 3Dfx graphics accelerator chip.
`AX25-HOWTO.gz`	How to install and configure Linux to support for AX.25 packet radio protocol used by Amateur Radio Operators worldwide.
`Access-HOWTO.gz`	How to make Linux useable by persons with disabilities.
`Alpha-HOWTO.gz`	Overview of Digital Equipment Corporation's 64-bit RISC architecture Alpha CPU (Linux runs on the Alpha).
`Assembly-HOWTO.gz`	How to program in assembly language in Linux.
`Benchmarking-HOWTO.gz`	How to benchmark a Linux system — measure how fast the system completes a computing task.
`BootPrompt-HOWTO.gz`	A list of arguments that can be passed to Linux at boot time (at the `LILO boot:` prompt).
`Bootdisk-HOWTO.gz`	How to create boot, root, and other utility disks for Linux.
`CD-Writing-HOWTO.gz`	How to record a CD-ROM using a CD Recorder installed in a Linux system.
`CDROM-HOWTO.gz`	How to install, configure, and use CD-ROM drives with Linux.
`Chinese-HOWTO.gz`	How to configure Linux for use with Chinese character set and Chinese version of X Window System.
`Commercial-HOWTO.gz`	A list of commercial software for Linux.
`Consultants-HOWTO.gz`	A list of consultants or consulting firms that provide support for Linux.
`Cyrillic-HOWTO.gz`	How to typeset and print documents in the Russian language on a Linux system.
`DNS-HOWTO.gz`	How to set up Domain Name Service (DNS) on a Linux system.
`DOS-to-Linux-HOWTO.gz`	How to apply your knowledge of MS-DOS in Linux.
`DOSEMU-HOWTO.gz`	How to set up and use the MS-DOS Emulator, DOSEMU.
`Danish-HOWTO.gz`	How to configure Linux for use with the Danish character set.
`Database-HOWTO.gz`	How to set up the PostgreSQL Relational Database System on Linux (PostgreSQL comes with Red Hat Linux on the companion CD-ROM).
`Disk-HOWTO.gz`	How best to use multiple disks and partitions in a Linux system.

HOWTO file name	Contents
`Distribution-HOWTO.gz`	A list of Linux distributions with particular focus on commercial CD-ROM distributions.
`ELF-HOWTO.gz`	How to compile and run programs in the ELF (Executable and Linking Format) binary format (the version of Linux on the companion CD-ROM is already configured to run ELF binaries).
`Emacspeak-HOWTO.gz`	How to support blind users using Emacspeak, an Emacs subsystem that provides feedback through synthesized speech.
`Ethernet-HOWTO.gz`	How to configure and use Ethernet network adapters with Linux.
`Finnish-HOWTO.gz`	How to set up Linux for the Finnish language (except for the initial paragraph, the HOWTO itself is in Finnish).
`Firewall-HOWTO.gz`	How to set up an Internet firewall on a Linux system.
`French-HOWTO.gz`	How to set up Linux for the French language (this HOWTO is in French).
`Ftape-HOWTO.gz`	How to set up and use floppy tape drives (QIC-40, QIC-80, QIC-3010, and QIC-3020 compatible tape drives that connect to your PC through the floppy disk controller) in Linux.
`GCC-HOWTO.gz`	How to set up and use the GNU C compiler and development libraries in Linux.
`German-HOWTO.gz`	How to use Linux with the German character set (this HOWTO is in German).
`Glibc2-HOWTO.gz`	How to install and use the GNU C Library Version 2 (libc 6) on Linux systems.
`HAM-HOWTO.gz`	Information about amateur radio software for Linux.
`Hardware-HOWTO.gz`	A list of hardware known to work with Linux and how to locate any necessary drivers.
`Hebrew-HOWTO.gz`	How to support Hebrew character set in X Window System and text-mode screens.
`IPX-HOWTO.gz`	How to obtain, install, and configure various software that use the Linux support for IPX protocol (IPX is used by Novell Netware).
`ISP-Hookup-HOWTO.gz`	How to connect a Linux system to an Internet service provider (ISP) via a dial-up modem connection.
`Installation-HOWTO.gz`	How to obtain and install Linux.
`Intranet-Server-HOWTO.gz`	How to use a Linux system in an Intranet Intranet that ties together UNIX, Novell Netware, Windows NT, and Windows 95 systems.

(continued)

Table 1-6 *(Continued)*

HOWTO file name	Contents
Italian-HOWTO.gz	How to set up Linux for the Italian language (this HOWTO is in Italian).
Java-CGI-HOWTO.gz	How to develop and use Java programs in Linux.
Kernel-HOWTO.gz	How to upgrade and rebuild the Linux kernel.
Keyboard-and-Console-HOWTO.gz	How to use various Linux utilities to configure the keyboard and the console (the text-mode screen).
MGR-HOWTO.gz	How to use the MGR (ManaGeR) graphical windowing system in Linux.
MILO-HOWTO.gz	How to set up and use the Miniloader (MILO) for Linux on Alpha AXP based systems (just as LILO loads and starts Linux on Intel-based PCs, MILO loads Linux on Alpha systems).
Mail-HOWTO.gz	How to set up and maintain Electronic Mail (e-mail) on a Linux system.
NET-3-HOWTO.gz	How to install and configure TCP/IP networking in Linux.
NFS-HOWTO.gz	How to set up NFS (Network File System) server and client in Linux.
NIS-HOWTO.gz	How to configure NIS (Network Information Service) in Linux.
News-HOWTO.gz	How to set up and maintain USENET news (discussion groups) in Linux.
Optical-Disk-HOWTO.gz	How to install and configure optical disk drives in Linux (includes detailed coverage of the Panasonic LF1000 PD Phase change optical drive with the SCSI-II interface).
PCI-HOWTO.gz	Information on Linux's support for the PCI (Peripheral Component Interconnect) bus architecture.
PCMCIA-HOWTO.gz	How to install and use PCMCIA (Personal Computer Memory Card International Association) Card Services in Linux.
PPP-HOWTO.gz	How to set up and use Point-to-Point Protocol (PPP) networking in Linux.
Pilot-HOWTO.gz	How to use the 3Com (U.S. Robotics) Palm Pilot personal digital assistant (PDA) with a Linux system.
Polish-HOWTO.gz	How to set up Linux for the Polish language (this HOWTO is in Polish).
Printing-HOWTO.gz	How to set up printing in Linux.
Printing-Usage-HOWTO.gz	How to use the print spooling system in Linux.
RPM-HOWTO.gz	How to use the Red Hat Package Manager (RPM) in Linux.

HOWTO file name	Contents
Reading-List-HOWTO.gz	List of books useful for a Linux user.
SCSI-HOWTO.gz	Information about support for SCSI devices in Linux.
SCSI-Programming-HOWTO.gz	Information on programming the SCSI device driver in Linux (useful for programmers who want to add support for a new SCSI device in Linux).
SMB-HOWTO.gz	How to use the Server Message Block (SMB) protocol, also called the NetBIOS or LAN Manager protocol, with Linux.
SRM-HOWTO.gz	How to boot an Alpha-based Linux system using the SRM (System Reference Manual) firmware, which is the firmware normally used to boot DEC UNIX on Alpha.
Serial-HOWTO.gz	How to set up serial communication devices in Linux.
Serial-Programming-HOWTO.gz	How to program the serial port in Linux.
Shadow-Password-HOWTO.gz	How to obtain, install, and configure the password Shadow Suite in Linux (password shadowing provides for more secure passwords than the ones stored in the /etc/passwd file).
Slovenian-HOWTO.gz	How to configure Linux for Slovenian users (this HOWTO is in Slovenian).
Sound-HOWTO.gz	How to enable support for sound hardware in Linux.
Sound-Playing-HOWTO.gz	Lists many sound file formats and the applications that can be used to play sound in Linux.
Spanish-HOWTO.gz	How to configure Linux for Spanish-speaking users (this HOWTO is in Spanish).
TeTeX-HOWTO.gz	How to install and use the teTeX TeX (pronounced *tech* as in technology) and LaTeX document formatting software in Linux.
Thai-HOWTO.gz	How to set up Linux for the Thai language.
Tips-HOWTO.gz	Hints and tips to make Linux more useful and fun.
UMSDOS-HOWTO.gz	How to install and use the UMSDOS file system that enables you to install Linux in an MS-DOS directory (not included in Red Hat Linux).
UPS-HOWTO.gz	How to use an uninterruptable power supply (UPS) with Linux.
UUCP-HOWTO.gz	How to set up and use the UNIX-to-UNIX Copy (UUCP) software in Linux.
User-Group-HOWTO.gz	How to establish and run a Linux User Group.
VAR-HOWTO.gz	Lists Linux Value Added Resellers (VARs).

(continued)

Table 1-6 *(Continued)*	
HOWTO file name	**Contents**
VMS-to-Linux-HOWTO.gz	How to transition from VMS to Linux (VMS is an operating system that runs on VAX systems from Digital Equipment Corporation).
XFree86-HOWTO.gz	How to install and configure the XFree86 (X Window System Version 11 Release 6 — X11R6) in Linux.
XFree86-Video-Timings-HOWTO.gz	How to create a modeline in the *X* configuration file for a specific video card (see Chapter 10 for a description of modeline and format of the X configuration file).

Shutting down Linux

When you are ready to shut down Linux, you should do so in an orderly manner. Even if you are the sole user of a Linux system, several other programs usually are running in the background. Also, operating systems such as Linux try to optimize the way they write data to the disk. Because disk access is relatively slow (compared with the time needed to access memory locations), data usually is held in memory and written to the disk in large chunks. If you simply turn the power off, therefore, you run the risk that some files won't be updated properly.

To shut down Linux, use the shutdown command. You have to log in as root to execute this command. If you are already logged in using another user name, you can become super user by typing the following command (your input is shown in boldface; my comments are in italic):

```
su
Password:    (type the root password and press Enter)
```

After you become super user, type the following command to halt the system:

```
/sbin/shutdown -h now
```

The following message appears:

```
The system is going down for system halt NOW !!
```

After a few moments, you see a many more messages about processes being stopped. Finally, you see the message:

```
System halted
```

Now you can safely turn the power off.

If you want to reboot the system instead of halting it, use the `shutdown`
command with the `-r` option, as follows:

```
/sbin/shutdown -r now
```

Tip Another way to restart the system quickly is to press Ctrl+Alt+Delete—
the same key combination you use to reboot a PC under DOS. Anyone
can use the Ctrl+Alt+Delete sequence to reboot a Linux system from
the console (this means the PC's keyboard, not a terminal connected to
the serial port).

Summary

Linux is a UNIX-like operating system for Intel 80×86, Pentium, and compatible
systems. The CD-ROM that accompanies this book includes the latest version of
the popular Red Hat Linux distribution. This chapter guides you through the
process of installing Linux on your PC. The next chapter shows you how to keep
your Linux kernel current and how to install new packages using the Red Hat
Package Manager (RPM).

The detailed information in this chapter includes:

▶ Information about your PC that you should gather before installing Linux

▶ The overall process of installing Red Hat Linux (including the X Window
System) from the companion CD-ROM

▶ Steps you perform under MS-DOS or Microsoft Windows before installing
Linux

▶ How to boot Linux initially, partition the hard disk, and load the various
software components from the companion CD-ROM using the Red Hat
installation program.

▶ How to start and shut down a Linux system

Part II:

Running Linux

Chapter 6

Secrets of Linux Commands

In This Chapter

▷ Understanding shells: the command interpreters of Linux

▷ Looking at Bash (the Bourne Again Shell)

▷ Understanding the Linux directory structure

▷ Using Linux commands to work with files

▷ Writing shell scripts

▷ Automating common chores with a shell script

▷ Understanding the basics of Perl programming

You saw how to work within a graphical environment in Linux in Chapter 5. Unfortunately, you can't do everything from the graphical environment. It's not impossible to design a graphical interface that enables you to perform most chores, but Linux does not come with such a comprehensive graphical user interface. Therefore, even in the graphical environment of *X*, you often have to work in an xterm window, where you can use Linux commands to accomplish specific tasks.

You can have a variety of command interpreters, or *shells*, in Linux. This chapter introduces you to *Bash*, the Bourne Again Shell, which is the default shell in Linux. You learn some important shell commands and also see how to write simple *shell scripts*, a collection of shell commands stored in a file. When it comes to writing scripts, a language called *Perl* is popular among UNIX system administrators. Because you probably are the system administrator of your Linux system, this chapter also introduces you to Perl.

I defer coverage of Tcl/Tk, another popular scripting language you can use to build applications with a graphical interface, to Chapter 8.

Cross-Reference

The Bash Shell

If you have used MS-DOS, you may be familiar with COMMAND.COM, the DOS command interpreter, the program that displays the infamous C:\> prompt. Linux provides a command interpreter similar to COMMAND.COM in DOS. In UNIX, the command interpreter traditionally is referred to as a *shell*. The original UNIX shell was called the *Bourne shell* and its executable program was named sh. The default Linux shell is Bash and its program name is bash (you find it in the /bin directory). Bash is compatible with the original sh but includes many desirable features of other well-known shells, such as the C shell and the Korn shell. For example, Bash lets you recall commands you entered previously; it even completes partial commands.

The purpose of a shell such as Bash is to display a prompt and execute the command you type at the keyboard. Some commands, such as cd (change directory) and pwd (print working directory), are built into Bash. Many more commands, such as cp (copy) and ls (list directory), are separate programs (meaning a file representing these commands resides in one of the directories on your system). As a user, however, you needn't know or care whether some command is built in or whether it is in the form of a separate executable program.

In addition to the standard Linux commands, Bash can execute any program stored in an executable file. Bash can even execute a shell script (a text file containing one or more commands). As you learn later in the section "Shell Scripts in Bash" of this chapter, you can actually use the shell's built-in commands to write programs (also known as *shell scripts*).

The next few sections give you a sense of the various features of a shell, ranging from the general command syntax to the basics of shell programming. After you go through the overview, you can read more about selected topics in later sections.

> The discussions in this chapter assume you are using Bash as your shell because Bash is the shell you get when you install Linux from the CD-ROM that accompanies this book.

Note

Command syntax

Because a shell interprets what you type, knowing how the shell processes the text you enter is important. All shell commands have the following general format:

```
command option1 option2 ... optionN
```

A single line of command is commonly referred to as a *command line*. On a command line, you enter a command followed by one or more options (or *arguments*) known as *command-line options* (or *command-line arguments*).

One basic rule is you must use a space or a tab to separate the command from the options. You must also separate options with a space or a tab. If you want to use an option that contains embedded spaces, you have to put that

option inside quotation marks. To search for my name in the password file, for example, I would enter the `grep` command as follows:

```
grep "Naba Barkakati" /etc/passwd
```

When `grep` prints the line with my name, it looks like this:

```
naba:Lgb7sOywtVswx:500:100:Naba Barkakati:/home/naba:/bin/bash
```

That line contains a great deal of information, the most interesting information (for the purposes of this discussion) being the field that follows the last colon (:). That field shows the name of the shell I am running.

The number and the format of the command-line options, of course, depend on the actual command. When you learn more about the commands, you see the command-line options that control the behavior of a command are of the form x, in which x is a single character.

If a command is too long to fit on a single line, you can press the backslash (\) key, followed by Enter. Then you can continue entering the command on the next line. Or, you can concatenate several shorter commands on a single line; just use the semicolon (;) as a separator between the commands. For example, when rebuilding the Linux kernel (as explained in Chapter 2) you can complete three sequential tasks by typing the following commands on a single line:

```
make dep; make clean; make boot
```

Command combinations

Linux follows the UNIX philosophy of giving the user a toolbox of many simple commands. You can, though, combine these simple commands to create a more sophisticated command. Suppose you want to determine whether a device file named `sbpcd` resides in your system's `/dev` directory. You look for the file because some documentation tells you that for a Sound Blaster Pro CD-ROM drive, you need that device file. You could use the command `ls /dev` to get a directory listing of the `/dev` directory and see whether anything containing `sbpcd` appears in the listing. Unfortunately, the `/dev` directory has a great many entries and it may be difficult to locate any item that has `sbpcd` in its name. You can combine the `ls` command with `grep` and come up with a command that does exactly what you want:

```
ls /dev | grep sbpcd
```

The shell sends the output of the `ls` command (the directory listing) to the `grep` command, which searches for the string `sbpcd`. That vertical bar (|) is known as a *pipe,* because it acts as a conduit between the two programs; the output of the first command becomes the input of the second one.

Most Linux commands are designed in a way that allows the output of one command to be fed into the input of another. All you have to do is concatenate the commands, placing pipes between them.

I/O redirection

Linux commands that are designed to work together have a common feature: they always read from the *standard input* (usually, the keyboard) and write to the *standard output* (usually, the screen). If you want a command to read from a file, you can redirect the standard input to come from that file. Similarly, to save the output of a command in a file, you can redirect the standard output to a file. These features of the shell are called *input* and *output redirection,* or *I/O redirection.*

Using the following command, for example, you can search through all files in the /usr/include directory for the occurrence of the string typedef and then save that list in a file called typedef.out:

```
grep typedef /usr/include/* > typedef.out
```

This command also illustrates another feature of Bash. When you use an asterisk (*), Bash replaces the asterisk with a list of all the filenames in the specified directory. Thus, /usr/include/* means all the files in the /usr/include directory.

Shell programs

If you are not a programmer, you may feel apprehensive about programming. But shell programming can be as simple as storing a few commands in a file. In fact, you can have a useful shell program that has a single command.

While writing this book, for example, I had to capture screens from the X Window System and use the screen shots in figures. I used the X screen-capture program, xwd, to store the screen images in the X Window Dump (XWD) format. The book's production team, however, wanted the screen shots in PCX format. So I used the Portable Bitmap (PBM) toolkit to convert the XWD images to PCX format. To convert each file, I had to run two programs and delete a temporary file, as follows:

```
xwdtopnm < file.xwd > file.ppm
ppmtopcx < file.ppm > file.pcx
rm file.ppm
```

These commands assume the programs xwdtopnm and ppmtopcx are in one of the directories listed in the PATH environment variable. By the way, xwdtopnm and ppntopcx are two programs in the PBM toolkit.

After converting a few XWD files to PCX format, I got tired of typing the same sequence of commands for each file. At that point, I prepared a file named topcx and saved the following lines in it:

```
xwdtopnm < $1.xwd > $1.ppm
ppmtopcx < $1.ppm > $1.pcx
rm $1.ppm
```

Then I made the file executable, using this command:

```
chmod +x topcx
```

The `chmod` command enables you to change the permission settings of a file. One of those settings determines whether the file is executable. The `+x` option means you want to mark the file as an executable file. You need to do this because Bash will run only executable files.

Finally, I converted the file `figure1.xwd` to `figure1.pcx` by using the following command:

```
topcx figure1
```

The `topcx` file is called a *shell program*. When you run this shell program with the command `topcx figure1`, the shell substitutes `figure1` for each occurrence of $1.

In a nutshell, this is why you might create shell programs: to have your Linux system perform repetitive chores.

Here is another interesting example of a shell program. Suppose you occasionally have to use MS-DOS text files on your Linux system. Although you might expect to use a text file on any system without any problem, there is one catch: DOS uses a carriage return followed by a linefeed to mark the end of each line, whereas Linux (and other UNIX systems) use only a linefeed. As a result, if you use the `vi` editor to open a DOS text file, you see ^M at the end of each line. That ^M stands for *Ctrl-M,* which is the carriage-return character.

On your Linux system, you can easily rid the DOS text file of the extra carriage returns by using the `tr` command with the `-d` option. Essentially, to convert the DOS text file `filename.dos` to a Linux text file named `filename.linux`, you type the following:

```
tr -d '\015' < filename.dos > filename.linux
```

In this command, `'\015'` denotes the ASCII code for the carriage-return character in octal notation.

You can use the `tr` command to translate or delete characters from the input. When you use `tr` with the `-d` option, it deletes all occurrences of a specific character from the input data. Following the `-d` option, you must specify the character to be deleted. Like many UNIX utilities, `tr` reads the standard input and writes its output to standard output. As the sample command shows, you must use input and output redirection to use `tr` to delete all occurrences of a character in a file and to save the output in another file.

If you don't want to remember all this information every time you convert a DOS file to UNIX, store the following in a file named `dos2unix`:

```
tr -d '\015' < $1 > $2
```

Then make the file executable by using this command:

```
chmod +x dos2unix
```

That's it! Now you have a shell program named `dos2unix` that converts a DOS text file to a UNIX text file. If you have the MS-DOS partition mounted as `/dosc`, you can try the `dos2unix` shell program with the following command:

```
dos2unix /dosc/autoexec.bat aexec.bat
```

The command creates a file named `aexec.bat` in the current directory. If you open this file with the `vi` editor, you should not see any ^M characters at the ends of lines.

> **Note**
>
> If you are familiar with MS-DOS, you'll note shell scripts are a lot like MS-DOS batch files. Except for some syntax differences, shell scripts are similar to DOS batch files.

Environment variables

The shell and other Linux commands need information to work properly. If you type a command that isn't one of that shell's built-in commands, the shell must locate an executable file (whose name matches the command you typed). The shell needs to know which directories to search for those files. Similarly, a text editor such as `vi` needs to know the type of terminal (even if the terminal happens to be `xterm`, which essentially emulates a terminal in a window).

One way to provide this kind of information to a program is through command-line options. If you use this approach, however, every time you start a program, you may have to enter many options. UNIX provides an elegant solution through environment variables.

An *environment variable* is nothing more than a name associated with a string. On my system, for example, the environment variable named PATH is defined as follows:

```
PATH=/usr/local/bin:/bin:/usr/bin:.:/usr/X11R6/bin
```

The string to the right of the equal sign is the value of the PATH environment variable. By convention, the PATH environment variable is a sequence of directory names, each name separated from the preceding one by a colon (:). The period in the list of directories also denotes a directory; it represents the current directory.

When the shell has to search for a file, it simply searches the directories listed in the PATH environment variable. The shell searches the directories in PATH in order of their appearance. If two programs have the same name, therefore, the shell executes the one it finds first.

In a fashion similar to the shell's use of the PATH environment variable, an editor such as vi uses the value of the TERM environment variable to determine how to display the file you are editing with vi. To see the current setting of TERM, type the following command at the shell prompt:

```
echo $TERM
```

If you type this command in an xterm window, the output is as follows:

```
xterm
```

To define an environment variable in Bash, use the following syntax:

```
export NAME=Value
```

Here, NAME denotes the name of the environment variable, and Value is the string representing its value. Therefore, you set TERM to the value xterm by using the following command:

```
export TERM=xterm
```

With an environment variable like PATH, you typically want to append a new directory name to the existing definition, rather than define the PATH from scratch. The following example shows how you can accomplish this task:

```
export PATH="$PATH:/usr/games"
```

This command appends the string :/usr/games to the current definition of the PATH environment variable. The net effect is to add /usr/games to the list of directories in PATH.

PATH and TERM are only two of a handful of common environment variables. Table 6-1 lists some of the useful environment variables in Bash.

Table 6-1   Useful Bash environment variables

Environment variable	Contents
DISPLAY	The name of the display on which the X Window System displays output (typically set to :0.0)
HOME	Your home directory
LOGNAME	Your login name
PATH	The list of directories in which the shell looks for programs
PS1	The shell prompt (the default is bash$ for all users except root; for root, the default prompt is bash#).
SHELL	Your shell (SHELL=/bin/bash for Bash)
TERM	The type of terminal

Processes

Every time the shell acts on a command you type, it starts a *process*. The shell itself is a process; so are any scripts or programs the shell executes. An example of such a program is the fvwm2 window manager. You can use the ps command to see a list of processes. When you type ps, for example, Bash shows you the current set of processes. Following is a typical report I get when I enter the ps command in an xterm window:

```
ps
PID TTY STAT  TIME COMMAND
  344   1 S    0:00 /bin/login -- naba
  352   1 S    0:00 -bash
  363   1 S    0:00 sh /usr/X11R6/bin/startx
  364   1 S    0:00 xinit /usr/X11R6/lib/X11/xinit/xinitrc --
  367   1 S    0:00 fvwm2 -cmd FvwmM4 -debug
/etc/X11/AnotherLevel/fvwm2rc.m4
  436   1 S    0:00 /usr/X11R6/lib/X11/fvwm2//FvwmTaskBar 9 4
/tmp/fvwmrca00376
  437   1 S    0:00 /usr/X11R6/lib/X11/fvwm2//FvwmButtons 11 4
/tmp/fvwmrca0037
  440   1 S N  0:00 xload -nolabel -geometry 32x20+0+0 -bg grey60 -
update 5
  441   1 S    0:00 /usr/X11R6/lib/X11/fvwm2//FvwmPager 11 4
/tmp/fvwmrca00376
  442  p0 S    0:00 bash
  501  p0 R    0:00 ps
```

In the default output format, the COMMAND column shows the commands that created the processes. This list shows the bash shell and the ps command as the processes. Other processes include all the programs the shell started when I typed startx to run *X*. In particular, the list include the fvwm2 (window manager) process.

The default ps command does not provide all the processes running on a Linux system. What ps shows are the commands you started, either directly or indirectly (through shell scripts that run automatically when you log in). To see the full complement of processes, use the a option of the ps command together with the x option, as follows:

```
ps ax
```

I won't show the output of the command because quite a few processes are running even when you are the only user on the system. You should expect to see anywhere from 30 to 45 processes in the list.

Secret

If you study the output of the ps command, you find the first column has the heading PID and it shows a number for each process. *PID* stands for *process ID* (identification), which is a sequential number assigned by the Linux kernel. If you look through the output of the ps ax command, you should see the init command is the first process; it has a PID or process number of 1. That's why init is referred to as the *mother of all processes.*

The process ID or process number is useful when you have to stop an errant process forcibly. Look at the output of the ps ax command and note the PID of the offending process. Then use the kill command with that process number. To stop process number 123, for example, type **kill -9 123**.

UNIX systems, including Linux, use signals to notify a process that a specific event has occurred. The kill command enables you to send a signal to a process (identified by a process number). The -9 part of the kill command indicates the signal to be sent; 9 is the number of the SIGKILL signal that causes a process to exit.

Background commands and virtual terminals

When you use MS-DOS, you have no choice but to wait for each command to complete before you enter the next command. (You can type ahead a bit, but the MS-DOS system can hold only a few characters in its internal buffer.) Linux, however, can handle multiple tasks at the same time. The only problem you may have is the terminal or console will be tied up until a command completes.

If you are working in an xterm window and a command takes too long to complete, you can open another xterm window and then continue to enter other commands and do your work. If you are working in text mode, however, and some command seems to take too long, you need another way to access your system.

You can continue working in several ways while your Linux system handles a lengthy task:

■ You can start a lengthy command *in the background,* which means the shell starts the process corresponding to a command and immediately comes back to accept more commands. The shell does not wait for the command to complete; the command runs as a distinct process in the background. To start a process in the background, all you must do is place an ampersand (&) at the end of a command line. When I want to run the topcx shell script to convert a large image file named image1.xwd to PCX format, for example, I run the script in the background by using the following command:

```
topcx image1 &
```

■ If a command (that you did not run in the background) seems to be taking a long time, press Ctrl-Z to stop it; then type **bg** to put that process in the background.

■ Use the *virtual-terminal* feature of Linux. Even though your Linux system has only one physical terminal (the combination of monitor and keyboard is called the *terminal*), it gives you the appearance of having multiple terminals. The initial text screen is the first virtual terminal. Press Alt-F2 to get to the second virtual terminal, Alt-F3 for the third

virtual terminal, and so on. From the X Window System, you have to press Ctrl-Alt-F1 to get to the first virtual terminal, Ctrl-Alt-F2 for the second one, and so on.

Tip To get back to the X display, press Ctrl-Alt-F7. You can use one of the virtual terminals to log in and kill processes that may be causing your X display screen to become unresponsive (if the mouse stops responding, for example).

Command completion in Bash

Many commands take a filename as an argument. When you want to browse through a file named `/etc/X11/XF86Config`, for example, you type the following:

```
more /etc/X11/XF86Config
```

That entry causes the `more` command to display the file `/etc/X11/XF86Config` one screen at a time. For the commands that take a filename as an argument, Bash includes a feature that enables you to type short filenames. All you must type is the bare minimum — just the first few characters — to identify the file uniquely in its directory.

To see an example, type `more /etc/X11/XF`, but don't press Enter yet; press Tab instead. Bash automatically completes the filename, so the command becomes `more /etc/X11/XF86Config`. Now press Enter to run the command.

Tip Whenever you type a filename, press Tab after the first few characters of the filename. Bash probably can complete the filename, so you needn't type the entire name. If you have not entered enough characters to identify the file uniquely, Bash beeps. Just type a few more characters and press Tab again.

Wildcards

Another way to avoid typing too many filenames is to use *wildcards*, special characters, such as the asterisk (*) and question mark (?), that match zero or more characters in a string. If you are familiar with MS-DOS, you may have used commands such as COPY *.* A: to copy all files from the current directory to the A drive. Bash accepts similar wildcards in filenames. In fact, Bash provides many more wildcard options than MS-DOS does.

Bash supports three types of wildcards:

- The asterisk (*) character matches zero or more characters in a filename. Therefore, * denotes all files in a directory.
- The question mark (?) matches any single character.
- A set of characters in brackets match any single character from that set. The string `[xX]*`, for example, matches any filename that starts with x or X.

Wildcards are handy when you want to perform a task on a group of files. To copy all the files from a directory named /mnt to the current directory, for example, type the following:

```
cp /mnt/* .
```

Bash replaces the wildcard character * with the names of all the files in the /mnt directory. The period at the end of the command stands for the current directory.

You can use the asterisk with other parts of a filename to select a more specific group of files. Suppose you want to use the grep command to search for the string typedef struct in all files in the /usr/include directory that meet the following criteria:

■ The filename starts with s.

■ The filename ends with .h.

The wildcard specification s*.h denotes all filenames that meet these criteria. Thus, you can perform the search with the following command:

```
grep "typedef struct" /usr/include/s*.h
```

The string contains a space you want the grep command to find, so you have to enclose that string in quotation marks. This method ensures that Bash does not try to interpret each word in the string as a separate command-line argument.

Although the asterisk (*) matches any number of characters, the question mark (?) matches a single character. Suppose you have four files — image1.pcx, image2.pcx, image3.pcx, and image4.pcx — in the current directory. To copy these files to the /mnt directory, use the following command:

```
cp image?.pcx /mnt
```

Bash replaces the single question mark with any single character and copies the four files to /mnt.

The third wildcard format — [...] — matches a single character from a specific set. Typically, you combine this format with other wildcards to narrow the matching filenames to a smaller set. To see a list of all filenames in the /etc/X11/xdm directory that start with x or X, type the following command:

```
ls /etc/X11/xdm/[xX]*
```

When expanding the [...] wildcard format, if Bash does not find any filenames matching the format, it leaves the wildcard specification intact.

Command history

To make it easy for you to repeat long commands, Bash stores up to 500 old commands. Essentially, Bash maintains a *command history* (a list of old commands). To see the command history, type `history`. Bash displays a numbered list of the old commands, including those you entered during previous logins. That list may resemble the following:

```
1   cd
2   ls -a
3   more /etc/X11/XF86Config
4   history
```

If the command list turns out to be too long, you may choose to see only the last few commands. To see the last ten commands only, type this command:

```
history 10
```

To repeat a command from the list the `history` command shows, all you must do is type an exclamation point (!), followed by that command's number. To repeat command number 3, type `!3`.

You also can repeat an old command without knowing its command number. Suppose you typed `more /usr/lib/X11/xdm/xdm-config` a while ago and now you want to look at that file again. To repeat the previous `more` command, simply type the following:

```
!more
```

Often, you want to repeat the last command you typed, perhaps with a slight change. You may, for example, have displayed the contents of the directory by using the `ls -l` command. To repeat this command, type two exclamation points, as follows:

```
!!
```

Sometimes, you want to repeat the previous command, but add extra arguments to it. Suppose `ls -l` shows too many files. You can simply repeat that command but pipe the output through the `more` command, as follows:

```
!! | more
```

Bash replaces the two exclamation points with the previous command and then appends `| more` to that command.

Tip
An easy way to recall previous commands is to press the up-arrow key, which causes Bash to go backward in the list of commands. To move forward in the command history, press the down-arrow key.

Command editing

After you recall a command, you needn't settle for the command as is; you can edit the command. Bash supports a wide variety of command-line editing commands. These commands are similar to those used by the `emacs` and `vi` editors.

Suppose you want to look at the file /etc/X11/XF86Config, but you typed the following:

```
more /etc/X11/XF86config
```

After you press Enter and see an error message saying no such file exsts, you realize the c in config should have been uppercase. Instead of typing the entire line, you can type the following editing command to fix the problem:

```
^con^Con
```

Bash interprets this command to mean it should replace the string con with Con in the previous command.

By default, Bash enables you to edit the command line using a small subset of commands supported by the emacs editor.

I am already familiar with emacs, so I use the emacs commands to edit commands. To bring back a previous command line, for example, I press Ctrl-P. Then I commonly use the following keystrokes to edit the command line:

>   Ctrl-B to go backward a character

>   Ctrl-F to go forward a character

>   Ctrl-D to delete the character on which the text cursor rests

To insert text, I type the text. Although emacs has a huge selection of editing commands, you can edit Bash command lines adequately with the preceding small set.

Aliases

While configuring the Linux kernel and creating a new Linux boot floppy, I changed the directory to /usr/src/linux/arch/i386/boot quite a few times. After typing that directory name twice, I immediately set up a shortcut, using Bash's alias feature:

```
alias goboot='cd /usr/src/linux/arch/i386/boot'
```

I intentionally did not use any underscore characters or uppercase letters in goboot, because I wanted the alias to be quick and easy to type (and it had to mean something to me only). After I defined the alias, I could go to that directory by typing the following at the Bash prompt:

```
goboot
```

As you can see, an *alias* is simply an alternative (and, usually, shorter) name for a lengthy command. Bash replaces the alias with its definition and performs the equivalent command.

Tip If you type the same long command often, you should define an alias for that command. To make sure the alias is available whenever you log in, place the definition in the .bash_profile file in your home directory.

Many users use the alias feature to give more familiar names to common commands. If you are a DOS user and you are used to the `dir` command to get a directory listing, you can simply define `dir` as an alias for `ls` (the Linux command that displays the directory listing), as follows:

```
alias dir=ls
```

Now you can type `dir` whenever you want to see a directory listing.

Another good use of an alias is to redefine a dangerous command, such as `rm`, to make it safer. By default, the `rm` command deletes one or more specified files. If you type `rm *` by mistake, `rm` deletes all files in your current directory. I learned this the hard way one day when I wanted to delete all files that ended with `.xwd`. (These files were old screen images I no longer needed.) I intended to type `rm *.xwd` but, somehow, I ended up typing `rm * .xwd`. I got the following message:

```
rm: .xwd: No such file or directory
```

At first I was puzzled by the message from `rm`, so I typed **ls** to see the directory's contents again. When the listing showed nothing, I realized I had an extra space between the * and `.xwd`. All the files in that directory, of course, were gone forever.

The `rm` command provides the `-i` option, which asks for confirmation before deleting a file. To make this option a default, add the following `alias` definition to the `.bash_profile` file in your home directory:

```
alias rm='rm -i'
```

That's it! From now on, when you use `rm` to delete a file, the command first asks for confirmation, as follows:

```
rm .bash_profile
rm: remove `.bash_profile'? n
```

Press **y** to delete the file; otherwise, press **n**.

Linux Commands

The shell enables you to run any Linux command, but you need to know the commands before you can run them. Because Linux is a UNIX clone, all Linux commands essentially are UNIX commands. Some of the most important commands are for moving around the Linux file system. This section summarizes these important commands. You can try these commands at the shell prompt in an `xterm` window.

Consult Appendix B for a Linux command reference to learn about many more Linux commands.

Cross-Reference

Linux directory layout

Like any other operating system, Linux organizes information in files. The
files are, in turn, contained in directories. A directory can have
subdirectories, giving rise to a hierarchical structure. Unlike MS-DOS, in
which you see individual drives, the Linux file system starts with a root
directory; a single slash (/) denotes the root directory. Figure 6-1 shows the
basic layout of the Linux file system, including the root directory and several
important subdirectories.
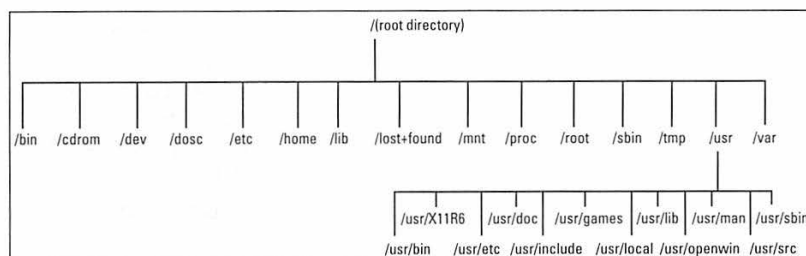
Figure 6-1: The Linux directory structure

Many directories have specific purposes. If you know the purpose of specific
directories, finding your way around the Linux directories is easier. Another
benefit of knowing the typical use of directories is you can guess where to
look for specific types of files when you face a new situation. Table 6-2 briefly
describes the directories in a Linux system:

Table 6-2   Linux system directories

Directory	Description
/	The root directory that forms the base of the file system. All files and directories are logically contained in the root directory, regardless of their physical locations.
/bin	Contains the executable programs that are part of the Linux operating system. Many Linux commands, such as `cat`, `cp`, `ls`, `more`, and `tar`, are located in `/bin`.
/boot	Contains the Linux kernel and other files needed by the LILO boot manager.
/mnt/cdrom	Directory where the CD-ROM drive typically is mounted.
/dev	Contains all device files. Linux treats each device as being a special file and all such files are located in the device directory `/dev`.
/dosc	Directory where the MS-DOS partition is typically mounted (provided your system's hard disk has an MS-DOS partition).

(continued)

Table 6-2 *(Continued)*

Directory	Description
/etc	Contains most system configuration files and the initialization scripts (in the /etc/rc.d subdirectory).
/home	Conventional location of the home directories of all users. User naba's home directory, for example, is /home/naba.
/lib	Contains library files for C and other programming languages.
/lost+found	Directory for lost files. Every disk partition has a lost+found directory.
/mnt	An empty directory, typically used to mount devices such as floppy disks and disk partitions temporarily. Also contains the /mnt/floppy directory for mounting floppy disks and the /mnt/cdrom directory for mounting the CD-ROM drive (of course, you can also mount the CD-ROM drive on another directory).
/proc	A special directory that contains information about various aspects of the Linux system.
/root	The home directory for the root user.
/sbin	Contains executable files representing commands typically used for system-administration tasks. Commands such as mount, halt, umount, and shutdown reside in the /sbin directory.
/tmp	Temporary directory any user can use as a scratch directory (meaning the contents of this directory are considered to be unimportant and are usually deleted every time the system boots).
/usr	Contains the subdirectories for many important programs, such as the X Window System and the online manual.
/var	Contains various system definition files, as well as directories for holding transient information.

The /usr directory also contains a host of useful subdirectories. Table 6-3 lists a few of the important subdirectories in /usr.

Table 6-3 Important /usr subdirectories

Subdirectory	Description
/usr/X11R6	Contains the XFree86 (X Window System) software.
/usr/bin	Contains executable files for many more Linux commands, including utility programs commonly available in Linux, but not part of the core Linux operating system.
/usr/doc	Contains the documentation files for the Linux operating system, as well as many utility programs (such as the Bash shell, mtools, the xfm File Manager, and the xv image viewer).

Subdirectory	Description
/usr/games	Contains the Linux game collection, which includes games such as doom, dungeon, hangman, and snake.
/usr/include	Contains the header files (files with names ending in .h) for the C and C++ programming languages. Also includes the X11 header files in the /usr/include/X11 directory.
/usr/lib	Contains the libraries for C and C++ programming languages; also contains the libraries for X and Tcl.
/usr/local	Contains local files. The /usr/local/bin directory, for example, is supposed to be the location of any executable program developed on your system.
/usr/man	Contains the online manual (which you can read by using the man command).
/usr/sbin	Contains many administrative commands, such as commands for electronic mail and networking.
/usr/src	Contains the source code for the Linux kernel (the core operating system).

Directory navigation

In Linux, when you log in as root, your home directory is /root. For other users, the home directory is usually in the /home directory. My home directory (when I log in as naba), for example, is /home/naba.

By default, only you have permission to save files in your home directory, but you can create subdirectories in your home directory to organize your files further.

A Linux shell such as Bash supports the concept of a *current directory*, which is the directory on which all file and directory commands operate. After you log in, for example, your current directory is the home directory. To see the current directory, use the pwd command.

To change the current directory, use the cd command. To change the current directory to /usr/doc, type the following:

```
cd /usr/doc
```

Then, to change the directory to the bash-1.14.7 subdirectory in /usr/doc, type this command:

```
cd bash-1.14.7
```

Now, if you use the pwd command, that command shows /usr/doc/bash-1.14.7 as the current directory. Therefore, you can refer to a directory's name in two ways:

- An absolute pathname (such as /usr/doc) that specifies the exact directory in the directory tree

- A relative directory name (such as bash-1.14.7, which means the bash-1.14.7 subdirectory of the current directory)

If you type cd bash-1.14.7 in /usr/doc, the current directory changes to /usr/doc/bash-1.14.7, but the same command in /home/naba tries to change the current directory to /home/naba/bash-1.14.7.

Tip

Use the cd command without any arguments to change the current directory to your home directory. Actually, the lone cd command changes the current directory to the directory listed in the HOME environment variable, but that environment variable contains your home directory by default.

Notice the tilde character (~) also refers to the directory specified by the HOME environment variable. Thus, the command cd ~ changes the current directory to whatever directory the HOME environment variable specifies.

Wizard

You can use a shortcut to refer to any user's home directory. Prefix a user's login name with a tilde (~) to refer to that user's home directory. Therefore, ~naba refers to the home directory of the user naba and ~root is the home directory of the root user. If your system has a user with the login name emily, you would type cd ~emily to change to Emily's home directory.

The directory names . and .. have special meanings. A single period (.) indicates the current directory, whereas two periods (..) indicate the parent directory. If the current directory is /usr/doc, for example, you can change the current directory to /usr by using this command:

```
cd ..
```

Directory listings and permissions

As you move around the Linux directories, you want to know the contents of a directory. You can get a directory listing by using the ls command. By default, the ls command, without any options, displays the contents of the current directory in a compact four-column format. If you log in as root and type ls, the ls command lists the contents of the /root directory in the following manner:

```
Mail        Xrootenv.0              .
```

From this listing, you cannot really tell whether an entry is a file or a directory. To see the complete details of a directory's contents, use the -l option with ls, as follows:

```
ls -l
```

For the /root directory, a typical output from ls -l is the following:

```
total 2
drwx------  2 root     root          1024 Jan 26 00:32 Mail
-rw-r--r--  1 root     root           345 Apr  5 01:07 Xrootenv.0
```

This listing shows considerable information about each directory entry, which can be a file or another directory. Looking at a line from the right column to the left, you see the rightmost column shows the name of the directory entry. The date and time before the name show the date and time of the last modification to that file. Before the date and time is the size of the file, in bytes.

The file's group and owner appear to the left of the column that shows the file's size. The next number indicates the number of links to the file. Finally, the leftmost column shows the file's *permission settings*, which determine who can read, write, or execute the file.

The first letter of the leftmost column has a special meaning, as the following list shows:

- If the first letter is l, the file is a symbolic link to another file.

- If the first letter is d, the file is a directory.

- If the first letter is a dash (-), the file is a normal file.

After that first letter, the leftmost column shows a sequence of nine characters, which appears as rwxrwxrwx when each letter is present. Each letter indicates a specific permission and a hyphen in place of a letter indicates no permission for a specific operation on the file. Think of these nine letters as three groups of three letters (rwx), interpreted as follows:

- The leftmost group of rwx controls the read, write, and execute permission of the file's owner. In other words, if you see rwx in this position, the file's owner can read (r), write (w), and execute (x) the file. A hyphen in the place of a letter indicates no permission. Thus, the string rw- means the owner has read and write permission but no execute permission. Typically, executable programs (including shell programs) have execute permission.

- The middle three rwx letters control the read, write, and execute permission of any user belonging to that file's group.

- The rightmost group of rwx letters controls the read, write, and execute permission of all other users (collectively referred to as the *world*).

Thus, a file with the permission setting rwx is accessible only to the file's owner, whereas the permission setting rwxr–r makes the file readable by everyone.

Use the chmod command to change the permission setting of a file. A shell program (or shell script), for example, typically is a text file you prepare with a text editor. After saving the file, you have to make the file executable before you can run the script. To add execute permission to a script file named convert_images, for example, type the following:

```
chmod +x convert_images
```

This command adds the execute permission for owner, group, and world.

An interesting feature of the `ls` command is it does not list any file whose name begins with a period. To see these files, you must use the `ls` command with the `-a` option, as follows:

```
ls -a
```

Tip The example of the `ls` command shows an interesting feature of Linux commands: Most Linux commands take single-character options, each with a minus sign (you might think of this sign as being a hyphen) as prefix. When you want to use several options, type a hyphen and concatenate the option letters one after another. Therefore, `ls -al` is equivalent to `ls -a -l`.

File manipulation

You may often copy files from one directory to another. Use the `cp` command to perform this task. To copy the file `/usr/X11R6/lib/X11/fvwm2/system.fvwm2rc` to the `.fvwm2rc` file in another directory (such as your home directory), type the following:

```
cp /usr/X11R6/lib/X11/fvwm2/system.fvwm2rc  .fvwm2rc
```

If you want to copy a file to the current directory and retain the same name, use a period (.) as the second argument of the `cp` command. Thus, the following command copies the `XF86Config` file from the `/etc/X11` directory to the current directory:

```
cp /etc/X11/XF86Config
```

The `cp` command makes a new copy of a file and leaves the original intact.

Another Linux command, `mv`, moves a file to a new location. The original copy is gone and a new copy appears at the specified destination. One use of `mv` is to rename a file. If you want to change the name of the `today.list` to `old.list`, use the `mv` command, as follows:

```
mv today.list old.list
```

On the other hand, if you want to move the `today.list` file to the subdirectory named `saved`, use this command:

```
mv today.list saved
```

Wizard An interesting feature of `mv` is you can use it to move entire directories — with all their subdirectories and files — to a new location. If you have a directory named `data` that contains many files and subdirectories, you can move that entire directory structure to `old_data` by using this command:

```
mv data old_data
```

Another common file operation is deleting a file. Use the `rm` command to delete a file. To delete a file named `old.list`, for example, type the following command:

```
rm old.list
```

Be careful with the `rm` command, in particular when you are logged in as `root`. Inadvertently deleting important files with `rm` is easy. (See the "Aliases" section earlier in this chapter to learn how you can avoid accidentally clobbering files with the `rm` command.)

In addition to copying, renaming, and deleting files, you may want to view a file's contents. Use the `more` command to look at a text file one page at a time. To view the file `/etc/X11/XF86Config`, for example, use this command:

```
more /etc/X11/XF86Config
```

The `more` command pauses after each page and you must press the spacebar to move to the next page. Press Enter to move forward one line at a time in the file.

Another useful Linux command for file viewing is `less`. (The name is a play on words, because `less` does more than `more`.) Whereas `more` enables you to move forward one page at a time, the `less` command enables you to move forward and backward through a file.

Directory manipulation

To organize files in your home directory, you have to create new directories. Use the `mkdir` command to create a directory. To create a directory named `images` in the current directory, type the following:

```
mkdir images
```

After you create the directory, you can use the `cd images` command to change to that directory.

When you no longer need a directory, use the `rmdir` command to delete it. You can delete a directory only when the directory is empty.

File finder

The `find` command locates files (and directories) that meet specified search criteria. The `find` command is one of the most useful Linux commands. The Linux version of the `find` command comes from GNU and it has more extensive options than the standard UNIX version. I show the syntax for the standard UNIX `find` command, however, because that syntax works in Linux and you can use the same format on other UNIX systems.

I must admit, when I began using UNIX many years ago (I started with Berkeley UNIX in the early 1980s), I was confounded by the `find` command. I must have stayed with one basic syntax of `find` for a long time before graduating to more complex forms. The basic syntax I learned first was for finding a file anywhere in the file system.

Suppose you want to find any file or directory with a name that starts with `fvwm`. You can use `find` to perform this search, as follows:

```
find / -name "fvwm*" -print
```

This command tells `find` to start looking at the root directory (/), to look for filenames that match `fvwm*`, and to display the full pathname of any matching file.

The `-print` option and the double quotation marks around the search string are not necessary in Linux as long as no files exist in the current directory that start with `fvwm`. You could have typed the preceding command as follows:

```
find / -name fvwm*
```

No harm exists in using the longer form, however, because that's how you must use `find` in other UNIX systems. I continue to show you `find` with the standard UNIX syntax.

You can use variations of this simple form of `find` to locate a file in any directory (as well as subdirectories contained in the directory). If you forget where in your home directory you stored all files named `y2k*` (names that start with the string `y2k`), you can search for the files by using the following command:

```
find ~ -name "y2k*" -print
```

When you become comfortable with this syntax of `find`, you can start using other options of `find`. To find only specific types of files (such as directories), use the `-type` option. The following command displays all top-level directory names in your Linux system:

```
find / -type d -maxdepth 1 -print
```

You probably do not have to use the complex forms of `find` in a typical Linux system, but you can always look up the rest of the `find` options by using this command:

```
man find
```

Shell Scripts in Bash

The fundamental philosophy of UNIX is to give the user many small and specialized commands, along with the necessary plumbing to connect these commands. By *plumbing*, I mean the way one command's output is used as a second command's input. Bash, the default shell in Linux, provides this plumbing in the form of I/O redirection and pipes. Bash also includes features such as the `if` statement, which runs commands only when a specific condition is True and the `for` statement, which repeats commands a specified number of times. You can use these features of Bash to write programs called *shell scripts*.

Shell scripts are popular among system administrators. If you are a system administrator, you can build up a collection of custom shell scripts that help you automate tasks you perform often. If the disk seems to be getting full, for example, you may want to find all files that exceed some size (say, 1MB) and that have not been accessed in the past 30 days. In addition, you may want to

send an e-mail message to all users who have large files, requesting they try to archive and clean up those files. You can perform all these tasks with a shell script. You might start with the following `find` command to identify the large files:

```
find / -type f -atime +30 -size +1000k -exec ls -l {} \; > /tmp/largefiles
```

This command creates a file named `/tmp/largefiles`, which contains detailed information about the old files that are taking up too much space. After you get a list of the files, you can use a few other Linux commands — such as `sort`, `cut`, and `sed` — to prepare and send mail messages to users who have large files they should try to clean up.

The following sections provide an overview of Bash programming.

As you try simple shell programs, don't name your sample programs `test`. Linux includes an important program named `test` (`/usr/bin/test`). This program is used in shell scripts to test for various conditions, such as whether a file exists and whether a file is readable. If you create a sample program named `test`, some scripts may end up calling your program instead of the system's `test` program.

Caution

A simple shell script

Earlier in this chapter, the section "Shell Programs" discussed how you can simply place often-used commands in a file and use the `chmod` command to make the file executable. Voila — you have a shell script. Just as most Linux commands accept command-line options, a Bash script accepts command-line options. Inside the script, you can refer to the options as $1, $2, and so on. The special name $0 refers to the name of the script itself.

Consider the following Bash script:

```
#!/bin/sh
echo "This script's name is: $0"
echo Argument 1: $1
echo Argument 2: $2
```

The first line causes Linux to run the `/bin/sh` program, which subsequently processes the rest of the lines in the script. The name `/bin/sh` traditionally refers to the Bourne shell — the first UNIX shell. In Linux, `/bin/sh` is a symbolic link to `/bin/bash`, which is the executable program for Bash. Therefore, in Linux, Bourne shell scripts are run by Bash (which happens to be compatible with the Bourne shell).

If you save this simple script in a file named `simple` and you make that file executable with the command `chmod +x simple`, you can run the script as follows:

```
simple
This script's name is: ./simple
Argument 1:
Argument 2:
```

Learning more about shell programming

Although these sections provide an overview of shell programming, there is much more to learn about shell programming than this single chapter can cover. *UNIX Power Tools*, coauthored by several UNIX gurus, is a good source of information on all aspects of UNIX commands in general and shell programming in particular.

The script file's name appears relative to the current directory, which is represented by a period. Because you ran the script without any arguments, the script does not display any arguments.

Now try running the script with a few arguments, as follows:

```
simple "This is one argument" second-argument third
This script's name is: ./simple
Argument 1: This is one argument
Argument 2: second-argument
```

As this example shows, the shell treats the entire string within double quotation marks as a single argument. Otherwise, the shell uses spaces as separators between arguments on the command line.

Bash programming overview

Like any programming language, Bash includes the following features:

■ Variables that store values, including special built-in variables for accessing command-line arguments passed to a shell script and other special values

■ The capability to evaluate expressions

■ Control structures that enable you to loop over several shell commands or to execute some commands conditionally

■ The capability to define functions that can be called in many places within a script. Bash also includes many built-in functions that you can use in any script.

The next few sections illustrate some of Bash's programming features through simple examples. Because you are already running Bash, you can try the examples by typing them at the shell prompt in an xterm window.

Variables

Define variables in Bash just as you define environment variables. Thus, you might define a variable this way:

```
count=12   # note no embedded spaces allowed
```

To use a variable's value, prefix the variable's name with a dollar sign ($). $PATH, for example, is the value of the variable PATH (yes, the famous PATH environment variable). To display the value of the variable named count, you would use the following command:

```
echo $count
```

Bash has some special variables for accessing command-line arguments. In a shell script, $0 refers to the name of the shell script. The variables $1, $2, and so on, refer to the command-line arguments. The variable $* stores all the command-line arguments as a single variable and $? contains the exit status of the last command executed by the shell.

You also can prompt the user for input and use the read command to read the input into a variable. Following is an example:

```
echo -n "Enter value: "
read value
echo "You entered: $value"
```

The -n option stops the echo command from automatically adding a new line at the end of the string it displays.

Control structures

In Bash scripts, the control structures — such as if, case, for, and while — depend on the exit status of a command to decide what to do next. When any command executes, it returns an *exit status*, a numeric value that indicates whether the command was successful. By convention, an exit status of zero means the command succeeded. (Yes, you read it right: zero indicates success.) A nonzero exit status indicates something went wrong with the command.

You might use a script that makes a backup copy of a file before opening it by using the vi editor in the following manner:

```
#!/bin/sh
if cp "$1" "#$1"
then
    vi "$1"
else
    echo "Failed to create backup copy"
fi
```

This script illustrates the syntax of the *if-then-else* structure and also shows how the exit status of the cp command is used by the if structure to decide the next action. If cp returns zero, the script invokes vi to edit the file; otherwise, the script displays a message and exits.

Don't forget the final fi that terminates the if structure. Forgetting fi is a common source of errors in Bash scripts.

Bash includes the `test` command to enable you to evaluate any expression and use the expression's value as the exit status of the command. Suppose you want a script that enables you to edit an existing file. Using `test`, you might write such a script as follows:

```
#!/bin/sh
if test -f "$1"
then
    vi "$1"
else
    echo "No such file"
fi
```

A shorter form of the `test` command leaves out `test` and places the command's options in brackets. Using this notation, you would write the preceding script as follows:

```
#!/bin/sh
if [ -f "$1" ]
then
    vi "$1"
else
    echo "No such file"
fi
```

Another common control structure is the `for` loop. The following script adds the numbers one through ten:

```
#!/bin/sh
sum=0
for i in 1 2 3 4 5 6 7 8 9 10
do
    sum=`expr $sum + $i`
done
echo "Sum = $sum"
```

This example also illustrates the use of the `expr` command to evaluate an expression.

Built-in functions in Bash

Bash has more than 50 built-in functions, including common functions such as `cd` and `pwd`, as well as many others that are not used frequently. You can use these built-in functions in any Bash script. This chapter does not have enough space to cover the built-in functions, but you can learn more about them from the online manual for Bash by using the `man bash` command.

Perl as a Scripting Language

Officially, *Perl* stands for *Practical Extraction Report Language*. Perl was created by Larry Wall to extract information from text files and then use that information to prepare reports. Programs written in Perl, the language, are interpreted and executed by perl, the program. This book's companion CD-ROM includes the perl program; it should be installed on your system in the /usr/bin directory.

Perl is available on a wide variety of computer systems because, like Linux, Perl can be distributed freely. Also, Perl is popular among many users and system administrators as a scripting language, which is why this section introduces Perl and shows its strengths. In Chapter 8, you learn about another scripting language (Tcl/Tk) that provides the capability to create graphical user interfaces for the scripts.

As you know by now, the term *script* simply is a synonym for *program*. Unlike programs written in programming languages such as C and C++, Perl programs needn't be compiled; the perl program simply interprets and executes the Perl programs. The term *script* often is used for such interpreted programs written in a shell's programming language or in Perl. (Strictly speaking, perl does not simply interpret a Perl program; it converts the Perl program to an intermediate form before executing the program.)

Note: If you are familiar with shell programming or the C programming language, you can pick up Perl quickly. If you have never programmed, becoming proficient in Perl may take a while. I encourage you to start with a small subset of Perl's features and ignore anything you do not understand. Then you can slowly add Perl features to your repertoire.

Do I have Perl?

Before you proceed with the Perl tutorial, check to see whether you have perl installed on your system. Type the following command:

```
which perl
```

The which command tells you whether it finds a specified program in the directories listed in the PATH environment variable. If perl is installed, you should see the following output:

```
/usr/bin/perl
```

If the which command complains that no such program exists in the current PATH, this message does not necessarily mean you don't have perl installed. You may not have the /usr/bin directory in PATH. Check to ensure /usr/bin is

in PATH: either type echo $PATH or look at the message displayed by the which command (this message includes the directories in PATH). If /usr/bin is not in PATH, use the following command to redefine PATH:

```
PATH=$PATH:/usr/bin; export PATH
```

Now try the which perl command again. If you still get an error, you may not have installed perl. You can always install Perl from the companion CD-ROM with the following steps:

1. Log in as root.

2. Insert the companion CD-ROM is in the CD-ROM drive and mount the CD-ROM using the following command:

   ```
   mount /dev/hdc /mnt/cdrom
   ```

   Replace /dev/hdc with the device name for your CD-ROM drive.

3. Type the following command to change directory to the location where the Red Hat packages are located:

   ```
   cd /mnt/cdrom/RedHat/RPMS
   ```

4. Type the following rpm (Red Hat Package Manager) command to install Perl:

   ```
   rpm -i perl*
   ```

If you have perl installed on your system, type the following command to see its version number:

```
perl -v
```

Following is typical output from that command:

```
This is perl, version 5.004_01

Copyright 1987-1997, Larry Wall

Perl may be copied only under the terms of either the Artistic License
or the GNU General Public License, which may be found in the Perl 4.0
source kit.
```

This output tells you that you have Perl Version 5.004, patch level 01, and Larry Wall, the originator of Perl, holds the copyright. Perl is freely distributed under the GNU General Public License, however.

The companion CD-ROM has Version 5.004_01 of Perl. You can get the latest version of Perl by pointing your Web browser to the Comprehensive Perl Archive Network (CPAN). The following address connects you to the CPAN site nearest to you:

```
http://www.perl.com/CPAN/
```

Your first Perl script

Perl has many features of C and, as you may be aware, most books on C start with an example program that displays `Hello, World!` on your terminal. Because Perl is an interpreted language, you can accomplish this task directly from the command line. If you enter:

```
perl -e 'print "Hello, World!\n";'
```

the system responds:

```
Hello, World!
```

This command uses the `-e` option of the `perl` program to pass the Perl program as a command-line argument to the Perl interpreter. In this case, the following line constitutes the Perl program:

```
print "Hello, World!\n";
```

To convert this line to a script, all you must do is place the line in a file and start the file with a directive to run the `perl` program (as you do in the shell scripts, in which you place a line such as `#!/bin/sh` to run the Bourne shell to process the script).

To try a Perl script, follow these steps:

1. Use a text editor, such as `vi` or `emacs`, to save the following lines in the file named `hello`:

   ```
   #!/usr/bin/perl
   # This is a comment.
   print "Hello, World!\n";
   ```

2. Make the `hello` file executable by using the following command:

   ```
   chmod +x hello
   ```

3. Run the Perl script by typing the following at the shell prompt:

   ```
   hello
   Hello, World!
   ```

That's it! You've just written and tried your first Perl script.

Secret

Notice the first line of a Perl script starts with `#!`, followed by the full pathname of the `perl` program. If the first line of a script starts with `#!`, the shell simply strips off the `#!`, appends the script file's name to the end, and runs the script. Thus, if the script file is named `hello` and the first line is `#!/usr/bin/perl`, the shell executes the following command:

```
/usr/bin/perl hello
```

More on Perl

This chapter devotes a few sections to give you an overview of Perl and show a few simple examples. This discussion doesn't do justice to Perl, though. If you want to use Perl as a tool, consult one of the following books on Perl:

> Larry Wall, Tom Christiansen, and Randal L. Schwartz, *Programming Perl, 2nd Edition* (O'Reilly & Associates, 1996)

> Randal L. Schwartz, *Learning Perl* (O'Reilly & Associates, 1993)

Naba Barkakati, *Discover Perl 5* (IDG Books Worldwide, 1997)

The book by Perl originator Larry Wall and Randal Schwartz is the authoritative guide to Perl (although it may not be the best resource for learning Perl). The later book by Randal Schwartz focuses more on teaching Perl programming. My recent book teaches Perl with step-by-step instructions backed by short, but complete, example programs.

A Perl overview

Most programming languages, including Perl, have some common features:

- *Variables* to store different types of data. You can think of each variable as a placeholder for data — kind of like a mailbox, with a name and room to store data. The content of the variable is its value.

- *Expressions* that combine variables by using *operators*. An expression might add several variables; another might extract a part of a string.

- *Statements* that perform some action, such as assigning a value to a variable or printing a string

- *Flow-control statements* that allow statements to be executed in various orders, depending on the value of some expression. Typically, flow-control statements include `for`, `do-while`, `while`, and `if-then-else` statements.

- *Functions* (also called *subroutines* or *routines*) that enable you to group several statements and give them a name. This feature enables you to execute the same set of statements by invoking the function that represents those statements. Typically, a programming language provides some predefined functions.

The next few sections provide an overview of these major features of Perl and illustrate the features through simple examples.

Basic Perl syntax

Perl is free-form, like C; no constraints exist on exact placement of any keyword. Perl programs often are stored in files with names that end in `.pl`, but there is no restriction on the filenames you can use.

As in C, each Perl statement ends with a semicolon (;). A pound sign (#) marks the start of a comment. The perl program disregards the rest of the line beginning with the pound sign.

Groups of Perl statements are enclosed in braces ({...}). This feature also is similar to C.

Variables

You don't have to declare Perl variables before using them, as you do in C. You can easily recognize a variable in a Perl script because each variable name begins with a special character: an at symbol (@), a dollar sign ($), or a percent sign (%). This special character denotes the variable's type. The three variable types are

- *Scalar variables*, which represent the basic data types: integer, floating-point number, and string. A dollar sign ($) precedes a scalar variable. Following are some examples:

```
$maxlines = 256;
$title = "Linux Secrets, Second Edition";
```

- *Array variables*, which are collections of scalar variables. An array variable has an at symbol (@) as a prefix. Thus, the following are arrays:

```
@pages = (62, 26, 22, 24);
@commands = ("start", "stop", "draw", "exit");
```

- *Associative arrays*, which are collections of key-value pairs in which each key is a string and the value is any scalar variable. A percent-sign (%) prefix indicates an associative array. You can use associative arrays to associate a name with a value. You might store the amount of disk space used by each user in an associative array such as the following:

```
%disk_usage = ("root", 147178, "naba", 28547, "emily", 55, "ivy", 60);
```

Because each variable type has its own special character prefix, you can use the same name for different variable types. Thus, %disk_usage, @disk_usage, and $disk_usage can appear within the same Perl program.

Scalars

Scalar variables are the basic data type in Perl. Each scalar's name starts with a dollar sign ($). Typically, you start using a scalar with an assignment statement that initializes it. You can even use a variable without initializing it. The default value for numbers is zero and the default value of a string is an empty string. If you want to see whether a scalar is defined, use the defined function as follows:

```
print "Name undefined!\n" if !(defined $name);
```

The expression (defined $name) is 1 if $name is defined. You can actually "undefine" a variable by using the undef function. You can undefine $name, for example, as follows:

```
undef $name;
```

Variables are evaluated according to context. Following is a script that initializes and prints a few variables:

```
#!/usr/bin/perl
$title = "Linux SECRETS, Second Edition";
$count1 = 550;
$count2 = 375;

$total = $count1 + $count2;

print "Title: $title -- $total pages\n";
```

When you run this Perl program, it produces the following output:

```
Title: Linux SECRETS, Second Edition -- 925 pages
```

As the Perl statements show, when the two numeric variables are added, their numeric values are used, but when the $total variable is printed, its string representation is displayed.

Another interesting aspect of Perl is it evaluates all variables in a string within double quotation marks ("..."). On the other hand, if you write a string inside single quotation marks ('...'), Perl leaves that string untouched. If you were to write

```
 print 'Title: $title -- $total pages\n';
```

with single quotes instead of double quotes, Perl would display

```
Title: $title -- $total pages\n
```

and not even generate a new line.

A useful Perl variable is $_ (dollar sign followed by the underscore character). This special variable is known as the *default argument*. The Perl interpreter determines the value of $_ depending on the context. When the Perl interpreter reads input from the standard input, $ holds the current input line. When the interpreter is searching, $_ holds the default search pattern.

Arrays

An *array* is a collection of scalars. The array name starts with an at symbol (@). As in C, array subscripts start at zero. You can access the elements of an array with an index. Perl allocates space for arrays dynamically.

Consider the following simple script:

```
#!/usr/bin/perl
@commands = ("start", "stop", "draw" , "exit");

$numcmd = @commands;
print "There are $numcmd commands. The first command is:
$commands[0]\n";
```

When you run the script, it produces the following output:

```
There are 4 commands. The first command is: start
```

As you can see, equating a scalar to the array sets the scalar to the number of elements in the array. The first element of the @commands array is referenced as $commands[0] because the index starts at zero. Thus, the fourth element in commands is $commands[3].

Two special scalars are related to an array. The $[variable is the current base index, which is zero by default. The scalar $#*arrayname* (in which *arrayname* is the name of an array variable) has the last array index as the value. Thus, for the @commands array, $#commands is 3.

You can print an entire array with a simple print statement like this:

```
print "@commands\n";
```

When Perl executes this statement, it displays the following output:

```
start stop draw exit
```

Associative arrays

Associative array variables, which are declared with a percent-sign (%) prefix, are unique features of Perl. Using associative arrays, you can index an array with a string such as a name. A good example of an associative array is the %ENV array that Perl automatically defines for you. In Perl, %ENV is the array of environment variables you can access by using the environment-variable name as an index. The following Perl statement prints the current PATH environment variable:

```
print "PATH = $ENV(PATH)\n";
```

When Perl executes this statement, it prints the current setting of PATH. In contrast to regular arrays, you must use braces to index into an associative array.

Perl has many built-in functions — such as delete, each, keys, and values — that enable you to access and manipulate associative arrays.

Predefined variables in Perl

Perl has several predefined variables that contain useful information you may need in a Perl script. Following are a few important predefined variables:

@ARGV is an array of strings that contains the command-line options to the script. The first option is $ARGV[0], the second one is $ARGV[1], and so on.

%ENV is an associative array that contains the environment variables. You can access this array by using the environment-variable name as a key. Thus, $ENV{HOME} is the home directory and $ENV{PATH} is the current search path the shell uses to locate commands.

$$ is the script's process ID.

$< is the user ID of the user who is running the script.

$? is the status returned by the last system call.

$_ is the default argument for many functions.

$0 is the name of the script.

Operators and expressions

Operators are used to combine and compare Perl variables. Typical mathematical operators are addition (+), subtraction (-), multiplication (*), and division (/). Perl provides nearly the same set of operators C has. When you use operators to combine variables, you end up with *expressions*. Each expression has a value.

Following are some typical Perl expressions:

```
error < 0
$count == 10
$count + $i
$users[$i]
```

These expressions are examples of the *comparison operator* (the first two lines), the *arithmetic operator*, and the *array-index operator*.

Caution

In Perl, don't use the == operator to find out whether two strings match; the == operator works only with numbers. To test the equality of strings, Perl includes the FORTRAN-style eq operator. Use eq to see whether two strings are identical, as follows:

```
if ($input eq "stop") { exit; }
```

Other FORTRAN-style string comparison operators include ne (inequality), lt (less than), gt (greater than), le (less than or equal), and ge (greater than or equal). You also can use the cmp operator to compare two strings. The return value is -1, 0, or 1, depending on whether the first string is less than, equal to, or greater than the second one.

Perl also provides the following unique operators. C lacks an exponentiation operator, which FORTRAN includes; Perl uses ** as the exponentiation operator. Thus, you can enter the following:

```
$x = 2;
$y = 3;
$z = $x**$y;  # z should be 8 (2 raised to the power 3)
$y **= 2; # y is now 9 (3 raised to the power 2)
```

You can initialize an array to null by using ( ) — the *null-list operator* — as follows:

```
@commands = ();
```

The dot operator (.) enables you to concatenate two strings, as follows:

```
$part1 = "Hello, ";
$part2 = "World!";
$message = $part1.$part2;  # Now $message = "Hello, World!"
```

A curious but useful operator is the *repetition operator*, denoted by x=. You can use the x= operator to repeat a string a specified number of times. Suppose you want to initialize a string to 65 asterisks (*). The following example shows how you can initialize the string with the x= operator:

```
$marker = "*";
$marker x= 65;  # Now $marker is a string of 65 asterisks
```

Another powerful operator in Perl is the *range operator*, represented by two periods (..). You can initialize an array easily by using the range operator. Following are some examples:

```
@numerals = (0..9); # @numerals is the list 0, 1, 2, 3, 4, 5, 6, 7, 8
, 9
@alphabet = ('A'..'Z'); # @alphabet is the list of capital letters A
through Z
```

Regular expressions

If you have used UNIX for a while, you probably know about the grep command, which enables you to search files for a pattern of strings. Following is a typical use of grep to locate all files with any occurrences of the string blaster or Blaster on any line of all files with names that end in .c:

```
cd /usr/src/linux/drivers/cdrom
grep "[bB]laster" *.c
```

These commands produce this output on my system:

```
cdu31a.c:   ( 0x230,     0 ),    /* SoundBlaster 16 card */
sbpcd.c: *             Works with SoundBlaster compatible cards and with
"no-sound"
sbpcd.c:        0x230, 1, /* Soundblaster Pro and 16 (default) */
sbpcd.c:        0x250, 1, /* OmniCD default, Soundblaster Pro and 16 */
sbpcd.c:        0x270, 1, /* Soundblaster 16 */
```

```
sbpcd.c:         0x290, 1, /* Soundblaster 16 */
sbpcd.c:static const char *str_sb = "SoundBlaster";
sbpcd.c:static const char *str_sb_l = "soundblaster";
sbpcd.c: *              sbpcd=0x230,SoundBlaster
sbpcd.c:              msg(DBG_INF," LILO boot: ...
sbpcd=0x230,SoundBlaster\n");
sjcd.c: *  the SoundBlaster/Panasonic style CDROM interface. But today, the
```

As you can see, grep found all occurrences of blaster and Blaster in the files whose names end in .c.

The grep command's "[bB]laster" argument is known as a *regular expression*, a pattern that matches a set of strings. You construct a regular expression with a small set of operators and rules similar to the ones for writing arithmetic expressions. A list of characters inside brackets ([...]), for example, matches any single character in the list. Thus, the regular expression "[bB]laster" is a set of two strings, as follows:

blaster   Blaster

Perl supports regular expressions just as the grep command does. Many other UNIX programs, such as the vi editor and sed (stream editor), also support regular expressions. The purpose of a regular expression is to search for a pattern of strings in a file. This is why editors support regular expressions.

Perl enables you to construct complex regular expressions, but the rules are fairly simple. Essentially, the regular expression is a sequence of characters in which some characters have special meaning. Table 6-4 lists the basic rules of interpreting the characters.

Table 6-4   Rules for interpreting regular expression characters

Expression	Meaning
.	Matches any single character except a new line
$x*$	Matches zero or more occurrences of the character x
$x+$	Matches one or more occurrences of the character x
$x?$	Matches zero or one occurrence of the character x
[...]	Matches any of the characters inside the brackets
$x\{n\}$	Matches exactly n occurrences of the character x
$x\{n,\}$	Matches n or more occurrences of the character x
$x\{,m\}$	Matches zero or, at most, m occurrences of the character x

Expression	*Meaning*
x{n,m}	Matches at least n occurrences, but no more than m occurrences of the character x
$	Matches the end of a line
\0	Matches a null character
\b	Matches a backspace
\B	Matches any character that's not at the beginning or the end of a word
\b	Matches the beginning or end of a word — (when not inside brackets)
\cX	Matches Ctrl-X (where X is any alphabetic character)
\d	Matches a single digit
\D	Matches a nondigit character
\f	Matches a form feed
\n	Matches a new-line (line-feed) character
\ooo	Matches the octal value specified by the digits ooo (where each o is a digit between 0 and 7)
\r	Matches a carriage return
\S	Matches a nonwhite-space character
\s	Matches a white-space character (space, tab, or new line)
\t	Matches a tab
\W	Matches a nonalphanumeric character
\w	Matches an alphanumeric character
\xhh	Matches the hexadecimal value specified by the digits hh (where each h is a digit between 0 and f)
^	Matches the beginning of a line

If you want to match one of the characters $, |, *, ^, [,], \, and /, you have to place a backslash before them. Thus, you would type these characters as \$, \|, *, \^, \[, \], \\, and \/. Regular expressions often look confusing because of the preponderance of strange character sequences and the generous sprinkling of backslashes. As with anything else, though, you can start slowly and use only a few of the features in the beginning.

So far, this section has summarized the syntax of regular expressions, but you haven't yet seen how to use regular expressions in Perl. Typically, you place a regular expression within a pair of slashes and use the match (=~) or not-match (!~) operators to test a string. You can write a Perl script that performs the same search as the one done with grep earlier in this section. Follow these steps to complete this exercise:

1. Use an editor to type and save the following script in a file named `lookup`:

```
#!/usr/bin/perl

while (<STDIN>)
{
    if ( $_ =~ /[bB]laster/ ) { print $_; }
}
```

2. Make the `lookup` file executable by using the following command:

```
chmod +x lookup
```

3. Try the script, using the following command:

```
cat /usr/src/linux/drivers/cdrom/sbpcd.c | lookup
```

My system responds with this:

```
*              Works with SoundBlaster compatible cards and with
"no-sound"
         0x230, 1, /* Soundblaster Pro and 16 (default) */
         0x250, 1, /* OmniCD default, Soundblaster Pro and 16 */
         0x270, 1, /* Soundblaster 16 */
         0x290, 1, /* Soundblaster 16 */
static const char *str_sb = "SoundBlaster";
static const char *str_sb_l = "soundblaster";
*                  sbpcd=0x230,SoundBlaster
             msg(DBG_INF,"   LILO boot: ...
sbpcd=0x230,SoundBlaster\n");
```

The `cat` command feeds the contents of a specific file (which, as you know from the `grep` example, contains some lines with the regular expression) to the lookup script. The script simply applies Perl's regular expression-match operator (=~) and prints any matching line.

The `$_` variable in the script needs some explanation. The `<STDIN>` expression gets a line from the standard input and, by default, stores that line in the `$_` variable. Inside the `while` loop, the regular expression is matched against the `$_` string. All the `lookup` script's work is done with this single Perl statement:

```
if ( $_ =~ /[bB]laster/ ) { print $_; }
```

This example illustrates how you might use a regular expression to search for occurrences of strings in a file.

After you use regular expressions for a while, you can better appreciate their power. The trick is to determine exactly what regular expression performs the task you want. Following is a search that looks for all lines beginning with exactly seven spaces and ending with a right parenthesis:

```
while (<STDIN>)
{
    if ( $_ =~ /\)\n/ && $_ =~ /^ {7}\S/ )  { print $_; }
}
```

Flow-control statements

So far, you have seen Perl statements that are meant to execute in a serial fashion — one after another. Perl also includes statements that enable you to control the flow of execution of the statements. You have already seen the `if` statement and a `while` loop. Perl includes a complete set of flow-control statements just like those in C, but with a few extra features.

In Perl, all conditional statements take the following form:

```
conditional-statement
{ Perl code to execute if conditional is true }
```

Notice you *must* enclose within braces ({...}) the code following the conditional statement. The conditional statement checks the value of an expression to determine whether to execute the code within the braces. In Perl, as in C, any nonzero value is considered true, whereas a zero value means false.

The following sections briefly describe the syntax of the major conditional statements in Perl.

if and unless

The Perl `if` statement is similar to the C `if` statement. For example, an `if` statement might check a count to see whether the count exceeds a threshold, as follows:

```
if ( $count > 25 ) { print "Too many errors!\n"; }
```

You can add an `else` clause to the `if` statement, as follows:

```
if ($user eq "root")
{
    print "Starting simulation...\n";
}
else
{
    print "Sorry $user, you must be \"root\" to run this program.\n";
    exit;
}
```

If you know C, you can see Perl's syntax looks like C. Conditionals with the `if` statement can have zero or more `elsif` clauses to account for more alternatives, such as the following:

```
print "Enter version number:"; # prompt user for version number
$os_version = <STDIN>;          # read from standard input
chop $os_version;               # get rid of the newline at the end of
the line
# Check version number
if ($os_version >= 10 ) { print "No upgrade necessary\n";}
```

```
elsif ($os_version >= 6 && $os_version < 9) { print "Standard
upgrade\n";}
elsif ($os_version > 3 && $os_version < 6) { print "Reinstall\n";}
else { print "Sorry, cannot upgrade\n";}
```

The unless statement is unique to Perl. This statement has the same form as if, including the use of elsif and else clauses. The difference is unless executes its statement block only if the condition is False. You could, for example, use the following:

```
unless ($user eq "root")
{
    print "You must be \"root\" to run this program.\n";
    exit;
}
```

In this case, unless the string user is "root", the script exits.

while

Use Perl's while statement for *looping*—repeating some processing until a condition becomes false. To read a line at a time from standard input and to process that line, you might use the following:

```
while ($in = <STDIN>)
{
# Code to process the line
    print $in;
}
```

If you read from the standard input without any argument, Perl assigns the current line of standard input to the $_ variable. Thus, you can write the preceding while loop as follows:

```
while (<STDIN>)
{
# Code to process the line
    print $_;
}
```

Perl's while statements are more versatile than those in C, because you can use almost anything as the condition to be tested. If you use an array as the condition, for example, the while loop executes until the array has no elements left, as in the following example:

```
# Assume %arg has the current set of command arguments
while (%arg)
{
    $arg = shift %arg;         # extract one argument
# Code to process the current argument
    print $arg;
}
```

The shift function removes the first element of an array and returns that element.

You can skip to the end of a loop with the next keyword; the last keyword exits the loop. The following while loop adds the numbers from 1 to 10, skipping 5:

```
while (1)
{
    $i++;
    if($i == 5) { next;}   # Jump to the next iteration if $i is 5
    if($i > 10) { last;}   # When $i exceeds 10, end the loop
    $sum += $i;            # Add the numbers
}
# At this point $sum should be 50
```

for and foreach

Perl's for statement has a similar syntax to C's for statement. Use the for statement to execute a statement any number of times, based on the value of an expression. The syntax is as follows:

```
for (expr_1; expr_2; expr_3) { statement block }
```

expr_1 is evaluated one time, at the beginning of the loop, and the statement block is executed until expression expr_2 evaluates to zero. The third expression, expr_3, is evaluated after each execution of the statement block. You can omit any of the expressions, but you must include the semicolons. Also, the braces around the statement block are required. Following is an example that uses a for loop to add the numbers from 1 to 10:

```
for($i=0, $sum=0; $i <= 10; $sum += $i, $i++) {}
```

In this example, the actual work of adding the numbers is done in the third expression, and the statement controlled by the for loop is an empty block ({}).

The foreach statement is most appropriate for arrays. Following is the syntax:

```
foreach Variable (Array) { statement block }
```

The foreach statement assigns to Variable an element from the Array and executes the statement block. The foreach statement repeats this procedure until no array elements are left. The following foreach statement adds the numbers from 1 to 10:

```
foreach $i (1..10) { $sum += $i;}
```

Notice I declare the array with the range operator (..). You also can use a list of comma-separated items as the array.

If you omit the Variable in a foreach statement, Perl implicitly uses the $_ variable to hold the current array element. Thus, you could use the following:

```
foreach (1..10) { $sum += $_;}
```

goto

The `goto` statement transfers control to a statement label. Following is an example that prompts the user for a value and repeats the request if the value is not acceptable:

```
ReEnter:
print "Enter offset: ";
$offset = <STDIN>;
chop $offset;
unless ($offset > 0 && $offset < 512)
{
    print "Bad offset: $offset\n";
    goto ReEnter;
}
```

Access to Linux

You can execute any Linux command from Perl in several ways:

- Call the `system` function with a string that contains the Linux command you want to execute.

- Enclose a Linux command within *backquotes* (`), which also are known as *grave accents*. You can run a Linux command this way and capture its output.

- Call the `fork` function to copy the current script and process new commands in the child process (if a process starts another process, then the new process is known as a *child process*).

- Call the `exec` function to overlay the current script with a new script or Linux command.

- Use `fork` and `exec` to provide shell-like behavior (monitor user input and process each user-entered command through a child process). This section presents a simple example of how to accomplish this task.

The simplest way to execute a Linux command in your script is to use the `system` function with the command in a string. After the `system` function returns, the exit code from the command is in the `$?` variable. You can easily write a simple Perl script that reads a string from the standard input and processes that string with the `system` function. Follow these steps:

1. Use a text editor to enter and save the following script in a file named `rcmd.pl`:

```
#!/usr/bin/perl
# Read user input and process command

$prompt = "Command (\"exit\" to quit): ";
print $prompt;

while (<STDIN>)
{
```

```
        chop;
        if ($_ eq "exit") { exit 0;}

# Execute command by calling system
        system $_;
        unless ($? == 0) {print "Error executing: $_\n";}
        print $prompt;
}
```

2. Make the `rcmd.pl` file executable, using the following command:

   `chmod +x rcmd.pl`

3. Run the script by typing `rcmd.pl` at the shell prompt. Following is some sample output from the `rcmd.pl` script (the output depends on what commands you enter):

```
Command ("exit" to quit): ps
PID TTY STAT  TIME COMMAND
  415  p0 S    0:00 /bin/login -h lnb486 -p
  416  p0 S    0:00 -bash
  585  p0 S    0:00 perl ./rcmd.pl
  586  p0 R    0:00 ps
Command ("exit" to quit): exit
```

Another way to run UNIX commands is to use `fork` and `exec` in your Perl script. Following is an example script — `psh.pl` — that uses `fork` and `exec` to execute commands entered by the user:

```
#!/usr/bin/perl

# This is a simple script that uses "fork" and "exec" to
# runs a command entered by the user

$prompt = "Command (\"exit\" to quit): ";
print $prompt;

while (<STDIN>)
{
    chop;    # remove trailing newline
    if($_ eq "exit") { exit 0;}

    $status = fork;
    if($status)
    {
# In parent... wait for child process to finish...
        wait;
        print $prompt;
        next;
    }
    else
    {
        exec $_;
    }
}
```

The following example shows how the `psh.pl` script executes the `ps` command:

```
Command ("exit" to quit): ps
  PID TTY STAT  TIME COMMAND
  415  p0 S     0:00 /bin/login -h lnb486 -p
  416  p0 S     0:00 -bash
  589  p0 S     0:00 perl ./psh.pl
  590  p0 R     0:00 ps
Command ("exit" to quit): exit
```

UNIX shells, such as Bash, use the `fork` and `exec` combination to run commands.

File access

You may have noticed the `<STDIN>` expression in various examples in this chapter. This is Perl's way of reading from a file. In Perl, a file is identified by a *file handle*, which is just another name for an identifier. Usually, file handles are in uppercase characters. `STDIN` happens to be a predefined file handle that denotes the standard input — by default, the keyboard. `STDOUT` and `STDERR` are the other two predefined file handles. `STDOUT` is used for printing to the terminal and `STDERR` is used for printing error messages.

To read from a file, you write the file handle inside angle brackets (`<>`). Thus, `<STDIN>` reads a line from the standard input.

You can open other files by using the `open` function. The following example shows you how to open the `/etc/passwd` file for reading and how to display the lines in that file:

```
open (PWDFILE, "/etc/passwd");  # PWDFILE is the file handle
while (<PWDFILE>) { print $_;}  # By default, input line is in $_
close PWDFILE;                  # Close the file
```

By default, the `open` function opens a file for reading. You can add special characters at the beginning of the filename to indicate other types of access. A > prefix opens the file for writing, whereas a > prefix opens a file for appending. Following is a short script that reads the `/etc/passwd` file and creates a new file, named `output`, with a list of all users without any shell (the password entries for these users has a : at the end of the line):

```
#!/usr/bin/perl
# Read /etc/passwd and create list of users without any shell

open (PWDFILE, "/etc/passwd");
open (RESULT, ">output");                  # open file for writing

while (<PWDFILE>)
{
    if ($_ =~ /:\n/) {print RESULT $_;}
}

close PWDFILE;
close RESULT;
```

After you execute this script, you should find a file named `output` in the current directory. Following is what the output file contains when this script is run on my Linux system:

```
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
```

One interesting filename prefix is the *pipe character*—the vertical bar (|). If you call `open` with a filename that begins with |, the rest of the filename is treated as a command. The Perl interpreter executes the command and you can use `print` calls to send input to this command. The following Perl script sends a mail message to a list of users:

```
#!/usr/bin/perl
# Send mail to a list of users

foreach ("root", "naba")
{
    open (MAILPIPE, "| mail -s Greetings $_");
    print MAILPIPE "Remember to send in your weekly report today!\n";
    close MAILPIPE;
}
```

If a filename ends with a pipe character (|), that filename is executed as a command, and you can read that command's output with the angle brackets, as shown in the following example:

```
open (PSPIPE, "ps -ax |");
while (<PSPIPE>)
{
# Process the output of the ps command—this example simply echoes
each line
    print $_;
}
```

Subroutines

Although Perl includes a large assortment of built-in functions, you can add your own code modules, in the form of subroutines. In fact, the Perl distribution comes with a large set of subroutines. Following is a simple script that illustrates the syntax of subroutines in Perl:

```
#!/usr/bin/perl
sub hello
{
```

```
# Make local copies of the arguments from the @_ array
    local ($first,$last) = @_;

    print "Hello, $first $last\n";
}

$a = Jane;
$b = Doe;

&hello($a, $b);     # Call the subroutine
```

When you run this script, it displays the following output:

```
Hello, Jane Doe
```

Following are some points to note about subroutines:

- The subroutine receives its arguments in the array @_ (the at symbol, followed by an underscore character).

- Variables used in subroutines are global by default. Use the `local` function to create a local set of variables.

- Call a subroutine by placing an ampersand (&) before its name. Thus, subroutine `hello` is called by &hello.

If you want, you can put a subroutine in its own file. The `hello` subroutine, for example, can be in a file named `hello.pl`. When you place a subroutine in a file, remember to add a return value at the end of the file—just type 1; at the end to return 1. Thus, the `hello.pl` file would be as follows:

```
sub hello
{
# Make local copies of the arguments from the @_ array
    local ($first,$last) = @_;

    print "Hello, $first $last\n";
}
1;      # return value
```

Then you have to write the script that uses the `hello` subroutine, as follows:

```
#!/usr/bin/perl
require 'hello.pl';   # include the file with the subroutine
definition

$a = Jane;
$b = Doe;

&hello($a, $b);     # Call the subroutine
```

This script uses the `require` function to include the `hello.pl` file that actually contains the definition of the `hello` subroutine.

Built-in functions in Perl

Perl has nearly 200 built-in functions (also referred to as *Perl functions*), including functions similar to the ones in the C Run-Time Library, as well as functions that access the operating system. You need to go through the list of functions to see the breadth of capabilities available in Perl. This chapter doesn't have enough space to cover these functions, but you can learn about the Perl functions by pointing your Web browser to the following address:

```
ftp://ftp.digital.com/pub/plan/perl/CPAN/doc/manual/html/perlfunc.html
```

Summary

At heart, Linux is still UNIX, and you must learn to use a shell — a command interpreter — to perform many common tasks. Even when you use a graphical interface, you usually have to open an `xterm` terminal window and type commands at the shell prompt. This chapter focuses on Bash — the Bourne Again Shell — as well as the scripting language Perl.

By reading this chapter, you learn the following:

▷ Even when you stay in the graphical environment of the X Window System, you have to type Linux commands in an `xterm` window to perform many routine tasks.

▷ A shell is a program that runs commands for you. Bash is the default shell in Linux. Bash is compatible with the Bourne shell that comes with all other UNIX systems.

▷ The Linux directory structure is logically organized as a single tree, regardless of the physical location of subdirectories. Linux includes many commands for navigating directories and manipulating files.

▷ The `find` command provides a powerful way to locate all files that meet specific search criteria.

▷ A shell script is nothing more than a sequence of Linux commands in a file. Typically, you have to place a special line at the beginning of the script file and make it executable. Then you can run the script by typing its name at the shell prompt.

▷ Perl is a popular scripting language that comes on this book's companion CD-ROM. You can use Perl to write powerful scripts on your Linux system.

▷ Perl contains features comparable to those of other programming languages, such as C. A powerful feature of Perl is its capability to use regular expressions and to search files for occurrences of a search pattern.

Chapter 7

Secrets of DOS Under Linux

━━

In This Chapter

▷ Accessing a DOS partition from Linux

▷ Using the mtools utility programs to access and use DOS floppy disks

▷ Exploring the capabilities of the DOSEMU DOS emulator

━━

Typically, you install Linux on a PC that previously had DOS and Microsoft Windows installed on it. If you happen to work in DOS and Windows as well as in Linux, you probably want to access the DOS files from Linux. This chapter shows you how to mount and access MS-DOS disks, including floppy disks.

You also learn about a package called mtools, which enables you to access and use (copy, delete, and format) MS-DOS files (typically, on a floppy disk) in Linux.

Mounting a DOS File System

If you have MS-DOS and Microsoft Windows installed on your hard disk, you probably already have the DOS partition mounted under Linux. During installation (see Chapter 1), the Red Hat installation program runs the Disk Druid program, which asks whether you want to access any DOS hard disk partition under Linux. Disk Druid finds these DOS partitions (as well as OS/2 partitions) by checking the hard disk's partition table.

Through the Disk Druid program you can specify where you want to mount each DOS partition. (Mounting makes the DOS directory hierarchy appear as part of the Linux file system.) A common choice is to mount the first DOS partition as /dosc, the second one as /dosd, and so on. If you specify these mount points, Disk Druid performs the necessary steps to ensure the DOS partitions are mounted automatically whenever you boot Linux.

To see whether you already have your DOS hard disk partition mounted automatically, follow these steps (you needn't be the root user to do this):

1. Use the `grep` command to look for the string `msdos` in the file `/etc/fstab`. Here is the result I get with `grep` on one of my Linux PCs:

```
grep msdos /etc/fstab
/dev/hda1  /dosc  msdos    defaults   0 0
```

I explain the file `/etc/fstab` in the "The `/etc/fstab` file" section of this chapter.

2. If the output shows one or more lines that contain `msdos`, your Linux system already mounts DOS hard disk partitions automatically. In this example, the output shows a matching line whose first field is the partition name `/dev/hda1` (the first partition on the first IDE disk); the second field, `/dosc`, shows where that partition is mounted.

3. If the `grep` command does not show any lines that contain the string `msdos` in `/etc/fstab`, your system does not mount any DOS hard disk partitions automatically. An explanation, of course, may be your hard disk does not have any DOS partitions.

Tip Another quick way to find out about the mounted devices is to type `mount` (without any arguments) at the shell prompt. Following is the output of the `mount` command on my system:

```
/dev/hda3 on / type ext2 (rw)
/dev/hda1 on /dosc type msdos (rw) (MS-DOS partition mounted on /dosc)
```

If you see any `msdos` in the output, those lines indicate MS-DOS file systems mounted on Linux. In this case, an MS-DOS partition is mounted on the Linux directory `/dosc`.

The following sections explain how the DOS partitions are mounted automatically. Even if you don't have any DOS partitions on your hard disk, you may want to learn how to access a DOS file system from Linux, because someday you may have to access a DOS floppy disk under Linux. Understanding the concept of mounting is the key to using a DOS file system under Linux.

The mount command

As Chapter 6 explains, Linux has a single file system that starts at the root directory, denoted by a single slash (/). Even if you have a separate hard disk (or multiple hard disk partitions on a single disk), the contents of those hard disks appear logically somewhere in the Linux file system. *Mounting* is the operation you must perform to cause a physical storage device (be it a hard disk partition or a CD-ROM) to appear as part of the Linux file system.

Many Linux systems have a small disk partition mounted on the root directory (/) and a larger partition mounted on the `/usr` directory. A larger partition is used for `/usr` because many software packages, including the X Window System, are installed under `/usr`.

You can use the mount command to mount a device manually on the Linux file system at a specified directory. This directory is referred to as the *mount point*. You can use any directory as the mount point. If you mount a device on a nonempty directory, though, you lose the ability to access the files in that directory until you unmount the device with the umount command. Therefore, you should always use an empty directory (such as /mnt) as the mount point.

Like any UNIX command, mount has numerous options. At the same time, you can get by with only a few options.

Because mounting makes a physical device part of the Linux file system, by default only the root user is allowed to run the mount command. However, the root user can set up entries in the /etc/fstab file to enable any user to mount a device. Consult the section "Mounting DOS floppy disks" to learn how you can enable any user to mount DOS floppy disks. If you try to mount a device when you are not logged in as root and you get the message

```
mount: only root can do that
```

this means you must log in as root first.

If you are not already logged in as root, use the su command to become root. When you type **su** without any argument, the shell assumes you want to become root and prompts you for the root password, as follows:

```
[naba@lnbp200 naba]$ su    (Become the root user)
Password:                   (Enter root password)
[root@lnbp200 naba]#        (Now you are root)
```

After you enter the root password, the prompt changes to indicate you are root.

As root, suppose you want to mount the CD-ROM device (the name is /dev/cdrom) on the mount point /mnt/cdrom (which should already exist on your system). To do so, type the following command:

```
mount /dev/cdrom /mnt/cdrom
```

The mount command will report an error if the CD-ROM device is mounted already. Otherwise, the mount operation succeeds and you can access the CD-ROM's contents through the /mnt/cdrom directory.

To mount a DOS partition, you use a similar format for the mount command, but you also should specify the type of file system on the DOS partition. If your DOS partition is the first partition on your IDE (Integrated Drive Electronics) drive and you want to mount it on /dosc, use the following mount command:

```
mount -t msdos /dev/hda1 /dosc
```

The `-t msdos` part of the `mount` command specifies the device you are mounting— `/dev/hda1` —has an MS-DOS file system. Linux has built-in support for MS-DOS files. Figure 7-1 illustrates the effect of this `mount` command.
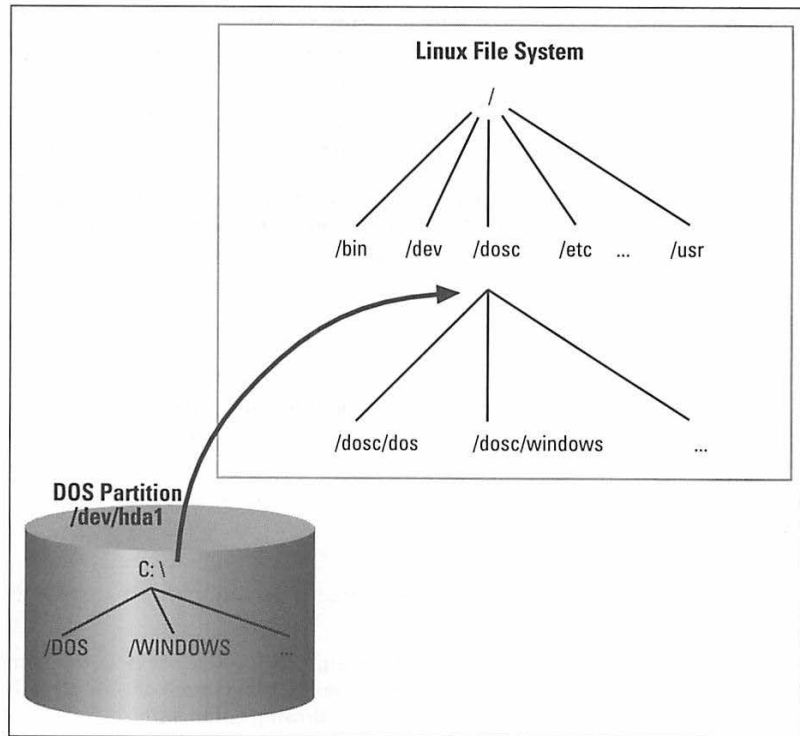
Figure 7-1: Mounting a DOS partition on the `/dosc` directory

Figure 7-1 also shows how directories in your DOS partition are mapped to the Linux file system. What used to be the `C:\DOS` directory under DOS becomes `/dosc/dos` under Linux. Similarly, `C:\WINDOWS` now is `/dosc/windows`. You probably can see the pattern now. To convert a DOS filename to Linux (for this specific case when you mount the DOS partition on `/dosc`), do the following:

■ Change the DOS names to lowercase.

■ Change `C:\` to `/dosc/`.

■ Change all backslashes (\) to slashes (/).

DOS floppy disks

Just as you can mount a DOS hard disk partition under Linux, you can mount a DOS floppy disk. Usually, you must log in as `root` to mount a floppy, but you can follow the steps shown in the latter part of this section and set up your system so any user can mount a DOS floppy disk. You also need to know the device name for the floppy drive. By default, Linux defines two generic floppy device names:

■ `/dev/fd0`, which is the A drive (the first floppy drive)

■ `/dev/fd1`, which is the B drive (the second floppy disk drive, if you have one)

As for the mount point, an existing directory named `/mnt/floppy` is specifically meant for this type of temporary mount operation. Thus, you can mount the DOS floppy disk on the `/mnt/floppy` directory with the following command:

```
mount -t msdos /dev/fd0 /mnt/floppy
```

After the floppy is mounted, you can copy files to and from the floppy using Linux's copy command (`cp`). To copy the file `xtmenu1.pcx` from the current directory to the floppy, type the following:

```
cp xtmenu1.pcx /mnt/floppy
```

Similarly, to see the contents of the floppy disk, type the following:

```
 ls /mnt/floppy
```

When you want to remove the floppy disk from the drive, you should first dismount the floppy drive. This action removes the association between the floppy disk's file system and the mount point on the Linux file system. Use the `umount` command to dismount a device, as follows:

```
umount /dev/fd0
```

You can set up your Linux system so any user can mount a DOS floppy disk. All you have to do is log in as `root` and add a line in the `/etc/fstab` file. For example, to enable users to mount a DOS floppy in the A drive on the `/a` directory, perform the following steps:

1. Log in as `root`.

2. Create the `/a` directory, with the following command:

   ```
   mkdir /a
   ```

3. Edit the `/etc/fstab` file in a text editor (such as `vi` or Emacs), insert the following line, save the file, and quit the editor:

   ```
   /dev/fd0    /a    msdos    noauto,user    0 0
   ```

You learn more about the /etc/fstab file in the next section; the user option (which appears next to noauto) enables all users to mount DOS floppy disks. The first field in that line is the device name (/dev/fd0), the second field is the mount directory (/a), and the third field shows the type of file system (msdos).

4. Log out and log in as a normal (not root) user.

5. To test that you can mount a DOS floppy disk without being root, insert a DOS floppy in the A drive and type the following command:

    mount /a

    Notice you use the mount directory as an argument for the mount command. The mount operation should succeed and you should see a listing of the DOS floppy when you type the command ls /a.

6. To unmount the DOS floppy, type umount /a.

The /etc/fstab file

In Linux, the /etc directory contains many text files that have configuration information for the system. As you learned in Chapter 5, for example, the /etc/inittab files contains information about what processes to start after Linux boots. The /etc/fstab file is one such configuration file — a text file containing information the mount and umount commands use. Each line in the /etc/fstab file provides information about a device to be mounted on a directory in the Linux file system.

Following is the /etc/fstab file from a typical Linux system:

```
/dev/hda3     /            ext2     defaults   1 1
/dev/hda1     /dosc        msdos    defaults   0 0
/dev/hda5     /dosd        msdos    defaults   0 0
/dev/hda4     swap         swap     defaults   0 0
/dev/fd0      /mnt/floppy  ext2     noauto     0 0
/dev/cdrom    /mnt/cdrom   iso9660  noauto,ro  0 0
none          /proc        proc     defaults   0 0
```

The first field on each line shows a device name, such as hard disk partition. The second field is the mount point and the third field indicates the type of file system on the device. You can ignore the last three fields for now.

The sample /etc/fstab file shows the /dev/hda4 device (the fourth partition on the first IDE hard disk) is used as a swap device for virtual memory, which is why both the mount point and the file-system type are set to swap. The last line shows another special file system — the proc file system — which Linux uses to store system information. The line on which msdos appears is the file-system type and specifies the DOS partition /dev/hda1 should be mounted on /dosc.

The contents of the /etc/fstab file are used to mount various file systems in Linux automatically. During Linux startup, the init process executes a shell script that invokes mount with the -a option. This script causes mount to read the /etc/fstab file and mount all listed file systems (except those with the noauto option). To mount a DOS partition automatically, therefore, you should add to the /etc/fstab file a line that contains the necessary information for mounting that partition. If you want to mount the DOS file system in the first partition of the first Small Computer System Interface (SCSI) disk on /dosd, for example, add the following line to /etc/fstab:

```
/dev/sda1    /dosd    msdos    defaults  0  0
```

The fourth field on each line of the /etc/fstab file shows a comma-separated list of options that apply to a specific device. Typically, you will find the defaults option in this field. The defaults option implies, among other things, that the device is mounted at boot time, only the root user can mount the device, and the device is mounted for reading and writing. If the options include noauto, the device is not automatically mounted when the system boots. Another useful option is user. Any user can mount a device in the /etc/fstab file that has the user option. For example, if you want to allow any user to mount the CD-ROM, log in as root and add the user option to the /dev/cdrom line in /etc/fstab as follows:

```
/dev/cdrom    /mnt/cdrom    iso9660 noauto,ro,user  0 0
```

With this line in place, any user can mount a CD-ROM with the following command:

```
mount /mnt/cdrom
```

Using mtools

The preceding sections show you one way to access the MS-DOS file system: mount the DOS hard disk or floppy disk using the mount command and then use regular Linux commands, such as ls and cp. This approach to mounting a DOS file system is fine for hard disks. Linux can mount the DOS partition automatically at startup and you can access the DOS directories on the hard disk whenever necessary.

If you want to get a quick directory listing of a DOS floppy disk, however, mounting can be tedious. First, you must mount the floppy drive, then you have to use the ls command, and, finally, you must use the umount command before taking the floppy disk out of the drive.

This is where the mtools package comes to the rescue. The mtools package implements most common DOS commands. The commands have the same names as in DOS, except you add an m prefix to each command. Thus, the command for getting a directory listing is mdir, and mcopy copies files. The best part of mtools is that you needn't mount the floppy disk to use the mtools commands.

Because the mtools commands write to and read from the physical
device (floppy disk), you must log in as root to perform these
commands. If you want any user to use the mtools commands, you
must alter the permission settings for the floppy drive devices. Use the
following command to enable anyone to read from and write to the first
floppy drive:

```
chmod o+rw /dev/fd0
```

Do I have mtools?

The mtools package comes with the Red Hat Linux distribution on this book's
companion CD-ROM. When you installed Linux, mtools was installed
automatically as part of the base Linux. The mtools executable files are in the
/usr/bin directory. To see whether you have mtools installed, type ls
/usr/bin/mdir at the shell prompt. If the ls command shows this file exists,
you should have mtools available on your system.

You can also type the following rpm command to verify mtools is installed on
your system:

```
rpm -q mtools
mtools-3.6-4
```

If mtools is installed, the output shows you the full name of the mtools
package. The sample output shows mtools Version 3.6 is installed on the
system.

To try mtools, follow these steps:

1. Log in as root; or, type **su** and then enter the root password.

2. Place a MS-DOS floppy disk in your system's A drive.

3. Type **mdir**. You should see the directory of the floppy disk (in the
   standard DOS directory listing format).

The /etc/mtools.conf file

The mtools package should work with the default setup, but if you get any
errors, you should check the /etc/mtools.conf file. This file contains the
definitions of the drives (such as A, B, and C) that the mtools utilities see.
Following are a few lines from a typical /etc/mtools.conf file:

```
drive a: file="/dev/fd0" exclusive
drive b: file="/dev/fd1" exclusive

# First IDE hard disk partition
drive c: file="/dev/hda1"

# First SCSI hard disk partition
# drive c: file="/dev/sda1"
```

The pound sign (#) starts comments. Each line defines a drive letter, the associated Linux device name, and some keywords that indicate how the device is accessed. In this example, the first two lines define drives A and B. The third noncomment line defines drive C as the first partition on the first IDE drive (/dev/hda1). If you have other DOS drives (for example, D), you can add another line that defines drive D as the appropriate disk partition.

If your system's A drive is a high-density 3.5-inch drive, you do not have to change anything in the default /etc/mtools.conf file.

Typically, you use the mtools utilities to access the floppy disks. Although you can define C and D drives for your DOS hard disk partitions, you may want to access those partitions by mounting them with the Linux mount command. Because the hard disk partitions can be mounted automatically at startup, accessing them through the Linux commands should be just as easy.

The mtools commands

As explained earlier in this chapter, the mtools package is a collection of utilities. So far, you have seen the command mdir — the mtools counterpart of the DIR command in DOS.

If you know the MS-DOS commands, using the mtools commands is easy. Type the DOS command in lowercase letters and remember to add m in front of each command. Because the Linux commands and filenames are case-sensitive, you must use all lowercase letters when you type mtools commands.

Tip

Table 7-1 summarizes the 13 commands available in mtools.

Table 7-1   The mtools commands

mtools *utility*	*MS-DOS command*	*Action*
mattrib	ATTRIB	Changes MS-DOS file-attribute flags
mcd	CD	Changes an MS-DOS directory
mcopy	COPY	Copies files between MS-DOS and Linux
mdel	DEL or ERASE	Deletes an MS-DOS file
mdir	DIR	Displays an MS-DOS directory listing
mformat	FORMAT	Places an MS-DOS file system on a low-level formatted floppy disk (use fdformat to low-level-format a floppy in Linux)
mlabel	LABEL	Initializes an MS-DOS volume label
mmd	MD or MKDIR	Creates an MS-DOS directory
mrd	RD or RMDIR	Deletes an MS-DOS directory

(continued)

Table 7-1	*(Continued)*	
mtools *utility*	**MS-DOS command**	**Action**
mread	COPY	Copies an MS-DOS file to a Linux file
mren	REN or RENAME	Renames an existing MS-DOS file
mtype	TYPE	Displays the contents of an MS-DOS file
mwrite	COPY	Copies a Linux file to MS-DOS

You can use the mtools commands just as you would use the corresponding DOS command. For example, the mdir command works like the DIR command in DOS. The same goes for all the other mtools commands shown in Table 7-1. Regarding wildcard characters (such as *), you must remember the Linux shell is the first program to see your command. If you do not want the shell to expand the wildcard character, therefore, you should use quotation marks around filenames containing any wildcard characters. To copy all *.txt files from the A drive to your Linux directory, for example, use this command:

mcopy "a:*.txt".

If you leave off the quotation marks, the shell tries to expand the string a:*.txt with filenames from the current Linux directory and then it tries to copy those files (if any) from the DOS floppy disk.

On the other hand, when you want to copy files from the Linux directory to the DOS floppy disk, you *do* want the shell to expand any wildcard characters. To copy all *.pcx files from the current Linux directory to the DOS floppy disk, for example, invoke mcopy this way:

mcopy *.pcx a:

The mtools utilities let you use the backslash character (\) as the directory separator, just as you would under DOS. Whenever you have a filename that contains the backslash character, you must enclose the string in double quotation marks. The following command copies a file from a subdirectory on the A drive to the current Linux directory:

mcopy "a:\test\sample.dat".

How to format a DOS floppy

Suppose you run Linux on your home PC and you no longer have MS-DOS installed on your system, but you have to copy some files on an MS-DOS floppy disk and take the disk to your office. If you already have a formatted MS-DOS floppy, you can simply mount that floppy disk and copy the file to the floppy using the Linux cp command. What if you don't have a formatted DOS floppy? The mtools package again comes to the rescue.

Wizard

The `mtools` package provides the `mformat` utility, which can format a floppy disk for use under MS-DOS. Unlike the DOS `format` command that formats a floppy in a single step, the `mformat` command requires you to follow a two-step process to prepare the floppy disk:

1. Use the `fdformat` command (a Linux command) to low-level-format a floppy disk. The `fdformat` command expects the floppy device name to be the argument; the device name includes all the parameters necessary for formatting the floppy disk.

   Figure 7-2 illustrates the device-naming convention for the floppy drive device. Based on the information shown in Figure 7-2, to format a 3.5-inch high-density floppy disk in your system's A drive, you use the following command:

   ```
   fdformat /dev/fd0H1440
   Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
   Formatting ... done
   Verifying ... done
   ```
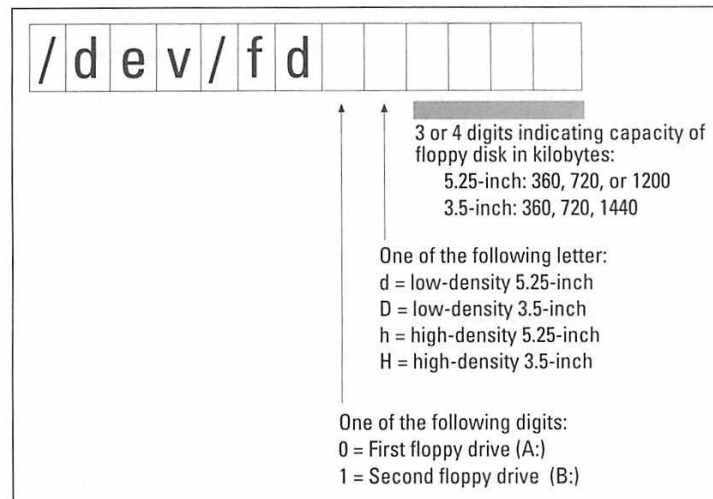

Figure 7-2: Naming convention for the floppy disk drive in Linux

2. Use the `mformat` command to put an MS-DOS file system on the low-level-formatted floppy disk. If the floppy is in drive A, type the following command:

   ```
   mformat a
   ```

Using DOSEMU

Accessing DOS file systems with the Linux `mount` command or through the `mtools` utilities enables you to exchange files with MS-DOS, but that capability does not help if you want to run a DOS program under Linux. As you might expect, Linux comes with a solution to this problem—you can use DOSEMU to create an MS-DOS environment within Linux.

Secret

Although DOSEMU stands for *DOS emulator*, it is really not an emulator. In fact, DOSEMU uses the same capabilities of the Intel 80×86 processor that Microsoft Windows uses to run MS-DOS in a window. Beginning with the 80386, Intel 80×86 and Pentium processors support a virtual-86 mode, in which multiple virtual 8086 machines can exist on the same 80×86 processor. As you may recall, the 8086 is the 16-bit microprocessor that originally ran MS-DOS. Because DOSEMU uses virtual-86 mode, it can run DOS programs in their native form, without any need to emulate the 80×86 instruction set. You need a copy of MS-DOS (any version from 3.3 to 6.2) to run DOSEMU.

DOSEMU must simulate a DOS environment for DOS applications that want to access the system's resources directly, such as the keyboard, printer, serial port, display, and disk. A configuration file, `/etc/dosemu.conf`, controls the way that DOSEMU accesses various system resources.

Caution

Although this section introduces DOSEMU, you should know DOSEMU is still under development. The DOSEMU version currently is 0.66 (it was not considered to be mature enough to have a 1.0 version number). Although DOSEMU is safe enough to try, you should be prepared to reboot your Linux system if anything goes wrong. Back up your important files before you try DOSEMU.

Installing DOSEMU

When you install Linux from the companion CD-ROM, if you select the DOS/Windows Connectivity component, the installation program installs DOSEMU. If you did not install DOSEMU during the initial installation, follow these steps to install DOSEMU now:

1. If you are not already logged in as `root`, type the **su** command and provide the root password to become `root`.

2. Make sure the companion CD-ROM is in the CD-ROM drive. If the CD-ROM is not mounted already, mount it by using the following command:

   `mount /dev/cdrom /mnt/cdrom`

3. Change the current directory to the CD-ROM directory in which the Red Hat packages are located, as follows:

   `cd /mnt/cdrom/RedHat/RPMS`

4. Type the following command to install DOSEMU:

   `rpm -i dosemu*`

   All the files necessary for running DOSEMU should be installed in the appropriate directories after this `rpm` command completes.

Reading the manual

After you install DOSEMU, you should read the documentation for information on how to set up and run it. You should start with the QuickStart file (a text file) in the /usr/doc/dosemu* directory. (The exact directory name depends on the version of DOSEMU you are using.) Use the following commands to browse this file:

```
cd /usr/doc/dosemu*
less QuickStart
```

Tip The less command enables you to view the file one page at a time. less is similar to more, except less enables you to go backward as well as forward.

Following are the key points explained in the QuickStart file:

■ The file /etc/dosemu.users contains the names of users who are allowed to run DOSEMU and what operations each user may perform in DOSEMU.

■ You must edit the DOSEMU configuration file /etc/dosemu.conf to reflect your system's configuration.

■ You need a hard disk *image file* (a Linux file DOSEMU treats as a DOS drive). This file is present when you install DOSEMU from this book's companion CD-ROM.

■ To set up DOSEMU for the first time, you need a MS-DOS boot floppy that contains the files FDISK.EXE and SYS.COM. You can create this floppy under DOS, which must have come with your PC.

The following section shows you how to use the DOS floppy disk to set up the hard disk image. (This image file is used as the C drive under DOSEMU.)

Configuring DOSEMU

The DOSEMU distribution on this book's companion CD-ROM comes nearly ready to run. You do have to go through some configuration steps, however, before you can run DOSEMU. The following sections cover these tasks.

Editing /etc/docemu.conf

You must be logged in as root when you perform these configuration tasks. The first task is to edit the /etc/dosemu.conf file. Like most Linux configuration files, /etc/dosemu.conf is a text file. Comments start with a pound sign (#). The file already contains many commented lines, as well as instructions for each section.

To begin, you should note the following items in the configuration file:

1. Search for the string hdimage, and make sure no comment symbol (#) is at the beginning of that line, so it appears as follows:

```
disk { image "/var/lib/dosemu/hdimage" }    # use diskimage file.
```

This line refers to a Linux file `/var/lib/dosemu/hdimage` that serves as a disk under DOSEMU. The name `hdimage` stands for *hard disk image*. The first `disk` line in the `/etc/dosemu.conf` line defines the C drive under DOSEMU. When you install DOSEMU, the installation program also creates a `/varlib/dosemu/hdimage` file you can use as the hard disk image.

2. If you have a DOS partition you want to access under DOSEMU, uncomment an appropriate line (or add a new one with the correct name of your DOS partition). My system's DOS partition is `/dev/hda1`, so I have the following disk line uncommented in `/etc/dosemu.conf`:

```
disk { partition "/dev/hda1" readonly }    # 1st partition on 1st IDE.
```

This partition will then become the D drive under DOSEMU.

3. Search for `fd0` and make sure the floppy disk devices `/dev/fd0` (first floppy drive) and `/dev/fd1` (second floppy drive) are the correct size (`threeinch` or `fiveinch`). My system has only one 3.5-inch floppy drive, so I have only one uncommented line in the FLOPPY DISKS section of the `/etc/dosemu.conf` file:

```
floppy { device /dev/fd0 threeinch }
```

The `/etc/dosemu.conf` file contains many more options, but you needn't configure everything right now. The first step is to get DOSEMU running. Then you can turn to specific items, such as how to make the printer work and how to access the serial port under DOSEMU.

Initializing the hard disk image

You can think of the hard disk image as being a raw hard disk: you have to format it and make it bootable before DOSEMU can boot off that hard disk image. To do this, you need a real MS-DOS boot floppy disk. You can create the boot floppy by using the MS-DOS installation that came with your PC.

To create a DOS boot floppy, you first have to boot your system under MS-DOS (or Windows 95) and then perform these steps:

1. Put a floppy disk in your A drive (all previous contents of the floppy will be destroyed when you format it) and type the command FORMAT A: /S. MS-DOS prompts you with the following message:

```
Insert new diskette for drive A:
and press ENTER when ready...
```

2. Press Enter, because you already have the floppy disk in the A drive. MS-DOS formats the disk and places the necessary operating-system files on it (that's what the /S option of the FORMAT command does).

3. MS-DOS also prompts you for a Volume label. You can simply press Enter in response.

4. DOS asks whether you want to format another floppy disk. Type N to indicate you are done formatting.

5. Copy `FDISK.EXE` and `SYS.COM` files to the floppy disk. You need these two programs to initialize the hard disk image under DOSEMU. Use the following commands to copy the `FDISK.EXE` and `SYS.COM` programs to the floppy in the A drive:
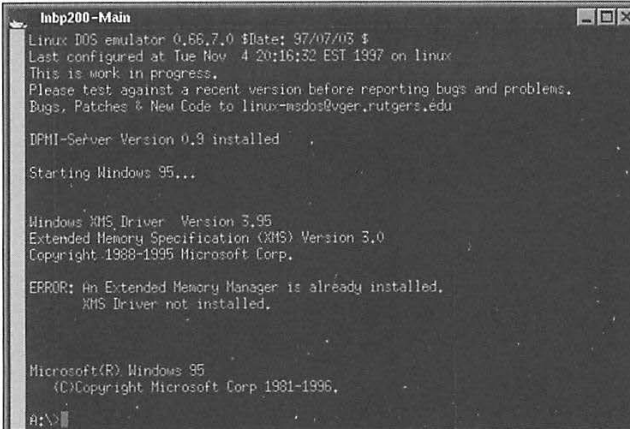
```
c:
cd \windows\command (typically, the MS-DOS files are in this
directory)
copy fdisk.exe a:
copy sys.com a:
```

After you prepare the DOS boot floppy, restart Linux on your system, using your favorite method (LILO or a Linux boot floppy). Then log in as `root`.

Next, put the DOS boot floppy in the A drive and type the following command at the shell prompt:

```
dos -A
```

After a brief pause, DOSEMU should boot off the A drive and you are left with an `A:\>` prompt, as shown in Figure 7-3.

Figure 7-3: Starting DOSEMU for the first time from a DOS boot floppy disk

Type the following DOS command:

```
A:\> dir c:
```

DOSEMU should display the contents of the C drive — actually, the hard disk image file DOSEMU treats as the C drive. Your next task is to make this C drive bootable. To initialize the hard disk image (the C drive), perform these steps:

1. To initialize the master boot record of the C drive, type the following command:

   ```
   A:\> fdisk /mbr
   ```

You might get a few warning messages; you can ignore those messages.

2. Next, transfer the necessary system files to the hard disk image using the following command:
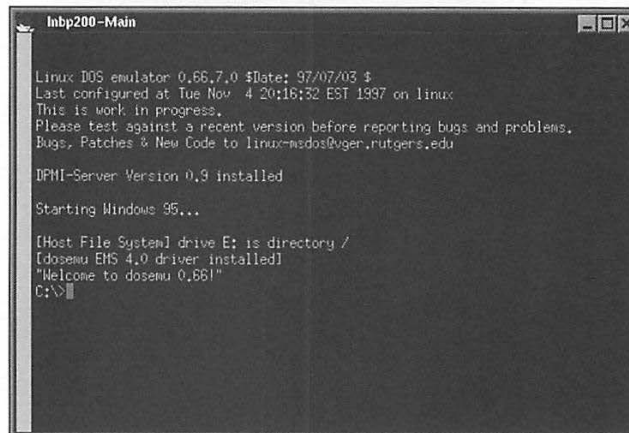
   ```
   A:\> sys c:
   System transferred
   ```

After these two steps, the hard disk image (C drive in DOSEMU) should be bootable. Before you can test the hard disk image, exit DOSEMU by using this command:

```
A:\> c:\exitemu
```

Starting DOSEMU from the hard disk image

After you get your hard disk image file set up properly, you should be able to boot directly from that image. Remove the DOS boot floppy from your system's A drive and type **dos** at the shell prompt. DOSEMU should boot from the hard disk image and display the familiar C:\> prompt, as shown in Figure 7-4.

Figure 7-4: Booting DOSEMU from the hard disk image

In Figure 7-4, you see DOSEMU displaying some information and then booting from drive C. You get the C:\> prompt, from which you can type DOS commands.

If you type dir, you see the hard disk image that works as the C drive is about 1MB. That's adequate, though, because you can access your real DOS partition as well as your Linux partition. The C drive is used only to boot MS-DOS.

Tip

The initial DOSEMU message indicates the drive letter assigned to the Linux file system (the / directory). For example, the message in Figure 7-4 says the / directory is drive E. This means if you type dir e:, you should see the contents of the Linux root (/) directory.

If you have a DOS partition on your hard disk and you added an appropriate disk line in the /etc/dosemu.conf file, that partition should be the D drive under DOSEMU. Try the command dir d: and see what DOSEMU shows you. You should be able to run some simple DOS programs (such as the MS-DOS editor) from your hard disk's DOS partition. Type d:\windows\command\edit sample.txt to start the MS-DOS editor and edit the sample.txt file. Figure 7-5 shows the MS-DOS editor running under a DOSEMU session in an xterm window.
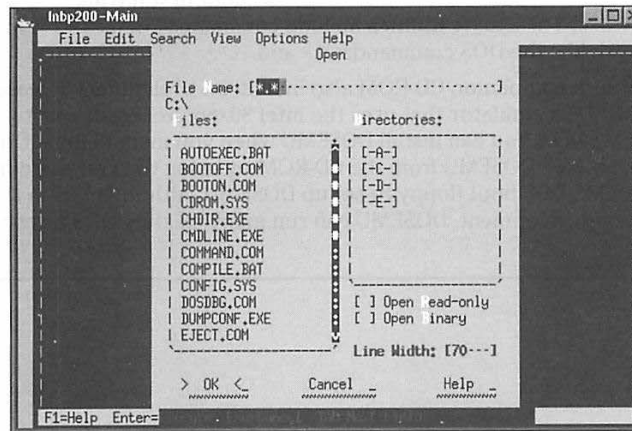
Figure 7-5: The Open dialog box of the MS-DOS editor running under DOSEMU in an xterm window

Figure 7-5 shows the editor's Open dialog box (press Alt+F and then type *o* to open this dialog). To quit the editor, press Alt+F and then type *x*.

You can get a DOS-like screen if you start DOSEMU from a text-mode console instead of from an xterm window.

When you are ready to quit DOSEMU, type **c:\exitemu** at the DOS prompt.

Summary

If your PC has a DOS partition in addition to the Linux partition or if you work with DOS floppy disks, you can access the DOS files directly from Linux. You also can use the mtools utility programs to format and use a DOS floppy disk directly from Linux. In addition, an ongoing project — DOSEMU — provides a DOS emulator that enables you to run DOS programs directly in Linux.

By reading this chapter, you learn the following:

▷ Linux has built-in support for the MS-DOS file system. You can use the mount command to access a DOS partition or a DOS floppy from Linux. After mounting a DOS file system at a directory in your Linux system, you can use Linux commands such as ls and cp to manipulate the DOS files.

▷ When you installed Linux, following the directions in Chapter 1, you also installed a set of utility programs known as mtools. The mtools programs provide a convenient way to access MS-DOS files, especially floppy disks, because you can use mtools commands without first having to mount the floppy disk. The mtools utilities include commands such as mdir and mcopy that work like the DOS commands DIR and COPY.

▷ This book's companion CD-ROM also includes a preliminary release of the DOSEMU DOS emulator that uses the Intel 80×86 processor's virtual-86 mode to run MS-DOS. You can install DOSEMU when you install Linux. Otherwise, you can install DOSEMU from the CD-ROM by using the rpm -i command. You need an MS-DOS boot floppy to set up DOSEMU. Although it is in its early stages of development, DOSEMU can run many existing DOS programs.

Part III:

Exploiting Your Hardware in Linux

Chapter 11

Disk Drives

In This Chapter

▷ Surveying the different types of disk controllers Linux supports

▷ Understanding disk drive concepts: cylinder, heads, and sectors (CHS)

▷ Understanding disk drive operations: partitioning and booting with LILO

▷ Understanding the 1,024-cylinder limit of BIOS

▷ Surveying SCSI disks Linux supports

▷ Troubleshooting SCSI

▷ Using Iomega Zip drives in Linux

▷ Looking at known bugs in EIDE disk controllers for the PCI bus

When it comes to disk drives and Linux, what matters is whether Linux supports the *disk controller*: the card that connects the disk drive to your PC's motherboard. Linux supports most common disk controllers. This chapter gives you information on different disk controllers and specific details on how to install and run Linux on a system that has one or more of these controllers.

Disk Controller Types

The disk controller is the adapter card that acts as an intermediary between your PC's motherboard and one or more hard disk drives. Typically, you can connect up to two hard drives and two floppy drives to a single disk controller. The Small Computer System Interface (SCSI) controller is an exception to this norm; you can connect up to seven SCSI devices (anything that has a SCSI interface, such as a disk drive, CD-ROM drive, tape drive, or scanner).

Over the years, several types of hard disk controllers have appeared for the PC. Following are some of the disk controllers you may find in a PC:

- *ST506* disk controllers, which originally appeared in IBM XT and AT computers, became the common disk controller of the PC industry in its early years (remember, the IBM PC-AT came out in 1984). Many PCs have disk controllers that are compatible with the ST506. Seagate's ST506 was the original hard drive for PCs; Western Digital's WD1003 was the controller card. Thus, these controllers are often referred to as being WD1003-compatible controllers. The original ST506 drives used a recording method known as Modified Frequency Modulation (MFM). Many ST506 disk controllers also support drives that use another type of data-recording technique, known as Run Length Limited (RLL).

- The *Integrated Drive Electronics* (IDE) interface emulates the ST506 interface. IDE drives, however, contain the necessary controller circuitry built into the drive itself. The motherboard typically contains an IDE interface for connecting the drive to the motherboard. By now, IDE drives are in widespread use in PCs. Nowadays, the term *AT Attachment* (ATA) is used to refer to IDE. The original IDE interface could support only two drives and it limited the maximum disk size to approximately 500MB. You needed third-party drivers to use disks larger than 500MB. Today's PCs with large disk drives use the Enhanced IDE interface (described next).

- The *Enhanced IDE* (EIDE) interface supports up to four internal IDE devices (which include hard drives as well as CD-ROM drives), higher-capacity drives, and higher speeds of data transfer. Typical EIDE interfaces consist of two IDE interfaces: primary and secondary, each of which is capable of supporting up to two drives. EIDE interfaces are popular because of their low cost. Many PCs use the EIDE interface to connect both the hard disk and the CD-ROM drive to the PC's motherboard.

- The *Enhanced Small Device Interface* (ESDI) controllers emulate the ST506 interface but provide higher data-transfer rates.

- *Small Computer System Interface* (SCSI) controllers provide a separate bus onto which you can connect up to seven SCSI devices. You can find hard drives, CD-ROM drives, tape drives, and scanners that support SCSI. You can connect multiple SCSI devices to the computer by daisy-chaining the devices with a SCSI cable. All UNIX workstations (such as the ones from Hewlett-Packard, IBM, and Sun Microsystems) use SCSI. Lately, SCSI is also becoming popular on PCs. The only drawback is that the SCSI controller is relatively expensive compared with EIDE controllers.

Linux supports all these common disk controllers. Even though several disk controllers appear in the preceding list, essentially two types of controllers exist: IDE (where IDE refers to all ST506-compatible interfaces) and SCSI.

Most Pentium systems use a motherboard that supports the Peripheral Component Interconnect (PCI) bus. These PCI systems need SCSI adapter cards that plug into PCI bus slots and IDE interface that connects to the PCI bus. Linux supports the PCI bus. Initially, problems occurred with specific PCI-interface hardware, but Linux can now work around some of these problems. The PCI bus seems poised to become the dominant bus in the PC marketplace, replacing the old and outdated ISA bus. (Chapter 9 discusses various buses.)

Disk Drive Concepts

When you read about disk drives, you'll run into some terms and concepts unique to the world of hard disks. This section explains some of these concepts.

Cylinders, heads, and sectors

The physical organization of the disk is expressed in terms of cylinders, heads, and sectors. A hard disk consists of several platters of magnetic material. In physical terms, you can think of *cylinder, head*, and *sector* as meaning the following:

- A *cylinder* is one of a series of concentric rings on one side of a disk platter. Each side of the platter is divided into cylinders.

- The total number of *heads* is the number of sides of all the magnetic platters.

- A *sector* is a pie-shape wedge on the platter. Each cylinder is divided into sectors.

Any location on the disk can be expressed in terms of the cylinder, the head, and the sector. Identifying a disk location in terms of cylinder (C), head (H), and sector (S) is known as *CHS addressing*.

The physical *geometry* of a hard disk usually is expressed in terms of cylinders, heads, and sectors. A disk may have a geometry of 528/32/63, which means the disk has 528 cylinders, 32 heads, and 63 sectors. Usually, each sector can store 512 bytes (or half a kilobyte) of data. Thus, the capacity of this disk is (528×32×63) / 2 kilobytes, or 532,224 kilobytes.

PC hard disk controllers include a read-only memory (ROM) Basic Input/Output System (BIOS) on the controller. By convention (and compatibility with the original IBM PC architecture), the BIOS uses CHS addressing to access the hard disk. The disk BIOS, however, uses a 10-bit value as the cylinder address. Because 10 bits can hold numbers between 0 and 1,023, the BIOS can address, at most, 1,024 cylinders.

Many large disks have more than 1,024 cylinders. To accommodate the 1,024-cylinder limit, the disk controllers have to resort to some tricks to handle disks with more than 1,024 cylinders. Later, in the section on "Disks with more than 1,024 Cylinders," you read more about problems with disks that have more than 1,024 cylinders; you should be able to relate to this problem.

Master Boot Record (MBR)

The first sector of the hard disk (cylinder 0, head 0, sector 1) is called the *Master Boot Record* (MBR). This 512-byte storage area contains important information about the disk, such as the partition table and a small amount of code the PC's BIOS loads and runs when you boot the PC.

The small program in the MBR reads the partition table; determines which partition is active (that's just an attribute of a partition); reads the active partition's first sector—or *boot sector*—and runs whatever program resides in that boot sector. The program in a partition's boot sector usually loads whatever operating system is installed on that partition.

When you install the Linux Loader (LILO) on the hard disk, the LILO program resides on the hard disk's MBR or the boot sector of the partition where the Linux root directory (/) is located.

Partitions

Partitions are a way of dividing a hard disk and treating each part separately. By dividing your PC's hard disk into partitions, you can install different operating systems in different partitions. Even if you use the entire disk for Linux, you would, at minimum, need a partition Linux can use as *swap space* —an extension of memory, so you can have more virtual memory than the physical memory on your system.

The Master Boot Record contains the partition table, starting at byte 446 (0x1be, which means 1be in hexadecimal). The partition table can have up to four 16-byte entries. Each 16-byte value defines a partition. Each partition is specified by a starting and an ending cylinder number. The partition entry also includes a type that identifies the operating system that created the partition. The concept of partitions is a convention all PC-based operating systems, ranging from MS-DOS to Linux, follow.

In MS-DOS, you use the FDISK program to manipulate the partitions. Linux includes a program with the same name— fdisk (lowercase)— to alter the disk partitions.

Linux device names for disks

In Linux, each device is represented by a device file in the /dev directory. The device name for the hard disk depends on the type of disk controller. For IDE and EIDE drives, the device name is /dev/hda for the first disk, /dev/hdb for the second disk, and so on.

On an EIDE interface, if you have a hard disk on the primary interface and a CD-ROM drive on the secondary interface, the device names are /dev/hda for the hard disk drive and /dev/hdc for the CD-ROM drive.

The Linux disk drivers treat each disk partition as being a separate device. The first partition in the first IDE disk is /dev/hda1, the second partition is /dev/hda2, the third one is /dev/hda3, and so on. Similarly, the device names for the partitions on the second IDE drive are /dev/hdb1, /dev/hdb2, and so on.

The SCSI disk devices are named /dev/sda, /dev/sdb, and so on. If a SCSI device is a hard disk, its partitions are named by appending the partition number to the device name. Thus, the partitions of the first SCSI hard disk are named /dev/sda1, /dev/sda2, /dev/sda3, and so on.

Floppy Disks in Linux

Chapter 7 describes several ways to access MS-DOS floppy disks under Linux. You can mount the floppy and use Linux commands or use the mtools utility programs to read from or write to the floppy. You also can create a Linux file system on a floppy disk. In fact, you'll find Linux file systems on the boot and root floppies you use to install Linux.

Formatting and creating a Linux file system on a floppy disk is a straightforward process. To format a 3.5-inch high-density floppy in the A drive, for example, use the following command:

```
fdformat /dev/fd0H1440
```

For a 5.25-inch high-density floppy, change the device name to /dev/fd0h1200. On the B drive, change the first 0 in the device name to 1.

After you format the floppy, use the following command to create a Linux file system on the floppy:

```
mke2fs -m 0 /dev/fd0H1440 1440
```

The -m option is used to specify what percentage of blocks should be reserved for use by the *super user* (root). By specifying the -m 0 option, you ensure mke2fs does not reserve any space on the floppy disk for the super user. If you do not explicitly specify the -m option, mke2fs reserves 5 percent of the disk space for the super user.

After you create the file system on the floppy drive, you can mount the floppy at a *mount point* (an empty directory) in the Linux file system. The following example shows how to mount the floppy drive at the /mnt directory:

```
mount /dev/fd0H1440 /mnt
```

Now you can use Linux commands, such as cp and mv, to copy or move files to the floppy disk. Before you eject the floppy disk from the drive, use the following command to dismount the floppy:

```
umount /dev/fd0H1440
```

Hard Disk Operations in Linux

You must perform some disk operations to install and use Linux on your system. Chapter 1 explains some of the disk operations you perform when you set up Linux. The next few sections provide some additional information about these disk operations.

When you first get a PC, the hard disk usually is set up as a one huge partition, and DOS and Windows are already installed on it. (If you bought your PC recently, it probably came with Windows 95 or Windows 98 preinstalled.) To install Linux, you have to start by creating at least two partitions for Linux: one for the swap space and the other for the Linux file system.

If you have original disks to reinstall the current operating system (be it Windows 95 or MS-DOS and Windows 3.1), simply go ahead and repartition the disk. To do this, boot the PC, using a DOS boot floppy, and run the MS-DOS version of FDISK program to delete the current partition and create new ones. Create three new partitions: One for DOS and Windows, one for Linux swap space, and one for the Linux file system. Then you have to reinstall DOS and Windows in their partition and install Linux on the other partition.

If you do not want to go through the trouble of reinstalling DOS and Windows or Windows 95, you have to alter the existing partition somehow. FIPS, which can split an existing DOS partition into two separate partitions, enables you to perform this task. Chapter 1 describes how to use FIPS.

Cross-
Reference

Partitioning with fdisk

Partitioning the disk involves creating several smaller logical devices within a single hard disk. Under MS-DOS, you would use the FDISK program to view and alter a disk's partition table. In Linux, the partitioning program is called fdisk.

Chapter 1 explains how to use Linux and MS-DOS FDISK during Linux installation. I mention it here for the sake of completeness, to jog your memory about the fdisk program.

Even if you have already partitioned your hard disk, you can always run fdisk just to see the current partition table of a hard disk. If you have a SCSI disk drive with the device name /dev/sda, for example, you can look at its partition table with fdisk as follows:

```
fdisk /dev/sda
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help): p
Disk /dev/sda: 64 heads, 32 sectors, 500 cylinders
Units = cylinders of 2048 * 512 bytes

   Device Boot   Begin    Start     End  Blocks   Id  System
/dev/sda1             1        1     301  308208    6  DOS 16-bit >=32M
/dev/sda2           302      302     499  202752   83  Linux native

Command (m for help): q
```

The m command shows you a list of the single-letter commands fdisk accepts. You can see the current partition table with a p command. This example's SCSI disk has two partitions: one for DOS and the other for Linux.

The Id field in the table of partitions printed by the fdisk program (when you type p at the fdisk prompt) is a number that denotes a partition type. If you want to see a list of all known partition types, type l (that's a lowercase *L*) at the fdisk prompt. Table 11-1 lists the partition types that Linux understands:

Table 11-1   Partition types known to Linux

Type	Description	Type	Description	Type	Description	Type	Description
0	Empty	9	AIX bootable	75	PC/IX	b7	BSDI fs
1	DOS 12-bit FAT	a	OS/2 Boot Manager	80	Old MINIX	b8	BSDI swap
2	XENIX root	b	Win95 FAT32	81	Linux/MINIX	c7	Syrinx
3	XENIX usr	40	Venix 80286	82	Linux swap	db	CP/M
4	DOS 16-bit <32M	51	Novell?	83	Linux native	e1	DOS access
5	Extended	52	Microport	93	Amoeba	e3	DOS R/O
6	DOS 16-bit >=32	63	GNU HURD	94	Amoeba BBT	f2	DOS secondary
7	OS/2 HPFS	64	Novell Netware	a5	BSD/386	ff	BBT
8	AIX	65	Novell Netware				

Booting from the hard disk with LILO

To boot Linux from a hard disk automatically, you need the Linux Loader
(LILO). LILO is a boot loader program; OS/2 has an equivalent program,
called *Boot Manager*. These programs usually reside in the Master Boot
Record of a disk and are the first to get loaded. The boot loader program
then, in turn, prompts you for the name of a operating system to start (which
typically means a disk partition from which to boot). Starting an operating
system basically involves loading that operating system's main program into
memory and running it. For Linux, this step involves loading the Linux kernel
into memory and giving control to the kernel.

Cross-Reference

LILO is much more than just Linux Loader; it also serves as a general-purpose
boot manager capable of booting MS-DOS, OS/2, or Windows 95. Chapter 1
describes how you can install LILO on your hard disk during the Linux
installation process. This section summarizes the LILO installation process if
you do it outside the Linux installation program described in Chapter 1.

Tip

The LILO documentation is in /usr/doc/lilo-0.20 on your system. You should
consult the README file in /usr/doc/lilo-0.20 for the latest word on installing
and configuring LILO. The following section provides an overview only.

Installing LILO

Typically, you install LILO as part of the Linux installation process described in Chapter 1. The only time you must repeat LILO configuration and installation is when you update the kernel or add a new operating system you want to boot through LILO.

Installing LILO involves two basic steps:

1. Prepare the LILO configuration file, which contains information necessary for installing LILO. The default LILO configuration file is `/etc/lilo.conf`.

2. Run the `/sbin/lilo` program (referred to as the *map installer* in LILO's README file) to update the boot sector and create the `/boot/map` file, which contains information that LILO uses during the boot process.

To prepare the LILO configuration file, you must log in as `root` and edit the `/etc/lilo.conf` file. Chapter 2 briefly describes how to edit this file. Essentially, you add or edit information about the operating systems you want to boot with Linux and the names of the disk partitions where those operating systems reside.

Cross-Reference

When the LILO configuration file — `/etc/lilo.conf` — is ready, you can install LILO with the following command:

`/sbin/lilo`

The `lilo` program looks for the `lilo.conf` file in the `/etc` directory, interprets its contents, prepares the necessary boot and map files, and initializes the boot record of the device specified (by the `boot` line) in the configuration file. If the boot device is specified as a hard disk partition, for example, LILO sets up the boot sector of that disk partition.

Tip

The files in `/usr/doc/lilo-0.20`, especially `/usr/doc/lilo-0.20/README`, contain the latest information on LILO. You may want to check out this directory if you have a unique arrangement of hard disk partitions and you want to find out whether you can use LILO to boot Linux.

Using LILO's boot prompt

As LILO starts, it displays the `boot:` prompt and waits for the name of a boot image to load. If you do not respond within a specified period (this value is stored in the `/etc/lilo.conf` file), LILO loads the default boot image (the first image in the `/etc/lilo.conf` file).

Wizard

When you use LILO to boot Linux, you can specify the name of the Linux boot image, followed by one or more options. LILO passes these boot command-line options directly to the Linux kernel. As you read descriptions of various disk controllers, you find information on boot command-line options you can use to specify various parameters of a device. These parameters typically include the I/O port address, IRQ, and DMA of a device. Notice boot command-line options are always case-sensitive.

Removing LILO

If you set up LILO on the hard disk's Master Boot Record, but later want to remove it, you can do so easily, provided you have MS-DOS Version 5.0 or later. All you must do is boot from a DOS boot floppy (place the boot floppy in drive A and then power up the PC) and run FDISK with the /MBR option. The FDISK /MBR command essentially restores the Master Boot Record to the format MS-DOS uses. The next time you boot the PC, MS-DOS should start immediately.

Another way to uninstall LILO is to use LILO itself as follows:

```
/sbin/lilo -u
```

LILO replaces the MBR from a saved copy of the old boot sector, which LILO saves when you first install LILO.

Use the `/sbin/lilo -u` command only if you have installed LILO on your hard disk's MBR. When you use the `-u` option, LILO simply copies a file named `/boot/boot.nnnn` (*nnnn* is the device number, such as 0300 for the `/dev/hda` and 0800 for `/dev/sda`) to the disk's MBR. If you have been playing with LILO and an old boot file happens to be left over in the `/boot` directory, LILO might copy that file to the MBR. A bad MBR, of course, makes the disk unbootable. In such a case, boot from a DOS floppy and use the FDISK /MBR command to restore the MBR to boot DOS.

Creating swap space

Swap space is a disk partition Linux uses as an extension of its memory. When some memory-resident data is not needed immediately, Linux stores this data in the swap space. To create the swap space, you have to create a disk partition, using `fdisk`. Make sure you set the type of this disk partition to Linux swap. Typically, you can set up the swap partition and turn on swapping as you install Linux from this book's Slackware Linux.

If you must set up the swap space outside the installation program, you have to follow the procedure outlined in Chapter 1. The basic steps are to use the `mkswap` command to initialize the swap partition. You need the size of the swap partition (in number of blocks) before you use the `mkswap` command. Use the Linux `fdisk` program to find this information.

After `mkswap` finishes, use the `swapon` command to turn on swapping. Linux then begins to use the swap space.

To ensure Linux uses the swap space every time it boots, you need a line in the `/etc/fstab` file that indicates the swap partition's name. If the swap partition is `/dev/hda2`, for example, you would need the following line to `/etc/fstab`:

```
/dev/hda2       swap       swap     defaults     0 0
```

If you created a swap space when you installed Linux, the installation program adds the appropriate line in the `/etc/fstab` file. You may need to add such a line if you create additional swap spaces later.

If you put a partition's name in /etc/fstab, but forget to run mkswap on that partition, Linux displays the following error message:

Secret

```
Unable to find swap-space signature
```

The fix is to run mkswap to initialize the swap partition.

Creating file systems

To use a disk partition in Linux, you have to create a file system on that partition. You can think of this procedure as formatting the partition for Linux. When you install Linux by following the steps described in Chapter 1, one of the steps creates the file system; the setup program actually asks you whether you want to format a partition. For a Linux partition, the setup program uses the mk2efs command to create a Linux file system.

Linux supports several types of file systems, including the following:

- *MS-DOS file system.* This DOS file system is based on the File Allocation Table (FAT). This type is designated by the keyword msdos.

- *Minix file system.* Minix is the original UNIX clone that inspired the creation of Linux. Linux started by using the Minix file system, which limits filenames to 14 or 30 characters. The minix keyword identifies this file-system type.

- *Extended file system.* This old Linux file system goes by the keyword ext. You should not use this file system anymore.

- *Second extended file system.* This system is the latest and greatest Linux file system. The keyword ext2 refers to this file-system type. You can use longer filenames in this file system.

Wizard

The installation program creates an ext2 file system on the Linux partition automatically. To create an ext2 file system manually, you have to use the mke2fs command. To use the mke2fs command, you need the number of blocks in the disk partition where you want to create the file system. Use the fdisk program to figure out the partition's size and then use the mke2fs command as follows:

```
mke2fs -c /dev/hda3 405040
```

This command creates an ext2 file system on the /dev/hda3 partition, which contains 308,040 blocks (each block is 1,024 bytes, or 1KB). The -c option forces mke2fs to check for bad blocks (using a fast read-only test) before creating the file system.

Tip

When you install Linux on a hard disk that uses standard IDE, MFM, or RLL controllers, always install the ext2 file system and use the bad-block-checking options when you create the file system.

Specific Disk Problems in Linux

For most hard drives, installing Linux amounts to booting a Linux kernel that supports your system's hard disk controller, partitioning the disk under Linux, and loading the operating system and associated software onto the hard disk. You notice the hard disk only when you create the partitions (as explained in Chapter 1).

Whenever some aspect of the hard disk is out of the ordinary, however, you may run into problems when installing Linux. This section covers some of these problems and suggests solutions.

Windows 95 and LILO

On a PC that has both Linux and Windows 3.1 installed in separate partitions, you can use the Linux Loader (LILO) program to boot one or the other operating system. If you upgrade Windows 3.1 to Windows 95, you'll notice Windows 95 wipes out LILO. Windows 95 overwrites the Master Boot Record with its own program and LILO typically resides on the Master Boot Record. In this case, all you have to do is boot Linux, using a boot floppy (you should have created the boot floppy during Linux installation), and then run `/sbin/lilo` to reinstall LILO.

If you bought a new PC with Windows 95 preinstalled, all you have to do is install Linux by following the steps described in Chapter 1 (start with repartitioning the hard disk). When you come to the step that installs LILO, specify the operating systems and the disk partitions that LILO should configure for booting. When you specify partitions to boot, treat the Windows 95 disk partition as a DOS partition. After you finish installing Linux and LILO, you should be able to boot Windows 95 (assuming you left it installed) or Linux by entering the name of the appropriate boot image at LILO's boot prompt.

Disks with more than 1,024 cylinders

Disks that have more than 1,024 cylinders were a problem in older Linux kernels. The version of Linux on the companion disk, however, should work fine with EIDE disks that have more than 1,024 cylinders, so you can safely ignore the discussions in this section.

You can view a disk as being a collection of sectors, each of which is 512 bytes. The sectors are addressed in two ways:

- Linear Block Address (LBA), in which the sectors are numbered sequentially, starting at zero.

- Cylinder, Head, Sector (CHS) address, which is based on the physical construction of a hard disk, consisting of a stack of magnetic platters that rotate at high speed under read-write heads.

Old PC disk controllers and the Basic Input/Output System (BIOS) use physical CHS addresses to access the sectors on the disk. This arrangement worked fine until large disks started to appear on the market. The BIOS uses a 10-bit value as the cylinder number, which means the BIOS can access, at most, 1,024 cylinders on a hard disk (because a 10-bit value lies between 0 and 1,023). Because of this limitation, any disk that has more than 1,024 cylinders is generally referred to as a *large disk* in the PC world.

Most current hard disks tend to have 16 heads, 63 sectors, and a large number of cylinders. With a 1,024-cylinder limitation, the maximum disk size under BIOS is 16×63×1,024 = 1,0321,192 sectors = 528,482,304 bytes (because each sector is 512 bytes long) = 504MB (1MB = 1,024×1,024 bytes). As you know, many current desktop PCs have 2 or 4GB disks. These disks typically come with the Enhanced IDE (EIDE) interface.

The EIDE interface handles large disks by playing with the number of cylinders and heads. One trick EIDE BIOS uses is to halve the actual number of cylinders and double the number of heads. When the BIOS processes a disk-access request specified by CHS address, it automatically adjusts the cylinders and heads to access the correct sector on the disk. This adjustment is called *address translation*.

Linux gets a disk's geometry (number of cylinders, heads, and sectors) from the BIOS. If the BIOS does address translation, the reported values won't match the actual physical parameters of the disk. Unfortunately, Linux does not go through the BIOS to read from or write to the disk. When accessing the disk controller, Linux has to provide the CHS address with physically correct values. In other words, Linux has to perform address translation like the BIOS. Luckily, the Linux kernel on this book's companion disk already takes care of all the details and handles large disks properly.

You still may run into problems with large disks, however, not because Linux cannot handle large disks, but because on MS-DOS PCs, many older large hard disks are handled by special software that may conflict with Linux. Some large systems come with special software known as Disk Manager. The Disk Manager usually resides in the disk's Master Boot Record and may perform some magic to allow DOS to access the entire disk.

The only clean solution for such problems involves backing up your DOS files. Set the BIOS disk parameters to the correct disk parameters and then partition the disk under Linux. You can set aside the first partition for use under DOS (because DOS won't be able to access more than the first 1,024 cylinders) and create the necessary partitions for Linux beyond the DOS partition.

Because the Linux Loader (LILO) relies on BIOS, which cannot access more than 1,024 cylinders, you should try to create the Linux boot partition within the first 1,024 cylinders.

EIDE problems on PCI systems

The current crop of Pentium PCs use the PCI bus. With this new bus come new interfaces for disks. A common interface is the PCI EIDE controller to connect EIDE devices, such as hard disks and CD-ROM drives, to the PCI motherboard.

When the PCI bus first became popular in 1995, some users reported data corruption in EIDE disks with some PCI EIDE controllers. Specifically, the affected systems have PCI motherboards with PCI EIDE controllers that use one of the following chips:

Secret

■ RZ 1000

■ CMD 640

To find out what type of PCI EIDE controller your system has, use the `cat /proc/pci` command (assuming, of course, your system has a PCI bus).

The version of Linux on the companion CD-ROM detects and works around these problems automatically. The only remaining indication of the problem is when you type `cat /proc/pci`, the listing shows the CMD 640 controller as being buggy. For example, here are the first few lines of output on one of my PCs with the CMD 640 interface:

```
cat /proc/pci
PCI devices found:
  Bus  0, device  13, function  0:
    IDE interface: CMD 640 (buggy) (rev 2).
      Medium devsel.  IRQ 14.
(other lines deleted)
```

Notice how the CMD 640 interface is reported to be buggy.

Error messages about inodes and blocks

If you get error messages about bad inodes or blocks during system startup, chances are you did not shut down the system properly. Before powering off a Linux system, you should always log in as `root` and use the shutdown command to halt the system, as follows:

```
/sbin/shutdown -h now
```

After this, you must wait until you see a message saying the system has halted. Only then you should turn off the power.

Because of our typical DOS experience of simply turning the power switch off, most us are tempted to reach for the power switch to shut down the system (although, somehow, I always remember to shut down other UNIX workstations properly, such as HP and Sun workstations).

If the system is not shut down properly, the file system may be damaged. When you boot the system the next time, it runs a file-system-check program (`fsck`) that may fix the file system and boot the system. In some cases, though, `fsck` won't be able to fix the file-system damage. You have no option but to reinitialize the file system, using the `mke2fs` command.

SCSI Disk Controllers and Linux

The remainder of this chapter explains using SCSI controllers under Linux.

SCSI (pronounced *scuzzy*) is an increasingly popular interface for connecting up to seven different devices on the SCSI bus. Each device, and the SCSI controller, has a unique SCSI identifier (ID) in the range 0 through 7. The controller usually is set to SCSI ID 7; the other devices use numbers between 0 and 6 (this means you can connect up to seven devices to a SCSI controller). Typically, a SCSI hard disk is set to SCSI ID 0.

Linux supports the following SCSI controllers:

Adaptec AHA-1510/152x (ISA)

Adaptec AHA-154x (ISA)

Adaptec AHA-174x (EISA)

Adaptec AHA-274x (EISA)/284x (VLB)

Adaptec AHA 2920

Adaptec AHA-2940/3940 (PCI)

Adaptec AVA-1505/1515 (ISA) (Adaptec 152x-compatible)

Always IN2000

AMI Fast Disk VLB/EISA (BusLogic-compatible)

BusLogic (ISA/EISA/VLB/PCI) (all models)

DPT PM2001 and PM2012A (EATA-PIO)

DPT Smartcache (EATA-DMA) (ISA/EISA/PCI)

DTC 329x (EISA) (Adaptec 154x-compatible)

Future Domain TMC-16x0 and TMC-3260 (PCI)

Future Domain TMC-8xx and TMC-950

ICP-Vortex (PCI/EISA)

Media Vision Pro Audio Spectrum 16 (ISA)

Media Vision Premium 3D

NCR 53c7x0 and 53c8x0 (PCI)

NCR 5380 generic cards

Qlogic FAS408

Quantum ISA-200S and ISA-200MG

Seagate ST-01/ST-02 (ISA)

Sound Blaster 16 SCSI (Adaptec 152x-compatible) (ISA)

Tekram DC-390, DC-390W/U/F

Trantor T128/T128F/T228 (ISA)

Trantor T130B (NCR 5380-compatible)

UltraStor 14F (ISA), 24F (EISA), and 34F (VLB)

Western Digital WD7000 SCSI

Linux currently does not support parallel-port SCSI adapters and non-Adaptec-compatible DTC boards (such as 327x and 328x).

Cross-Reference When you configure the Linux kernel for SCSI support, the configuration program asks a number of questions in regard to SCSI controller cards. Consult Chapter 2 for a listing of the questions about SCSI support. You should answer Yes (type **y**) only to the question that pertains to your make and model of SCSI card.

Cable and termination problems

The SCSI bus needs terminators at both ends to work reliably. A *terminator* is a set of resistors that indicate the end of the SCSI bus. One end is the controller card itself, which typically has the terminator. Each SCSI device has two SCSI connectors, so you can *daisy-chain* (connect one device to the next) several SCSI devices. You are supposed to place a terminator on the last connector on the chain.

Secret Some SCSI controllers — such as Adaptec AHA 154xC, 154xCF, and 274x (*x* is any digit) — are sensitive to the type of cable and terminator you use. If the cables are not perfect or the terminator is not used properly, these SCSI cards may fail intermittently or may not work at all.

To avoid problems with overly sensitive SCSI cards, use cables that come from a reputable vendor and use cables from the same vendor to connect all SCSI devices. The cables should be SCSI-2-compliant and should have an impedance of 132 ohms (a characteristic of the cable; all you have to do is make sure the specified value is 132 ohms).

Adaptec AHA151x, AHA151x, and Sound Blaster 16 SCSI

These ISA-bus SCSI cards include all SCSI controllers based on the AIC 6260 or 6360 chipset. Typical hardware parameters for these controllers include the following:

BIOS addresses: 0xd8000, 0xdc000, 0xd0000, 0xd4000, 0xc8000, 0xcc000, 0xe0000, and 0xe4000

I/O ports: 0x140 and 0x340

IRQs: 9, 10, 11, and 12

DMA channels: not used

Autoprobe works with boards that have a BIOS installed. For other boards, such as Adaptec 1510 and Sound Blaster 16 SCSI, use the boot option in this format:

```
aha152x=IOPORT,IRQ,SCSI-ID,RECONNECT
```

All right-side arguments are numbers. *IOPORT* is the I/O address, and if *RECONNECT* is nonzero, the driver is allowed to disconnect and reconnect the device. Usually, the *SCSI-ID* is 7, and *RECONNECT* is specified as 1.

Usually, *SCSI-ID* is 7, and *RECONNECT* is nonzero. If an Adaptec AHA1510 card has I/O address 0x340 and IRQ 11, the boot option is the following:

```
aha152x=0x340,11,7,1
```

Adaptec AHA154x, AMI FastDisk VLB, BusLogic, and DTC 329x

Typical hardware parameters of these ISA-bus SCSI controllers include the following:

> I/O ports: 0x330 and 0x334
>
> IRQs: 9, 10, 11, 12, 14, and 15
>
> DMA channels: 5, 6, and 7

Autoprobe works with these controllers; there is no need for a BIOS on the controller. The BusLogic SCSI controllers are software-compatible with the Adaptec 1542. ISA, VLB, and EISA versions of BusLogic cards are available.

Adaptec AHA154xC and AHA154xCF controller cards often generate unexpected errors; these controllers are sensitive to the cable and termination details.

If you encounter infinite timeout errors Adaptec AHA154xC and 154xCF controllers, you may have to run the Adaptec setup program (which you do by pressing a specified key during power up) and enable synchronous negotiation.

Adaptec AHA 174x

This controller is an EISA-bus SCSI controller Adaptec no longer sells. Older EISA bus systems may have this card. Following are the hardware parameters of the AHA174x card:

> Bus slots: 1–8
>
> I/O ports: EISA bus does not require preassigned I/O ports
>
> IRQs: 9, 10, 11, 12, 14, and 15
>
> DMA channels: EISA bus does not require preassigned I/O ports

The Linux driver can detect the card automatically without any problems. The driver also expects the card to be running in enhanced mode, as opposed to standard AHA1542 mode.

Adaptec AHA274x, AHA284x, and AHA294x

The Adaptec AHA274x controller is an EISA bus card; AHA284x is a VLB card; AHA294x is a new PCI bus card. The AHA274x driver supports all three cards. You should enable the BIOS on these controller cards.

For the PCI controller to work, you must answer Yes to the following question during kernel configuration:

```
PCI bios support (CONFIG_PCI) [Y/n/?] y
```

Allways IN2000

The Allways IN2000 is an ISA-bus controller card with the following parameters:

I/O ports: 0x100, 0x110, 0x200, and 0x220

IRQs: 10, 11, 14, and 15

DMA: not used

The driver can detect the card automatically without any need for BIOS.

EATA DPT Smartcache

The Linux `eata_dma` SCSI driver supports all SCSI controllers that support the EATA-DMA protocol. The controllers include DPT PM2011, PM2012A, PM2012B, PM2021, PM2022, PM2024, PM2122, PM2124, PM2322, PM3021, PM3222, and PM3224.

The driver's autoprobe function works with all supported DPT cards. A common problem, however, is the IDE driver detects the ST-506 interface of the EATA controller. If the IDE driver has a problem with the detected parameters and fails, you won't be able to access your IDE hardware. In this case, you should change the EATA board's parameters, such as the I/O address and the IRQ. In particular, don't use IRQs of 14 or 15, which are the IRQs of the primary and secondary IDE interfaces.

If you have a PCI controller such as DPT PM2024, PM2124, or PM3224, remember to enable PCI BIOS when you configure the kernel.

Future Domain 16x0

The Future Domain 16x0 SCSI controller uses the TMC-1800, TMC-18C30, TMC-8C50, or TMC-36C70 chip. These ISA-bus cards typically have the following configurations:

BIOS addresses: 0xc8000, 0xca000, 0xce000, and 0xde000

I/O ports: 0x140, 0x150, 0x160, and 0x170

IRQs: 3, 5, 10, 11, 12, 14, and 15

DMA: not used

The driver can probe and detect the hardware automatically, provided the controller has a BIOS installed.

NCR53c8xx SCSI Chip (PCI)

NCR53C8xx refers to NCR53c810, NCR53c815, NCR53c820, and NCR53c825 — a series of low-cost SCSI chips for PCI motherboards. The version of Linux on the companion CD-ROM supports the NCR53c8xx. The driver can detect SCSI devices automatically, provided the PCI BIOS is present. In fact, the driver needs the BIOS because it uses BIOS-initialized values in the registers of the NCR53c8xx.

A reported problem with the NCR53c8xx is the chip works under DOS but fails under Linux, because it times out on a test due to a lost interrupt. A typical cause of this error is a mismatch between the IRQ setting in the hardware (typically set with a jumper) and the IRQ value stored in the CMOS setup (the setup program of your PC, the one you can run during power-up). To correct the problem, check the following things:

- Make sure the hardware IRQ setting matches that in the CMOS setup.

- If the NCR 53c8xx is on a board that has jumpers for selecting PCI interrupt lines (PCI has interrupt lines INTA, INTB, INTC, and INTD), make sure only INTA is being used.

- If the PCI board has jumpers for selecting level-sensitive and edge-triggered interrupts, make sure the board is using "level-sensitive" interrupts.

Another reported problem is system lockup when an S3 928 or Tseng ET4000/W32 PCI video chipset is used, due to problems in the video chipsets.

On a system that has an NCR53c8xx SCSI chip, you may encounter the following message:

```
scsi%d: IRQ0 not free, detaching
```

This message indicates the PCI configuration register contains a zero. The reason may be a mismatch between the hardware IRQ and the value in CMOS, or a defective BIOS.

Because the NCR53c8xx is a PCI device, you must enable PCI support when you rebuild the Linux kernel.

Seagate ST0x and Future Domain TMC-8xx and TMC-9xx

Typical hardware parameters of these ISA-bus SCSI controllers include the following:

> BIOS addresses: 0xc8000, 0xca000, 0xcc000, 0xce000, 0xdc000, and 0xde000

> IRQs: 3 and 5

> DMA channels: not used

When it tries to probe for the controller automatically, the driver probes only the BIOS addresses; it assumes the IRQ is 5. Also, autoprobe works only if a BIOS is installed.

During boot, you can provide one of the following command lines to force detection of the controller:

```
st0x=BIOS-ADDRESS,IRQ
tmc8xx=BIOS-ADDRESS,IRQ
```

BIOS_ADDRESS is the BIOS address of the board, and *IRQ* is the interrupt-request channel.

Secret

Common problems with the ST01 or ST02 SCSI controller are time-outs when Linux accesses the disk connected to the controller, because the board's default settings disable interrupts. You should set jumpers (W3 on ST01 and JP3 on ST02) on the board to reenable interrupts. You also should select IRQ 5.

Secret

If you get errors when you try to run `fdisk` on a drive connected to Seagate or Future Domain controllers, you should use `fdisk`'s extra functions menu to specify the disk geometry (cylinders, heads, and sectors).

Pro Audio Spectrum PAS16 SCSI

The PAS16 SCSI refers to the SCSI interface on a Pro Audio Spectrum sound card. Following are the hardware-configuration parameters for the PAS16 SCSI:

> I/O ports: 0x388, 0x384, 0x38x, and 0x288

> IRQs: 10, 12, 14, and 15 (must be different from the IRQs used for sound)

> DMA: not used for the SCSI portion of the card

The autoprobe function does not require BIOS. You can specify a command line at the boot prompt to specify the parameters of your PAS 16 SCSI. For a PAS 16 SCSI at I/O address 0x388 and IRQ 10, for example, use the following command line:

`pas16=0x388.10`

Trantor T128, T128F, and T228

These Trantor ISA-bus SCSI cards have the following configuration parameters:

> BIOS addresses: 0xcc000, 00xc8000, 0xdc000, and 0xd8000
>
> IRQs: on all boards, none, 3, 5, and 7; on T128F, 10, 12, 14, and 15
>
> DMA: not used

The driver can autoprobe as long as a BIOS is installed. If one of these SCSI controllers does not have BIOS, or if the BIOS is disabled, you can specify the controller through a command line like the following:

`t128=BIOS-ADDRESS.IRQ`

`BIOS-ADDRESS` is the base address (not I/O address). For a controller with a BIOS address 0xcc000 and IRQ 5, for example, the command line is:

`t128=0xcc000.5`

Use -1 for the IRQ if a controller does not have an IRQ; use -2 to make the driver probe the IRQ.

Ultrastor 14f (ISA), 24f (EISA), and 34f (VLB)

The Ultrastor SCSI cards have the following configuration parameters:

> I/O ports: 0x130, 0x140, 0x210, 0x230, 0x240, 0x310, 0x330, and 0x340
>
> IRQs: 10, 11, 14, and 15
>
> DMA channels: 5, 6, and 7

The autoprobe function works in all cases except when the I/O port address is 0x310. Because I/O port 0x310 is not supported by the autoprobe code, you should select a different I/O port address for the Ultrastor controller.

If you have a sound card, I/O port address 0x330 typically is used by the MIDI device. Use a different I/O port address for the Ultrastor card if you have a sound card in your PC. A good I/O port for the Ultrastor cards is 0x340.

The Ultrastor controllers support a WD1003 emulation mode, in which they can work with ST-506-interface disk drives. If you have your Ultrastor controller in WD1003 mode, the Ultrastor SCSI driver will fail, displaying the following error message:

```
hd.c: ST-506 interface disk with more than 16 heads detected,
   probably due to non-standard sector translation.  Giving up.
   (disk %d: cyl=%d, sect=63, head=64)
```

You can fix this problem by setting the Ultrastor controller to its native SCSI mode.

Western Digital 7000

The hardware configurations for this ISA-bus SCSI controller are as follows:

> BIOS address: 0xce000
>
> I/O port: 0x350
>
> IRQ: 15
>
> DMA channel: 6

The driver can probe and detect the SCSI controller automatically, provided the BIOS is installed.

Some revisions of the Western Digital 7000 controller may not work with the driver. Reportedly, Revision 5 and later controllers work fine. Also, on the working controllers, the onboard SCSI chip should have an A suffix.

Iomega Zip drive (SCSI)

The Iomega Zip drive is a low-cost, removable disk drive that enables you to use floppy-like disks, each of which is capable of holding 100 million characters. If you consider a megabyte to be 1,024×1,024 = 1,048,576 bytes, 100 million characters will be about 95MB. Unlike floppies, the Zip disks do not have a hardware write-protect tab, which means you must be careful when you initialize a disk or delete files.

The Zip drive comes in three versions: SCSI interface for Macintosh or MS-DOS and a parallel-port version. The current Linux kernel supports the parallel-port version of the Zip drive.

Both the DOS and Macintosh versions of the SCSI Iomega Zip drive should work under Linux. You can use an existing Linux-supported SCSI card (preferably, a simple Adaptec AHA152x-compatible card) to connect the Zip drive to your PC. The Zip drive has switches to turn on termination (if it's the last device in the chain) and select SCSI ID (one of 5 or 6).

You also can connect the Zip drive to a separately sold Zip Zoom interface card — a low-end SCSI card compatible with the Adaptec AHA152x card. If you have a kernel that supports the AHA152x and you use LILO, you can use the Zip drive with the following line added to the /etc/lilo.conf file:

```
append="aha152x=0x340,11,7,1"
```

To prepare the Zip disk for use under Linux, you should try it under DOS. Run the \SCSI\INSTALL program on the Iomega Tools disk. After that, the Zip disk should work under Linux.

Log in as `root`, and run `/sbin/fdisk` on the SCSI device that represents the Zip drive. If the Zip drive is the only SCSI device, it'll be `/dev/sda`, so you type the following:

```
/sbin/fdisk /dev/sda
```

Check the current partition and set the type to Linux native. Then create an `ext2` file system with the following command:

```
/sbin/mke2fs /dev/sda1
```

This command assumes you are using the first partition of the Zip disk. Create the a `zip` subdirectory in `/mnt` with the `mkdir /mnt/zip` command. You then can mount the Zip disk at an appropriate mount point and use the disk, as follows:

```
mount -text2 /dev/sda1 /mnt/zip
```

The contents of the Zip disk will be in the `/mnt/zip` directory. Before ejecting the disk, use the command `umount /mnt/zip` to dismount the Zip disk.

SCSI troubleshooting

Most SCSI problems are due to bad cables or improper termination. You should check the cables and the terminator before you try anything else. The following sections list other common SCSI problems and their suggested fixes:

Problem booting with LILO

When booting from a SCSI hard disk, LILO may hang after displaying the letters `LI`. This problem occurs if the SCSI controller's BIOS and the Linux SCSI driver interpret the disk geometry differently.

To fix this problem, add the `linear` keyword on a single line in the `/etc/lilo.conf` file. This keyword causes LILO to use Linear Block Addresses (LBA) instead of the physical cylinder-head-sector (CHS) addresses when it accesses the disk. LILO uses the disk geometry supplied by the BIOS and compute physical addresses at run time, which should work properly.

SCSI device at all SCSI IDs

If a SCSI device shows up at all possible SCSI IDs, you must have configured that device with the same SCSI ID as the SCSI controller (usually, 7). Change the ID of that device to another value. (Many devices have a simple switch for setting the SCSI ID; on many other devices, you have to change a jumper.)

SCSI device at all LUNs

If a SCSI device shows up at all possible SCSI Logical Unit Numbers (LUNs), the device probably has errors in the *firmware*—the built-in code in the device's SCSI interface. To verify these errors, first use the following command line during boot:

```
max_scsi_luns=1
```

If the device works with this option, you can add it to the list of blacklisted SCSI devices in the array of structures named `device_list` in the file `/usr/src/linux/drivers/scsi/scsi.c`. The definition of this structure and the current contents of the array are as follows:

```
struct dev_info{
    const char * vendor;
    const char * model;
    const char * revision; /* Latest revision known to be bad.  Not used yet */
    unsigned flags;
};


/*
 * This is what was previously known as the blacklist.  The concept
 * has been expanded so that we can specify other types of things we
 * need to be aware of.
 */
static struct dev_info device_list[] =
{
{"CHINON","CD-ROM CDS-431","H42", BLIST_NOLUN}, /* Locks up if polled for lun
!= 0 */
{"CHINON","CD-ROM CDS-535","Q14", BLIST_NOLUN}, /* Locks up if polled for lun
!= 0 */
{"DENON","DRD-25X","V", BLIST_NOLUN},           /* Locks up if probed for lun
!= 0 */
{"HITACHI","DK312C","CM81", BLIST_NOLUN},        /* Responds to all lun - dtg */
{"HITACHI","DK314C","CR21" , BLIST_NOLUN},       /* responds to all lun */
{"IMS", "CDD521/10","2.06", BLIST_NOLUN},        /* Locks-up when LUN>0 polled.
*/
{"MAXTOR","XT-3280","PR02", BLIST_NOLUN},        /* Locks-up when LUN>0 polled.
*/
{"MAXTOR","XT-4380S","B3C", BLIST_NOLUN},        /* Locks-up when LUN>0 polled.
*/
{"MAXTOR","MXT-1240S","I1.2", BLIST_NOLUN},      /* Locks up when LUN>0 polled
*/
{"MAXTOR","XT-4170S","B5A", BLIST_NOLUN},        /* Locks-up sometimes when
LUN>0 polled. */
{"MAXTOR","XT-8760S","B7B", BLIST_NOLUN},        /* guess what? */
{"MEDIAVIS","RENO CD-ROMX2A","2.03",BLIST_NOLUN},/*Responds to all lun */
{"MICROP", "4110", "*", BLIST_NOTQ},     /* Buggy Tagged Queuing */
{"NEC","CD-ROM DRIVE:841","1.0", BLIST_NOLUN}, /* Locks-up when LUN>0 polled.
*/
{"RODIME","RO3000S","2.33", BLIST_NOLUN},        /* Locks up if polled for lun
!= 0 */
{"SANYO", "CRD-250S", "1.20", BLIST_NOLUN},      /* causes failed REQUEST SENSE
on lun 1
                * for aha152x controller, which causes
```

```
             * SCSI code to reset bus.*/
{"SEAGATE", "ST157N", "\004|j", BLIST_NOLUN},   /* causes failed REQUEST SENSE
on lun 1
             * for aha152x controller, which causes
             * SCSI code to reset bus.*/
{"SEAGATE", "ST296","921", BLIST_NOLUN},        /* Responds to all lun */
{"SEAGATE","ST1581","6538",BLIST_NOLUN},   /* Responds to all lun */
{"SONY","CD-ROM CDU-541","4.3d", BLIST_NOLUN},
{"SONY","CD-ROM CDU-55S","1.0i", BLIST_NOLUN},
{"SONY","CD-ROM CDU-561","1.7x", BLIST_NOLUN},
{"TANDBERG","TDC 3600","U07", BLIST_NOLUN},     /* Locks up if polled for lun
!= 0 */
{"TEAC","CD-ROM","1.06", BLIST_NOLUN},          /* causes failed REQUEST SENSE
on lun 1
             * for seagate controller, which causes
             * SCSI code to reset bus.*/
{"TEXEL","CD-ROM","1.06", BLIST_NOLUN},         /* causes failed REQUEST SENSE
on lun 1
             * for seagate controller, which causes
             * SCSI code to reset bus.*/
{"QUANTUM","LPS525S","3110", BLIST_NOLUN},      /* Locks sometimes if polled
for lun != 0 */
{"QUANTUM","PD1225S","3110", BLIST_NOLUN},      /* Locks sometimes if polled
for lun != 0 */
{"MEDIAVIS","CDR-H93MV","1.31", BLIST_NOLUN},   /* Locks up if polled for lun
!= 0 */
{"SANKYO", "CP525","6.64", BLIST_NOLUN},        /* causes failed REQ SENSE,
extra reset */
{"HP", "C1750A", "3226", BLIST_NOLUN},          /* scanjet iic */
{"HP", "C1790A", "", BLIST_NOLUN},              /* scanjet iip */
{"HP", "C2500A", "", BLIST_NOLUN},              /* scanjet iicx */

/*
 * Other types of devices that have special flags.
 */
{"SONY","CD-ROM CDU-8001","*", BLIST_BORKEN},
{"TEXEL","CD-ROM","1.06", BLIST_BORKEN},
{"IOMEGA","Io20S        *F","*", BLIST_KEY},
{"INSITE","Floptical   F*8I","*", BLIST_KEY},
{"INSITE","I325VM","*", BLIST_KEY},
{"NRC","MBR-7","*", BLIST_FORCELUN | BLIST_SINGLELUN},
{"NRC","MBR-7.4","*", BLIST_FORCELUN | BLIST_SINGLELUN},
{"NAKAMICH","MJ-4.8S","*", BLIST_FORCELUN | BLIST_SINGLELUN},
{"PIONEER","CD-ROM DRM-602X","*", BLIST_FORCELUN | BLIST_SINGLELUN},
{"PIONEER","CD-ROM DRM-604X","*", BLIST_FORCELUN | BLIST_SINGLELUN},
{"EMULEX","MD21/S2     ESDI","*", BLIST_SINGLELUN},
{"CANON","IPUBJD","*", BLIST_SPARSELUN},
{"MATSHITA","PD","*", BLIST_FORCELUN | BLIST_SINGLELUN},
{"YAMAHA","CDR100","1.00", BLIST_NOLUN},  /* Locks up if polled for lun != 0 */
{"YAMAHA","CDR102","1.00", BLIST_NOLUN},  /* Locks up if polled for lun != 0 */
{"nCipher","Fastness Crypto","*", BLIST_FORCELUN},
/*
 * Must be at end of list...
 */
{NULL, NULL, NULL}};
```

From this list, you get an idea of the types of SCSI devices with the problem of showing up at all LUNs. If you have the same problem with a SCSI device, you can add that device's name to this list before the last line. You may not want to do this, however, if you are unfamiliar with the C programming language.

Sense errors on error-free SCSI device

The cause of this problem usually is bad cables or improper termination. Check all cables and make sure the SCSI bus is terminated at both ends.

Networking kernel problems with SCSI device

If a Linux kernel with networking support does not work with SCSI devices, the problem may be the autoprobe function of the networking drivers. The *autoprobe capability* is meant to detect the type of networking hardware automatically. The network drivers read from and write to specific I/O addresses during autoprobing. If an I/O addresses is the same as that used by a SCSI device, the system may have a problem. In this case, you have to check the I/O address, IRQ, and DMA values of the network cards and of the SCSI controller, and make sure no conflicts exist. Most SCSI controllers (and even network adapters) enable you to configure these parameters (I/O address, IRQ, and DMA) through setup software that comes with the adapter.

Device detected but not accessible

If the kernel detects a SCSI device (as reported in the boot messages, which you can see with `dmesg | more`) but you cannot access the device, the device file is missing from the `/dev` directory.

To add the device file, log in as `root`, change the directory to `/dev`, and then use the MAKEDEV script to create the device file. To add a device for a SCSI tape drive, for example, you would use the following command:

```
cd /dev
./MAKEDEV st0
```

When you log in as `root`, the current directory is typically not in the PATH environment variable. This is why you need to add the `./` prefix when executing the MAKEDEV script. You can find more information about the MAKEDEV script with this command:

```
man MAKEDEV
```

SCSI lockup

If the SCSI system locks up, check the SCSI controller card, using any diagnostic software that came with the card (usually, the diagnostic software runs under DOS). Look for conflicts in I/O address, IRQ, or DMA with other cards. Some sound cards, for example, use a 16-bit DMA channel in addition

to an 8-bit DMA; make sure you did not inadvertently use the same 16-bit DMA for the SCSI card.

Secret

The Linux SCSI driver for some SCSI cards supports only one outstanding SCSI command at a time. With such a SCSI card, if a device such as a tape drive is busy rewinding, the system may be unable to access other SCSI devices (such as a hard disk or a CD-ROM drive) that are daisy-chained with that tape drive. A solution to this problem is to add a second SCSI controller to take care of the tape drives.

SCSI devices not found

If the Linux kernel does not detect your SCSI devices at startup, you get the following message when Linux boots:

```
scsi : 0 hosts
```

If you see this message, but you know the SCSI devices are there (and they work under DOS), the problem may be the lack of a BIOS on the SCSI controller; the autoprobe routines that detect SCSI devices rely on the BIOS.

This problem occurs for the following SCSI cards:

- Adaptec 152x, 151x, AIC-6260, and AIC-6360

- Future Domain 1680, TMC-950, and TMC-8xx

- Trantor T128, T128F, and T228F

- Seagate ST01 and ST02

- Western Digital 7000

Secret

Even if a SCSI controller has a BIOS, jumpers often are available for disabling the BIOS. If you disabled the BIOS for some reason, you may want to re-enable it (read the documentation of your SCSI controller for directions) so Linux can detect the SCSI devices automatically.

For a SCSI card, such as the Adaptec 151x, that does not have any BIOS, use the following command line during boot to force detection of the card:

```
aha152x=0x340,11,7,1
```

Summary

You need a hard disk to install Linux on your PC. In particular, to install Linux successfully, your PC's hard disk controller must be supported by Linux. This should not be a problem because Linux supports the popular IDE and Enhanced IDE interfaces, as well as the SCSI interface. This chapter describes how to install and use hard disks in Linux. By reading this chapter, you learn the following things:

▷ Linux's support for a specific hard disk drive depends on the disk controller used to connect that drive to the PC's motherboard. Linux supports the popular IDE (Integrated Drive Electronics), Enhanced IDE (EIDE), and SCSI disk controllers for ISA, EISA, VLB, and PCI buses. These disk controllers cover nearly all types of disks that PCs typically use.

▷ You must partition a disk to install Linux and when you install Linux, you must create a swap space and set up a mechanism to boot Linux. You can use LILO to manage the boot process, even if you have multiple operating systems (such as DOS and Windows 95) on the disk.

▷ The disk controller sees the disk as being a collection of 512-byte sectors on magnetic platters mounted on a spindle and rotating under read-write heads. This physical construction of a hard disk gives rise to the view of a disk as a collection of cylinders, heads, and sectors (CHS). Linux likes to see the disk as being a sequence of sectors that can be addressed sequentially (Linear Block Address, or LBA).

▷ The PC's BIOS uses CHS addressing and limits the cylinder address to a 10-bit value, thus giving rise to a 1,024-cylinder limit on hard disk geometries. Because hardware constraints prevent disks from having more than 16 heads and 63 sectors, the BIOS can handle disks up to 504MB (1,024×16×63 sectors; each sector is 512 bytes). Newer large disks use some tricks to handle larger disks, but these tricks sometimes conflict with the way Linux addresses disks. You generally are safe if you keep the DOS settings for the disk parameters such as number of cylinders, heads, and sectors.

▷ SCSI is popular because you can connect up to seven devices through a single SCSI controller. The only drawback is the SCSI controller is relatively expensive compared with EIDE controllers.

▷ Linux supports a wide variety of SCSI controllers, including the popular Adaptec and BusLogic SCSI controllers.

▷ Many recent PCs have the PCI bus and PCI-bus EIDE controller chips built into the motherboard. Two of these EIDE controllers — RZ 1000 and CMD 640 — are reported to have some bugs that occur when you have multiple IDE devices on the EIDE controller. Linux can work around these bugs.

▷ With a little effort, you should be able to use an Iomega Zip drive with Linux.

Appendixes

Appendix B

Linux Commands

This appendix presents an alphabetically arranged reference of the most important Linux commands. The goal is to provide you with an overview of all commands needed to manage files and directories, start and stop processes, find files, work with text files, and access online help.

If you are looking for a command for a specific task, but don't know which command to use, you may find it helpful to browse through the commands by category. Table B-1 shows the Linux commands organized by category.

Table B-1   Linux commands grouped by category

Command Name	Action
Getting online help	
apropos	Find man pages for a specified keyword
info	Display online help information about a specified command
man	Display online help information
whatis	Similar to apropos, but searches for complete words only
Making commands easier	
alias	Define an abbreviation for a long command
type	Show the type and location of a command
unalias	Delete an abbreviation defined using alias
Managing files and directories	
cd	Change current directory
chmod	Change file permissions
chown	Change file owner and group
cp	Copy files
ln	Create symbolic links to files and directories
ls	Display the contents of a directory
mkdir	Create a directory
mv	Rename file as well as move file from one directory to another
rm	Delete files
rmdir	Delete directories

(continued)

Table B-1 *(Continued)*

Command Name	Action
Managing files and directories	
pwd	Display the current directory
touch	Update a file's time stamp
Finding files	
find	Find files based on specified criteria such as name, size, and so on.
locate	Find files using a periodically updated database
whereis	Find files based in the typical directories where executable (also known as binary) files are located
which	Find files in the directories listed in the PATH environment variable
Processing files	
cat	Display a file on standard output (can be used to concatenate several files into one big file)
cut	Extract specified sections from each line of text in a file
dd	Copy blocks of data from one file to another (used to copy data from devices)
diff	Compare two text files and finds any differences
expand	Convert all tabs into spaces
file	Display the type of data in a file
fold	Wrap each line of text to fit a specified width
grep	Search for regular expressions within a text file
less	Display a text file, one page at a time (can go backward also)
lpr	Print files
more	Display a text file, one page at a time (goes forward only)
nl	Number all nonblank lines in a text file and print the lines to standard output
paste	Concatenate corresponding lines from several files
patch	Update a text file using the differences between the original and revised copy of the file
sed	Copy a file to standard output while applying specified editing commands
sort	Sort lines in a text file

Processing files

split	Break up a file into several smaller files with specified size
tac	Reverse a file (last line first, and so on)
tail	Display last few lines of a file
tr	Substitute one group of characters for another throughout a file
uniq	Eliminate duplicate lines from a text file
wc	Count the number of lines, words, and characters in a text file
zcat	Display a compressed file (after decompressing)
zless	Display a compressed file one page at a time (can go backward also)
zmore	Display a compressed file one page at a time

Archiving and compressing files

compress	Compress files
cpio	Copy files to and from an archive
gunzip	Uncompress files compressed with GNU ZIP (gzip) or compress
gzip	Compress files (more powerful than compress)
tar	Create an archive of files in one or more directories (originally meant for archiving on tape)
uncompress	Uncompress files compressed with compress

Managing processes

bg	Run an interrupted process in the background
fg	Run a process in the foreground
free	Display amount of free and used memory in the system
halt	Shut down Linux and halts the computer
kill	Send a signal to a process (usually used to terminate a process)
ldd	Display the shared libraries needed to run a program
nice	Run a process with lower priority (referred to as nice mode)
ps	Display list of currently running processes
printenv	Display the current environment variables
pstree	Similar to ps, but shows parent-child relationships clearly
reboot	Stop Linux and then restarts the computer
shutdown	Shut down Linux
top	Display list of most processor- and memory-intensive processes
uname	Display information about the system and the Linux kernel

(continued)

Table B-1   *(Continued)*

Command Name	Action
Managing users	
chsh	Change the shell (command interpreter)
groups	Print the list of groups that includes a specified user
id	Display the user and group ID for a specified user name
passwd	Change the password
su	Become another user or root (when invoked without any argument)
Managing the file system	
df	Summarize free and available space in all mounted storage devices
du	Display disk usage information
fdformat	Format a diskette
fdisk	Partition a hard disk
fsck	Check and repair a file system
mkfs	Create a new file system
mknod	Create a device file
mkswap	Create swap space for Linux in a file or a disk partition
mount	Mount a device (for example, the CD-ROM) on a directory in the file system
swapoff	Deactivate a swap space
swapon	Activate a swap space
sync	Write buffered data to files
tty	Display the device name for the current terminal
umount	Unmount a device from the file system
Working with the date and time	
cal	Display a calendar for a specified month or year
date	Show current date and time or set a new date and time

The rest of this appendix presents individual reference entries for each command shown in Table B-1. Each reference entry has a standard appearance with the following sections:

■ *Purpose*: Tells you when to use the command.

■ *Syntax*: Shows the syntax of the command with a few common options. Typical option values are also shown. All optional items are shown in square brackets.

■ *Options*: Lists most options, along with a brief description of each option. For many commands, you will find all options listed in this section. However, some commands have too many options to list. For those commands, I show the most commonly used options.

■ *Description*: Describes the command and provides more details about how to use the command.

alias	
Purpose	Define an abbreviation for a long command or view the current list of abbreviations
Syntax	`alias [abbrev=command]`
Options	None
Description	If you type `alias` alone, you get a listing of all currently defined abbreviations. Typically, you use `alias` to define easy-to-remember abbreviations for longer commands. For example, if you type `ls -l` often, you might add a line with `alias ll='ls -l'` in the `.bashrc` file in your home directory. Then you can type `ll` instead of `ls -l` to see a detailed listing of a directory. `alias` is a built-in command of the Bash shell and is described in Chapter 6.

apropos	
Purpose	View a list of all man pages containing a specific keyword
Syntax	`apropos keyword`
Options	None
Description	The `apropos` command looks up the keyword in a database (known as the `whatis` database) created by the `/usr/sbin/makewhatis` program. The `whatis` database is an index of keywords contained in all the man pages in the system. Unfortunately, when you try `apropos` with a simple keyword such as *find,* you may end up with a long listing of man pages because the word *find* appears in many man pages.

bg	
Purpose	Run an interrupted process in the background
Syntax	`bg`
Options	None
Description	After you type a command that takes a long time to finish, you can press Ctrl+Z to interrupt the process. Then, you can type `bg` to continue that command in the background while you type other commands at the shell prompt. `bg` is a built-in command of the Bash shell.

---

cal

Purpose	View the calendar of any month in any year
Syntax	`cal` `cal [-jy] [[month_number] year]]`
Options	`-j` displays Julian dates (the day number between 1 and 366) `-y` displays the calendar for all months of the current year

Description   If you type `cal` without any options, it prints a calendar for the current month. If you type `cal` followed by a number, `cal` treats the number as the year and prints the calendar for that year. To view the calendar for a specific month in a specific year, provide the month number (1 = January, 2 = February, and so on) followed by the year. Thus, to view the calendar for January 2000 type the following:

cal 1 2000
```
   January 2000
Su Mo Tu We Th Fr Sa
               1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

---

cat

Purpose	Copy contents of a file to standard output (the screen)
Syntax	`cat [-benstvA] files`
Options	`-b` numbers nonblank lines `-e` shows end of line (as $) and all nonprinting characters `-n` numbers all output lines starting with number 1 `-s` replaces multiple blank lines with a single blank line `-t` shows tabs as ^I `-v` shows nonprinting characters `-A` shows all characters (including nonprinting ones)
Description	Typically, `cat` is used to display the contents of a file or to concatenate several files into a single file. For example, `cat file1 file2 file3 > all` combines three files into a single file named `all`.

---

cd

Purpose	Change current directory
Syntax	`cd [directory]`
Options	None
Description	Typing `cd` without a directory name changes the current directory to your home directory. Otherwise, `cd` changes to the specified directory. `cd` is a built-in command of the Bash shell.

chmod	
Purpose	Change the permission settings of one or more files
Syntax	`chmod [-cfvR]` *`permission files`*
Options	`-c` lists only files whose permissions changed `-f` stops any error message displays `-v` verbosely displays permission changes `-R` recursively changes permission of files in all subdirectories
Description	To use `chmod` effectively, you have to learn how to specify the permission settings. One way is to concatenate one letter from each of the following tables in the order shown (*who, action, permission*):

Who		Action		Permission	
u	user	+	add	r	read
g	group	-	remove	w	write
o	others	=	assign	x	execute
a	all			s	set user ID

To give everyone read access to all files in a directory, you would type `chmod a+r *`. On the other hand, to make a specific file executable by everyone, you would type `chmod +x filename`.
Another way to specify permission settings is to use a three-digit sequence of octal numbers. In a detailed listing, the read, write, and execute permission settings for the user, group, and others appear as the sequence `rwxrwxrwx` (with dashes in place of letters for disallowed operations). Think of `rwxrwxrwx` as three occurrences of the string `rwx`. Now, assign the values $r=4$, $w=2$, and $x=1$. To get the value of the sequence `rwx`, simply add the values of `r`, `w`, and `x`. Thus, `rwx` $= 7$. Using this formula, you can assign a three-digit value to any permission setting. For example, if the user can read and write the file and everyone else can only read the file, the permission setting is `rw-r—r—` (that's how it appears in the listing) and the value is 644. Thus, if you wanted all files in a directory to be readable by everyone, but writable by only the user, you would use the command `chmod 644 *`.

chown	
Purpose	Change the user and group ownership of a file
Syntax	`chown [cvfR]` *`username.groupname files`*
Options	`-c` lists only files whose ownership changed `-f` stops any error message displays `-v` verbosely displays ownership changes `-R` recursively changes ownership of files in all subdirectories
Description	To make a user the owner of one or more files, invoke `chown` with the user name followed by the file names. To change the group ownership as well, append the new group name to the user name with a period as separator. For example, to make user `naba` the owner of all files in a directory, I type `chown naba *`. Note that you have to be logged in as `root` to change the ownership of files.

chsh	
Purpose	Change the default shell that is started at login
Syntax	`chsh [-s shell] [username]`
Options	`-s` shell specifies the name of the shell executable to use (shell can be any program listed in the `/etc/shells` file, such as `/bin/bash` and `/bin/csh`)
Description	A user's default shell is stored in the `/etc/passwd` file. The `chsh` command lets you change the default shell to any of the shells listed in the `/etc/shells` file. If you type `chsh` without any arguments, `chsh` prompts you for the name of a shell.

compress	
Purpose	Compress one or more files using Lempel-Ziv compression
Syntax	`compress [-cdrvV] files`
Options	`-c` writes compressed file to the standard output and retains original `-d` decompresses the file `-r` recursively compresses files in all subdirectories `-v` displays a message as each file is compressed `-V` prints version number and exits
Description	The `compress` command compresses each specified file and replaces the original with the compressed version (with a `.Z` suffix appended to the name). You can uncompress the file with the `compress -d` command or the `uncompress` command.

cp	
Purpose	Copy files and directories
Syntax	`cp [options] source_files destination_directory` `cp [options] source_file destination_file`
Options	`-a` preserves all file attributes `-b` makes backup copy before copying `-d` copies a link but not the file pointed to by the link `-i` asks for confirmation before overwriting files `-l` creates hard links instead of copying files `-p` preserves ownership, permissions, and file time stamp `-R` recursively copies files in all subdirectories `-s` creates soft links instead of copying files `-u` copies only when the file being copied is newer than the destination `-v` displays verbose messages as copying progresses `—help` displays a help message about `cp`
Description	The `cp` command copies one file to another. You can also copy several files from one directory to another.

cpio	
Purpose	Copy files in from or out to an archive that can be on a storage medium such as tape or a file on the disk

Syntax	```cpio [-icdv] pattern```
	```cpio [-ocBv] pattern```
	```cpio [-padm] pattern```

Options	-i extracts files whose names match the *pattern* -o copies to archive files whose names are provided on standard input -p copies files to another directory on the same system -a resets access times of input files -B copies using 5,120 bytes per record (default is 512 bytes per record) -c reads or writes header information as ASCII characters -d creates directories as needed -m retains previous file modification time -v prints a list of file names
Description	The cpio command copies files in from and out to archives. There are three distinct variants of the cpio command: cpio -o creates an archive; cpio -i extracts from an archive; and cpio -p copies from one directory to another. cpio is not that popular among Linux users; tar is much more commonly used. However, some installation programs use cpio during the installation process.

cut	
Purpose	Copy selected parts of each line of text from a file to standard output
Syntax	```cut [options] file```
Options	-b *list* extracts the characters at positions specified in the *list* -f *list* extracts the fields (assumed to be tab-separated) specified in *list* -d *char* specifies the character that delimits the fields (default is the tab) -s skips lines that do not contain delimited fields (see -f option)

Description	The cut command specifies parts from each line of text in a file and writes those lines out to standard output. You can either extract a range of characters (specified by their positions) from each line or specific fields, where the fields are separated by a special character such as the tab. For example, to extract characters 1 through 11 and the 56th character onward from a detailed directory listing, use the following command:
	
	``ls -l
		cut -b 1-11,56-	more```
			```-rw-r--r-- DIR_COLORS```
			```-rw-r--r-- HOSTNAME```
			```-rw-r--r-- Muttrc```
			```-drwxr-xr-x X11```
			```-rw-r--r-- adjtime```
			```-rw-r--r-- aliases```
			```-rw-r--r-- aliases.db```
			```-rw------- at.deny```
			```(... lines deleted)```

date	
Purpose	Display current date and time or set new date and time
Syntax	`date [options] [+format]` `date [-su] [MMDDHHMM[[CC]YY][.SS]]`
Options	`-s` sets the date and/or time `-u` displays or sets time using Greenwich Mean Time
Description	The `date` command alone displays the current date and time. Using the `+format` argument, you can also specify a display format for the date and time. For a complete listing of the format specification, type `man date`. To set the date, use date -s followed by the date and time in the `MMDDHHMM` format, where each character is a digit (`MM` is the month number, `DD` is the day, `HH` is the hour, and `MM` are the minutes). You can optionally specify the year (`YY`) and century (`CC`) as well.

dd	
Purpose	Copy blocks of data from standard input to standard output (and optionally convert the data from one format to another).
Syntax	`dd option1=value1 option2=value2 option3=value3 ...`

Options	`if=file` reads from specified file instead of standard input
	`of=file` writes to specified file instead of standard output
	`ibs=nbytes` reads blocks of *n* bytes at a time
	`obs=nbytes` writes blocks of *n* bytes at a time
	`bs=nbytes` reads and writes blocks of *n* bytes at a time
	`cbs=nbytes` converts blocks of *n* bytes at a time
	`skip=nblocks` skips *n* input blocks from beginning of input file
	`seek=nblocks` skips *n* output blocks in the output file
	`count=nblocks` copies *n* blocks from input to output
	`conv=code` performs conversion; *code* can be one of following:
	    `ascii` converts EBCDIC to ASCII
	    `ebcdic` converts ASCII to EBCDIC
	    `lcase` converts to lowercase
	    `ucase` converts to uppercase
	    `swab` swaps every pair of input bytes
	    `noerror` continues after read errors
	    (Note: EBCDIC is an encoding format used in IBM mainframes.)

Description	The `dd` command copies blocks of data from standard input to standard output, optionally converting the data as the copying proceeds. Typically, `dd` is used to copy data directly from one device to another. For example, you can copy the Linux kernel (`/boot/vmlinuz`) to a diskette with the following command: `dd if=/boot/vmlinuz of=/dev/fd0`

df	
Purpose	Display the amount of free and used storage space on all mounted file systems.
Syntax	df [options] [*filesystem*]
Options	-a displays information for all file systems -i displays inode information (the disk is organized into inodes) -T prints the type of file system -t *type* displays information about specified types of file systems only -x *type* excludes specified types of file systems from the output —help displays a help message
Description	The df command returns the amount of free and used space on a specified file system. If you want to know how full your disks are, use the df command without any arguments. The df command then displays information about used and available storage space on all currently mounted file systems.

diff	
Purpose	Show the difference between two text files (or all files with same names in two directories)
Syntax	diff [options] *from_file to_file*
Options	-a treats all files as text even if they do not seem to be text files -b ignores blank lines and repeated blanks -c produces output in a different format -d tries to find a smaller set of changes (this makes diff slower) -e produces a script for ed editor to convert from_file to to_file -f produces output similar to that in -e, but in reverse order -i ignores case -l passes the output to the pr command to paginate it -n works like -f, but counts the number of changed lines -r recursively compares files with same name in all subdirectories -s reports when two files are the same -t expands tabs to spaces in the output -u uses the unified output format -v displays version of diff -w ignores spaces and tabs when comparing lines
Description	The diff command compares *from_file* with *to_file* and displays the lines that differ. The output can be in a format that the patch command can use to convert *from_file* to *to_file*.

du	
Purpose	Display summary information about disk usage (in kilobytes)
Syntax	`du [options] [directories_or_files]`
Options	`-a` displays usage information for all files (not just directories) `-b` displays usage in bytes (instead of kilobytes) `-c` displays a grand total of all usage information `-k` displays usage information in kilobytes (default) `-s` displays total disk usage without per-directory details

Description	The `du` command displays the disk space (in kilobytes) used by the specified files or directories. By default, `du` displays the disk space used by each directory and subdirectory. A common use of `du` is to type `du -s` to view the total space used by the current directory. For example, here is how you might check the details of disk space used by the `/usr/doc/HOWTO` directory:
	`du /usr/doc/HOWTO`
	`480      /usr/doc/HOWTO/mini/other-formats/html`
	`278      /usr/doc/HOWTO/mini/other-formats/sgml`
	`762      /usr/doc/HOWTO/mini/other-formats`
	`2998     /usr/doc/HOWTO/mini`
	`4742     /usr/doc/HOWTO`

expand	
Purpose	Write files to standard output after expanding each tab into an appropriate number of spaces
Syntax	`expand [options] [files]`
Options	`-n` (where *n* is a number) sets the tabs *n* spaces apart `-n1 [n2, ...]` (where *n1, n2,...* are numbers) specifies the tab stops `-i` converts only the initial tab into spaces
Description	The `expand` command reads from the specified files (or standard input, if no files are specified) and writes them to standard output, with each tab character expanded an appropriate number of spaces. By default, `expand` assumes that the tab positions are eight spaces apart (this is equivalent to the `-8` option).
fdformat	
Purpose	Format a diskette specified by device name (such as `/dev/fd0H1440` for a 3.5-inch, high-density diskette in drive A).
Syntax	`fdformat [-n] device_name`
Options	`-n` disables the verification performed after formatting
Description	The `fdformat` command formats a diskette. Use an appropriate device name to identify the diskette drive (see Chapter 7 for naming conventions for diskette drives). After formatting a diskette with `fdformat`, you can use `mkfs` to install a Linux file system, or `tar` to store an archive.

fdisk	
Purpose	Partition a disk or display information about existing partitions
Syntax	`fdisk [options] [device_name]`
Options	`-1` displays partition tables and exits `-s device` displays the size of the specified partition `-v` displays the version number of the `fdisk` program
Description	The `fdisk` command partitions a specified hard disk (see Chapter 11 for disk-drive naming conventions). You can also use `fdisk` to display information about existing partitions. You should never run `fdisk` and alter the partitions of a hard disk while one or more of its partitions are mounted on the Linux file system. Instead, you should boot from an installation diskette (or a boot diskette) and then perform the partitioning. Remember that partitioning typically destroys all existing data on a hard disk.
fg	
Purpose	Continue an interrupted process in the foreground
Syntax	`fg`
Options	None
Description	After you interrupt a process by typing Ctrl+Z, you can continue that process in foreground by typing the `fg` command. Note that `fg` is a built-in command of the Bash shell.
file	
Purpose	Display the type of data in a file based on rules defined in the `/usr/lib/magic` file (also known as the "magic file")
Syntax	`file [options] files`
Options	`-c` displays a parsed form of a magic file (or the default one) and exits `-m file1[:file2:...]` specifies other magic files `-v` displays version number and exits `-z` looks inside compressed files
Description	The `file` command uses rules specified in the /usr/lib/magic file to determine the type of data in the specified files. For example, you can use the `file` command to check the type of each file in the `/usr/lib` directory, as follows: `file *	more` `Mcrt1.o: ELF 32-bit LSB relocatable, Intel 80386, version 1, not stripped` `X11: symbolic link to ../X11R6/lib/X11` `autoconf: directory` `bison.hairy: C program text` `bison.simple: C program text` `cracklib_dict.hwm: ASCII text` `cracklib_dict.pwd: ASCII text` `cracklib_dict.pwi: ASCII text` `(... lines deleted)`

find	
Purpose	Display a list of files that match a specified set of criteria
Syntax	`find [path] [options]`

Options	`-depth` processes current directory first, and then subdirectories
	`-maxdepth` *n* restricts searches to *n* levels of directories
	`-follow` processes directories included through symbolic links
	`-name` *pattern* finds files whose names match the *pattern*
	`-ctime` *n* matches files modified exactly *n* days ago
	`-user` *uname* finds files owned by the specified user
	`-group` *gname* finds files owned by the specified group
	`-path` *pattern* finds files whose pathname matches the *pattern*
	`-perm` *mode* finds files with specified permission setting
	`-size` +*n*K finds files bigger than *n* kilobytes
	`-type` *x* finds files of specified type where *x* is one of the following:
	    f matches files
	    d matches directories
	    l matches symbolic links
	`-print` displays the name of files found
	`-exec` *command* `[options] {} \;` executes specified command by passing it the name of the found file

Description	The `find` command is useful for finding all files that match a specified set of criteria. If you type `find` without any arguments, the output is a listing of every file in all subdirectories of the current directory. To view all files whose names end with `.gz`, you would type `find . -name "*.gz"`.
fold	
Purpose	Wrap lines of text to a specified width (default is 80 characters)
Syntax	`fold [options] [files]`
Options	`-b` counts bytes instead of columns, so backspaces and tabs are counted `-s` breaks lines at word boundaries `-w` *N* (where *N* us a number) sets line width to *N* characters
Description	The `fold` command wraps each input line to a specified number of characters and displays the results on the screen (standard output). If you do not specify any file name, `fold` reads lines from the standard input.
free	
Purpose	Display amount of free and used memory in the system
Syntax	`free [options]`
Options	`-b` displays memory in number of bytes `-k` displays memory in kilobytes (default) `-m` displays memory in megabytes `-s` *n* repeats the command every *n* seconds `-t` displays a line containing a summary of the total amounts
Description	The `free` command displays information about the physical memory (RAM) and the swap area (on the disk). The output shows the total amount of memory as well as the amount used and the amount free.

fsck	
Purpose	Check and repair a Linux file system
Syntax	`fsck [options] device_name`
Options	`-A` checks all file systems listed in the `/etc/fstab` file `-R` skips the root file system (when checking all file systems) `-T` does not show the title on startup `-N` shows what might be done, but does not actually do anything `-V` produces verbose output `-t fstype` specifies the file system type (such as `ext2`) `-n` answers all confirmation requests with no (only for `ext2` file system) `-p` carries out all repairs without asking for confirmation (for `ext2`) `-y` answers all confirmation requests with no (only for `ext2` file system)
Description	The `fsck` command checks the integrity of a file system and carries out any necessary repairs. Depending on the type of file system, `fsck` runs an appropriate command to perform the actual task of checking and repairing the file system. For example, to check an `ext2` file system, `fsck` runs the `e2fsck` program. You have to run `fsck` when you power down your system without running the shutdown command. Typically, `fsck` is automatically run during system startup.

grep	
Purpose	Search one or more files for lines that match a regular expression (a search pattern)
Syntax	`grep [options] pattern  files`

Options	`-N` (where N is a number) displays N lines around the line containing `pattern`
	`-c` shows the number of lines that contain the search pattern
	`-f file` reads options from specified file
	`-i` ignores case
	`-l` displays the filenames that contain `pattern`
	`-n` displays the line number next to lines that contain `pattern`
	`-q` returns a status code, but does not display any output
	`-v` displays the lines that do not contain `pattern`
	`-w` matches only whole words

Description	The `grep` command searches the specified files for a pattern. The pattern is a regular expression, which has its own rules. Typically, you use `grep` to search for a specific sequence of characters in one or more text files.

groups	
Purpose	Show the groups to which a user belongs
Syntax	`groups [username]`
Options	None
Description	The `groups` command displays the names of groups to which a user belongs. If you do not specify a username, the command displays your groups.

gunzip	
Purpose	Uncompress files compressed by the `gzip` or the `compress` command
Syntax	`gunzip [options] files`
Options	See the options for `gzip`
Description	The `gunzip` command uncompresses compressed files (these files have the .gz or .Z extension). After uncompressing, `gunzip` replaces the compressed files with their uncompressed versions and removes the `.gz` or `.Z` extension in the filenames. The `gunzip` command is the same as `gzip` with the `-d` option.
gzip	
Purpose	Compress one or more files
Syntax	`gzip [options] files`
Options	`-c` writes output to standard output and retains original file `-d` uncompresses file (same as `gunzip`) `-h` displays a help message `-l` lists contents of a compressed file `-n` does not save original name and time stamp `-r` recursively compresses files in all subdirectories `-v` displays verbose output `-V` displays version number
Description	The `gzip` command compresses files using Lempel-Ziv (LZ77) coding, which produces better compression than the algorithm used by the `compress` command. After compressing a file, `gzip` replaces the original file with the compressed version and appends a `.gz` to the filename.
halt	
Purpose	Terminate all processes and halt the system (you must log in as `root`)
Syntax	`halt [options]`
Options	`-n` does not flush out in-memory buffers to disk before halting system `-f` forces halt without calling the `/sbin/shutdown` command `-i` shuts down all network interfaces before halting system
Description	The `halt` command lets the super user (root) terminate all processes and halt the system. The `halt` command invokes `/sbin/shutdown` with the `-h` option.
id	
Purpose	List the user ID, group ID, and groups for a user
Syntax	`id [options] [username]`
Options	`-g` displays group ID only `-n` displays group name instead of ID `-u` displays user ID only
Description	The `id` command displays the user ID, group ID, and groups for a specified user. If you do not provide any user name, `id` displays information about the current user.

info	
Purpose	View online help information about any Linux command
Syntax	`info [options] command`
Options	`-d dirname` adds a directory to the list of directories to be searched for files `-f infofile` specifies file to be used by `info` `-h` displays usage information about info
Description	The `info` command displays online help information about a specified command in a full-screen text window. You can use Emacs commands to navigate the text displayed by `info`. To learn more about `info`, type `info` without any arguments.

kill	
Purpose	Send a signal to a process
Syntax	`kill [options] process_id`
Options	`-signum` (where `signum` is a number or name) sends the specified signal `-1` lists the signal names and numbers
Description	The `kill` command sends a signal to a process. Typically, the signal is meant to terminate the process. For example, `kill -9 123` terminates the process with ID `123`. To see process IDs, use the `ps` command. To see a list of signal names and numbers, type `kill -1` (that's lowercase "ell").

ldd	
Purpose	Display names of shared libraries required to run a program
Syntax	`ldd [options] programs`
Options	`-v` prints the version number of `ldd` `-V` prints the version number of the dynamic linker (`ld.so`) `-d` relocates functions and reports missing functions `-r` relocates both data and functions and reports missing objects

Description	The `ldd` command lets you determine which shared libraries are needed to run the specified programs. For example, to determine what you need to run the Bash shell (/bin/bash), type the following:
	`ldd /bin/bash`
	`     libtermcap.so.2 => /lib/libtermcap.so.2 (0x40003000)`
	`     libc.so.6 => /lib/libc.so.6 (0x40006000)`
	`     /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00000000)`

less	
Purpose	View text files one screen at a time (and scroll back if needed)
Syntax	`less [options] filenames`
Options	`-?` displays a list of commands you can use in `less` `-p text` displays the first line where `text` is found `-s` reduces multiple blank lines to a single blank line
Description	The `less` command displays the specified files one screen at a time. Unlike `more`, you can press b, Ctrl+B, or Esc+V to scroll backward. To view the commands you can use to interact with `less`, press h while you are viewing a file in `less`.

ln	
Purpose	Set up a hard or symbolic link (pseudonyms) to files and directories
Syntax	`ln [options] existing_file new_name`
Options	`-b` makes backup copy of files about to be removed `-d` creates a hard link to a directory (only `root` can do this) `-f` removes existing file with `new_name` `—help` displays a help message `-s` creates a symbolic link `-v` displays verbose output
Description	The `ln` command assigns a new name to an existing file. With the `-s` option, you can create symbolic links that can exist across file systems. Also, with symbolic links, you can see the link information with the `ls -l` command. Otherwise, `ls -l` shows two distinct files for a file and its hard link.
locate	
Purpose	From a periodically updated database, list all files that match a specified pattern
Syntax	`locate pattern`
Options	None

Description	The `locate` command searches a database of files for any name that matches a specified pattern. Your Linux system is set up to periodically update the file database. If you are not sure about the location of a file, just type `locate` followed by a part of the filename. For example, here's how you might search for the `XF86Config` file:
	`locate XF86Config`
	`/usr/X11R6/lib/X11/XF86Config`
	`/usr/X11R6/lib/X11/XF86Config.eg`

lpr	
Purpose	Print one or more files
Syntax	`lpr [options] [files]`
Options	`-Pprinter` prints to the specified printer (the name appears in `/etc/printcap`) `-#N` (where `N` is a number) prints that many copies of each file `-h` suppresses the burst page (the first page with user information) `-m` sends mail upon completion of print job `-r` removes the file after printing `-J jobname` prints this job name on the burst page `-U username` prints this user name on the burst page
Description	The `lpr` command prints the specified files by using your system's print spooling system. If no filenames are specified, `lpr` reads input from the standard input. You can print to a specific printer with the `-P` option.

ls	
Purpose	List the contents of a directory
Syntax	`ls [options] [directory_name]`
Options	-a displays all files, including those that start with a period (.) -b displays unprintable characters in filenames with octal code -c sorts according to file creation time -d lists directories like any other file (rather than listing their contents) -f lists directory contents without sorting (exactly as they are in the disk) -i shows the inode information -l shows the file listing in the long format with detailed information -p appends a character to the filename to indicate type -r sorts listing in reverse alphabetical order -s shows the size (in kilobytes) of each file next to the filename -t sorts listing according to file's time stamp -1 displays a one-column listing of filenames -R recursively lists the files in all subdirectories
Description	The `ls` command displays the listing of a specified directory. If you omit the directory name, `ls` displays the contents of the current directory. By default `ls` does not list files whose names begin with a period (.); to see all files, type `ls -a`. You can see full details of files (including size, user and group ownership, and read-write-execute permissions) with the `ls -l` command.

man	
Purpose	View online manual pages (also called *man pages*).
Syntax	`man [options] [section] command`
Options	-C *cfile* specifies man configuration file (default is `/etc/man.config`) -P *pager* specifies program to use to display one page at a time (for example, `less`) -a displays all man pages matching a specific *command* -h displays a help message and exits -w shows the location of man pages to be displayed
Description	The `man` command displays the man pages for the specified command. If you know the section for a man page, you can provide the section as well. For example, all Tcl/Tk man pages are in section n. Thus, you can view the man page for the Tcl/Tk command pack with the command `man n pack`.

mkdir	
Purpose	Create a directory
Syntax	`mkdir [options] directory_name`
Options	-m *mode* assigns the specified permission setting to the new directory -p creates the parent directories if they do not already exist
Description	The `mkdir` command creates the specified directory.

mkfs	
Purpose	Create a Linux file system on a hard disk partition or a diskette
Syntax	`mkfs [-V] [-t] [options] device_name [blocks]`
Options	`-V` produces verbose output needed for testing `-t fstype` specifies the file system type (such as `ext2`) `-c` checks the device for bad blocks before creating the file system `-l filename` reads bad block list from specified file
Description	The `mkfs` command creates a Linux file system on the specified device. The device is typically a hard disk partition or a diskette (that has been formatted with `fdformat`).
mknod	
Purpose	Create a device file with specified major and minor numbers
Syntax	`mknod device_file {b|c} major minor`
Options	None
Description	The `mknod` command creates a device file (such as the ones in the `/dev` directory) through which the operating system accesses physical devices such as the hard disk, serial port, keyboard, and mouse. To create a device file, you have to log in as `root` and have to know the major and minor numbers of the device for which you are creating the device file. Additionally, you must specify one of the letters b or c to indicate whether the device is block- or character-oriented. Typically, you perform this step following specific instructions in a HOWTO document.
mkswap	
Purpose	Create a swap space for Linux
Syntax	`mkswap device_or_file  numblocks`
Options	None
Description	The `mkswap` command creates a swap space for use by the Linux kernel. If you are creating swap space in a disk partition, specify the partition's device name (such as `/dev/hda2`) as the second argument to `mkswap`. If you want to use a file as swap space, create the file with a command such as `dd if=/dev/zero of=swapfile bs=1024 count=16384`. Then type `mkswap swapfile 16384` to create the swap space. You have to use the command `swapon` to activate a swap space.
more	
Purpose	View text files one screen at a time
Syntax	`more [options] filenames`
Options	`+N` (where *N* is a number) displays file starting at specified line number `+/pattern` begins displaying 2 lines before the `pattern` `-s` reduces multiple blank lines to a single blank line

more	
Description	The `more` command displays the specified files one screen at a time. To view the commands you can use in `more`, press h while you are viewing a file using `more`. For more advanced file viewing, use the `less` command.

mount	
Purpose	Associate a physical device to a specific directory in the Linux file system
Syntax	`mount [options] device  directory`
Options	`-a` mounts all devices listed in the `/etc/fstab` file `-h` displays a help message and exits `-r` mounts the device for read-only (no writing allowed) `-t  fstype` specifies the file system type on the device `-v` displays verbose messages `-V` displays the version number and exits
Description	The mount command attaches the contents of a physical device to a specific directory on the Linux file system. For example, you may mount a CD-ROM at the `/cdrom` directory. Then, you can access the contents of the CD-ROM at the `/cdrom` directory (in other words, the root directory of the CD-ROM appears as `/cdrom` after the mount operation). To see the listing of the `RedHat` directory on the CD-ROM, you would type `ls /cdrom/RedHat`.

mv	
Purpose	Rename files and directories or move them from one.directory to another
Syntax	`mv [options] source  destination`
Options	`-b` makes backup copies of files being moved or renamed `-f` removes existing files without prompting `-i` prompts before overwriting any existing files `-v` displays name of file before moving it
Description	The `mv` command either renames a file or moves it to another directory. The command works on either plain files or directories. Thus, you could rename the file `sample` to `sample.old` with the command `mv sample sample.old`. On the other hand, you can move the file `/tmp/sample` to `/usr/local/sample` with the command `mv /tmp/sample /usr/local/sample`.

nice	
Purpose	Run a program at a lower or higher priority level
Syntax	`nice [options] program`
Options	$+n$ (n = number) adds n to nice value (positive values are lower priority) $-n$ (n = number) subtracts n to nice value (negative means higher priority)
Description	The `nice` command allows you to run a program at lower or higher priority. By default, programs run at the nice value of zero. Adding to the nice value decreases the priority while subtracting from the nice value increases the program's priority. Only `root` can decrease the nice value.

nl

Purpose	Add line numbers to nonblank lines of text in a file and write to standard output
Syntax	`nl [options] [file]`
Options	`-ba` numbers all lines `-bt` numbers text lines only (default) `-sc` separates text from line numbers with the character `c` (default is tab) `-wn` uses `n` columns to show the line numbers

Description	The `nl` command adds a line number to each nonblank line of text from a file and writes the lines to standard output. Suppose the file `sample.txt` has the following lines:
	`A line followed by a blank line and`
	
	`then another non-blank line.`
	
	Applying the `nl` command to this file produces the following result:
	`nl sample.txt`
	    `1   A line followed by a blank line and`
	
	    `2   then another non-blank line.`

passwd

Purpose	Change password
Syntax	`passwd [username]`
Options	None
Description	The `passwd` command changes your password. It prompts for the old password followed by the new password. If you log in as `root`, you can change another user's password by specifying the user name as an argument to the `passwd` command.

paste

Purpose	Write to standard output corresponding lines of each file, separated by a tab
Syntax	`paste file1 file2 [...]`
Options	`-s` pastes the lines from one file at a time instead of one line from each file `-d delim` uses delimiters from the list of characters instead of a tab `-` causes paste to use standard input as a file
Description	The `paste` command takes one line from each of the listed files and writes them out to standard output, separated by a tab. With the `-s` option, the `paste` command can also concatenate all lines from one file into a single gigantic line.

patch	
Purpose	Apply the output of the `diff` command to an original file
Syntax	`patch [options] < patch_file`

Options	`-c` causes patch file to be interpreted as a context `diff`
	`-e` forces patch file to interpret the patch file as an `ed` script
	`-f` forces patch file to be applied regardless of any inconsistencies
	`-n` causes patch file to be interpreted as a normal `diff`
	`-pN` strips everything up to *N* slashes in the pathname
	`-R` indicates that patch file was created with new and old files swappped
	`-u` causes patch file to be interpreted as a unified `diff`
	`-v` displays the version number

Description	The `patch` command is used to update an original file by applying all the differences between the original and revised version. The differences are in the form of an output from the `diff` command, stored in the `patch_file`. Changes to Linux kernel source code is distributed in the form of a patch file. Chapter 2 discusses how to apply patches to the kernel source.

printenv	
Purpose	View a list of environment variables
Syntax	`printenv`
Options	None
Description	The `printenv` command displays a list of all current environment variables.

ps	
Purpose	Display status of processes (programs) running in the system
Syntax	`ps [options]`
Options	[Note: Unlike other commands, `ps` options do not have a - prefix.] `a` displays processes of other users `f` displays family tree of processes `j` displays output using jobs format `l` displays in long format with many details for each process `m` displays memory usage information for each process `u` displays user name and start time `x` displays processes not associated with any terminal
Description	The `ps` command displays the status of processes running in the system. Typing `ps` alone produces a list of processes you are running. To see a list of all processes in the system, type `ps ax` (or `ps aux`, if you want more details about each process).

pstree	
Purpose	Display all running processes in the form of a tree
Syntax	`pstree [options] [pid]`
Options	-a shows command-line arguments -c does not compact subtrees -l displays long lines -n sorts processes by process ID (instead of name) -p shows process IDs

Description	The `pstree` command shows all processes in the form of a tree, which makes it easy to understand the parent-child relationships among the processes. The following is typical output from `pstree`:
	
	`pstree`
	`init-+-atd`
	`      |-crond`
	`      |-gpm`
	`      |-httpd---10*[httpd]`
	`      |-inetd-+-in.telnetd---bash---pstree`
	`
		`-2*[in.telnetd---bash---su---bash]`
		`      |-kerneld`
		`      |-kflushd`
		`      |-klogd`
		`      |-kswapd`
		`      |-10ve`
		`      |-lpd`
		`      |-lpnetd`
		`      |-6*[mingetty]`
		`      |-nmbd`
		`      |-postmaster`
		`      |-pppd`
		`      |-sendmail`
		`      |-smbd`
		`      |-syslogd`
		`      `-update`

pwd	
Purpose	Display current working directory
Syntax	pwd
Options	None
Description	The pwd command prints the current working directory. pwd is a built-in command of the Bash shell.

reboot	
Purpose	Terminate all processes and reboot the system (you must log in as `root`)
Syntax	`reboot [options]`
Options	`-n` does not flush out in-memory buffers to disk before rebooting system `-f` forces halt without calling the `/sbin/shutdown` command `-i` shuts down all network interfaces before rebooting system
Description	The `reboot` command lets the super user (`root`) terminate all processes and reboot the system. The `reboot` command invokes `/sbin/shutdown` with the `-r` option.

rm	
Purpose	Delete one or more files
Syntax	`rm [options] files`
Options	`-f` removes files without prompting `-i` prompts before removing a file `-r` recursively removes files in all subdirectories, including the directories `-v` displays name of each file before removing it
Description	The `rm` command deletes the specified files. To remove a file you must have write permission to the directory containing the file.

rmdir	
Purpose	Delete a specified directory (provided the directory is empty)
Syntax	`rmdir [options] directory`
Options	-p removes any parent directories that become empty
Description	The `rmdir` command deletes empty directories. If a directory is not empty, you should use the `rm -r` command to delete the files as well the directory.

sed	
Purpose	Copy a file to standard output after editing according to a set of commands
Syntax	`sed [options] [editing_commands] [file]`
Options	-e'instructions' applies the editing instructions to the file `-f scriptfile` applies editing commands from `scriptfile` -n suppresses default output
Description	The `sed` command is known as the *stream editor* — it copies a file to standard output while applying specified editing commands. If you do not specify a file, it reads from the standard input. To use the stream editor, you have to learn its editing commands, which are very similar to the ones used by the `ed` editor (described in Chapter 24).

shutdown	
Purpose	Terminate all processes and shutdown (or reboot) the system
Syntax	`shutdown [options]` *time* `[`*messages*`]`
Options	`-t` specifies the time between the message and the kill signal `-h` halts the system after terminating all the processes `-r` reboots the system after terminating all the processes `-f` performs a fast reboot `-k` sends warning messages but does not actually shut down system `-c` cancels a shutdown in progress
Description	The `shutdown` command (the full pathname is `/sbin/shutdown`) brings down the Linux system in an orderly way. You must specify a time when shutdown begins; use the keyword `now` to shutdown immediately. You must be logged in as `root` to run the `shutdown` command.

sort	
Purpose	Sort or merge lines from a file and then write to standard output
Syntax	`sort [options]` `[`*files*`]`
Options	`-c` checks if files are already sorted and prints error message if not `-m` merges files by sorting them as a group `-b` ignores leading blanks `-d` sorts in phone directory order (uses only letters, digits, and blanks) `-f` treats lowercase letters as equivalent uppercase letters `-k` *POS1*`[`*.POS2*`]` specifies the sort field as characters between the two positions *POS1* and *POS2* `-o` *file* writes output to specified file instead of standard output `-r` sorts in reverse order `-g` sorts numerically but uses conversion to real number `-i` ignores unprintable characters `-n` sorts numerically (used when number begins each line) `-t`*C* specifies the separator character `+`*N* only considers characters from position *N* onwards (0=first position)
Description	The `sort` command sorts the lines in one or more files and writes them out to the standard output. The same command can also merge several files (when used with the `-m` option) and produce an appropriately sorted and merged output.

split	
Purpose	Split a file into several smaller files
Syntax	`split [options]` *file* `[`*prefix*`]`
Options	`-l` *N* (where *N* is a number) puts *N* lines in each file `-N` (where *N* is a number) puts *N* lines in each file `-b` *N*`k` (where *N* is a number) splits the file every *N* kilobytes `-C` *N*`k` (where *N* is a number) puts as many lines as possible without exceeding *N* kilobytes per file

split

Description	The `split` command breaks up a large file into smaller files. By default, `split` puts 1,000 lines into each file. The files are named by groups of letters `aa`, `ab`, `ac`, and so on. You can specify a prefix for the filenames. For example, to split a large archive into smaller files that fit into several high-density 3.5-inch diskettes, use split as follows: `split -C 1440k bigfile.tar disk.` This creates files named `disk.aa`, `disk.ab`, and so on.

su

Purpose	Become another user
Syntax	`su [options] [username]`
Options	`-c COMMAND` passes the `COMMAND` to the shell `-f` prevents reading the startup file (`.cshrc`) when the shell is csh or tcsh `-l` makes this a login shell by reading the user's startup file `-p` preserves the environment variables `HOME`, `USER`, `LOGNAME`, and `SHELL` `-s SHELL` runs the specified `SHELL` instead of the user's default shell
Description	The `su` command lets you assume the identity of another user. You have to provide the password of that user before you can continue. If you do not provide a user name, `su` assumes you want to change to the `root` user.

swapoff

Purpose	Deactivate the specified swap device or file
Syntax	`swapoff device`
Options	None
Description	The `swapoff` command stops Linux from using the specified device or file as a swap space.

swapon

Purpose	Activate the specified swap device or file
Syntax	`swapon [-a] device`
Options	`-a` enables all swap devices listed in the `/etc/fstab` file
Description	The `swapon` command activates the specified device or file as a swap space. During system startup, all swap spaces are activated by the `swapon -a` command, which is invoked by the script file `/etc/rc.d/rc.sysinit`.

sync

Purpose	Write buffers to disk
Syntax	`sync`
Options	None
Description	When you cannot shut down your Linux system in an orderly manner (for example, when you cannot execute `shutdown`, `halt`, or `reboot` commands), you should type `sync` before switching off the computer.

tac	
Purpose	Copy a file, line by line, to the standard output in reverse order (last line first)
Syntax	`tac file`
Options	`-b` places the separator at the beginning of each line `-r` treats the separator string specified by -s as a regular expression `-s sep` specifies a separator (instead of the default newline character)
Description	The `tac` command displays the specified text file in reverse order, copying the lines to standard output in reverse order. By default `tac` treats each line as a record and uses the newline character as the record separator. However, you can specify a different separator character, in which case, `tac` will copy those records to standard output in reverse order.
tail	
Purpose	View the last few lines of a file
Syntax	`tail [options] file`
Options	`-N` (where N is a number) displays last N lines `-n N` (where N is a number) displays last N lines `-f` reads the file at regular intervals and displays all new lines
Description	The `tail` command displays lines from the end of the specified file. By default, `tail` displays the last 10 lines from the file. To view the last 24 lines from a file named `messages`, type `tail -24 messages`.
tar	
Purpose	Create an archive of files or extract files from an archive
Syntax	`tar [options] files_or_directories`

Options	`-c` creates a new archive
	`-d` compares files in an archive with files in current directory
	`-r` extends the archive with more files
	`-t` lists contents of an archive
	`-x` extracts from the archive
	`-C directory` extracts files into the specified directory
	`-f file` uses the specified file as the archive instead of a tape
	`-L n` specifies capacity of tape as n kilobytes
	`-N date` only archives files newer than the specified date
	`-T file` archives or extracts the filenames specified in `file`
	`-v` displays verbose messages
	`-z` compresses or uncompresses archive with `gzip`

Description	The `tar` command creates an archive of files or extracts files from an existing archive. By default, `tar` assumes the archive to be on a tape. However, you can use the `-f` option to specify a file as the archive.

top	
Purpose	List currently running processes, arranged in order of their share of CPU time
Syntax	`top [q] [d delay]`
Options	q causes `top` to run with highest possible priority (you have to be `root`) d `delay` specifies the delay between updates in seconds
Description	The `top` command produces a full-text screen with the processes arranged according to their share of CPU time. By default, `top` updates the display every five seconds. Press q or Ctrl+C to quit `top`.

touch	
Purpose	Change a file's time stamp
Syntax	`touch [options] files`
Options	-c stops `touch` from creating a file that does not exist -d `time` uses the specified time -r `file` uses the time stamp from the specified file -t `MMDDhhmm[[CC]YY][.ss]` uses the specified date and time
Description	The `touch` command lets you change the date and time of a file's last modification (this information is stored with the file). If you use `touch` without any options, the current date and time is used as the time stamp for the file. If the specified file does not exist, `touch` creates a new file of size 0 bytes.

tr	
Purpose	Copy from standard input to standard output while substituting one set of characters with another
Syntax	`tr [options] string1 [string2]`
Options	-c complements characters in `string1` with ASCII codes 001–377 -d deletes from the input all characters specified in `string1` -s replaces repeated sequences of any character in `string1` with a single character

Description	The `tr` command substitutes all characters in `string1` in the input with the corresponding characters in `string2`. For example, to convert the file `sample.lc` to all uppercase and store in `sample.uc`, you would type:
	`tr [a-z] [A-Z] < sample.lc > sample.uc`
	
	To replace repeated occurrences of newlines in a file with a single newline character, you would type
	`tr -s '\n' < infile > outfile`

tty	
Purpose	Display the device name of the terminal
Syntax	`tty`
Options	None
Description	The `tty` command displays the name of the terminal connected to the standard input. This is useful in shell scripts that may need the terminal name.
type	
Purpose	Display the type of a command (whether it is a built-in shell command or a separate executable program)
Syntax	`type command`
Options	None

Description	The `type` command tells you the type of a command — whether it is a built-in shell command, an alias, or an executable program. For example, I used `alias ll='ls -l'` to define the alias ll. Here is what I get when I check the type of the command ll:
	 `type ll`
	 `ll is aliased to 'ls -l'`

umount	
Purpose	Disassociate a device from the Linux file system
Syntax	`umount device`
Options	None
Description	The `umount` command removes the association between a device and a directory in the Linux file system. Only the `root` can execute the `umount` command.
unalias	
Purpose	Delete an abbreviation defined earlier with `alias`
Syntax	`unalias abbreviation`
Options	None
Description	The `unalias` command removes an abbreviation defined earlier with the `alias` command. `unalias` is a built-in command of the Bash shell.
uname	
Purpose	Display system information such as type of machine and operating system
Syntax	`uname [options]`

---

uname

Options	-a displays all information -m displays the hardware type (for example, i586) -n displays machine's host name -p displays the processor type (this appears as unknown) -r displays the operating system release (for example, 2.0.32) -s displays the operating system name -v displays the operating system version (shown as date of compilation)
Description	The uname command displays a variety of information about your machine and the operating system (Linux).

---

uncompress

Purpose	Uncompress one or more files that have been compressed using the compress command
Syntax	uncompress [-cdrvV] files
Options	-c writes result to the standard output and retains original -r recursively uncompresses files in all subdirectories -v displays a message as each file is uncompressed -V prints version number and exits
Description	The uncompress command uncompresses each specified file and replaces the compressed version with the original (and removes the .Z suffix appended to the name of the compressed file). The uncompress command is the same as running compress with the -d option.

---

uniq

Purpose	Write all unique lines from an input file to standard output
Syntax	uniq [options] file
Options	-N (where N is a number) ignores the first N fields on each line +N (where N is a number) ignores the first N characters on each line -c writes number of times each line occurred in file -d writes only duplicate lines -u writes only unique lines (default)
Description	The uniq command removes duplicate lines from an input file and copies the unique lines to the standard output. If you do not specify an input file, uniq reads from standard input.

---

wc

Purpose	Display byte, word, and line count of a file
Syntax	wc [options] [files]
Options	-c displays only the byte count -w displays only the word count -l displays only the line count
Description	The wc command displays the byte, word, and line count of a file. If you do not specify an input file, wc reads from standard input.

whatis	
Purpose	Search whatis database (see `apropos`) for complete words
Syntax	`whatis keyword`
Options	None
Description	The `whatis` command searches the whatis database (see the entry for `apropos`) for the specified keyword and displays the result. Only complete word matches are displayed.

whereis	
Purpose	Find the source, binary, and man page for a command
Syntax	`whereis [options] command`
Options	-b searches only for binaries -m searches only for man pages -s searches only for sources

Description	The `whereis` command searches the usual directories (where binaries, man pages, and source files are located) for binaries, man pages, and source files for a command. For example, here is the result of searching for the files for rpm command:
	`whereis rpm`
	`rpm: /bin/rpm /usr/include/rpm /usr/man/man8/rpm.8`

which	
Purpose	Search the directories in the PATH environment variable for a command
Syntax	`which command`
Options	None
Description	The `which` command searches the directories in the PATH environment variable for the file that will be executed when you type the command. This is a good way to check what you would execute when you type a specific command.

zcat, zless, zmore	
Purpose	View the contents of a compressed text file without having to first decompress the file

Syntax	`zcat filename`
	`zless filename`
	`zmore filename`

Options	None
Description	The `zcat`, `zless`, and `zmore` commands work the same way as `cat`, `less`, and `more`. The only difference is that the z commands can directly read files compressed with `gzip` or `compress` (without having to first uncompress the files with `gunzip`). These commands are particularly useful for reading the compressed HOWTO files in `/usr/doc/HOWTO` directory.

Red Hat Hat® LINUX® Secrets

2nd EDITION

About the Author

Naba Barkakati is an expert programmer and successful computer book author who has experience in a wide variety of systems, ranging from MS-DOS and Windows to UNIX and the X Windows System. Over the past ten years, Naba has written 22 computer books on a number of topics ranging from Windows programming with Visual C++ to LINUX, including Visual C++® Developer's Guide, and Borland® C++ 4 Developer's Guide.

Stay on Top of the Latest LINUX Developments

Since its introduction as freeware in 1994, LINUX has grown into a robust, full-fledged operating system that rivals any commercial UNIX system. Take full advantage of the raw performance and core functionality of LINUX. From installing the kernel to setting up LINUX as a Web server to developing add-ons, *Red Hat LINUX Secrets*, Second Edition, brings you specific, real-world solutions for all your computing needs — whether at home or at the office — plus expert tips and little-known facts you won't find in any other book!

The Ultimate Guide to Mastering LINUX

- Install and customize LINUX to fit your needs
- Set up LINUX for hardware peripherals — including video and network cards, CD-ROM drives, IDE/EIDE drives, and more
- Master C++, Motif, and X programming
- Use LINUX to connect to SLIP and PPP dial-up networks
- Configure LINUX as a Web or anonymous FTP server
- Discover the best resources for ongoing information on LINUX

http://www.idgbooks.com

System Requirements	Intel 386 or better; 40-100MB hard drive space; 16MB RAM; CD-ROM drive
Reader Level	Intermediate to Advanced
Shelving Category	PCs/Operating Systems/LINUX

$49.99 USA • $69.99 Canada • £42.99 UK Incl. VAT

Red Hat LINUX 5.1 on CD-ROM, complete with:

- LINUX kernel 2.2.34
- Installation and configuration tools
- Graphical user interface
- Full TCP/IP networking
- Tools for ISP access
- Complete suite of Internet applications, including e-mail, news, and telnet
- Apache Web Server 1.2.6
- Samba 1.9
- Text editors
- Graphics software
- Programming languages, including GNU C, C++, Perl, Tcl/Tk, Python, and GNU AWK
- Support for ELF and IBCS
- Complete suite of standard UNIX utilities
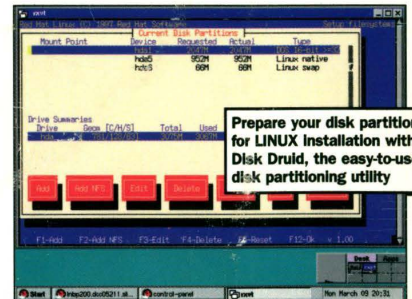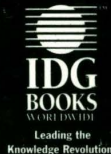- Text formatting and typesetting software
- Games

Shareware programs are fully functional, free trial versions of copyrighted programs. If you like particular programs, register with their authors for a nominal fee and receive licenses, enhanced versions, and technical support. Freeware programs are free, copyrighted games, applications, and utilities. You can copy them to as many PCs as you like — free — but they have no technical support.

Prepare your disk partitions for LINUX installation with Disk Druid, the easy-to-use disk partitioning utility

IDG BOOKS WORLDWIDE
Leading the Knowledge Revolution

INFO WORLD

IDG .net

ISBN 0-7645-3175-1

54999

7 85555 53175 8

9 780764 531750