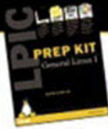
LPIC




LEARN | REVIEW | PRACTICE | PASS

TUTORIAL
Text is designed for self-study with testing tips and memory boosters

FLASH NOTES
BONUS tear-out booklet reviews the key concepts and terms from the book

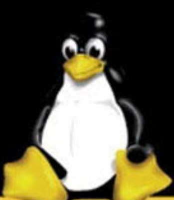
LAB EXERCISES
Lab exercises provide hands-on training and troubleshooting practice

TEST ENGINE
Random-access test engine (250+ questions written for use on a Linux platform

PREP KIT

General Linux I

COVERS EXAM 101



LINUX PROFESSIONAL INSTITUTE CERTIFICATION

QUE®

Theresa Hadden Martinez

LPIC Prep Kit 101 General Linux 1

Written by Theresa Hadden Martinez

QUE®

201 W. 103rd Street
Indianapolis, Indiana 46290

Contents at a Glance

LPIC Prep Kit 101 General Linux 1

International Standard Book Number: 0-7897-2292-5

Library of Congress Catalog Card Number: 99-068691

Printed in the United States of America

First Printing: June, 2000

02 01 00 4 3 2 1

Trademarks

Warning and Disclaimer

Credits

Associate Publisher
Greg Wiegand

Acquisitions Editor
Tracy Williams

Senior Development Editor
Jill Hayden

Development Editor
Valerie Perry

Managing Editor
Thomas Hayes

Project Editor
Natalie Harris

Copy Editor
Megan Wade

Indexer
Aamir Burki

Proofreader
Benjamin Berg

Technical Editor
Coletta Witherspoon

Team Coordinator
Vicki Harding

Media Developer
Michael Hunter

Interior Designer
Anne Jones

Cover Designers
Anne Jones
Kevin Spear

Production
Cyndi Davis-Hubler
Lisa England

Composed in ***AGaramond*** and ***Futura*** by Que Corporation.

I would like to dedicate this book to my husband, Ron, who overlooks my behavior when I am writing, and to our dog, Edwina, who will not let me forget that playing catch is also important.

Acknowledgments

I want to take this opportunity to thank my acquisitions editor, Tracy Williams, and my development editors, Jill Hayden and Valerie Perry, who worked so diligently to make this the best book possible. In addition, I want to thank everyone else at Pearson who worked together to present you, the reader, with the material you need to pass your certification exam in a format that helps, rather than hinders, your learning experience.

Theresa

About the Author

Theresa Hadden, MCSE, MCT, and SAIR Linux certified, teaches Microsoft server products, Linux, and security courses. She presents both Microsoft official curriculum and custom prepared courses to prepare students for certification exams, as well as post-exam reviews. In addition, she teaches Linux Intro and system administration courses for Compaq and has been active in LPI's Linux community in the mailing list and other venues, providing input and suggestions for the objectives and questions on the test. Theresa has more than 15 years system administration experience using Novell, UNIX, and Windows NT. She works with small- to medium-sized businesses needing LAN/WAN capabilities or Internet access using both Linux and Microsoft products. Theresa also provides installation, administration, and educational services to her clients.

About the Technical Editors

Craig and Coletta Witherspoon have been working with and writing about computers since the 1970s. They began their careers writing training and promotional materials for multinational corporations. They are the authors of about 20 books (including four books on the Linux operating system) and technical editors for about two dozen other books, and have been involved in the LPIC 101 General Linux I test since its beta-test stage.

Tell Us What You Think!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As an associate publisher for Que, I welcome your comments. You can fax, email, or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and author as well as your name and phone or fax number. I will carefully review your comments and share them with the author and editors who worked on the book.

Fax: 317-581-4666

Email: `office_que@mcp.com`

Mail: Associate Publisher
Que
201 West 103rd Street
Indianapolis, IN 46290 USA

Table of Contents

INTRODUCTION

The Linux Professional Institute (LPI) is a group of volunteers who have come together with a belief in the need for a certification in Linux that will be accepted worldwide. The other Linux certifications available are distribution-specific. The goal of LPI is to demonstrate the candidate's abilities to administer any Linux system.

As you review the objectives for each exam, you will see how LPI has implemented their goal. Each objective covers essential information applicable to any Linux installation. As a result, the LPI certification will become the Linux certification of choice for those who want to demonstrate their Linux abilities instead of their abilities to manage one particular distribution of Linux.

The first level of certification covers the skills needed by a junior Linux system administrator. This certification will cover basic abilities in the following areas:

- Documentation
- GNU and UNIX commands
- Boot, initialization, shutdown, and runlevels
- Devices, Linux filesystem
- Administrative tasks

This certification consists of two exams. Each of these two exams covers general Linux topics. Originally, LPI intended to offer a third exam as part of the first level of certification. This third exam was to be distribution-specific, with the candidate choosing among several different offerings.

Based on the psychometric evaluation of the available topics that were to be included in these distribution exams, it was decided that there was not enough material to justify the third exam. Some distribution-specific material was then rolled over into the objectives for the second test (102).

This book is intended to help you prepare for the first exam (101) offered by the Linux Professional Institute on your road to becoming Linux certified.

The 101 exam covers material common to all distributions. Areas covered on the exam include documentation, GNU/UNIX commands, filesystems, booting, runlevels, and administrative tasks. This test is required for all candidates for LPIC Level 1 certification.

How This Book Is Organized

The chapters in this book have been written based on the objectives published by LPI for the 101 exam. You can review the objectives at `http://www.lpi.org`.

The objectives are thoroughly covered in this book, although not in the order listed on the LPI Web site. Rather, they are organized in a logical way so that each chapter builds on the preceding chapter. For a complete list of exam objectives and where they are discussed, refer to Appendix G, "Objectives Index."

Chapter 1, "Finding and Creating Documentation," covers the use and maintenance of system and application documentation installed on your system.

Chapter 2, "Booting Linux," takes you through the steps of booting Linux, managing runlevels, and shutting down your system.

Chapter 3, "Linux Filesystem," covers partitioning as well as disk utilities used to maintain the filesystem.

Chapter 4, "File Management," discusses the directory hierarchy and file management. It also discusses utilities used for text searches.

Chapter 5, "GNU and UNIX Commands," covers typing at the command line and history. In addition, it discusses processes and priorities.

Chapter 6, "Maintaining the Filesystem," explains disk quotas and disk management.

Chapter 7, "Users and Groups," examines how to organize and manage your users and groups.

Chapter 8, "Text Streams," covers the most common text filters.

Chapter 9, "Permissions," discusses how to control the resources on your system.

Chapter 10, “Administrative Tasks,” provides an introduction to basic system administration.

Chapter 11, “Backup and Restore,” outlines how to plan for disaster recovery and discusses the more common utilities.

Appendix A, “Glossary,” provides definitions of terms presented in this book.

Appendix B, “Certification Process,” discusses the road to LPI certification.

Appendix C, “Testing Tips,” presents some hints to help you be successful on your exams.

Appendix D, “Alternate Resources,” lists other places to go for information.

Appendix E, “Using the CD-ROM,” explains how to use the CD-ROM that comes with this book.

Appendix F, “Lab Exercises,” is a comprehensive test for you to use to gauge your readiness for this exam.

Appendix G, “Objectives Index,” includes a list of LPIC Test 101 objectives and a discussion of each objective.

Even if you feel comfortable with the material covered in a specific chapter, please be sure to review that chapter before proceeding to the next one. Because each chapter builds on the preceding ones, you’ll want to have a firm grasp of the foundation chapter before moving on.

A bonus that you received when you purchased this book is the tear-out booklet. This booklet contains a brief synopsis of each chapter covering the most important points. Use it in conjunction with the prerequisites at the beginning of each chapter to ensure you understand the foundation chapters before proceeding.

You also can use the tear-out booklet for your review before taking the final test at the end of the book. In addition, you can take the tear-out booklet with you the day of your exam to review in the parking lot right before your test.

Each chapter begins with a series of “While You Read” questions that test your preconceptions of how chapter topics are put to task in an administrative environment. To gauge your own learning curve, write down your answers before moving on to the rest of the chapter. While you read the chapter, refer to your answers and revise them as necessary. The answers are provided at the end of the chapter.

A collection of Practice Test questions also appears at the end of each chapter. Both the correct and incorrect answers are explained in an effort to optimize the benefits you get from this practice test. After completing the end-of-chapter test, review the answers to see how well you did. Not doing this is only cheating yourself.

If you find that you have missed several of the end-of-chapter questions, go back and review that chapter before proceeding to the next chapter. If you still are unclear about a concept, try another resource such as the Linux Documentation Project or any of the other resources listed in Appendix D.

Preparing to Test

This exam is intended for the individual who has experience with Linux. You need to have a Linux system available that you can use. Practice each of the concepts covered in this book until you are comfortable with them.

To get the most from this book, you should start with Chapter 1 and complete each chapter in the order in which they are presented. When starting a chapter, the first thing you should do is to read the prerequisites. If you don't feel you have a good grasp of these principles, go back and review before proceeding.

Next, read the "While You Read" questions and write down your answers. You also might want to make note of any other questions that come to mind. After you have finished reading the chapter, look over the answers. Make sure you understand why the answers are correct.

The next thing you will want to do is any associated lab exercise. These are designed to reinforce the didactic material you have read. The hands-on experience will be invaluable when you are sitting there taking the exam.

The last thing you should do is answer the end-of-chapter questions. Write down your answer to each question before checking it. If you miss any, review the rationale for each possible answer. You also should review the appropriate section in the chapter. If you still are unclear, consult Appendix D for other places that will provide additional explanations.

While you are studying for your first exam, take the tear-out booklet with you wherever you go. It is designed to be portable. You can use it to review whenever you have a few minutes, such as when standing in line, waiting at the auto repair shop, and any other time you are stuck with a few minutes to fill.

After you have completed the book and the end-of-book test, you will want to use the practice test engine provided on the CD-ROM that accompanies this book. Enough questions are provided for four completely different tests, although by using the randomizing feature, you'll get a "new" test every time you sit down. This is excellent experience to get you ready for the real thing. See Appendix E for information on how to use the test engine.

The test will contain questions that require you to evaluate and diagnose the situation. The answer you choose will be the result of this evaluation. It is in this situation that the hands-on experience you get from the exercises in Appendix F and your own experience will be invaluable.

Study well and best of luck on your certification endeavor!

CHAPTER PREREQUISITE

Before beginning this chapter you should have access to a computer running Linux that you can use to practice these concepts and skills. You should be able to log on to your system and understand the basics of typing at the command line.

CHAPTER 1

Finding and Creating Documentation

WHILE YOU READ

1. You ask a co-worker how to locate a file on your Linux system and he tells you to check the `man` pages for `find` and `locate`. What does he mean by the term `man`?
2. You are looking at the man page about using `tar` to find the option to compress the `tar` file. How can you find it without having to read the entire page?
3. Why does using `man command` not find the correct manual page?
4. How can you configure your computer to make Spanish the default language for manual pages?
5. Where can you find information on setting up your new zip drive?
6. What is the Linux Documentation Project?
7. You find a file in `/usr/bin` but do not know what it does. How can you find this out?
8. How do you find the exact name of the command to change your password?

Getting Help

Usually there is no hard copy manual supplied with Linux unless you purchase a distribution that includes a manual. However, there is an almost unlimited amount of information available from many sources. These may be categorized as

- Local documentation
- Online via the Internet
- Newsgroups
- Mailing lists
- Books

The ability to find and use this documentation will be tested on the exam.

Local Documentation

There are several types of locally installed documentation. Table 1.1 shows the types and their locations.

Table 1.1 Installed Documentation and Their Locations

Documentation	*Location*
manual pages	`/usr/man`
info	`/usr/info`
HOWTOs	`/usr/doc/HOWTO`
Frequently Asked Questions (FAQ)	`/usr/doc/FAQ`
program documentation	`/usr/doc/<program name>`

Each one of these has its advantages and disadvantages. Usually information on the same topic can be found in more than one source. However, one source may be easier for you to understand than the other.

man Pages

There is extensive local online help available on almost every utility, system call, or command on your system called the *manual pages*. The manual pages are not user-friendly and are more helpful for looking up syntax than using as a tutorial.

They are frequently referred to as man pages and are invoked using the `man` utility. `man` searches for a man page on a specified topic and sends the output through a pager. The pager then prints the man page to your display. How you navigate through the man page is dependent on which pager is being used.

The syntax for `man` is

```
man [options] [section] [topic]
```

You invoke a man page by passing the topic to the `man` command. If you type `man man` the manual page describing how the manual pages work is displayed. If you are looking for information on the command `who`, you could get it by typing `man who`. The displayed man page will present a summary of what the `who` command does and its syntax.

Each man page is divided into sections, although not all sections appear in every page. Table 1.2 lists the sections and what they might contain.

Table 1.2 Sections of man Pages

Section	*Description*
NAME	The name and a brief description of the command
SYNOPSIS	How to use the command and its command-line options
DESCRIPTION	Explanation of the program and its options
OPTIONS	All options with a brief description of each
SEE ALSO	Related man pages
DIAGNOSTICS	Description of error messages
FILES	List of files used by the command and their locations
BUGS	Known problems
HISTORY	Milestones of program's development
AUTHOR	Program's author and contributors

Not only are the individual manual pages internally organized into sections, the pages themselves are arranged in a specific directory structure. Table 1.3 lists the names of these directories and what type of man pages each contains. This is information you may need to know for the exam.

Table 1.3 Directory Organization of man Pages

Directory	Contents
/usr/man/man1	Commands you can run from within a shell
/usr/man/man2	Documentation on system calls
/usr/man/man3	Manual pages for libc functions
/usr/man/man4	Information about files in the /dev directory
/usr/man/man5	Details of formats for special files such as /etc/passwd
/usr/man/man6	Games
/usr/man/man7	Descriptions of Linux file system, man pages, and so on
/usr/man/man8	Pages for root operator utilities
/usr/man/man9	Documentation on Linux kernel source routines

After a man page is displayed, there are specific commands for navigating around the document. And these commands vary depending on the pager used to display the man pages.

The two pagers that are usually used to display man pages are `more` and `less`. For both pagers, all you need to do is press the spacebar to display the next screen or `b` to back up one screen. `less` also allows you to use the <PgUp> and <PgDn> keys to scroll up or down.

The Enter key will advance the display one line at a time. When you are finished, type `q` to quit in either `less` or `more`.

Because man pages can be quite long and you cannot easily scroll up and down, a method of search is extremely valuable.

In order to search for a string inside the man page, type

```
/<string>
```

which will appear on the last line of the screen. The cursor will advance to the first occurrence of the search string after you press the Enter key. To continue searching for the next occurrence of the search string, type `n`. If no match was found or the last instance has been found, then the pager will notify you by placing a message on the last line of the screen.

When searching for a man page on a specific topic, the first page found will be displayed.

The sections (directories) are not searched in numerical order, but rather in the following order:

1, 8, 2, 3, 4, 5, 6, 7, 9

Each section has its own introduction man page that presents what is in that section. For example, if you want to find out information on what the game section has, you would type `man intro`; however, you would be presented with the intro page from Section 1.

`man` quits searching as soon as it finds a match. This might not be the page you were looking for as the search ends as soon as one page is found that satisfies your criteria. Typing `man intro` would never get you to the games section, Section 6. You can override this behavior by specifying which section to search. Typing

```
man 6 intro
```

would display the intro page for the games section.

But say you need information on using `write` in an application you are writing. When you type `man write` you get information on the `write` utility.

You know that there are additional man pages on `write` but do not know which section they are in. You can search all man pages for the string `write` by typing

```
man -k write
```

Each man page that discusses `write` will be listed with a short description. Two lines of the output would be

```
write (1)           - send a message to another user
write (2)           - write to a file descriptor
```

You can use the `-k` option if the term you are searching for is not contained in the name section. For example, say you want information on how to change your password. If you type `man password` you get the error

```
No manual entry for password
```

However, if you type `man -k password`, `man` searches the short description and displays a listing of all man pages that contain the string `password` in its short description, as in the following results:

```
chage (1)       - change user password expiry information
chpasswd (8)    - update password file in batch
conflict (8)    - search for alias/password conflicts
crypt (3)       - password and data encryption
fgetpwent (3)   - get password file entry
getpass (3)     - get a password
getpw (3)       - Re-construct password line entry
```

Because none of the descriptions actually give you the syntax for what you do to change your password you can try `man -K password`. This causes `man` to search each man page for the term `password` and display the title of the page with a query asking if you want to view that page.

```
/usr/man/mann/entry.n? [ynq] n
/usr/man/mann/TclX.n? [ynq] n
/usr/man/mann/interp.n? [ynq] n
/usr/man/man8/smbpasswd.8? [ynq] n
/usr/man/man8/smbmount.8? [ynq] n
/usr/man/man8/uucico.8? [ynq] n
/usr/man/man8/vipw.8? [ynq] n
```

If you respond with a `y`, then that page is displayed. When you quit viewing that page, `man` continues its search and presents the next match until you tell it to quit. Although this produces the most complete results, it can take a long time to finish.

Another useful option to use with `man` is `-a`. This option causes `man` to search the entire page and display each one it locates, one after another. When you quit one man page, then the next page that satisfies your criteria is displayed.

Key Concept

When searching for information, be sure you understand the `-k`, `-K`, and `-a` options. `-k` searches the `whatis` database; `-K` searches the entire contents of the `man` pages; and `-a` finds all matching entries.

The man pages are usually stored in compressed, unformatted form. Before displaying, each page needs to be uncompressed and formatted. The `man.config` file contains configuration data for `man`, including how to construct the search path for `man`; the path for various programs used by `man` such as `less`, `more`, or `nroff`; and a list of uncompressors.

A different configuration file can be used by typing

```
man -C MyMan.conf command
```

Which pager the `man` command uses is contained in the `PAGER` environmental variable. To define a different one that you want to use, edit your `.profile` file and add the line

```
export PAGER=/bin/less
```

The `man` command needs to know where the manual pages are located. This information is contained in the `MANPATH` environmental variable—the value of which comes from the `man.config` file. You can redefine this variable by adding to your `.profile` file the following

```
export MANPATH=<path to manual pages>
```

Key Concept

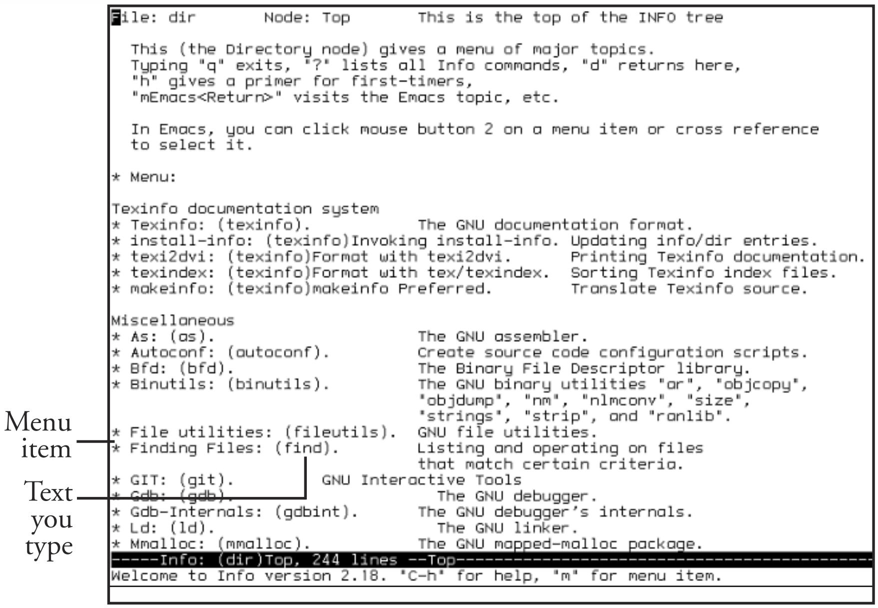
The man pages provide information on commands, utilities, and system calls. The output is displayed via a pager. The `MANPATH` or pager can be changed by editing either the `man.config` or `.profile` files.

info

`info`, which is a reader for the GNU hypertext documentation, is easier to use than man pages. `info` can be called either from `emacs`, a text editor, or as a standalone program.

Although the information provided by `info` is often much easier to understand, navigation can be difficult. It uses key combinations like those used in `emacs`.

If you start `info` by typing `info` you are presented with the screen shown in Figure 1.1, which presents a directory of available documentation.



```
File: dir       Node: Top       This is the top of the INFO tree

  This (the Directory node) gives a menu of major topics.
  Typing "q" exits, "?" lists all Info commands, "d" returns here,
  "h" gives a primer for first-timers,
  "mEmacs<Return>" visits the Emacs topic, etc.

  In Emacs, you can click mouse button 2 on a menu item or cross reference
  to select it.

* Menu:

Texinfo documentation system
* Texinfo: (texinfo).           The GNU documentation format.
* install-info: (texinfo)Invoking install-info. Updating info/dir entries.
* texi2dvi: (texinfo)Format with texi2dvi.      Printing Texinfo documentation.
* texindex: (texinfo)Format with tex/texindex.  Sorting Texinfo index files.
* makeinfo: (texinfo)makeinfo Preferred.        Translate Texinfo source.

Miscellaneous
* As: (as).                     The GNU assembler.
* Autoconf: (autoconf).         Create source code configuration scripts.
* Bfd: (bfd).                   The Binary File Descriptor library.
* Binutils: (binutils).         The GNU binary utilities "ar", "objcopy",
                                "objdump", "nm", "nlmconv", "size",
                                "strings", "strip", and "ranlib".
* File utilities: (fileutils).  GNU file utilities.
* Finding Files: (find).        Listing and operating on files
                                that match certain criteria.
* GIT: (git).           GNU Interactive Tools
* Gdb: (gdb).                     The GNU debugger.
* Gdb-Internals: (gdbint).      The GNU debugger's internals.
* Ld: (ld).                       The GNU linker.
* Mmalloc: (mmalloc).           The GNU mapped-malloc package.
-----Info: (dir)Top, 244 lines --Top-----------------------------------------
Welcome to Info version 2.18. "C-h" for help, "m" for menu item.
```


Figure 1.1
The introductory screen is displayed when starting `info`.

The top part of the screen lists some helpful keys followed by the menu. Each menu item begins with an asterisk (*), followed by the description of the menu item, and ending with a colon (:). After the colon is the text of the menu item in parentheses, as shown in Figure 1.1. It is this text that you type to go to that menu item.

You use the spacebar to scroll through the list of available menu items. Typing the letter `m` places your cursor at the bottom of the screen where you can enter the menu item. After typing `<enter>`, you are taken to that menu item.

For example, if you want to get information on finding a file, you can go to it by typing `mfind` while in `info` or invoke `info` with the word `find` as an argument, such as

```
info find
```

Either method will bring you to the screen shown in Figure 1.2.

Figure 1.2
The `info` screen on the topic find lists the sections available in the `find` documentation.

As you can see, there is no menu text on this screen that you can type to go to one of the menu items. However, you can use the Tab key to highlight the topic of interest and then press Enter to go to it.

`info` also has a tutorial that is well worth your time if you want to get the full value of `info`. From the main screen, type `h` to start the tutorial.

Navigating through the information is fairly straightforward. The spacebar scrolls down one screen and the Delete key scrolls back one screen. You can jump forward to the next topic by typing `n` and jump back by typing `p`. Typing `l` will return you to your last location.

Key Concept

The `info` utility provides much the same information as the man pages; however, it is usually easier to understand. The spacebar scrolls up and the Delete key scrolls down. Type `n` for the next topic and `p` for the previous one. Typing `q` will quit.

HOWTOs

Linux HOWTOs are detailed documents that describe the methods of performing tasks in Linux. For example, you may need to find out how to configure a piece of hardware or how to complete a simple task such as printing a document.

A HOWTO document usually covers a complex subject and as such tends to be rather long. Less complex subjects such as LILO and printing are covered in mini HOWTOs.

Versions (usually compressed) of HOWTOs are usually installed in the `/usr/doc/HOWTO` directories. Check online to find the latest version(s).

You can download the latest version of a HOWTO via ftp (from `ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO` or one of its mirrors). They also can be browsed via HTML from several sites, including `http://sunsite.unc.edu/LDP/HOWTO/` or `http://www.linuxdoc.org/HOWTO/`.

Mirror sites can be located by consulting `http://sunsite.unc.edu/LDP/mirrors.htm`. These documents are also available in several different languages.

Key Concept

HOWTOs are written by users to describe how they solved a problem. They are easier to use than man pages and `info` and cover different topics. Because HOWTOs are frequently updated and new ones published, you should periodically verify that you have the latest versions.

HOWTOs are easier to understand than using man pages or info and frequently address issues that are not discussed in other locations. They include topics that other users have researched and published that would be useful to other users.

For example, are you trying to run the game Quake on Linux? Checking the man pages and `info` does not help. However, there is a HOWTO that tells you how to do it.

FAQ

FAQ (Frequently Asked Questions) is a collection of documents that are intended to be read in conjunction with the HOWTO documents. They are written in a question-and-answer format, addressing those questions that users ask most often.

FAQs are available in HTML, PostScript, and text formats. Copies of the FAQs in all three formats are usually installed in the `/usr/doc/FAQ` directory by almost all distributions. Check online to find the latest version(s) at `http://linuxdoc.org/FAQ/`.

Program Documentation

Text files that provide more information about specific programs can be found in `/usr/doc` in subdirectories named for the program and version number. These provide information on installation, configuration, and use of the application. You can usually read them with either `less` or `more`.

In addition, applications usually install a manual page that discusses how to use the program. These are found by using the `man` command discussed earlier in this chapter.

Internet Resources

As you know, all the source code for Linux is available on the Internet for download. Linux documentation is also online in many locations. The primary source of information related to Linux is the Linux Documentation Project at `http://metalab.unc.edu/LDP/`. Here you'll find all the HOWTOs, mini-HOWTOs, and FAQs. This site also provides the ability to search all of the material from one place.

The Linux Documentation Project also has made available several guides that cover specific topics ranging from the kernel to network administration. They are available from `http://sunsite.unc.edu/LDP`. Available guides include

The Linux Documentation Project Manifesto, by Matt Welsh

Installation and Getting Started Guide, by Matt Welsh

The Linux Kernel, by David Rusling

The Linux Kernel Hackers' Guide, by Michael K. Johnson

The Linux Kernel Module Programming Guide, version 1.1.0 by Ori Pomerantz

The Network Administrator's Guide, by Olaf Kirch

The Linux System Administrators' Guide, by Lars Wirzenius

The Linux Users' Guide, by Larry Greenfield

The Linux Programmer's Guide, by Sven Goldt, Sven van der Meer, Scott Burkett, and Matt Welsh

A more recent project to make Linux documentation more user-friendly is under way by Linuxnewbie.org (`http://www.linuxnewbie.org/`). Their goal is to create user-friendly documents out of existing HOWTOs, FAQs, and other resources.

Newsgroups

A wealth of information is available via many Usenet newsgroups. The `comp.os.linux.announce` newsgroup is a moderated group that contains up-to-date information about software updates, new ports, user group meetings, and commercial products. Submissions to this group should be emailed to `linux-announce@news.ornl.gov`.

`comp.os.linux.answers` contains all the FAQs, HOWTOs, and other important documentation. HOWTOs are posted near the beginning of the month. You then can use the NewstoHOWTO program available from `ftp://sunsite.unc.edu/pub/Linux/system/news/misc/` to assemble the posts.

Other groups worth reading are in the `comp.os.linux.*` hierarchy. Often, you can have a problem that is not in a HOWTO or FAQ but is answered in one of the following newsgroups:

`comp.os.linux.setup`

`comp.os.linux.hardware`

`comp.os.linux.networking`

`comp.os.linux.x`

`comp.os.linux.development.apps`

`comp.os.linux.development.system`

`comp.os.linux.advocacy`

`comp.os.linux.misc`

Most newsgroups are archived at `http://www.deja.com` and `http://www.altavista.com` where you can search postings relevant to a specific topic or forum.

Mailing Lists

Several mailing lists are available on the subject of Linux. Some of these are generic while others apply to a specific distribution.

You can join a mailing list mirror of `comp.os.linux.announce` by sending email to `Linux-Announce-Request@NEWS-DIGESTS.MIT.EDU` with the word `subscribe` in the body of the message. Sending an empty message to the same address will get you help on using the mailing list (including instructions on how to leave).

There are several Linux mailing lists hosted at `majordomo@vger.rutgers.edu`. Send a message with the word `lists` in the body to find out what lists are available. Sending a message with the word `help` will get you instructions for subscribing and unsubscribing to the lists. Most of these lists are technical and are not intended for new users' questions.

Be sure to follow the rules of Net etiquette. Don't pose a question to a newsgroup or mailing list until after you have researched it. Often your question has already been posed and answered. Check the documentation (HOWTOs, FAQs, and man pages) first, then check the archives at `http://www.deja.com/` before you post.

If you do post, include as much relevant information about your system and your problem as possible, such as general information on the version(s) of your software and a brief summary of your hardware configuration. Learn to appreciate the value of hacking your system, and how to fix problems yourself.

Searching for Information

There are many other ways to find information about commands when you need help. Some of the most frequently used commands are `whereis`, `whatis`, `which`, and `apropros`.

whereis

The `whereis` command is used to find files but also will show you the location of the file's binary, source files, and any associated manual pages. One limitation of `whereis` is that the search is limited to known directories that are coded into the program. Its syntax is

```
whereis [options] filename
```

If you wanted to see where to find `passwd` or its related man pages, you would type

```
whereis  passwd

passwd: /usr/bin/passwd /etc/passwd /etc/passwd.OLD /usr/man/man1/passwd.1
/usr/man/man5/passwd.5
```

If the file is not found then only the searched for filename is displayed.

```
whereis foo

foo:
```

To find only the binary, use the `-b` option; to find only the man pages use `-m`; or to find only the source files use `-s`.

which

Use the `which` command to find where a command is located. It searches your path until a match is found. If a match is not found, `which` displays an error message.

```
which command
```

Multiple commands can be passed on the same command line. The `which` command is helpful when a command is not responding as you expect. Occasionally, there might be more than one command with the same name in different locations. Using `which` verifies that you are running the command you thought you were. After a match is found, it quits the search and displays the full path to it.

whatis and *apropos*

Both `whatis` and `apropos` derive their information from the man pages.

whatis

The `whatis` command displays a summary line about a program that is derived from the program's manual page.

```
whatis whereis

whereis (1) - locate the binary, source and manual page files for a command
```

This information is stored in a database called whatis located in the `/usr/man/man1` directory and is created/updated by using the command `makewhatis`.

apropos

This command helps out when you cannot remember how to do something. You can search for almost anything. Say you can't remember how to change a user's password; you could type `apropos password to find out how to change it.`

The difference between `apropos` and `whatis` has to do with how they perform the search. `apropos` searches for the argument string anywhere on the line—similar to `man -k`.

On the other hand, `whatis` (equivalent to `man -f`) tries to match a complete command name only on the part before the hyphen. Consequently, `whatis cc` will report if there is a cc manual and remain quiet for gcc, while `apropos` would successfully find both.

Key Concept

Be sure to know the usage of `whereis`, `which`, `whatis`, and `apropos` as listed in Table 1.4. You will see all four on the exam.

Table 1.4 The Usage of `whereis`, `which`, `whatis`, and `apropos`

`whereis`	Returns the path to the command
`which`	Tells you which command is being executed
`whatis`	Gives a short description of what a command does
`apropos`	Searches short descriptions from man pages

Writing Documentation

One of the softer skills that you will be tested on is your ability to document your system. This is the one task that most people like the least; however, it is one of the most important tasks that administrators do. You can expect to see one or two questions on the exam covering the importance of system documentation.

The first step in creating system documentation is to create and maintain a system log. For each computer you should record type of hard drive controller, number and size of disks, partitioning scheme, hostname, IP address, attached peripherals, and special keystrokes or commands for the machine such as how to configure the BIOS.

Whenever a problem occurs, you should document the details of the problem, what you tried, and what finally worked. For example, if you have a `cronjob` that fails, you should record the symptoms of the failure and what you did to correct it. That way when the same problem occurs at a later date, you will be able to solve it in a minimum amount of time.

As changes are made to each computer, record what they are and why they were made. This should include new hard disks, new partitions, applications you install, changes to configuration files, and so on. In other words, record everything you do. This documentation should be complete enough that the system can be rebuilt in a minimum amount of time.

You also will need to write documentation on how to use the systems at your site—both from an administrator's point of view as well as from a user's. You might want to write a document telling users how to log on or how to run a specific application. You should also have documentation on how your system is constructed that another administrator could use to rebuild or troubleshoot your system.

The next question is what format to use for this documentation. There are two formats, either electronically or on paper. Table 1.5 compares the pros and cons of each format.

Table 1.5 Comparisons of Paper Versus Electronic Documentation

Type	*For*	*Against*
Electronic	Easy to update and search. Easy to include command output and screen shots.	No access if computer is down. Hard to carry around.
Paper	Available if system is down. Can include screen shots and command output.	Hard to update or search. Can be messy and hard to read.

If you chose to keep electronic documentation, be sure to plan its format so that you can find the information. If you elect to use the paper format, organize it as well. Documentation is not helpful if you cannot find the right entry.

User Support

Providing user support can be one of the most challenging tasks you ever have to perform. This task requires not just technical proficiency but also good interpersonal and communication skills. You must be able to present information to the user in such a way that he understands what you are saying. But you must also not make the user feel as though you are talking down to him.

Because providing user support is time-consuming, you need to be efficient without giving the impression that the request for help is a waste of your time. One of the best ways to be efficient as well as helpful is by having an organized approach to every problem.

You may receive requests for technical support in one of several ways. Perhaps you have a method of submitting trouble tickets or the user calls you on the phone. No matter how you are approached, your ultimate goal is to make the user more productive.

You also should respond to all requests for assistance in a timely manner. Be sure to determine how urgent the problem is and take that into account. You also should notify the user if you cannot address her problem immediately and give her an estimate of when you will be available.

First, you must be able to clearly identify the parameters of the problem. Determine when and how the problem came about. You always will want to ask when the problem first appeared. It is possible that you might get a trouble call that is actually a request for new functionality.

This is where that documentation can come in handy. After you have identified the problem, you will want to check your documentation to see if this problem has been addressed before. If you are lucky, you will have the solution right before you.

If your contact is via the phone, you can ask questions to better define the problem. When a problem is submitted in writing, important information might be missing. An exchange of emails might be necessary in order to define the problem appropriately.

When communicating with a user, be sure to be courteous. Avoid the use of jargon; instead, use language the user will understand without talking down to him. If written instructions are available, be sure and share them as is appropriate.

Remember, the trouble ticket is not closed until the user says the problem is solved. Even though you think you have answered the user's questions and that the problem is solved, the user might not feel that way. If you need to research the problem further, or ask another administrator for help, tell the user what you are going to do and when you will get back with her. Making the user feel as though she is part of the solution and not just the problem will go a long way in making your job easier.

Summary

By now you are comfortable with using the various forms of locally installed documentation. When preparing for the exam, be sure to understand the difference between `man -k` and `man -K`. The `man -K` command searches the entire contents of the `man` page while `man -k` searches only the description section of the `man` page.

Searching within a man page is accomplished using `/`; and continued with `n`.

You should know when to use `info` versus the HOWTOs and FAQs and how to keep the last two up-to-date. Understand the usefulness of newsgroups and mailing lists while abiding by the appropriate rules of Net etiquette.

You should also understand the differences between `whatis`, `which`, `whereis`, and `apropos`. The `whatis` command searches the `whatis` database and displays a short description of what the command or utility does. The `which` command lets you know what version of a command you are running and its location. The `whereis` command gives the location of the command, its source, and its manual pages. The `apropos` command displays the name of the man page that has your term in the description. You can also expect to see questions on maintaining the various types of documentation and dealing with users.

QUESTIONS AND ANSWERS

1. You ask a co-worker how to locate a file on your Linux system and he tells you to check the `man` pages for `find` and `locate`. What does he mean by the term `man`?

A: When he mentioned the `man` pages he was referring to the installed manual pages that document every command, utility, and application installed on your system.

2. You are looking at the man page on using `tar` to find the option to compress the `tar` file. How can you find it without having to read the entire page?

A: You can search for a term on the page by typing `/compress` and pressing the Enter key, which will take you to the correct option.

3. Why does using `man command` not find the correct manual page?

A: There are two different man pages on the command. Because `man` displays the first page it finds, that page might not be the page you are looking for.

4. You have a user who wants to use manual pages in Spanish. How can you configure his computer to make this the default?

A: First, install the Spanish version of the man pages on either a server or the user's computer. Next, edit the user's `.profile` file to set the `MANPATH` variable to this new location.

5. Where can you find information on setting up your new Zip drive?

A: Searching the man pages does not result in any information. You should check the HOWTOs—those installed on either your system or the Internet.

6. What is the Linux Documentation Project?

A: The Linux Documentation Project oversees all Linux documentation, including manual pages, HOWTOs, FAQs, and tutorials, in an effort to maintain its reliability and accuracy.

7. You find a file in `/usr/bin` but do not know what it does. How can you find this out?

A: Use the `whatis` command, which provides a one-line summary of the command's function.

8. How do you find the exact name of the command to change your password?

A: Use the `apropos` command to search the man pages for the password command. `apropos` will then list the man pages that contain it in the headers. If your result is not successful, use `man -K password` to search the entire contents of each man page.

PRACTICE TEST

1. How can you find all the man pages that discuss compression?
   a. `apropos compression`
   b. `man -k compression`
   c. `info compression`
   d. `man 3 -K compression`

Answer a is incorrect because this would produce a listing of all the man pages that have "compression" in the header. Answer b is incorrect; this would produce a listing of all the man pages that have "compression" in the header. Answer c is incorrect; `info` is a separate collection of documentation from the manual pages. **Answer d is correct; the `-K` option causes `man` to search the entire contents of each man page.**

2. When using `info` how could you go to the menu item `find`?
   a. Type `f` and press Enter.
   b. Type `mfind` and press Enter.
   c. Click `find` with your mouse.
   d. Press Alt+f to select `find`.

Answer a is incorrect; typing `f` would not select the `find` menu item. **Answer b is correct; typing `m` places your cursor at the bottom of the screen where you can type the menu item and then press Enter.** Answer c is incorrect; `info` does not let you click on menu items to select them. Answer d is incorrect; the Alt, or metakey, is not used to select menu items.

3. Where could you find additional information on the boot process?
   a. Check for a HOWTO on the boot process.
   b. Post a message on a newsgroup asking for more information.
   c. Look in the `/usr/doc/boot` directory.

Answer a is correct; the HOWTOs discuss topics rather than specific commands. Answer b is incorrect; newsgroups are not for researching a problem, but rather for posing specific questions that have not been previously answered. Answer c is incorrect; the `/usr/doc` directory has a subdirectory for each installed application and boot is not an application.

4. How could you find information on how to configure `trn` (a newsreader program) that is installed on your system?
   a. Type `trn --help`.
   b. Perform an online search at the Linux Documentation Project site.

c. Check the /usr/doc/trn.version directory.
d. Send an email message to the author.

Answer a is incorrect; the --help option is used with commands, not applications. Answer b is incorrect; the LDP site provides search capability when looking for information on Linux, not applications. **Answer c is correct; application documentation is stored in the /usr/doc directory in a subdirectory named for the application and its version**. Answer d is incorrect; the documentation should be consulted first because it was written by the author of the application.

5. You use whereis to search for a file you created last week but do not find it. Why?
   a. You must have deleted it.
   b. The whereis command did not search the correct directory.
   c. You did not pass the correct directory to search to whereis.
   d. You incorrectly spelled the name of the file.

Answer a is incorrect; although you may have deleted the file, this is not the problem. **Answer b is correct; the whereis command searches a predefined list of directories that do not include your home directory.** Answer c is incorrect; you cannot define the directory in which whereis searches. Answer d is incorrect; although you may have misspelled the name, that is not why the search failed.

6. When running a command, the results are unexpected. How do you find out what command is being executed?
   a. Use the which command.
   b. Use the what command.
   c. Use the man command.
   d. Use the whatis command.

Answer a is correct; the which command will indicate which version of a command is being run. Answer b is incorrect; there is no what command. Answer c is incorrect; the man command will display the appropriate man page. Answer d is incorrect; the whatis command will tell you what the command does.

7. How can you get a summary of what the who command does?
   a. Type who --help.
   b. Type whereis who.
   c. Type whatis who.
   d. Type man who.

Answer a is incorrect; the --help option presents help on syntax. Answer b is incorrect; the whereis command gives the path to the command. **Answer c is correct; the whatis command gives a short description of the command's function.** Answer d is incorrect; this would show the man page on who, but you are looking for a short description of the command's function.

8. What would be an equivalent command to `apropos sort`?

   a. `man sort`

   b. `man -K sort`

   c. `man -k sort`

   d. `man -f sort`

Answer a is incorrect; this would display the first man page on sort. Answer b is incorrect; this would search the content of all man pages for the word `sort`. **Answer c is correct; like `apropos`, `man -k` searches the header of the man pages for the search string.** Answer d is incorrect; `man -f` would search for only a complete command name.

CHAPTER PREREQUISITE

Before beginning this chapter you should have a solid understanding of using documentation as covered in Chapter 1, "Finding and Creating Documentation." In addition, you need to have a working system that successfully boots into Linux.

CHAPTER

2

Booting Linux

WHILE YOU READ

1. How can you boot multiple operating systems on your Linux computer?
2. What is the purpose of the file `vmlinuz-2.0.36-0.7`?
3. How should you approach doing some system maintenance work?
4. What is the purpose of the file `/etc/inittab`?
5. You install a UPS on your computer. What else should you do?
6. When booting your machine, you get `LIL` and then your system hangs. What could the problem be?
7. Can you use Ctrl-Alt-Del to reboot your computer?
8. You need to bring down your system to install a new hard disk. What should you do?

Starting Linux

You can start your Linux system just by turning on the power, using a program that is a boot loader, or booting from a floppy disk. When you power up your computer, certain tasks are performed during the power up phase. This period is referred to as POST: power on self test. After this has happened, the operating system can be loaded. A *boot loader* is a program such as `LILO` that manages the loading of your kernel. `LILO` is covered later in this chapter.

The method of starting Linux is dependent on how you have configured your system. No matter how you start your system, the steps you take to get the operating system up and running are basically the same.

Key Concept

Understanding the process of booting Linux is necessary in order to troubleshoot any problems you might have during the boot process. These concepts will be covered on the exam.

The kernel is located either on your boot floppy or on the hard drive in the `/boot` directory. The compressed kernel is usually called `vmlinuz-`*`version-number`*. However, the name and location are not important—only that the kernel can be located upon booting is.

First, the kernel is loaded into memory. Usually, your kernel is compressed; however, the code necessary to uncompress it is not compressed. You can use the `gunzip` utility to uncompress your kernel. See Chapter 11, "Backup and Restore," for information on this utility. Several parameters are included in the kernel image that you are loading, including where the root filesystem is located.

While loading, the kernel prints messages to the display and saves them to the `/var/log/messages` file. You can review these messages by looking at the log or by using the `dmesg` command. `dmesg` takes no arguments and prints out the last messages in the kernel message buffer. These messages include

- The console type and font
- Detection of PCI bus and any PCI cards present
- An estimate of the processor speed
- Amount of available system memory
- CPU type
- The version number of the kernel

- Any device drivers that are loaded
- The amount of swap space
- Network adapters and settings

See Chapter 10, "Administrative Tasks," for more information on this log file and using the `dmesg` command.

After the kernel has started, it mounts your root filesystem, usually located on your hard disk. After mounting the root, the control is handed over to the hard disk with the kernel remaining in memory.

The init Daemon

The last thing the kernel does is invoke the init daemon, which stays active until you shut down your system. It is responsible for creating processes that the rest of the system will use, such as login shells. The init daemon will also restart certain processes when they exit. For example, when you log out, init restarts the console so you can log in again.

The init daemon's actions are controlled by its configuration file, `/etc/inittab`.

The */etc/inittab* File

Each line of the `/etc/inittab` file contains four fields separated by a colon:

```
ID:runlevel:action:process
```

Table 2.1 shows what each field contains.

Table 2.1 Fields and the Purpose of the Lines in the **/etc/inittab** File

Field	*Contents and Purpose*
`ID`	One or two characters identifying the entry, usually the device name
`runlevel`	Indicates to which runlevels this line will apply; multiple runlevels can be listed; if blank, will apply to all runlevels
`action`	How to handle the entry; see Table 2.2 for valid entries
`process`	Command to execute

The action field indicates how to handle the command, such as whether to restart the command if it stops. Table 2.2 shows the valid entries for this field.

Table 2.2 Valid Entries for the `action` Field in the `/etc/inittab` File

`boot`	Runs when `/etc/inittab` is first read
`bootwait`	Runs when `/etc/inittab` is first read after the boot entries
`initdefault`	Sets initial runleveloff; stops the process if it is running; starts process only once after
`ondemand`	Keeps process running; restarts if stops
`powerfail`	Executed upon receipt of a power failure signal
`sysinit`	Execute before accessing console
`respawn`	Keeps process running; restarts the process if it stops
`wait`	Starts process once

The `init` process is used to start the kernel and any processes it needs to get your system up and running. If you want an application to start every time your system boots, enter that command in one of the `rc` scripts, which are discussed in the next section, "Runlevels."

Runlevels

A *runlevel* defines a set of processes to start as part of booting the system. This can range from a minimal amount of processes such as those used for system administration to a configuration supporting all configured devices.

The numbers associated with various runlevels differ depending on the distribution you're using and are listed in the `/etc/inittab` file. For example, the listing might look like

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
```

The runlevels 0 and 6 are reserved only for `halt` and `reboot`, respectively. The runlevels can vary from distribution to distribution. No matter what distribution you have installed, the init daemon processes only those lines associated with the current runlevel.

The single-user mode is a little different from all the rest of the available runlevels. This runlevel is often used for system administration tasks such as recovering a corrupt filesystem. No other users can log in while the system is in this runlevel. However, you can still run multiple applications at the same time. This is the only time that init does not read the `/etc/inittab` file. Rather, the default action is to run the `/bin/su` command and you are logged in as the superuser or root.

When starting at a runlevel higher than the single-user mode, the system starts up in multiuser mode. In this mode, init executes those lines with an action of `sysinit`, `boot`, or `bootwait`. Frequently, these lines mount the filesystems.

Next, the init daemon will execute all the other lines that are listed with the present runlevel in the runlevel field.

Your initial runlevel is defined by the line

```
id:3:initdefault:
```

Notice that this line does not define any processes to start, but rather defines the runlevel with which to start. You can change the runlevel by using the command `init`. Its syntax is

```
init [runlevel]
```

The `init` command takes a runlevel as an argument. You can specify how long to wait before changing the runlevel by using the `telinit` command with the `-t` *`seconds`* option. The default delay is 5 seconds. The `telinit` command is a link to the `init` command. Be sure that you notify any logged on users before changing the runlevel.

For example, the command

```
init 5
```

will change the runlevel to level five. `init` first sends a signal to stop any processes running that are not defined for the new runlevel. After the specified delay, `telinit` then sends a kill signal to these processes. It then starts any processes defined for the new runlevel that are not already running.

You also can change to single-user mode by passing either `s` or `S` to the `init` or `telinit` commands.

The next section in the `/etc/inittab` file defines the startup scripts to run, as in

```
si::sysinit:/etc/rc.d/rc.sysinit
```

The ID `si` at the beginning of the line is used internally by the init daemon. This line starts the script `rc.sysinit`, which handles tasks such as activating the swap partition, running `fsck`, and mounting your filesystems. The `fsck` command checks the filesystem for corrupt entries and is discussed in more detail in Chapter 6, “Maintaining the Filesystem.” This script can vary depending on your distribution. Read the one located on your system to get the details of what it accomplishes.

The next section in the `/etc/inittab` file runs scripts appropriate to your runlevel, which are located in the `/etc/rc.d` directories:

```
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

This is accomplished by calling the `rc` script with an argument identifying the runlevel. The `rc` script then runs the scripts located in the `rcn.d` directory, in which *n* represents the runlevel. Your system can execute other startup scripts, such as `/etc/rc.local`, depending on your distribution.

Your `rc` scripts also can call *modules.* These are pieces of kernel code that can be loaded and unloaded while your system is running. These modules are most frequently device drivers for devices you can't use all the time.

You can use the `/etc/conf.modules` file to pass parameters to each module at the time it is loaded. These parameters are frequently I/O addresses and interrupts. A sample `conf.modules` file is

```
alias eth0 3c59x
alias sound es1370
```

This loads the modules necessary for the Ethernet card and sound card.

Next are the processes that should be run at every runlevel. These usually include starting a `getty` process, which creates each of the virtual terminals.

Your `inittab` file also can contain instructions as to what to do if a power failure occurs and what to do if the power comes back up. An example is

```
# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
# Run gettys in standard runlevels
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
```

```
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

The `shutdown` command is discussed later in this chapter.

The init daemon stays active after reading the `/etc/inittab` file. It monitors the system for instructions to change the runlevel or when one of the processes it started stops. If one of init's child processes does stop, it does two things:

- First, it rereads the `/etc/inittab` file. If the action is to restart the process, then it does.
- Second, it logs in the `/etc/wtmp` and `/etc/utmp` files the fact that the process terminated and why. These files are discussed in Chapter 10, "Administrative Tasks."

The init daemon will also reread the `/etc/inittab` file if it is instructed to change the runlevel or if it is notified of a power failure.

If you want to make changes to the `/etc/inittab` file, you can use any text editor to alter the file while the system is running. These changes, however, will not be implemented until the init daemon rereads the file. This occurs when

- The runlevel is changed
- A power failure occurs
- You issue the command `init q`

The `init q` command causes `init` to reread the `/etc/inittab` file.

Exercise great care when editing the `/etc/inittab` file because errors can prevent you from logging in to your system. Make a copy of the original, correct file, and a boot floppy in case you make a mistake. You might be able to recover by starting in single-user mode, but that's not always possible. Occasionally, you might make a typo when editing `/etc/inittab` that could cause your system to enter a loop by starting and stopping a process repeatedly. To prevent this from happening, the init daemon tracks how frequently it has to restart a process. If a process is restarted more than 10 times in 2 minutes, the init daemon sends an error message and waits for 5 minutes before it will restart that process.

LILO

LILO (LInux LOader) is a boot loader that can reside on your hard disk or your boot floppy. It is executed when the system boots. Features of `LILO` include the following:

- Will work with DOS, UNIX, OS/2, Windows 95/98, and Windows NT
- Replaces the master boot record on your hard drive
- Can use up to 16 different boot images on different partitions, each of which can be password protected
- Allows the boot sector file, map file, and boot images to reside on different partitions

LILO has the capability to act as a boot loader for other operating systems. It will allow you to select the operating system you wish to boot.

Key Concept

LILO provides an efficient method for booting Linux as well as other operating systems. LILO can handle up to 16 different boot images on a single hard disk. When you install LILO, you can select to overwrite the Master Boot Record or not.

If no selection is made, the default operating system is usually booted. You can bypass this action by either pressing a key (the Alt, Shift, or Ctrl key) or requesting a prompt in the `/etc/lilo.conf` file. Either of these actions will cause LILO to show a boot prompt:

```
BOOT:
```

At this point, you can type the name of the operating system to boot. If you press Tab or type `?` at the LILO boot prompt, a list of configured operating systems will be displayed. If you do not make a selection, LILO will boot the first operating system it finds.

When typing a selection at the boot prompt, you can pass options to the kernel. This is often used to start your system in single-user mode by typing

```
kernelname single
```

at the boot prompt. Other options that can be passed include `root=`*`device`*, `ro` for read-only, and `rw` for read-write.

How LILO behaves is controlled by its default configuration file, `/etc/lilo.conf`. The `LILO` command has several different uses; its syntax is

```
lilo [options]
```

To install LILO, simply call it without any options. The options that can be used with LILO are listed in Table 2.3.

Table 2.3 Options That Can Be Used with *LILO*

Option	*Action*
`-b` *`bootdev`*	Specifies boot device
`-C` *`filename`*	Use specified configuration file
`-c`	Speeds boot by merging reads from adjacent sectors
`-D` *`label`*	Use kernel with given label as default
`-d` *`deciseconds`*	Indicates how long to wait for selection
`-f` *`filename`*	Specifies disk parameter file; default is `/etc/disktab`
`-I`	Asks for path to kernel
`-i` *`filename`*	Identifies file to be used as boot sector; default is `/boot/boot.b`
`-l`	Generates linear sector addresses
`-m` *`filename`*	Uses specified map file instead of default
`-P [fix|ignore]`	Fixes or ignores corrupted partition tables
`-q`	Lists the names and locations of kernel(s) to boot from the `/boot/map` file
`-R` *`command`*	Sets default for next reboot
`-r` *`directory`*	Changes root before anything else; used for troubleshooting
`-S` *`filename`*	Overwrites existing file
`-s` *`filename`*	Specifies where to store old boot sector; if used with `-u` specifies boot sector to restore
`-t`	Tests your configuration without installing
`-u` or `-U`	Uninstalls LILO
`-v`	Causes LILO to display information; multiple `-v`'s can be used to increase the amount of information displayed

The majority of the time you will use a configuration file when calling LILO instead of using the command line. When creating the configuration file, you use keywords instead of the options shown in Table 2.3. The available keywords are listed in Table 2.4.

Table 2.4 Keywords and Their Corresponding Options to Use in the Configuration File for LILO

Keyword	*Option*
`-b bootdev`	`boot=bootdev`
`-c`	`compact`
`-d deciseconds`	`delay=dsec`
`-D` *`label`*	`default=label`
`-i` *`filename`*	`install=bootsector`
`-f` *`filename`*	`disktab=file`
`-l`	`linear`
`-m` *`filename`*	`map=mapfile`
`-P fix`	`fix-table`
`-P ignore`	`ignore-table`
`-s` *`filename`*	`backup=file`
`-S` *`filename`*	`force-backup=file`
`-v`	`verbose=level`

The default configuration file for LILO is `/etc/lilo.conf`. A sample `lilo.conf` file is

```
boot=/dev/hda                     the partition that you will boot from
map=/boot/map                     the mapping file to use
install=/boot/boot.b              the boot sector file
prompt                            display a prompt
timeout=50                        How long to wait before staring the default
image=/boot/vmlinuz-2.0.36-0.7    The name of the kernal to start
         label=linux              the label or name you assigned
         root=/dev/hda1           the partition to mount as root
```

You can pass configuration parameters to the kernel, such as `append`, `ramdisk`, `read-only`, `read-write`, `root`, and `vga`, in the options section for each operating system listed. So, if you have an operating system that you boot to troubleshoot, you could create a `ramdisk` and load everything there. You also could make a filesystem read-only when booting.

If you modify your configuration file, you will need to write it to the boot sector of the hard disk. After you have completed your changes, issue the command

```
/sbin/lilo
```

Be sure to run this command every time you make any changes to the configuration file. If you do not run `/sbin/lilo` then the changes you made will not be read. You also will need to run it if you change the configuration of your system, including changing any partitions of your hard disk.

Key Concept

Any changes made to LILO do not take effect until you reinstall LILO by executing the `/sbin/lilo` command.

Troubleshooting LILO

As LILO is loaded, it displays the word LILO one character at a time, which serves as a way to identify errors with LILO. By noting how many characters get displayed, you will have a good starting point to determine what went wrong. Table 2.5 shows the possible problems that can prevent LILO from fully loading.

Table 2.5 Using the Characters Displayed for Troubleshooting LILO

Characters	*Meaning*
nothing	LILO was not loaded; could not be installed or partition is not active
`L` *`errorcode`*	Usually a media failure
`LI`	First stage completed but second stage was not executed; can be caused by geometry mismatch or inability to find `/boot/boot.b`
`LIL`	Both the first and second stage loaders started but cannot load the map file; usually a media failure
`LIL?`	Second stage loader loaded at incorrect address; geometry mismatch or inability to locate `boot.b` file
`LIL-`	Corrupt descriptor table; geometry mismatch or inability to locate the map file
`LILO`	LILO successfully loaded

Occasionally you might see hexadecimal digits after the first L. These are randomly generated and usually indicate a temporary disk problem. If LILO loads, this is not a severe problem but you should take note of it and run `fsck` as soon as possible.

Shutting Down Linux

You should never just turn off your Linux system. If the kernel is not allowed to write memory buffers to the disk, you could end up with a corrupted filesystem.

Linux maintains the most recent changes to inode tables and the hard disk in RAM due to its faster speed. If you power off before this information is written to disk then you will lose this data.

If you power off during a write process, this can result in lost files or incorrect information on the status of disk space. The same thing can happen if active processes are writing to disk and are not stopped cleanly, resulting in corrupted data and possible disk crashes.

Key Concept

Do not ever just turn off your computer. You could end up with a computer that will not boot depending on what the operating system is doing at the time you power down.

To shut down your system, you should use the `shutdown` command. Its syntax is

```
shutdown [options] when [message]
```

The `shutdown` command terminates all processes by sending them a SIGTERM signal. Next, `/etc/init` is called to change the runlevel and unmount the filesystems.

The `shutdown` command requires time to commence shutting down the system. This can be in the format hh:mm, an amount of time to wait such as +15, or the word `now` to shut down immediately.

The message is sent to all logged on users periodically until the shutdown process begins. If a message is not specified, a default message is sent.

There are several options that can be used with the `shutdown` command. They are listed in Table 2.6.

Table 2.6 Options That Can Be Used with the `shutdown` Command

Option	*Action*
`-c`	Cancel a shutdown process that is in progress
`-f`	Do not run `fsck` on reboot
`-F`	Force `fsck` on reboot

Option	*Action*
`-h`	Halt the system after shutdown is finished
`-k`	Send a warning message but do not shut down the system
`-n`	Shut down without calling `init`
`-r`	Reboot after shutdown is complete
`-t` *`seconds`*	Indicates period of time to delay after killing processes before calling `init`

Only root can run the `shutdown` command unless the `/etc/shutdown.allow` file exists. Any user listed in this file can shut down the system.

Do not turn off the power until the shutdown process has completed and you see the message

```
The system is halted
```

Key Concept

You can control who has the ability to shut down your system by using the `/etc/shutdown.allow` file. By default, no one except root can shut down your system. This file gives you the ability to allow others to shut it down.

There are two other commands that you can use to bring down your system. They are the `halt` and `reboot` commands. The options that can be used with these commands are listed in Table 2.7.

Table 2.7 Options That Are Used with the `halt` and `reboot` Commands

Option	*Action*
`-d`	Do not write to the `/etc/log/wtmp` file; implied with `-n`
`-f`	Do not call `shutdown` but force a halt or reboot
`-i`	Shut down the network interfaces before shutting down
`-n`	Do not run sync before reboot or halt
`-p`	Do a power off after the shutdown
`-w`	Do not bring the system down, just write to the `/var/log/wtmp` file

When you call either the `halt` or `reboot` command, it first checks the runlevel. If the system is in runlevel 0 or 6, the command runs. If the system is in any other runlevel, the command calls `shutdown -nf`. Neither `halt` nor `reboot` sends a message to logged on users or delays before executing.

Because Linux runs on PCs, pressing the Ctrl-Alt-Del key combination is the same as pressing the power button, leading to an unstable system. This can be prevented by trapping this key combination and calling the `shutdown` command. The lines

```
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

in the `inittab` file does just that. Be sure to verify that this capability is included in your version of Linux before using it.

Summary

This chapter has covered some important information. You will need to understand the boot process for the test. The steps are

- Uncompress the kernel
- Load the kernel into RAM
- Write boot messages to the system log
- Call `init`
- Process any `rc` scripts

LILO is a boot loader that many people use to control the booting process for Linux. You configure it with the file `/etc/lilo.conf`. If you are using modules, these are defined in either `/etc/modules.conf` or `/etc/conf.modules`, depending on your distribution.

You need to understand the format of the `/etc/inittab` file. Its fields are `ID`, `runlevel`, `action`, and `process`. In order to make changes, first edit your configuration file, stop the `syslogd` daemon, and then restart it. You can specify a configuration file other than the default if you want.

Runlevels are a set of processes necessary for your system to run in a certain mode. The default runlevel is defined in `syslogd`'s configuration file and can be changed using either the `init` or `telinit` command.

When shutting down your system, do not just power off. Use the `shutdown`, `reboot`, or `halt` command instead. Because incorrectly shutting down your system can lead to a corrupted filesystem, you will probably see at least one question covering this issue.

You have several files that are used for saving system messages. The most important one is `/var/log/messages`. You can use the file when troubleshooting the boot process.

QUESTIONS AND ANSWERS

1. How can you boot multiple operating systems on your Linux computer?

A. You can have different operating systems installed on different partitions. You will then need some method of selecting which operating system to start. One of the most common is LILO.

2. What is the purpose of the file `vmlinuz-2.0.36-0.7`?

A. This is the compressed kernel. The numbers refer to the version and patch level.

3. How should you approach doing some system maintenance work?

A. You want to change the runlevel to single-user mode. So, you need to send a message to all logged on users telling them to log off, and then you can use either the `init` or `telinit` command to change the runlevel.

4. What is the purpose of the file `/etc/inittab`?

A. The `/etc/inittab` file is the configuration file for `init`. It provides the information `init` needs to boot the system.

5. You install a UPS on your computer. What else should you do?

A. After installing the UPS and any needed software, you will need to tell `init` what to do when it receives a power failure signal. Do this by adding the appropriate lines to the `/etc/inittab` file.

6. When booting your machine, you get `LIL` and then your system hangs. What could the problem be?

A. The `LIL` indicates the progress that LILO has made in loading. The three letters indicate that there is a corrupt descriptor table preventing LILO from locating the map file or there is a geometry mismatch.

7. Can you use Ctrl-Alt-Del to reboot your computer?

A. Yes and no. The ability to use the Ctrl-Alt-Del key sequence exists if this key combination is trapped by `init`.

...continues

...continued

8. You need to bring down your system to install a new hard disk. What should you do?

A. Changing out hardware requires a complete power shutdown of your system. Before powering down your computer, you will want to notify any logged on users that they need to log off. You can do both by using the `shutdown` command and specifying how long to wait before starting the shutdown process.

PRACTICE TEST

1. Which of the following methods cannot be used to boot Linux?

- **a.** A floppy disk
- **b.** Your hard drive
- **c.** A boot loader
- **d.** Your CD-ROM

Answer a is incorrect; a floppy disk is often used to boot your system during troubleshooting. Answer b is incorrect; of course you can boot from your hard disk. Answer c is incorrect; LILO is just one boot loader that can be used to boot Linux. **Answer d is correct; you cannot boot Linux from your CD-ROM, although installation CDs are often bootable.**

2. In which directory is the kernel stored?

- **a.** `/`
- **b.** `/root`
- **c.** `/boot`
- **d.** `/kernel`

Answer a is incorrect; this is the root of the filesystem. Answer b is incorrect; this is root's directory. **Answer c is correct; the `/boot` directory is used to store the kernel.** Answer d is incorrect; normally there is not a `/kernel` directory.

3. Which of the following is not a valid field for the `/etc/inittab` file?

- **a.** `Date`
- **b.** `Runlevel`
- **c.** `Process`
- **d.** `ID`

Answer a is correct; there is no date field. Answer b is incorrect; each line requires a runlevel. Answer c is incorrect; each line specifies the process to be run, although this is optional. Answer d is incorrect; each line needs an ID.

4. What is meant by the term runlevel?
   a. The processes to run to boot Linux
   b. The processes to run before shutting down the system
   c. The processes that are needed to run mail
   d. The processes defined in the `rc` scripts

Answer a is correct; each runlevel is a set of processes necessary to boot Linux. Answer b is incorrect; processes are stopped when shutting down, not started. Answer c is incorrect; the mail daemon is just one process that might or might not be defined for a specific runlevel. Answer d is incorrect; `rc` scripts are called based on the specified runlevel.

5. What is a module?
   a. Add-ins for applications
   b. Kernel code that can be loaded and unloaded on-the-fly
   c. The same as the `rc` scripts
   d. Processes listed in the `/etc/utmp` file

Answer a is incorrect; although applications can have code that can be run to add additional features, these are not modules. **Answer b is correct; modules provide extra functionality to the kernel rather than compiling it into the kernel itself.** Answer c is incorrect; the `rc` scripts detail processes that should be started at boot based on runlevel. Answer d is incorrect; the `/etc/utmp` file has information on terminated processes.

6. Which of the following statements are true about LILO?
   a. The bootsector file, mapfile, and boot images must be on the same partition.
   b. You can only boot four different operating systems on the same computer.
   c. It works with UNIX and Windows operating systems but not with the OS/2 operating system.
   d. It replaces the master boot record on your disk.

Answer a is incorrect; each of these files can be located on separate partitions. Answer b is incorrect; you can boot up to 16 different boot images. Answer c is incorrect; LILO does work with the OS/2 operating system. **Answer d is correct; LILO does replace the master boot record.**

7. Which of the following options is valid with LILO?
   a. `-D` *`deciseconds`*
   b. `-d` *`label`*
   c. `-C` *`filename`*
   d. `-t` *`filename`*

Answer a is incorrect; the `-D` option is used to specify the label as the default. Answer b is incorrect; the `-d` option is used to specify the amount of time to delay. **Answer c is correct; use the `-C` option to specify a configuration file.** Answer d is incorrect; the `-t` option is used to test your configuration.

8. Which of the following is the default configuration file for LILO?
   a. `/etc/conf.lilo`
   b. `/etc/lilocf`
   c. `/etc/lilo.conf`
   d. `/etc/liloconf`

Answer a is incorrect; this is not the default file. Answer b is incorrect; this is not the default file. **Answer c is correct; the default file is called `lilo.conf` and is located in the `/etc` directory.** Answer d is incorrect; this is not the default file.

9. Which of the following commands will reboot your computer in 15 minutes?
   a. `shutdown +15`
   b. `reboot +15`
   c. `halt +15`
   d. `shutdown now +15`

Answer a is correct; this tells the `shutdown` command to wait 15 minutes before starting the shutdown process. Answer b is incorrect; the `reboot` command restarts your computer. Answer c is incorrect; the `halt` command stops your computer now. Answer d is incorrect; you do not need to specify the time relative to now.

10. What does the command `shutdown -w now` do?
    a. Shuts down the system after all users have logged out
    b. Sends a message to the logged on users but does not actually shut down the system
    c. Sends a message to the logged on users before shutting down the system
    d. Nothing, because this is not valid syntax

Answer a is incorrect; the `shutdown` command does not have the ability to monitor logged on users. **Answer b is correct; the `-w` option causes the `shutdown` command to notify all users that the system is being shut down without actually shutting down.** Answer c is incorrect; although the command would send a message to logged on users, it would restart the computer. Answer d is incorrect; this is valid syntax for the `shutdown` command.

CHAPTER PREREQUISITE

Before beginning this chapter, you should have a good grasp of the concepts covered in Chapter 2, "Booting Linux." You also should be comfortable finding additional information using locally installed documentation as well as the Internet.

CHAPTER

3

Linux Filesystem

WHILE YOU READ

1. After installing a new hard disk, what is your first step in preparing this hard disk for use by your system?
2. Before installing Linux on your computer, what should you do to prepare the hard disk?
3. Although you have 2GB of free space on your hard disk, when you use `fdisk` to create the partition, you receive an error and the action fails. Why?
4. What can you do if you want to install Linux on a computer that is running Windows 95 and does not have any free space?
5. What do the terms `hda1` and `sdc3` mean?
6. After creating a new partition, what might keep you from installing a new application on it?
7. Your system will not boot and you suspect that you have a corrupted filesystem. What should you do?
8. How can you determine whether or not the system you administer has an adequate amount of disk space?

Partitions

Partitions are used to divide hard drives into sections. A hard drive must have at least one partition and can have up to four. After creating your partitions, you need to format them, which creates the filesystem your operating system uses to store files. These steps are covered in more detail later in this chapter in the section "fdisk."

There are two types of partitions that can be used to store data, primary and extended. A *primary* partition is formatted and then used to store files. In addition, a computer can be booted only from a primary partition.

Key Concept

There are two types of partitions: primary and extended. An extended partition is divided into one or more logical drives that are used to store data.

An *extended* partition, on the other hand, cannot be used to store data directly. It is only a container for another structure called a *logical drive*. An extended partition is divided into one or more logical drives. Each logical drive is then formatted and used for data storage. Logical drives can exist only on extended partitions. You cannot directly use an extended partition. Rather, you use the logical drives contained in the extended partition.

A hard disk can have up to four primary partitions. Or, it can be divided into up to three primary partitions plus one extended partition. Figure 3.1 shows two different partitioning schemes.

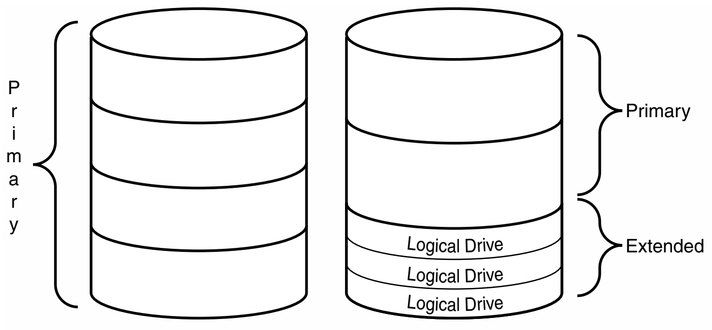


Figure 3.1
On the left is a hard drive scheme divided into four primary partitions. The right side shows a hard drive scheme divided into two primary partitions and one extended partition. The extended partition has been divided into logical drives.

Key Concept

There can be a total of no more than four partitions on each disk. These can be all primary or you can have up to three primary and one extended partition.

One more issue must be addressed before you finalize your partition plan. As mentioned in Chapter 2, "Booting Linux," when LILO is used to boot Linux, it does not recognize beyond the 1023rd cylinder of your hard drive. Therefore, if you are using LILO to boot your system, the partition where LILO resides must exist entirely within the first 504MB.

Interpreting Partition Names

Partitions are named according to the type of disk controller the hard disk contains, the partition, and which partition it is on that disk. Under DOS, the first partition of the first drive would be `disk(0) partition(1)`. Linux refers to the same partition as `hda1` if it is IDE and `sda1` if it's SCSI.

When interpreting partition names, the first letter identifies the type of controller (SCSI or IDE). The `d` stands for disk; the third letter numbers the disks (the first disk is a, the second is b, and so on); and the last character is the partition number. Table 3.1 shows some partitions.

Table 3.1 Naming Partitions Under Linux

Name	*Controller*	*Disk*	*Partition*
`hda1`	IDE controller	first disk	first partition
`sda1`	SCSI controller	first disk	first partition
`hdc3`	IDE controller	third disk	third partition
`sdb4`	SCSI controller	second disk	fourth partition

Key Concept

Partition names are based on the type of controller and partition number. Partitions on SCSI drives are referred to as `sdan` and partitions on IDE drives are named `hdan`.

How Many and What Size?

Linux will run just fine with only two partitions, root (`/`) and swap (`swap`). All files, applications, and so on, are installed on the *root* partition. The *swap* partition is used as virtual memory.

Virtual memory refers to a combination of the amount of physical RAM on your computer added to the amount of swap space you have configured. The operating system uses the swap space so it can function more efficiently.

If your system has 8MB of physical memory, a swap partition can give the operating system more memory, which results in a faster system. If you create a swap partition of 16MB, your system will have a total of 24MB of memory to use.

The type of partition you create is specified when using the fdisk utility as discussed later in this chapter. The advantages of using multiple partitions are as follows:

- If one partition is damaged, the other partitions remain isolated from the damage.
- If you have to reformat a partition, data on the other partitions are unaffected.
- Files that grow rapidly, such as log and mail files, can be kept on separate partitions to prevent them from taking all available space.
- Upgrades are easier to perform because the system and data files are isolated from each other.
- Backups are easier if system files and data files are located on separate partitions.
- The time required for filesystem checks at boot can be reduced.

Key Concept

The minimum number of partitions used to install Linux is two: one primary partition as the root (/) and a swap partition.

Your swap partition should be twice the amount of physical RAM installed on your system. The maximum size of a swap partition is 128MB, but you can have up to 16 separate swap partitions. The recommended minimum size is 16MB for the operating system to function at its best.

If space is tight, you can make the size of your swap partition equal to the amount of physical RAM, as long as it is at least 16MB.

Key Concept

A swap partition should be at least 16MB in size. Its maximum allowable size is 128MB. You can have up to 16 separate swap partitions on a single system.

Although a swap partition is more efficient, you can use a *swap file* if you do not have enough room to create a separate swap partition.

How all this is accomplished is relatively simple. Linux moves unused pages of memory out to the swap partition, allowing more applications to run simultaneously on your system. However, swapping is slower than using physical RAM. If you have 4MB of RAM or less, you will need to have a swap partition or the installation will fail.

One possible partitioning plan is shown in Table 3.2.

Table 3.2 Sample Partitioning Plan

Partition	*Contents*
`/`	Contains applications and configuration files
`/var`	Contains log files
`/usr`	Contains system software
`/tmp`	Contains temp files
`/home`	Contains users' home directories
`swap`	Acts as virtual memory

One disadvantage of multiple partitions is that you cannot enlarge a partition dynamically if it should get full. Rather, you would have to delete the partition and then re-create it with a larger size. With the larger hard drives available now, many system administrators use fewer partitions.

Plan your partitioning scheme out on paper. Although you can always blow away your partitions and start all over, having a well thought-out plan before starting the partitioning process will keep you from having to reinstall because of space constraints.

Key Concept

An advantage of multiple partitions is that you can better control the amount of space dynamic files such as log files can consume. On the other hand, a partition cannot be dynamically enlarged if it becomes full.

fips

If you are installing Linux on a computer running another operating system and there is no free space to create a new partition, you can use `fips.exe` to non-destructively divide a single FAT16 partition in two.

This program does not work with any filesystem except FAT16. Although fips usually works without losing any existing data, be sure to make a complete backup of your system before using it.

`fips.exe` can be downloaded at `http://sunsite.unc.edu/pub/Linux/system/install`. Whenever you use a program such as fips, you should always make a full backup and read the documentation before starting.

Copy `fips.exe` and `restorbb.exe` to a floppy. A copy of your old Master Boot Record (MBR) will be copied to this floppy during the installation program. This backup copy of your MBR can be used in case you ever need to restore your system to its original state. Be sure to label and date the diskette.

A copy of the old MBR also will be saved on your Linux partition as `/boot/boot.0300` (for IDE drives) or `/boot/boot.0800` (for SCSI drives). Do not *ever* delete the original file from your hard disk. If you delete it, you will not be able to restore the old MBR.

Key concept

You can use the fips utility to dynamically divide an existing `FAT` partition to create space for installing Linux.

fdisk

Be sure to keep good notes while partitioning your drive. You will want to note starting and ending cylinders, partition size in blocks, and any error messages you might get.

Cylinder refers to the total of the tracks that have the same location on each disk's surface. *Tracks* are the concentric circles on a disk.

When using fdisk, you must enter starting and ending cylinders for each partition. To create a partition, follow these steps:

1. Size the partition.
2. Designate its type.
3. Write it to the partition table.

Using fdisk and each of the previous steps are covered in the following sections.

fdisk takes the name of the disk you want to partition as an argument or value.

For example, to create partitions on the second SCSI drive in your system, you would type

```
fdisk /dev/sdb
```

If you use fdisk without an argument, it assumes you want to partition `/dev/hda`. Some operating systems require that they be installed on the first partition of the first disk. Because Linux does not care what disks you use, you might want to create partitions on different hard disks. Just run fdisk for each hard disk you need to partition.

When using fdisk, you can get a menu of available options by typing `m`, as in the following example:

```
Command (m for help): m
Command action
   a   toggle a bootable 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   extrafunctionality

Command (m for help):
```

If you create logical drives, you must reboot your system and go through the same options as before. However, when fdisk asks if you want to partition your drive the second time, say no.

You should use the Linux version of fdisk only to create or delete partitions for the Linux operating system. If you try to create or delete partitions for other operating systems with this version of fdisk, the other operating system will not recognize the partition.

After creating your partitions, you need to activate your swap partition(s), select the partition to be used as root, and format your partitions.

Key Concept

The fdisk utility can be used to create or delete partitions. It also can be used to view your partition table.

Let's walk through creating a couple partitions. If you have any existing partitions you will not be using, you can use the `d` command to delete them.

Step 1: Create a Primary Partition

First, type `n` to create a new partition. The Linux root partition will be 80MB in size. The following is an example:

```
Command (m for help):  n
Command action
     e   extended
     p   primary partition (1-4)
```

When creating a new partition, you are first asked whether you want to create a primary or extended partition.

> ## Key Concept
> If you have existing partitions on your drive, your options might be different. Remember, you can have only a maximum of four primary partitions or three primary partitions and one extended partition per hard disk. If you already have four partitions, you will not be able to create a new partition. In addition, if you already have an extended partition, you will not have this option.

For your first partition, you must create a primary partition because operating systems can boot only from primary partitions. After entering the `p` command, you will be prompted for the partition number and starting and ending cylinders.

```
Partition number (1-4): 1
First cylinder (0-788):  0
Last cylinder or +size or +sizeM or +sizeK (0-788): +80M
```

You can specify either the ending cylinder or the size of the partition. In our example, we are instructing fdisk to create an 80MB partition to use as our first partition. If you type `+80k`, the partition will be 80KB in size. Typing `+80` creates an 80-byte partition.

Now, when you issue the `p` command, fdisk displays the following:

```
Command (m for help):   p

Disk /dev/hda: 255 heads, 63 sectors, 788 cylinders
Units = cylinders of 16065 * 512 bytes
      Device Boot  Begin   Start     End   Blocks   Id  System
   /dev/hda1   *       1       1     459  3686759   83  Linux native
Command (m for help):
```

Step 2: Create a Swap Partition

Next, you need to create a swap partition. When creating successive partitions, they should start with the next cylinder after the last one ended. Because our primary partition ended at `459`, we will start the next one at `460`:

```
Command (m for help): n
Command action
     e   extended
     p   primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (460-788):  460
Last cylinder or +size or +sizeM or +sizeK (474-683):  +128M
```

Display your partition scheme again using the `p` command and write down this information, especially the size of each partition in blocks.

```
Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 788 cylinders
Units = cylinders of 16065 * 512 bytes

      Device Boot   Begin    Start     End   Blocks   Id  System
   /dev/hda1   *        1        1     459  3686759   83  Linux native
   /dev/hda2          460      460     475   128488   83  Linux native
```

In our example, the swap partition, `/dev/hda2`, is listed as a `Linux native` partition. To complete the creation of a swap partition, you must change the type to `swap` by using the `t` command:

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 788 cylinders
Units = cylinders of 16065 * 512 bytes

      Device Boot   Begin    Start     End   Blocks   Id  System
   /dev/hda1   *        1        1     459  3686759   83  Linux native
   /dev/hda2          460      460     475   128488   82  Linux swap
```

If you typed `L` instead of `82`, you will see that Linux swap is type `82` and Linux native is `83`. The type for extended partitions is `5`.

Be sure to review your partition table after each change to ensure that the changes you made are what you intended.

It is customary to create your primary and swap partitions before creating an extended partition. However, you can create a primary partition after creating an extended partition as long as there is free space available and you have not reached the four-partition limit.

Step 3: Create an Extended Partition

Next, you should create an extended partition. Use the following:

```
Command (m for help):  n
Command action
    e   extended
    p   primary partition (1-4)
e
First cylinder (476-788):  476
Last cylinder or +size or +sizeM or +sizeK (474-788):  788
```

When your partition table is displayed again, you'll see the following:

```
Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 788 cylinders
Units = cylinders of 16065 * 512 bytes

     Device Boot  Begin   Start     End   Blocks   Id  System
  /dev/hda1   *       1       1     459  3686759   83  Linux native
  /dev/hda2         460     460     475   128488   82  Linux swap
  /dev/hda3         476     476     788  2513751    5  Extended
```

Next, you will need to create logical drives in your extended partition. When you issue the `n` command, you will have a new choice:

```
Command (m for help): n
Command action
   l   logical (4 or over)
   p   primary partition (1-4)

Command (m for help):
```

You can elect to divide your extended partition into one or more logical drives by using the `l` command and then specifying the appropriate size. The starting and ending cylinders for each logical drive must be completely contained within the starting and ending cylinders of the extended partition.

After you have finished defining your partitions, you can quit fdisk and save the changes to the partition table by using the `w` command. If you decide you don't want to make these changes, use the `q` command to quit fdisk without saving your changes.

Step 4: Assign Filesystem Names

Next, you must associate the appropriate Linux filesystem names with each of the partitions you are going to use. This is done through the installation program for your specific distribution. Remember, you can install Linux with only the `/` and swap partitions.

The Linux Filesystem

The Linux filesystem organizes files and directories into a hierarchical structure. A filesystem provides a way to store files that can be randomly accessed, including hard disk partitions, floppy disks, and CD-ROMs. Because tape drives are accessed sequentially, they do not contain a true filesystem. Some of the common filesystems, not all of which are Linux filesystems, are

- Extended Filesystem—ext (Linux, has been replaced by ext2)
- Fast Filesystem—ffs (Amiga)
- HPFS Filesystem—hpfs (used with the OS/2 operating system; supported in real-mode only)
- ISO9660 Filesystem—iso9660 (CD-ROM)
- Linux Swap Filesystem—swap
- Minix Filesystem—minix (Minix; this was the first filesystem used for Linux)
- MS-DOS Filesystem—FAT16 msdos (DOS)
- Network File System—nfs (Data is stored on any machine on a network and access is granted via the network)
- Novell Filesystem—NCPFS (Novell servers)
- NT Filesystem—NTFS (Windows NT)
- proc Filesystem—proc (Virtual filesystem used by the Linux kernel to provide information to the user about processes)
- Second Extended Filesystem—ext2 (Linux standard filesystem)
- System V Filesystem—sysc (System V variants; commercial UNIX systems for PCs)
- Uniform Filesystem—ufs (Used by BSD, SunOS, and NeXTstep; is supported in read-only mode)

- UMSDOS Filesystem—umsdos (UNIX on MS-DOS; applied on top of a FAT16 filesystem to provide Linux functionality by creating special files; is slow)
- Virtual FAT Filesystem—VFAT (an extension of the FAT filesystem to support long filenames)
- Xenix Filesystem—xenix
- Xia Filesystem—xiafs (An old filesystem; hardly used anymore)

Within the Linux filesystem, the partition or hard drive you are accessing is invisible. Instead, each partition or hard drive is shown relative to the root of the filesystem. The root is represented as `/`.

If you have the following partitions

```
hda1        /
hda2        swap
hda3        /var
hda4        /home
```

and issue an `ls /` command, the directory listing will appear and the `/var` and `/home` filesystems will show up as subdirectories residing with the `/` directory, even though they are located on different partitions.

Formatting

Now that you have created your partitions, you must create a filesystem on each one before you can install Linux. Creating a filesystem on your hard disk is similar to formatting a floppy. You create a filesystem by using the `mkfs` command. Its syntax is

```
mkfs -t fs-type device blocks
```

`mkfs` actually calls another command based on the type of filesystem you specify with the `-t fs-type` option. The various `mkfs` commands associated with specific filesystem types are shown in Table 3.3.

Table 3.3 Commands Associated with Specific Filesystem Types Called by the mkfs Utility

Command	*Filesystem Type*
`mkfs.ext2`	Create an ext2 filesystem; same as mke2fs
`mkfs.msdos`	Create an MS-DOS filesystem
`mkfs.minix`	Create a Minix filesystem

You can use any of these commands instead of using the mkfs utility as a front end.

To create an ext2 filesystem on the first partition from the earlier example, you would issue the following command:

```
mkfs -t ext2 /dev/hda1 3686759
```

The notes you took while doing the partitioning are used here because you must provide the mkfs utility with the number of blocks on the partition you are formatting. If you did not note those numbers, you can rerun the `fdisk` command and display your partition table to find them.

When creating a filesystem, be very careful to ensure the device and block arguments are correct. If you designate the wrong device, you can destroy data on another partition. If you pass the wrong number of blocks, you might actually format part of another partition and destroy any data that is there.

The options you can use with the mkfs utility are listed in Table 3.4.

Table 3.4 Options to Use with the `mkfs` Command

Option	*Action*
`-t` *`fs-type`*	Defines the type of filesystem to create
`-v`	Displays all commands used to create the filesystem
`-c`	Checks for bad blocks before creating the filesystem
`-l` *`filename`*	Uses the named file as a list of known bad blocks

You also can use the `mkfs` command to create a filesystem on a floppy drive; however, the `fdformat` command is a better choice. It performs a low-level format to create the sector and track information on the floppy. The syntax for the `fdformat` command is as follows:

```
fdformat [option] device
```

You can use the `-n` option to prevent verification of the format. Floppy devices are usually `/dev/fd0` or `/dev/fd1`.

Key Concept

Before installing Linux, the disk must be prepared. You do this by creating your partitions, activating your swap partition, and designating the filesystem for each partition. After the partitions have been created, they must have a filesystem created by using the `format` command.

Filesystem Organization

Linux is organized in an hierarchical manner. It considers every file, directory, device, and link as a file that is placed on this structure. The most common organization for the directory structure is shown in Figure 3.2.

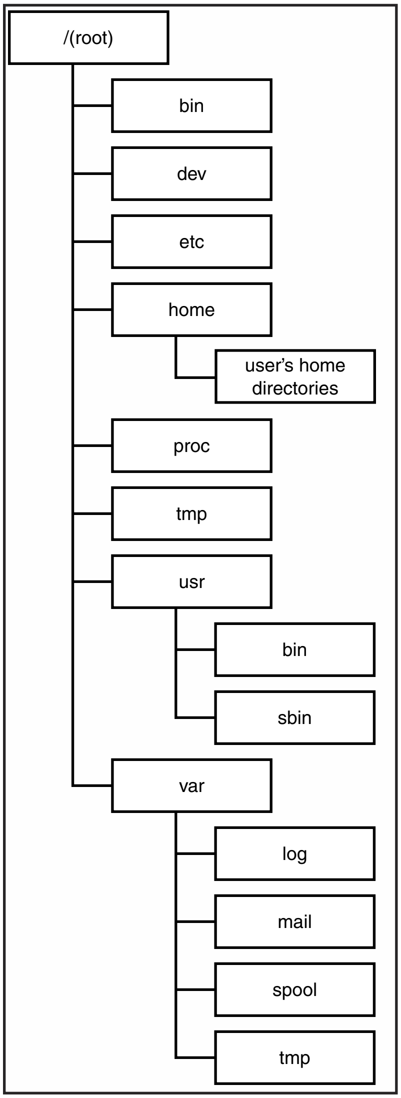


Figure 3.2
This figure shows a graphical representation of the organization of the Linux filesystem. This is independent of the actual partitions.

Each of the directories on the Linux filesystem conventionally contains certain types of files. Table 3.5 shows some of the more common directories and their contents.

Table 3.5 Linux Directories and Their Contents

Directory	*Contents*
`/`	The beginning or root of the directory tree
`/bin`	Command binaries
`/dev`	Peripheral device files
`/etc`	Configuration files for the system
`/home`	Users' home directories
`/lib`	Shared libraries
`/mnt`	Temporary partitions used for mounting
`/proc`	Virtual filesystem containing information about kernels and processes
`/tmp`	Application temporary files
`/usr`	Several subdirectories containing user commands, documentation, and other system information that does not change
`/var`	Log files and other files that change while the system is running

Inodes

The Linux operating system stores a lot of information about each file, including the following:

- Pointers to the file's physical location on the disk
- The file's name
- The file's owner and group IDs
- Rules governing access to the file
- The file's size
- The date and time the file was last accessed
- The date and time the file was last modified
- The date and time the inode was last modified
- The number of links to the file

A data structure called an *inode* is used to store the above information pertaining to each file with the exception of the file's name. The file's name is stored in its directory with an associated inode.

Each and every file has an inode associated with it. Even though every file can have more than one name through the use of links, each file has only one inode. (See Chapter 6, "Maintaining the Filesystem," for information on links.) Use `ls -l` to determine the number of the inode for any file. For example, issuing the command `ls -l MyFile` returns the following:

```
4144 MyFile
```

This means that the information about the file `MyFile` has an inode number of 4144. If an inode becomes corrupted, the file will no longer be accessible. Another way to see the inode number for a file is to use the `stat` command. Its syntax is

```
stat filename
```

Its output should look similar to the following:

```
  File: "mywhich"
  Size: 101          Filetype: Regular File
  Mode: (0775/-rwxrwxr-x)         Uid: (  503/  hadden)  Gid: (  500/  hadden)
Device:  3,0   Inode: 147474    Links: 1
Access: Tue Oct 19 16:16:58 1999(00019.19:52:38)
Modify: Fri May 28 09:02:41 1999(00164.03:06:55)
Change: Fri Sep  3 17:20:37 1999(00065.18:48:59)
```

The `stat` command reads the inode structure associated with the specified filename.

Key Concept

Each file has an associated inode that contains the file's physical location and all its attributes except for its filename. Use either the `ls -l` or `stat` command to determine a file's inode.

File Types

Linux supports several file types and uses the first character in the permission block to name the file type. Linux does not differentiate between files and directories in the inode table. The operating system knows whether the file is a regular file or a directory by the file type character. The most common file type characters used by Linux are listed in Table 3.6.

Table 3.6 The First Character of the Permission Block Identifies the File Type

Character	*Type of File*
`-`	Ordinary file
`b`	Block device
`c`	Character device
`d`	Directory
`l`	Link

Most files on the Linux system are ordinary files, including text files and applications. Any file a user creates is an ordinary file.

To Linux, directories are just empty files. They are organized in an hierarchical manner to provide holding places for other file types.

Block and character mode devices are composed of the instructions necessary for Linux to communicate with peripherals. These files are usually located in `/dev`.

Links are identified in the file type character as an `l`. See Chapter 6 for more information on links.

Keeping Your Disks Healthy

After your partitions are created and formatted and you have installed Linux, you will want to make sure they stay healthy. You can use several different utilities to determine the state of your filesystem or fix problems with it.

fsck

The most important utility for maintaining your filesystem is the `fsck` command. Like the `mkfs` command, `fsck` is a front end to the `check disk` command for the specified filesystem type. It is used to verify the filesystem's structure and repair any inconsistencies. Problems most frequently occur as a result of a system crash.

When a system crashes, the kernel is unable to sync the cache with the hard disk's contents. This can result in blocks being marked as in use when they are really empty. Other errors can occur from directly writing to a device to an area containing existing data.

The syntax for the `fsck` command is

```
fsck -t fs-type device
```

For example, to check the first partition created as an ext2 filesystem in our earlier example, you would type the following:

```
fsck -t ext2 /dev/hda1
```

The previous command calls the `e2fsck` command to check the ext2 filesystem on the first partition of the first IDE hard disk. Because this is the root partition, you will receive a message that the filesystem is mounted and be asked whether to continue. If you answer `yes`, `fsck` checks the following:

- inodes
- blocks
- file sizes
- directory structure
- links

Any partition other than root can be unmounted while your system is running. To run the `fsck` command, you must first unmount that filesystem. See Chapter 6 for more information on mounting and unmounting filesystems. To run `fsck` on your root partition, you can

- Start in read-only, single-user mode.
- Boot with a floppy and then scan the root partition.
- Run `fsck` during system boot.

When running `fsck` during system boot, the root filesystem is mounted in read-only mode, `fsck` runs, and then the root filesystem is mounted as read-write. All other filesystems are checked before they are mounted. Refer to Chapter 2, "Booting Linux," for details on booting and passing parameters to your boot loader.

Key Concept

The `fsck` command is used to verify the integrity of your filesystem. Unmount your filesystem before running it.

Options that are filesystem-specific for the `fsck` command are listed in Table 3.7.

Table 3.7 The Filesystem-Specific `fsck` Options and Their Actions

Option	*Action*
`-a`	Causes fsck to run non-interactively
`-c`	Checks for bad blocks
`-v`	Reports on fsck's progress

The filesystem keeps information about the filesystem as a whole in the *superblock*. If the superblock becomes corrupted, the filesystem cannot be mounted; however, backup copies of it are spaced at regular intervals across the filesystem. By default, these copies are saved every 8,192 blocks. Because the first block is number 1 and not number 0, copies would be stored at block 8193, 16385, 24577, and so on. To verify that this is the block group size on your system, you can issue the following command:

```
dumpe2fs device
```

The previous command produces a wealth of information about the filesystem; just look for the following line:

```
Blocks per group:          8192
```

If a different number is given, use this number to locate the copies of the superblock. You can then tell the `e2fsck` command to use a copy of the superblock to check a partition you can't mount because of a corrupted superblock. So, if you can't mount a `/dev/hda1` that has an ext2 filesystem because of a corrupted superblock, issue the following:

```
e2fsck -f -b 16385 /dev/hda1
```

In the previous command, the `-f` option is used to force the check. If you do not use the `-f` option, `e2fsck` will look at the copy of the superblock and might think that the filesystem is clean. The `-b` option identifies the copy of the superblock that is to be used.

Disk Usage

As files are created and saved to your system and log files are written to, your free disk space shrinks. It is important to monitor how much space is used on each disk. Linux runs best when there is adequate free space available.

To get optimal performance from your operating system, the amount of free space available should be 5–30 percent of each filesystem. In addition, if there is no free space, you will not be able to write to that filesystem.

Not only do you need to monitor the amount of free space, but you also will want to monitor what files are occupying the most space. By seeing what files are growing the fastest, or who is using the most space, you will be able to take steps to prevent your filesystem from becoming too full.

du

The `du` command reports on the amount of space occupied by a single file or a directory and all its files and subdirectories. The syntax for the `du` command is as follows:

```
du [options] [filename]
```

In the previous syntax, `filename` can also be the name of a directory. If `du` is called without a filename, it will report on the working directory and its contents. Table 3.8 lists the options you can use with `du`.

Table 3.8 The Options Used with the `du` Command

Option	*Action*
`-a`	Displays file usage
`-b`	Displays usage in bytes
`-c`	Prints grand total
`-h`	Appends a letter to identify the measurement being used for the report, such as M for megabytes
`-k`	Displays usage in kilobytes, the default
`-m`	Displays usage in megabytes
`-l`	Counts links
`-s`	Prints only the grand total

Unless you use the `-a` option, `du` displays only the directories, even though the space occupied by all the files is counted. If you do not have permission to a file, `du` will display an error message and not count that file when displaying disk usage.

df

Both the `du` and `df` commands report on object size. Whereas `du` reports on the size of objects such as files, `df` reports on the size of devices such as partitions. The syntax for the `df` command is

```
df [options] [filename]
```

The `df` command returns the following information values:

- Size of the device
- Number of free blocks on the device
- Number of occupied blocks on the device
- Percent of total blocks that are free
- Name of the device

If you pass an ordinary file's name to `df`, it will report on the device containing that file. If you do not name a file, `df` reports on all mounted filesystems. The options used with the `df` command are listed in Table 3.9.

Table 3.9 Options Used with the `df` Command

Option	*Action*
`-a`	Displays information on all filesystems; default action
`-h`	Appends a letter to identify the measurement being used for the report, such as `M` for megabytes
`-i`	Lists inode usage
`-k`	Displays usage in kilobytes
`-m`	Displays usage in megabytes
`--sync`	Calls the sync command before gathering information
`-t` *`fs-type`*	Displays only filesystems of *`fs-type`*
`-T`	Displays the filesystem type for each entry
`-x` *`fs-type`*	Excludes filesystems of *`fs-type`*

For example, if you type

```
df
```

your output might look like the following:

```
Filesystem          1024-blocks  Used Available Capacity Mounted on
/dev/hda1              1018298   93756   871931      10%    /
/dev/hda9              2555450   26494  2396823       1%   /home
/dev/hda7               132185      31   125328       0%    /tmp
/dev/hda5              2035606  468660  1461722      24%   /usr
/dev/hda6               256592    9435   233905       4%    /var
```

Key Concept

Use the `df` and `du` commands to determine how much free space is available.

What do you do if a filesystem is becoming too full? You will want to identify any large files that are being created but are no longer necessary. Log, mail, and news files are often big offenders. Other large files that might be deleted are core dumps.

Check whether the temp directories are not being cleaned out. In fact, this should be a regular part of your system maintenance.

Are there files that are not being used? Try compressing them and perhaps relocating them to another device. Directories also should not get too large because very large directories can affect system performance. Move files to several smaller directories.

One of the problems with directories is that they grow but do not shrink. So, if you move or delete a large number of files from a directory, you will not recover all the free space.

The only way to shrink the size of a directory is by deleting it. Therefore, copy the files you want to keep to another directory and then delete the original directory. After you've done this, you can re-create the directory with the same name and move the needed files to the new directory.

If you are outgrowing a partition, you have a couple choices for fixing the problem.

One possibility is to move some files to another partition and then place links to them on the original partition. See Chapter 6 for information on how to use links.

A better way would be to actually increase the size of the partition. However, this is also a more complex process. The first thing you must do is to make sure you have an up-to-date backup of the information on that partition. See Chapter 11, "Backup and Restore," for information on performing backups.

Next, you must decide where the extra space will come from. You can use any free space that exists on the same hard disk or free space from another hard drive, or you can add another hard disk.

You must delete the old partition and then create a new partition to take its place. Partitions can't be dynamically expanded under Linux; they must be deleted and re-created with a larger size. After you have created a larger partition, you will need to restore the original information. The restore operation is also covered in Chapter 11.

Summary

In this chapter, we have covered the types of partitions you will create. A primary partition is necessary for an operating system to boot. An extended partition is a container for logical drives that are then used to store data. Linux also has a special type of partition called a swap partition, which are used as virtual memory.

The fdisk utility is used to create these partitions. When creating a new partition, you must designate the beginning and ending cylinders. After the partitions are created, you must create a filesystem by using the `format` command.

The `fsck` command is used to examine the filesystem for errors and can be used to repair any errors found. Other utilities you can use to monitor your disk's health are the `df` and `du` commands, which are used to determine partition size and usage.

QUESTIONS AND ANSWERS

1. After installing a new hard disk, what is your first step?

   A. You first will need to partition the hard disk. You can create up to four partitions on a single hard disk.

2. Before installing Linux on your computer, what should you do to prepare the hard disk?

   A. After partitions have been created, you must associate each one with the appropriate filesystem and format the partitions so you will be able to save files to it.

3. Although you have 2GB of free space on your hard disk, when you use `fdsk` to create the partition, you receive an error and the action fails. Why?

   A. One of two things might be the problem. First, if you already have four partitions on the hard disk, you will not be able to create any more partitions no matter how much free space you have. The second possibility is that you are trying to create an extended partition and one already exists. You can have only one extended partition at a time.

4. What can you do if you want to install Linux on a computer that is running Windows 95 and does not have any free space?

   A. You can shrink the partition by using the fips program, provided that not all the partition is occupied by files. This will work as long as the partition is formatted with FAT16.

5. What do the terms `hda1` and `sdc3` mean?

   A. These are both names of partitions. The first example, `hda1`, refers to the first partition on the first IDE drive. The second example, `sdc3`, is the name of the third partition on the third SCSI disk.

6. After creating a new partition, what might keep you from installing a new application on it?

   A. One potential problem would be if there were not enough unoccupied space on the partition to house the application. Also, you must format a partition before you can save files to it.

...continues

7. Your system will not boot and you suspect that you have a corrupted filesystem. What should you do?

A. The most common problem is that the superblock is corrupted. First, you will need to boot your system from a floppy. Then, use the `e2fsck` command and one of the copies of the superblock to repair the damaged superblock.

8. How can you determine whether or not the system you administer has an adequate amount of disk space?

A. You can use the `du` command with the `-s` option to determine how much space is being used and then subtract that number from the size of each partition. A better solution, however, is to use the `df` command, which provides you with the total size of each partition, amount of space being used, and amount of free space available.

PRACTICE TEST

1. On a system with 8MB of RAM, what is the minimum and maximum recommended sizes of the swap partition?

- **a.** 8MB minimum, 16MB maximum
- **b.** 8MB minimum, 128MB maximum
- **c.** 16MB minimum, 16MB maximum
- **d.** 16MB minimum, 128MB maximum

Answer a is incorrect; the swap partition should be equal to twice the amount of RAM and can be up to 128MB. Answer b is incorrect; the swap partition should be twice the amount of RAM. Answer c is incorrect; although the minimum size is correct, the maximum allowable size is 128MB. **Answer d is correct; the minimum should be twice the amount of RAM (or 2×8MB) and the maximum allowable size is 128MB.**

2. When using the `fdisk` command, how do you indicate what size to make the partition?

- **a.** Indicate the starting and ending cylinders, which can be located anywhere on the hard disk.
- **b.** Pass the size of the partition as `+nM` to the `fdisk` command from the command line.
- **c.** Enter the starting cylinder, which should immediately follow the last partition and the ending cylinder or size.
- **d.** After selecting `n` to indicate a new partition, indicate its size as `+nM`.

Answer a is incorrect; the starting cylinder must be the next one after the ending cylinder of the prior partition. Answer b is incorrect; `fdisk` takes only the name of the partition to divide as an argument. **Answer c is correct; you must enter the starting cylinder, which has to be the next one after the ending cylinder of the last partition. You can then enter either the ending cylinder or the desired partition size, such as `+80M`, to create an 80MB partition.** Answer d is incorrect; you must first specify the starting cylinder before the size of the partition.

3. You have three partitions on your hard disk and 2GB of free space. You need to install an application that needs two partitions, a 50MB and a 300MB partition. What should you do?
   a. Create two new partitions in the free space.
   b. Create a new primary partition of 350MB and then use `mkfs` to divide it.
   c. Create a new extended partition of 2GB in size and then create two logical drives, one 50MB and one 350MB in size.
   d. Back up one of your partitions, delete it, and then re-create it, making it 350MB bigger.

Answer a is incorrect; you can have only a total of four partitions. Answer b is incorrect; the `mkfs` command is used to create a filesystem on the new partition. **Answer c is correct; because you can have only a total of four partitions, you must create an extended partition that can be divided into logical drives.** Answer d is incorrect; although you could do this, it would not create the separate partitions you need.

4. After creating your swap partition, what do you need to do to make it available to use for swapping by the operating system?
   a. Format the partition with the `mkfs` command.
   b. Change the type of partition to type `82` using the `fdisk` command.
   c. Format the partition with the `mkfs.swap` command.
   d. Pass the name of the partition to the kernel at boot.

Answer a is incorrect; the swap partition does not need to be formatted. **Answer b is correct; the swap partition needs to be designated as such by changing its type.** Answer c is incorrect; there is no `mkfs.swap` command. Answer d is incorrect; you must change the type of a partition to make it a swap partition, not designate it as such on boot.

5. Which of the following pieces of information is not necessary to pass to the `mkfs` command?
   a. Filesystem type
   b. Size of partition in bytes
   c. Partition name
   d. Size of partition in blocks

Answer a is incorrect; you must tell `mkfs` the type of the filesystem to apply to the partition. **Answer b is correct; the `mkfs` command does not need to know the number of bytes on the partition.** Answer c is incorrect; you must tell `mkfs` which partition to format. Answer d is incorrect; you must tell `mkfs` how many blocks are present on the partition you are formatting.

6. The inode structure contains all of the following, except
   - a. Filename
   - b. Access permissions
   - c. The file's owner
   - d. Date and time the file was last accessed

Answer a is correct; the filename is stored in the directory, not the inode structure. Answer b is incorrect; the permission block is stored in the inode. Answer c is incorrect; both the owner and group names are stored in the inode. Answer d is incorrect; the times the file was last accessed and last modified are stored in the inode.

7. You want to run `fsck` on your root partition. Which of the following will not enable you to do this?
   - a. Boot with a floppy and then run `fsck` on root.
   - b. Start in single-user mode and then run `fsck`.
   - c. Use `fsck` with the `-f` option to force the check.
   - d. Configure `fsck` to run at boot.

Answer a is incorrect; if you boot from a floppy, you will be able to run `fsck`. Answer b is incorrect; booting into read-only, single-user mode will enable you to run `fsck` on root. **Answer c is correct; there is no `-f` option to force `fsck` to run on a mounted filesystem.** Answer d is incorrect; the most common time to run `fsck` on root is during boot.

8. You need to prepare a report for your supervisor with the total space used by each user. What command would help you prepare this report?
   - a. `du`
   - b. `df`
   - c. `ls -l`
   - d. `wc`

Answer a is correct; the `du` command displays the amount of disk space being used. Answer b is incorrect; the `df` command shows the amount of space used on an entire partition. Answer c is incorrect; this displays a list of all the files and their attributes. Answer d is incorrect; the `wc` command counts the words, characters, and lines in a file.

9. You need to prepare a report that will be the basis of planning for needed hard disk space expansion. What command could you use to help prepare it?

   a. `du`
   b. `df`
   c. `ls -l`
   d. `wc`

Answer a is incorrect; the `du` command displays the amount of disk space being used. **Answer b is correct; the `df` command shows the amount of space used on each partition.** Answer c is incorrect; this displays a list of all the files and their attributes. Answer d is incorrect; the `wc` command counts the words, characters, and lines in a file.

10. The partition containing users' home directories is full. Which of the following solutions would not work?

    a. Create a new partition and move some users' home directories to the new partition.
    b. Back up the data, delete and re-create the partition with a bigger size, and then restore the data.
    c. Delete or move unused files.
    d. Instruct users to save files in another partition.

Answer a is incorrect; by creating a new partition and moving some users' home directories you will solve your space problem. Answer b is incorrect; this is the only method available to enlarge a partition. Answer c is incorrect; this would be a good stopgap measure, but your space problem will likely recur. **Answer d is correct; by instructing users to save files to a different partition, your organization scheme becomes chaotic, making files harder to find.**

CHAPTER PREREQUISITE

You should be comfortable with typing on the command line, which is covered in Chapter 5, "GNU and UNIX Commands." You should have a computer running Linux and have a clear understanding of the Linux filesystem as discussed in Chapter 3, "Linux Filesystem." In addition, you should have read and understood Chapter 1, "Finding and Creating Documentation."

CHAPTER

4

File Management

WHILE YOU READ

1. How can you get a list of all the directories in your home directory sorted by time with the newest listed at the top?
2. What command would you use to combine the contents of three files?
3. How can you monitor system messages as they are written to the log file?
4. How do you change the access time on a file to the present time?
5. What command would you use to copy a group of files including its directory hierarchy from one location to another?
6. After editing several files in different directories you don't remember where you are in the directory hierarchy. How can you discover where you are? How can you return to your home directory?
7. One of your users has a directory with over 300 files and cannot find the letter he wrote about the new medical plan. What would you tell him to do?
8. You have almost filled your disk quota but do not want to delete any files. What should you do?

Basic File Management Commands

This chapter discusses the basic commands used to manage your files and directories. These commands are the foundation necessary for you to perform the day-to-day tasks associated with the job of system administration. Table 4.1 lists these commands along with a short description of each command's purpose.

Table 4.1 Commands Covered in This Chapter and Their Purpose

Command	*Purpose*
`cd`	Sets location in filesystem
`ls`	Displays contents of directory
`file`	Determines file's type
`cat`	Displays file's contents
`more`	Displays file's contents one screen at a time
`less`	Displays file's contents one screen at a time
`wc`	Shows character, word, and line counts
`head`	Displays first few lines of a file
`tail`	Displays last few lines of a file
`touch`	Changes file's timestamp; create an empty file
`cp`	Copies a file
`dd`	Copies a file from one device to another
`mv`	Changes a file's name or location in the filesystem
`rm`	Deletes a file
`mkdir`	Creates a directory
`rmdir`	Deletes a directory

Many commands are used in performing file management, and you will need to be familiar with them for the exam. Rather than being asked what a command does, you will be asked about its functionality, or you will be given an example containing a command and have to predict the results of running that command. You also might see these commands used to solve a scenario question, so you need to be sure you understand each one.

cd

When you first log on to your computer, you are placed in your home directory. To change to other directories, you use the command `cd` (change directory) followed by the name of the new directory. The new directory then becomes your working directory. The syntax for `cd` is

```
cd [dir]
```

When designating a new directory, you can use either a *relative* or *absolute* path. The absolute path begins at the root or top of the filesystem and defines the complete route to your destination. For example

```
cd /home/hadden/letters
```

would locate the destination by starting at the root (`/`), then the `home` directory, then the `hadden` directory, and finally the `letters` directory. A relative path starts looking at the working directory and traverses down the path. So if your working directory is `/home/hadden` you could type

```
cd letters
```

or

```
cd letters/Aug
```

which would change to the `letters` subdirectory of `/home/hadden`. However, if the working directory is `/home/hadden/memos` you would receive an error message that the directory `letters` does not exist.

If you use a relative path with `cd` and the command fails, check to see that you are really where you think you are. To do this, use the command `pwd` (present working directory), which does not take any arguments. The `pwd` command outputs the absolute path to the working directory to your screen.

If you want to return to your home directory, you have several options. You can use the `cd` command followed by the absolute path to your home directory. A better way, however, is to use the `cd` command without any arguments. For example

```
$ pwd
/home/hadden/letters/Aug
$ cd
$ pwd
/home/hadden
```

Key Concept

As the structure of your home directory becomes more complex, the use of the `cd` command to return to your home directory and the `pwd` command to determine where you are will prevent you from forgetting where you are or saving a file to the wrong directory.

ls

File listings can be obtained using several different commands; the most often used command is `ls`. It has many options that alter its behavior.

If you are using `bash` (the Bourne Again Shell), you can get a file listing of the present working directory (`pwd`) unless it is otherwise specified with either the `ls` or `dir` command. The syntax for these two commands is

```
ls [options] [name]
dir [options] [name]
```

You should be familiar with the following options that work with either `ls` or `dir`.

`-a`	Lists all files, including hidden files
`-A`	Lists all files, except the current and parent directory
`-c`	Sorts file by time or chronologically (oldest first)
`-d`	Lists only the name of a directory, not its contents
`-l`	Lists in long format (showing permissions and other details)
`-r`	Lists in reverse order
`-t`	Sorts files by time (newest first)
`-x`	Lists all files across the page in columns
`-m`	Lists files in a single line separated by commas
`-F`	Appends a `/` after each directory and a `*` after binaries
`-R`	Lists all files recursively down the tree
`--color={never, always, auto}`	Lists colors based on file type
`--help`	Displays a synopsis of options

When specifying names, you can use wildcards, which are discussed later in this chapter and also in Chapter 8, "Text Streams." A *wildcard,* also called a *metacharacter,* is a character that represents zero or more characters.

One of the most commonly used wildcards is the `?`, which results in a match to any single character, including the `?`. The second most commonly used character is the asterisk (`*`). It matches to any number of any characters; in other words, it matches to anything and everything.

Some other things you need to know about the resulting file listings using the `ls` command are

- The default order is to list files beginning with numeric characters first and then all upper- and lowercase filenames last. Directory names are treated as filenames.
- Using either `ls` or `dir` without any options will display the listing in sorted, vertical columns.
- Hidden files, those starting with a "`.`", will not be listed unless you use the `-a` or `-A` option. These are usually configuration files.
- Colors are assigned by the file `/etc/DIR_COLORS`. Copy this file to your home directory as `.dir_color` to customize your colors.

A listing that results from issuing `ls -l` could look like this:

```
# ls -l
.
..
drwxr-xr-x 23 root root   2048 May  4 10:13 ./
drwxr-xr-x 17 root root   1024 Apr 29 01:26 ../
lrwxrwxrwx  1 root root     11 Dec 31 17:07 rmt ->
2m../sbin/rmt*
-rw-r--r--  1 root root 434898 Jul 31 1998  yp.conf
```

This produces a lot of information. The current directory is represented by a single dot (`.`) and the parent directory is represented by two dots (`..`). Reading from left to right, the columns are as follows:

```
lrwxrwxrwx  1 root root     11 Dec 31 17:07 rmt ->2m../sbin/rmt*
```

Using the previous line as an example

- File type (first character of first column) is `l` for link
- Permissions (next nine characters of first column) is `rwxrwxrwx`
- Number of links (column 2) is `1`

CH 4

- Owner (column 3) is `root`
- Group (column 4) is `root`
- File size (in bytes) (column 5) is `11`
- Date and time last modified (columns 6-8) is `11 Dec 31 17:07`
- Filename (column 9) is `rmt`
- Linked file (column 10) is `->2m../sbin/rmt*`

You can display the contents of other directories or further limit what files are listed by providing a name or names. You can use wildcards or metacharacters when specifying what to list. Some examples are listed and explained in Table 4.2.

Table 4.2 Some Examples of Using the `ls` Command

Command	*Output*
`ls /usr/sbin`	Lists all the files in the `/usr/sbin` directory
`ls -m *.txt`	Lists all files in the `pwd` with an extension of txt in a single line separated by commas
`ls -t memo?`	Lists all files whose name begins with `memo` plus any single character in reverse chronological order
`ls -R /`	Lists all files starting at root (/) and recursing through all subdirectories
`ls -Rd /*/*/*`	Lists only directories starting at root (/) and recursing through three levels of directories

You also can use these options with the `dir` command. Another file listing command you might see is `vdir`, which is the same as `ls -l`.

Notice the use of `*` and `?` to represent characters. The `*` matches zero or any number of characters, while the `?` represents any single character.

A wildcard or metacharacter is any character that represents zero or more characters. A wildcard has a special meaning to a command, utility, or program, and this meaning can be different depending on the command being used. The two most common uses of metacharacters are for ambiguous file references by the shell or as part of regular expressions, which are introduced later in this chapter.

Throughout this book you will see different wildcards or metacharacters used. Their usage is dependent on the command, utility, or shell interpreting the command. Examples of how to use metacharacters will be given as each command interpreter is discussed.

file

Previously when you looked at the output of the `ls -l` command, I mentioned that the character in column one designates the type of file. A file can be thought of as a collection of information referred to by a name. Linux views peripheral devices as files, enabling you to read and write to them the same as you do with any other type of file. Types of files include

- Ordinary files, such as text or binary
- Directories
- Printers
- Hard disk
- Floppy disk
- CD-ROM

The `file` utility enables you to get information about the contents of a file without having to examine the file directly. The output classifies the file by type. Although `file` presents information on any file, this information is not always correct. The syntax for `file` is

```
file [options] filename
```

By default, the output of the `file` command displays the name of the file followed by a brief classification. To prevent listing the filename, use the `-b` option. A list of files to be examined can be passed to the file command by using the `-f` file list option.

`file` will attempt to examine compressed files if you use the `-z` option. A discussion of compressed files is presented later in this chapter.

The following is sample output from the `file` command

```
$ file letters dog shout
letters: directory
dog: ascii text
shout: executable
```

cat

The `cat` (concatenate file) command can be used to create new files; however, it is primarily used to send the contents of one or more files to your display or other output device. `cat`'s functionality can be increased by using either > or >>.

The > is called a pipe or redirector. It is used to direct output to a location other than the standard output device, such as the display. The > can be used to write the output of a command to a file or other device, such as a printer.

The use of >> also redirects the output, but it appends to the existing file rather than overwrites it. See Chapter 8 for more details on using pipes.

The syntax for `cat` is

```
cat [OPTION] [FILE]...
```

To create a new file, type

```
cat > newfilename
file contents
Ctrl-D
```

`Ctrl-D` is the end of file (EOF) character and will close the file. To display the contents of one or more files to the standard output, type

```
cat file1 file2
```

This displays the contents of `file1` followed by the contents of `file2`. You also can do this by typing

```
cat file?
```

This lists the contents of all files whose name is `file` plus one additional character. If the only files present that satisfy this reference are `file1` and `file2`, the output will be the same as in the first example.

To combine multiple files into one, type

```
cat file1 file2 file3 > file4
```

This command creates a new file called `file4`, which contains the contents of `file1`, `file2`, and `file3`. If, however, you want to add the contents of `file1` to the end of `file2`, type

```
cat file1 >> file2
```

You also can use `cat` to overwrite an existing file.

```
cat > existing_filename
This is the new contents
<ctrl>D
```

The end result is to remove the old contents and insert the new contents into the existing file. Table 4.3 shows the options used with `cat`. You need to be familiar with these for the exam.

Table 4.3 An Explanation of `cat` Options

Option	*Function*
`-A`	Shows all nonprinting characters, prints `$` at the end of each line, and shows tabs as `^I` (equivalent to `-vET`)
`-b`	Numbers all nonblank output lines starting with 1
`-e`	Shows all nonprinting characters and prints `$` at the end of each line; equivalent to `-vE`
`-E`	Displays `$` at the end of each line
`-n`	Numbers all output lines, including blank ones
`-t`	Shows all nonprinting characters, prints `$` at the end of each line, and shows tabs as `^I`; equivalent to `-vT`
`-T`	Displays Tab characters as `^I`
`--help`	Displays help and exit
`-v`	Shows nonprinting characters
`--version`	Outputs version information and exits

Key Concept

`cat` has three uses: creating a new file; displaying a file; and appending to an existing file. `cat` is one of the few commands that lets you review the contents of a file or make changes to it.

more and *less*

Pagers are utilities that are used to display files in such a way that you can scroll back and forth one screen at a time through their contents. They also enable you to search the file for specific information.

The two most common pagers are `more` and `less`. Both enable you to scroll forward using the spacebar and scroll backward using the `b` key. `less`, however, has more features than `more`. Additional features of `less` include

- The capability to use cursor (arrow) keys to scroll forward and backward
- The capability to navigate with bookmarks, line numbers, or percentage of file
- `less` does not exit at the end of the file, although `more` does
- `lesskey` is used to customize the navigation keys used with `less`

Both utilities enable you to search a file while viewing it. These two utilities are used most frequently to display the man (manual) pages. Additional information on using the manual pages and these two pagers is presented in Chapter 1.

wc

Using `cat`, `more`, or `less` enables you to see the contents of a file. The `wc` command gives you additional information about the file but does not display its contents. `wc` will tell you the number of words, lines, or characters in a file. Its syntax is

```
wc [options] [filename(s)]
```

If you specify multiple filenames on the command line, the number of words, lines, and characters will be displayed for each file followed by the total number of words, lines, and characters for the group.

If you are interested in only one value, you can use one of the options in Table 4.4.

Table 4.4 Options to Define Output of `wc`

Option	*Output*
`-c`	Number of characters only
`-w`	Number of words only
`-l`	Number of lines only

Key Concept

The `wc` command does not change or display a file. Instead, its purpose is to count characters, words, and lines.

The output appears as the number of lines, number of words, number of characters, and filename:

```
wc dog
    4    22    132  dog
```

If you do not want the name of the file to be displayed, use the `-h` option.

head and *tail*

Like their names imply, these two commands let you look at either the beginning or end of one or more files. Use the `-n` option to designate how many lines to display; the default is 10. You also can use the `-v` or `—verbose` option to print filename headers.

`head` is helpful for identifying the beginning lines of multiple files. Type

```
head -5 /home/hadden/let*
```

to display the first five lines of all files starting with `let` in the directory `/home/hadden`. `tail` is especially helpful in reading log files where messages are appended. The command

```
tail -9 /var/log/messages
```

will display the last nine lines of the file `/var/log/messages`. You also can use `+n` option with `tail`:

```
tail +25 /var/log/messages
```

This starts at line 25 and displays each line to the end of the file. If you use the `-f` option with `tail`, you can watch as new messages are added to a log file. Type

```
tail -f /var/log/messages
```

to show each new message as it is written to the log. Type `CTRL-C` to end the display.

touch

`touch` is used to change the date and time of the last access or modification. The syntax is

```
touch [options] [date] filename
```

If the file does not exist, `touch` will create a new file of 0 length. If no date or time is specified, the current system time is used. To change the file's timestamp, use one of the options in Table 4.5.

Table 4.5 Options to Use with `touch` to Modify the Access or Modification Information

Option	*Action*

`-a`
`--time=atime`
`--time=access`
`--time=use`
	Modify the access time.

`-m`
`--time=mtime`
`--time=modify`
	Change the modification time.

`-t` *`time`*	Use specified time rather than system time. Must be in format MMDDhhmm[[CC]YY][ss].
`-c` `--no-create`	Do not create files that don't exist.
`-d` *`time`*	Update modification.Time can be in one of many formats including month names, time zones, and a.m./p.m.

cp

The `cp` command copies both files and directories. The copy operation will overwrite any existing file with the same name, so be careful. To prevent this, you can use the `-b` (backup target file) or the `-i` (interactive) option.

You also can use the `-v` option to have the `cp` command display what files were copied and the location to which they were copied. The `-f` option can be used to force the copy operation.

> **Key Concept**
>
> The `cp` command has the capability to copy files and directories. It also has the potential to create havoc on your system. Use the `-f` option carefully. The use of either the `-i` or `-v` option is great insurance against disaster.

To recursively copy one directory's contents to another, use either the `-r` or `-R` option. This also will recursively copy the directory structure.

To preserve file attributes, including permissions, owner, group, and timestamps, use the `-p` option. The `-P` option will copy the directory hierarchy as well as the file(s). For example

```
cp -P home/user/filename newdir
```

will copy the file `home/user/filename` to `newdir/home/user/filename` and create any nonexistent directories in the new path.

You also can use metacharacters with the `cp` command; however, be careful because you might get surprising results. To ensure that the results are not devastating, consider using the `-i` or `-b` options. This is especially helpful if you also are using the `-R` or `-P` options.

In addition, the `cp` command can be used to create links rather than copies of the files. The `-s` option makes symbolic links, whereas the `-l` option makes hard links. You cannot use `cp -l` to create links to directories. See Chapter 6, "Maintaining the Filesystem," for more details on links.

dd

The `dd` (device to device copy) is a special kind of copy utility. Its primary use is to copy files to other devices such as tape drives or floppies. It is often successful copying from one operating system to another when other methods fail. The syntax for `dd` is

```
dd [arguments]
```

Several arguments are available. One of the features of dd is the capability to specify block size as part of the cp command, giving you considerable control over the copy operation.

Some of the more common arguments are

if=*filename*	Specifies the source file
of=*filename*	Specifies the output file
bs=*block size*	Specifies how many bytes to read/write at one time

An example of making a backup of a floppy disk is

```
dd if=/dev/fd0 bs=512 of=MyFloppy
```

This command makes a copy of the floppy disk saved to a file named MyFloppy.

mv

The mv command is used to rename or move files to another location on the directory tree. Exercise caution: If you use the mv command to move a file to a location that contains an existing file of the same name, the existing file will be overwritten *without* warning. When evaluating questions on using either the mv or cp command, be sure to keep this in mind.

The syntax for mv is

```
mv [option] [source file] [target file]
```


One way to prevent the loss of a file is to use the -b option, which creates a backup of the file that is overwritten. For example

```
mv -b test1 test2
ls test*
test1 test2~ test2
```

Another helpful option to prevent inadvertent loss of files is the -i option, which forces mv to ask before overwriting an existing file. You will be prompted to reply y or n for each file that is replaced. This prevents you from inadvertently overwriting a critical file.

Key Concept

The mv command has the capability to relocate files and directories. Be careful using it. Be sure you understand the effect of using the -f option. Consider using the -i option, especially when moving executable files.

If you are moving a large number of files, you might want to use the `-v` option. This causes the name of each file to be displayed before it is moved. Because metacharacters can give you big problems, you should consider using the `-v` or `-i` options to specify which files are to be moved.

A powerful option is `-f`, which will force the move/copy operation no matter what. Use this only when you are sure you know that you will not lose anything you want to keep; you will not be prompted no matter what file is overwritten.

The `mv` command also can be used to move directories. If the target directory does not exist, the directory is renamed. However, if the target directory does exist, the source directory will be moved to be a subdirectory of the existing directory.

rm

The `rm` command can be used to remove individual or multiple files and directories. After a file has been deleted, it is gone, so use it cautiously.

Use the `-i` option to require confirmation before files are deleted. The `-i` option is an excellent safety feature when using metacharacters with `rm`, especially if working as the superuser. Otherwise, you might be left with a nonbootable system. The `-f` option will force deletion of write-protected files.

The `-r` option will recursively delete files and directories. You will be asked to confirm each deletion. To recursively delete files and directories without confirmation use `-fr`. You cannot use `rm` to delete directories, even empty ones, without the `-r` option.

You must have write permission to the directory containing the file you want to delete, but you do not need to have permission to the file you want to delete. If you do not have permission to the file you are trying to delete, you will be prompted to confirm the deletion.

mkdir

The `mkdir` command is used to create one or more directories. If no options are used, the parent directory must exist to create a child directory, as in the following examples

```
mkdir testing
mkdir testing/child
```

It also can create the parent and child directories in a single command by using the `-p` option.

```
mkdir -p testing/parent/child
```

You must have write permission to the parent directory (in this case the testing directory) to create a child directory. The default on the new directory is 0777, which is in turn modified by the system or user's `umask` as part of the directory creation operation. The default permissions can be overridden by using the `-m` option.

```
mkdir -m 444 testing/newchild
```

Permissions and the `umask` command are discussed in detail in Chapter 9, "Permissions."

rmdir

The `rmdir` command will delete only empty directories. Use the parent (`-p`) option to remove directory hierarchies.

```
rmdir -p testing/parent/child
```

This command deletes the `child` directory only. If there are more subdirectories in the `parent` directory, they will not be deleted. However, if you type

```
rmdir -p testing/parent/*
```

all empty subdirectories of the directory `parent` will be deleted.

Regular Expressions

Regular expressions are patterns with specific syntax that are used to match strings. They define a set of one or more characters. Regular expressions are used by many utilities, such as editors, when searching through files. In addition, each shell uses regular expressions in somewhat different ways when searching for files.

Pattern characters that you should be familiar with are listed in Table 4.6.

Table 4.6 Regular Expression Patterns

Pattern	*Function*
`*`	Matches zero or any character(s)
`?` or `.`	Matches any single character
`[xxx]` or `[x-x]`	Matches a character in a set or range of characters
`[^xxx]`	Matches any character except in the set
`\x`	Matches special characters
`^pattern`	Matches pattern to the beginning of a line
`pattern$`	Matches pattern to the end of a line
`x\{n,\}`	Matches a range of instances
`+`	Matches one or more characters

Depending on how you are using regular expressions, you might need a special character or delimiter to mark the beginning and end of your pattern. The most common delimiter is `/`, but `vi` and some other text editors allow the use of other characters as a delimiter.

Most utilities use common delimiters, but `grep` does not. (`grep` is discussed later in this chapter.) Details on using regular expressions will be discussed throughout this book in conjunction with specific utilities.

The most basic usage is `/file/`, which requires an exact match to the string `file`. This results in matches to `file`, `file1`, `MyFile`, and `file.old`. However, it will not match `MyFile` because searches are case-sensitive.

The use of *sets* overcomes this limitation. Therefore, using `/[Ff]ile/` will find all the files in the prior example as well as `MyFile`. `[Ff]ile` means either `File` or `file`.

Sets can explicitly list the acceptable characters in the preceding example or define a range of characters. So if you wanted to find a string that begins with any digit, you could use either `/[0123456789]*/` or `/[0-9]*/`.

Another way to designate a pattern is through the use of the pipe (`|`) character. You use the `|` character to separate possible choices, just like saying "or". So, looking at the earlier example, using `/file | File/` would find either `file` or `File`.

Use the caret (`^`) to find a pattern at the beginning of a line. Using `/^[0-9]/`, would result in finding any line starting with a digit. The dollar sign (`$`) can be used to find lines ending with a pattern. For example, `/$[0-9]/` would find all lines ending with a digit.

As each utility is discussed, the use of regular expressions will be presented. The first one we will cover is the use of `grep` for searching the contents of files.

Searching File Content

Numerous utilities give you the ability to find specific patterns in a file with or without opening the file. Some of these utilities display only what was found, while others will make changes to the file's contents. Two of these utilities discussed here are `grep` and `sed`.

grep

The **g**lobal **r**egular **e**xpression **p**rint (`grep`) command searches a file for a pattern or simple string. It does not change the file but rather displays each line that contains the desired string. The syntax for `grep` is

```
grep [options] pattern [file list]
```

There are three related commands—`grep`, `egrep`, and `fgrep`—that can be used for searching files line by line. The differences among them are minor. `grep` uses regular expressions. `egrep`, which uses extended expressions, has a slightly different syntax; and `fgrep` uses simple strings rather than regular expressions.

The utilities `egrep` and `fgrep` are not used as frequently as `grep`; therefore, `grep` will be covered here and on the exam. The most basic use of the `grep` command is to search a single text file for a regular expression, such as

```
grep cat MyFile
```

This would return each line in the file `MyFile` containing the string `cat`, whether it stands alone or as part of another word, such as "category".

Three major options can be used to define how `grep` handles the search pattern. Only one of these options can be used at a time. If none of them are used, the default, `-G`, is assumed. Table 4.7 lists these options and how they affect `grep`'s behavior. Be sure you understand the differences and when to use which one.

Table 4.7 Major `grep` Options

Option	*Action*
`-G`	Interprets as regular expression; default action
`-E`	Interprets as extended regular expression; such as `egrep`
`-F`	Interprets as a simple string; same as `fgrep`

To search for special characters, such as spaces or tabs, you need to enclose the pattern in quotes to prevent the shell from interpreting any special characters such as the space. For example

```
grep 'my files' MyFile
```

Without the quotes, this command would search for the pattern `my` in two files called `files` and `MyFile`. You cannot go wrong using quotes to enclose the pattern you are searching for, so get in the habit of using them. Although most of the time double quotes will work, using single quotes will always work.

Key Concept

The `grep` command searches a file for a pattern. It outputs the line containing the match to the display. Regular expressions can be used within the search pattern, but you should enclose them in single quotes (`'`).

You can use the `-n` option to show the line numbers of each line containing the search string. This option works with `grep`, `egrep`, and `fgrep`. If you only want to know the number of lines that contain the pattern, use the `-c` option. The output will not show each line but rather a count of the lines containing the pattern. If several files are searched, a total number of occurrences for each file will be displayed.

Occasionally you might need to see lines that do not contain a certain word or phrase. This also can be accomplished using `grep` and the `-v` option. Each line that does not contain the quoted pattern will be displayed. You can modify this further by adding the `-c` option. You then will see the number of lines that do not contain the pattern.

You also might use a file containing the names of the files to be searched by using the `-f` option and the name of the file containing the list. When using a file list, `grep`'s default action is to list the name of each file as a header before listing the results. This behavior can be suppressed by using the `-h` option.

`grep` performs case-sensitive searches, so if you search for lines containing `money` you will not get the lines containing `Money` or `MONEY`. To find the pattern irrespective of case, use the `-i` option, which causes `grep` to ignore case.

When `grep` is searching for a pattern, it does not care if the pattern is found as a whole word or part of a word. If you want to find only occurrences where the pattern is a word unto itself, use the `-w` option. For example, say you are looking for the letter you wrote about the upcoming New Year's Eve party but you do not remember what you named it. Instead of having `grep` return the line containing "New Year's Eve party", you can use the `-l` option so that only the name of the file containing the pattern is returned rather than the line itself.

`grep` will display errors if it encounters files that it cannot read, such as directories or device files. You can suppress these messages by using the `-s` option.

sed

The `sed` (stream editor) is a non-interactive line editor. Although it is useful for processing large amounts of text, it is somewhat daunting for the beginner. You pass all your commands to `sed` at one time, and then `sed` processes a file or files one line at a time. Commands are usually entered via a program or file you write, but it also will accept input from the command line.

`sed` does not change the original file but instead copies it, edits it, and either sends it to standard output (your display) or saves it to another file.

The syntax for `sed` is

```
sed [-n] [-e commands] [-f scriptname] textfile [>file]
```

The `-n` option prevents `sed` from sending the changed file to the standard output, except as instructed by the `p` (print) flag if it is present.

The `-e` option alerts `sed` that commands are immediately following.

The `-f` option names the file containing the `sed` commands.

The `textfile` is the file to be processed, and `>file` tells `sed` to write the output to the file specified.

A `sed` script consists of

```
[address[,address]] instruction [argument-list]
```

Addresses are optional and can be either the line number or a pattern. They specify the lines to which the instruction is applied. If no addresses are specified, `sed` processes all lines of the input file. If only one address is specified, all lines that correspond to that address are processed. Two addresses define a range where processing occurs, starting at the first address and ending at the second.

`sed` reads one line at a time. If the line satisfies the specified address or no address is given, `sed` reads the first command from the script or command line and processes it accordingly. `sed` then reads the next command and applies that. This process continues until all commands have been processed. Then the next line is read and the process continues.

`sed` can be used to search a file for a pattern and print the lines containing the pattern to standard output by typing

```
sed /term1/p file
```

Because `sed` displays all the lines of the input file by default, every line of the file will be printed to the display. In addition, the `p` argument will print any line containing the search term so those lines will appear twice.

Key Concept

`sed` is a line editor, but its changes are not applied to the original file. Instead the output is sent to the standard output. By default, each line is printed whether or not a change is made. `sed` also can take regular expressions as part of its search pattern.

If you want to see only the lines containing the search pattern, use the `-n` option:

```
sed -n /term1/p file
```

This will print only the lines that contain the pattern. If you do not use the `-n` instruction, every line will be displayed and the lines containing the pattern will be displayed twice.

You also can display a portion of the file based on line number:

```
sed '2 q' file
```

This will display the first two lines of the file. The `q` then causes `sed` to quit processing the file. Although the quotes are not necessary here, they are in the next example. If you want to see a portion of the file, you can specify a range of lines to display:

```
sed -n '3,7 p' file
```

This will print lines 3–7 and then quit.

`sed` is an extremely powerful text processor that can do much more complex options than are discussed here. See Chapter 8 for a more detailed discussion of `sed`.

Compressing and Uncompressing Files

Several utilities are available to compress and uncompress files. The most commonly used include `tar`, `gzip`, `gunzip`, and sometimes `compress`. The `zcat` utility enables you to view the contents of a compressed file. You need to know how these work and how to use them because you will see them on the exam.

tar

The `tar` (tape archive) utility is used to combine multiple files into one large archive while retaining the original directory structure. Although originally intended to create archives on a tape, `tar` files can be saved to any media. `tar` also has the capability to compress this file at the same time it creates it.

When creating a `tar` file, you should use only relative paths. That is, construct the path to the files you want to archive relative to you where you are in the file system hierarchy. If your location is `/home/hadden` and you want to archive all the files in the `/home/hadden/test` directory, you should enter the path as `test/*`.

If you try to use absolute pathnames with `tar`, the leading slash will be stripped off. This prevents overwriting existing files when a `tar` file is unarchived. To override this behavior, use the `-P` option.

Key Concept

The `tar` utility does not compress files, but instead can be used to create a single archive containing many files and their directory structure. This can be helpful when creating backups.

The tarfiles end in a `.tar`, and the syntax is

```
tar [options] [tarfile] [files]
```

The command to create a tarfile containing all the files in the `pwd` would be something like the following:

```
tar -cf myarchive.tar .
```

This command will cause the contents of the `pwd` to be tarred into the file `myarchive.tar`. The `-c` option creates the tarfile and the `-f` option gives the name for this new archive. The trailing dot is used to specify the current working directory.

You can view the contents of your tarfile using

```
tar -tf myarchive.tar
```

The `-t` option causes `tar` to print the names of the files contained in the archive.

`tar` not only enables you to create an archive, it also enables you to restore the contents of an archive. To extract your tarfile, type

```
tar -xvf myarchive.tar
```

Here the `-x` option tells `tar` to extract the contents of the tarfile called `myarchive.tar`. The `-v`, or `verbose`, option causes tar to print what files are being extracted to the standard output. The `-f` option names the tarfile to be extracted.

There are two different ways to restore a single file. First you can choose which file to extract from a list of the contents of the tarfile by typing

```
tar -xvwf myarchive.tar
```

The `-w` option places `tar` in interactive mode. You then are asked on a file-by-file basis whether to extract each file. If the tarfile contains a large number of files, this can take a long time.

Remember that `tar` has the capability to list the filenames of the tarfile. If you know the name of the file you want to extract, you can pass it directly to `tar`:

```
tar -xf myarchive.tar myfile
```

Be careful when using this option. Although the `tar` command will not overwrite directories, it will overwrite any existing file with the same name. To prevent this behavior, use the `-k` option, which will leave any existing files intact.

Many people use `tar` to manage their backups. Two options that make this so functional are the `-u` and `-r` options. The `-u` option causes `tar` to update or append the specified file to an existing tarfile. The `-r` option appends a new file to an existing tarfile. Both of these options require you to specify the existing tarfile name as well as the file you want to update or append.

gzip

Files are frequently compressed to save storage space or transmission time when copying or moving from one computer to another. In the DOS/Windows world, this is done using `pkzip`/`pkunzip` utilities. In the Linux world, the preferred format is `gzip`/`gunzip`, which creates a file with the `.gz` extension. There are some differences between `gzip`/`gunzip` and `pkzip`/`pkunzip`, however:

- `gzip` does not compress more than one file.
- `gzip` and `gunzip` automatically delete the original file.

After using the `tar` command to create an archive, it might be desirable to compress your file. You can use `tar` and `gzip` to create a compact backup, leaving you with the original directory structure and files intact inside a tarred and gzipped file.

The syntax for `gzip` is

```
gzip [options] [file-to-zip]
```

You can use `gzip` to compress the tarfile `myarchive.tar` by typing

```
gzip myarchive.tar
```

You also can combine these on the same command line using the pipe (`|`) character:

```
tar -cf myarchive.tar . | gzip myarchive.tar
```

This command first creates the tarfile `myarchive` containing all files in the `pwd`. Then `gzip` compresses `myarchive.tar`, creating a new compressed file called `myarchive.tar.gz`. Lastly, `gzip` deletes the original file, `myarchive.tar`. For this reason, you must have enough room for both the compressed file and the uncompressed file or the compression operation will fail.

You also can use another feature of the `tar` utility to `gzip` on-the-fly by utilizing the `-z` option. The following example does the same thing, but you type fewer characters:

```
tar -cvzf myarchive.tar.gz
```

Key Concept

Remember: You will `tar` first, and then `gzip` to create a compressed archive. `gzip` deletes the original file, in this case, `myarchive.tar`. The name of the file can end with `.tar.gz` or `.tgz`.

compress

The `compress` utility is one of the oldest compression utilities. The compressed file must end in `.z` and the syntax is

```
compress filename
```

This is a utility that is not used much anymore because its compression is not as good as newer utilities such as `gzip`. To uncompress the file, type

```
uncompress filename.Z
```

gunzip

Okay, you have all these compressed files sitting there. What do you do if you need to extract files? First, you must uncompress the tarfile using `gunzip`.

The syntax for `gunzip` is

```
gunzip filename
```

After `gunzip` has done its thing, you will have the original, uncompressed file. However, the compressed file is gone because `gunzip` deletes the compressed file after uncompressing it. You also can use the `gzip` command to uncompress the file by using the `-d` option.

Use `tar` to uncompress a compressed `tar` archive. Do this by using the `-z` option. So, if you type

```
tar -xvzf MyArchive.tar.gz
```

All the files contained in `MyArchive.tar.gz` will be restored, and the compressed file will still be there because `tar` does not erase the original file.

zcat

The `zcat` utility works like `cat`, except it enables you to see the contents of a compressed file. It works with files compressed using `gzip` (`.gz` extension) or compress (`.Z` extension). Its syntax is

```
zcat filename
```

After `zcat` is through displaying the file's contents, the original file is not changed. It is still there on your disk and still compressed.

Summary

In this chapter you have learned the basics of managing your files and navigating the file system hierarchy. Be sure you know how to use `cd` and `pwd`. You have learned how to get a listing of files in various formats and showing a variety of information. Review the various options for the `ls` command right before going for your exam.

Topics you are now familiar with are copying, moving, renaming, and deleting files. In addition, you know how to apply the same actions to directories including their hierarchy. Know which options to use to ensure that you don't lose critical files, such as `-i` and `-v`. Other options to be comfortable with before your exam are `-f` and `-R`.

Additional commands that you will probably use every day include `more`, `less`, `head`, `tail`, `touch`, and `cat`. `more` and `less` are pagers that display large files one screen at a time. `head` and `tail` are used to display portions of files. `touch` can create empty files or change access times on existing files.

`cat` is one command that you will be sure to see on the exam. Be sure you know how to create a new file with `cat`, how to overwrite an existing file, and how to append to the end of that file.

A very basic introduction to regular expressions has been presented as well as the use of the metacharacters `*` and `?`. `grep` is another command that you can count on seeing on the exam. Be sure to know the main options for `grep`. You also will probably see the use of regular expressions with `grep`.

QUESTIONS AND ANSWERS

1. How can you get a list of all the directories in your home directory sorted by time with the newest listed at the top?

   A. Use the `ls` command. To sort by time with the newest at the top, use the `-t` option. You also will need to use the `-d` option because you do not want to have the files listed, only the directories and the `-R` option will recurse down the directory tree. So you would type

   ```
   ls -tdR *
   ```

2. What command would you use to combine the contents of three files?

   A. Although `cat` is used primarily to list the contents of files, it is also great for combining files. `cat file1 file2 file3 > file4` will combine the contents of `file1`, `file2`, and `file3` into a new file it creates called `file4`.

3. How can you monitor system messages as they are written to the log file?

   A. Use the `tail` command to look at the most recent messages that have been written to the log. But this shows only the requested content and stops. To watch new messages as they are written, use the `-f` option:

   ```
   tail -f /var/log/messages
   ```

   When you are through, type `Ctrl-C` to end the display.

4. How do you change the access time on a file to the present time?

   A. Use the `touch` command. All you need to do is type

   ```
   touch -a filename
   ```

5. What command would you use to copy a group of files including its directory hierarchy from one location to another?

   A. Because you are making a copy you will use the `cp` command. However, it does not let you create a new directory hierarchy at the same time without using the `-r` option. If you also want to retain file attributes, use the `-p`. So, the command would be

   ```
   cp -rp /MyDir/* /NewDir
   ```

...continues

6. After editing several files in different directories you don't remember where you are in the directory hierarchy. How can you discover where you are? How can you return to your home directory?

   A. To determine what directory you are presently pointing toward, use the `pwd` command. This will return the complete path of your present directory. To return to your home directory, type `cd` without any arguments.

7. One of your users has a directory with over 300 files and cannot find the letter he wrote about the new medical plan. What would you tell him to do?

   A. It is probably a little late to suggest that he get organized and create a workable directory structure. However, you can use `grep` to search all his files for a pattern that he knows the document contains, such as

   ```
   grep -i 'medical insurance'
   ```

8. You have almost filled your disk quota but do not want to delete any files. What should you do?

   A. Use a compression utility such as `gzip` on documents that are not accessed daily, and then use `zcat` if you want to look at them. You also could use `tar` with or without `gzip` to create an archive of files such as graphics, perhaps moving them to a tape or another removable media. For example

   ```
   tar -cf MyLetters.tar *.doc | gzip MyLetters.tar
   ```

PRACTICE TEST

1. What is the output from the command `ls -ax *.txt`?
   - **a.** A listing of all files that have an extension of `txt` in reverse order.
   - **b.** A multiple column listing of all files that have `txt` in their names.
   - **c.** A listing of all files that have an extension of `txt` and their inode numbers in the present working directory.
   - **d.** A multiple column listing of all files in the present working directory with an extension of `txt` including the file `.txt`.

Answer a is incorrect; reverse order listing requires the `-r` option. Answer b is incorrect; `*txt*` would find all files containing `txt`. Answer c is incorrect; the option `-i` shows inode numbers. **Answer d is correct; the `-x` option shows the listing in multiple columns, the `-a` option shows hidden files (`.txt`) and because no path is defined, the listing would contain only files from the `pwd`.**

2. You want to append new messages to an existing log file. Which command would work?

   a. `cat MyOldLog MyNewLog`
   b. `cat MyOldLog >> MyNewLog`
   c. Use a text editor to add the new content
   d. `cat MyOldLog > MyNewLog`

Answer a is incorrect; this would display the contents of both files. **Answer b is correct; the >> causes the content of the first file to be appended to the second.** Answer c is incorrect; although you could do this, it would not be the best way. Answer d is incorrect; this would replace the contents of `MyNewLog` with the contents of `MyOldLog`.

3. You have a directory of 100 memos and you want to find the ones addressed to your boss. How would you do that?

   a. `grep -i 'Boss Name' *`
   b. `head -2 memo*`
   c. `touch memo*`
   d. `cat -n memo*`

Answer a is correct; this would display every line containing `Boss Name` along with the filename from which the line came. Answer b is incorrect; this would display the first two lines of every file in the `pwd` starting with memo. Answer c is incorrect; this would update the access time on every file beginning with memo to the present system time. Answer d is incorrect; this would display the contents of every file beginning with memo and number each line.

4. Which of the following commands will remove the directory `/home/adam` and all its files and subdirectories?

   a. `rmdir -p /home/adam/*`
   b. `rm -fr /home/adam`
   c. `rmdir -p /home/*`
   d. `rm -fr /home/adam/*`

Answer a is incorrect; `rmdir` will remove only empty directories. **Answer b is correct; the `-r` option will delete recursively and the `-f` option will force the deletion.** Answer c is incorrect; `rmdir` will remove only empty directories. Answer d is incorrect; this will remove all the files and subdirectories under `/home/adam` but leave the `/home/adam` directory.

5. You need to move all your users' directories from the `/user` directory to the `/home` directory. What would be the best way to do it?
   a. `mv -p /user/* /home`
   b. `cp /user/* /home` and then remove the originals
   c. `cp -P /user /home` and then remove the originals
   d. `cp -P /user/* /home` and then remove the originals

Answer a is incorrect; the `-p` option is used to preserve file attributes but the subdirectories would not be moved. Answer b is incorrect; this would only copy the directories immediately under `/user`. Answer c is incorrect; this would copy the hierarchy but it would be placed under `/home/user`. **Answer d is correct; the `-P` option will copy both the files and the directory structure under `/user` to `/home`.**

6. You have multiple subdirectories in your home directory. When you try to change to the `AugLetters` directory using the `cd` command you get an error message that the directory does not exist. What is the most likely cause of the problem?
   a. You used an incorrect relative path.
   b. You have to use an absolute path with the `cd` command.
   c. You are not in your home directory.
   d. There is no `AugLetters` directory.

Answer a is correct; if you use a relative path, you must make sure that you type the relative path from the present working directory. Answer b is incorrect; the `cd` command can take either absolute or relative paths. Answer c is incorrect; your present working directory does not have to be your home directory when using the `cd` command. Answer d is incorrect; although there is a possibility you did not create the directory, it is not likely.

7. The regular expression to find all lines beginning with `cat` is
   a. `/cat/`
   b. `/[cat]/`
   c. `/^cat/`
   d. `/cat*/`

Answer a is incorrect; this would find the occurrence of `cat` anywhere. Answer b is incorrect; this looks for `c` or `a` or `t`. **Answer c is correct; the caret (`^`) specifies that the line should start with `cat`.** Answer d is incorrect; this also would find the occurrence of `cat` followed by zero or any character.

8. How would you find every occurrence of the word `dog` in a file called `pets`?
    - a. `grep -w dog pets`
    - b. `grep -iw dog pets`
    - c. `grep -i dog pets`
    - d. `grep -c dog pets`

Answer a is incorrect; although it will find only lines containing `dog` as a complete word it will not find `Dog`. **Answer b is correct; the `-i` causes `grep` to ignore case, thereby finding all occurrences—even those that are capitalized.** Answer c is incorrect; it would ignore case but would return lines containing the string `dog` contained in other words as well. Answer d is incorrect; this would return the number of lines containing the pattern `dog`.

9. As part of your backup strategy, you want to back up the `/home` directories to `/backups` on another partition. You would
    - a. `tar -cf /backups/HomeBackUp.tar home/*`
    - b. `cp /home/* /backups/`
    - c. `cp -P /home /backups`
    - d. `tar -xf /backups/HomeBackUp.tar home/*`

Answer a is correct; the `-c` option creates the `tar` file and `-f` gives its name. Answer b is incorrect; this would copy the first level of directories only. Answer c is incorrect; although this would make a copy of the files it is not the best way. Answer d is incorrect; this would restore the files in the `/backups/HomeBackUp.tar` to their original location.

10. You have compressed your marketing report to save space but want to look up a statistic in it. What would be the quickest way to do this?
    - a. Uncompress the file and open it with a text editor.
    - b. Use `zcat` to display the file's contents.
    - c. Uncompress the file, and then use `cat` to display its contents.
    - d. Use `more` to display the file's contents.

Answer a is incorrect; although this would work, it would be the most time-consuming. **Answer b is correct; `zcat` lets you see the contents without uncompressing the file.** Answer c is incorrect; although this would work, it would be time-consuming. Answer d is incorrect; `more` cannot read a compressed file.

CHAPTER PREREQUISITE

Before beginning this chapter you should have a functioning Linux system. You should be comfortable typing commands at the command line, as discussed in this chapter. In addition, many commands have been presented that you should understand. You should also know how to find additional information on any command you want to use.

CHAPTER

5

GNU and UNIX Commands

WHILE YOU READ

1. What command is used to change your shell to another one?
2. How do you prevent error messages from appearing on your screen when running a command?
3. You have a file containing the names of your child's teachers. How could you display them in alphabetical order?
4. How can you review the command-line syntax you used to find the file `AcctData`?
5. What is the difference between a process and a job?
6. How can you change the priority of a process?
7. How can you change your path?
8. What is meant by command-line completion?

What Is a Shell?

This chapter presents an introduction to shells and particulars of the bash shell. It also covers input and output redirection and job control.

A *shell* is just another Linux program. It acts as the intermediary between the user and the operating system, and interprets what you type at the command line into a form the operating system can understand.

Table 5.1 shows some of the available shells and compares some of their features.

Table 5.1 Features of Some of the More Common Shells

Shell	*Built-In Commands*	*Command-Line Options*
ash (smallest)	24	10
bash (Bourne Again Shell)	48	12
ksh (Public domain Korn Shell)	42	20
tcsh (emulates the csh shell)	53	18
zsh (one of the largest)	84	50

By default, when you log on to your Linux computer, you are using the bash shell. Your default login shell is defined in the `/etc/passwd` file, which is covered in Chapter 7, "Users and Groups."

The name of the shell you are using is contained in the `SHELL` environment variable. To verify which shell you are using, examine the value of the environment variable `SHELL` by typing

```
echo $SHELL
```

You also can get the same information using the `finger` command, as in

```
finger hadden
```

The output of the `finger` command has a lot of information in addition to what shell you are using, such as your home directory, your username and full name, and other personal information.

This information is defined in the contents field of the line in the `/etc/passwd` file that defines your account. You can change this information with the `chfn` command.

If you want to try out another shell, you can type the shell's command along with its absolute path, such as

```
/bin/csh
```

This will start a child process running the new shell. When you are finished, type `exit` to return to the parent shell. If you decide you like the new shell better, you can permanently change your shell by using the `chsh` command, as in

```
chsh -s csh
```

You then will be prompted for your password before the change takes effect. A user can change only his own shell but root can change any user's shell.

For this exam, you do not need to know the details of configuring your shell; however, you do need to know how to use it. When you boot up your system and log on, you are in a bash shell unless you have changed it. If you issue the command `ls`, the shell interprets what you typed at the command line and runs the specified command.

Typing at the Command Line

The Linux operating system, not the shell, watches what you type at the command line. If you erase a word or character, the operating system does not send these corrections to the shell. After you press the Enter key, the operating system sends the entire line to the shell to be interpreted.

Multiple commands can be entered on a single line and separated by a semicolon (;). These commands are executed sequentially and the shell waits for each command to terminate before processing the next one. An example of this is

```
ls ; cat myfile
```

This command will first display a list of the files in the working directory. After finishing the file listing, the contents of the file `myfile` will be displayed by the `cat` command.

When you type on the command line and reach the end of the display, you can type `/<enter>` to have what you type appear on the next line of the display. The `/` tells the operating system to ignore the next character, in this case the newline character. For example, if you typed

```
ls /
cat myfile
```

This would work the same as the previous command. The `/` character tells the shell to ignore the next character. In this case, the next character is the newline character, which is generated when you press the Enter key.

First, the shell reads the command line as a whole, and then it breaks the command line into its parts or words. A *word* is a set of nonblank characters separated by either spaces or tabs. The first word is considered a command. This is true whether the command name is entered by itself or the complete path to the command is also entered. If the path to the command is not given then the value of the variable `PATH` is used to define the search parameters. The `PATH` variable is discussed later in this chapter.

Key Concept

The shell does not know what you type on the command line until you press the Enter key. Instead, the operating system monitors what you type. You can enter multiple commands on a single line by separating them with semicolons (;).

If your command line becomes too long for the display, you can type `/<enter>` and it will wrap to the beginning of the next line. When the operating system sends the command line to the shell, the newline character is dropped and the shell treats the commands as if they were typed on one continuous line.

The shell does not have any knowledge of whether or not the options or parameters being passed to the command are acceptable. Instead, any options that are entered are passed on to the command. Any error messages you receive are generated by the command itself and not by the shell.

If an executable is found with the same name as the command, the shell starts a new process for execution of the program. While the program is running the shell goes to sleep. After the program is complete, the shell wakes up and is ready to run another command.

The Readline Library

When you type at the command line, the editor you use is the Readline Library, which was developed by the Free Software Foundation. By default, the `emacs` mode is used and you use the same key strokes as when editing with `emacs`.

You can use the arrow keys to move back and forth through the characters you have typed. You also can use the keys as shown in Table 5.2.

Table 5.2 Keys to Be Used for Command-Line Editing

Key	*Action*
Ctrl+B	Move backward one character
Ctrl+F	Move forward one character
Esc+b	Move cursor to beginning of word to the left
Esc+f	Move cursor to beginning of word to the right
Ctrl+A	Move to beginning of line

Key	Action
Ctrl+E	Move to end of line
Del	Delete one character to left of cursor
Ctrl+D	Delete one character to right of cursor
Esc+Del	Delete word to left of cursor
Esc+d	Delete word to right of cursor
Ctrl+K	Delete from cursor to end of line

inputrc Just as you can configure the editor you want to use with `fc`, you also can configure the editor that is used when you are typing at the command line. To change to using the `vi` editor at the command line instead of the Readline Library, you would type

```
set -o vi
```

To change back to using `emacs` mode, type

```
set -o emacs
```

By default, the `/etc/inputrc` file contains the configuration information. You can override these settings by placing them in the `.inputrc` file in your home directory.

You also can use the `.inputrc` file to define key mappings. These mappings can include issuing commands or inserting text. For example, to assign the `F1` key to insert your name, add the following to your `.inputrc` file:

```
"\e[11~": "Your Name"
```

Now every time you press the `F1` key, your name will be inserted without the quotes. To see what key bindings have been defined, use the `bind -v` command.

To point to another file containing the configuration, set the `INPUTRC` environment variable to that file. You must include the entire path to the file. Environment variables are covered later in this chapter.

Command-Line Completion

Another helpful feature of the bash shell is command-line completion. You can use the Tab key to complete words you are entering on the command line.

If you are typing the first word on the command line when you press the Tab key, bash looks for a command that starts with what you have typed so far. If one is found, it completes the word. If more than one is found, it beeps. Press Tab again to see a list of available commands.

If you are not typing the first word (command), bash looks for a filename that matches what you have typed. If there is only one match, it completes the word for you. If there are multiple matches, it beeps, and pressing Tab a second time shows you a list of possible matches.

You also can press the Esc key twice at any time to complete your typing, just like using Tab. Like Tab, Esc will beep if multiple completions are available. Press Esc twice more to see a list of possible completions.

Key Concept

Bash uses the Tab and Esc keys to assist in completing what you type at the command line. If only one match occurs, it is placed on the command line. If more than one match is available, a list is displayed.

User Variables and Environment Variables

There are two types of variables, environment and user variables. A *user variable* is one that you can name and then assign a value to. An *environment variable* is already named but you can change its value.

User Variables

User variables are most often used in scripts. They provide a lot of power. One type of user variable that you are apt to use on a regular basis is an alias. Aliases are discussed later in this section.

Variable names can consist of letters, digits, and underscores. The first character of a variable name can't be a digit. Variables are created by naming them and assigning a value in a single statement, such as

```
variable_name=value
```

A variable exists as long as the shell in which it was created exists. You can remove the value of the variable by making it null, as in

```
variable_name=
```

The variable itself continues to exist and have memory assigned to it, as long as its parent shell exists.

Scope refers to the availability of a variable to other processes. The default is that a variable is seen only by the process where the variable was defined. So, if you define a variable in a script, it is visible to the child process that was spawned when the script was run. It is not visible to the parent process that originally called the script. These variables are called *private*, or *local*, variables.

For example, if you call a script that in turn calls another script, any variables declared in the second script will be invisible to the first script. To make the variable created in the second script visible to the first script, you need to make it a global variable. You use the `export` built-in command to make a variable global.

You can use the `echo` built-in command to see the value of a variable if you type

```
echo variable_name
```

By default, `echo` prints exactly what you typed to the standard output. In the prior example, `variable_name`, what you typed would be displayed. To see the value of a variable, preface the variable's name with `$`, as in

```
echo $variable_name
```

This will cause `echo` to print the value of the variable to your display. Use the same syntax in a script when you are interested in the value and not the variable's name.

Aliases

Aliases are user variables that are used to modify your environment. They are short names that have a value of a command. Often the value also contains arguments to be sent to the command.

Aliases can shorten the number of keys you have to type to enter frequently used commands or enable you to use an easier name for any command. The syntax for the `alias` command is

```
alias [name=command]
```


Aliases can be set at the command line but will be in effect only for that login session. Some helpful aliases that are often set systemwide are

```
alias rm="rm -i"
alias cp="cp -i"
alias mv="mv -i"
```

These commands set `rm`, `cp`, and `mv` to interactive mode every time they are invoked. In order to be systemwide, aliases must be entered by root in the `/etc/bashrc` file.

You can define your own personal aliases by editing the `.bashrc` file in your home directory, such as adding

```
alias lsl="ls -l"
alias lsc="ls --color"
alias lsa="ls -AF"
```

If you add aliases to either `/etc/bashrc` or `.bashrc` they do not take effect until you log off and log back on. However, you can cause them to take effect immediately by typing

```
source .bashrc
```

or

```
. .bashrc
```

at a command prompt. This causes the `.bashrc` file to be reread and any changes that were made to environment variables to be set. The second method of preceding the name of a script with a period and a space can be used to run any script, not just startup files.

Environment Variables

The look and function of your shell results from the use of environment variables. These parameters are saved in various configuration files located either in `/etc` or `/home/<username>`. The default environment variables for bash are located in the `/etc/profile` file.

To see some of your environment variables use either the `printenv` or `env` command. Both of these commands are typed at the command line without any options. Part of the output of this might look like

```
PATH=/opt/kde/bin:/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:
➥/usr/local/bin/blender:/home/
hadden/bin
KDEDIR=/opt/kde
HOME=/home/hadden
SHELL=/bin/bash
PS1=[\u@\h \W]\$
USER=hadden
BASH_ENV=/home/hadden/.bashrc
OSTYPE=Linux
```

This is a list of environment variables and their values in the form name=value. Think of a variable as a storage area that contains a value. You also can see all the possible environment variables by using the `set` command without any parameters. Table 5.3 lists some of the more common environment variables and what they contain.

Table 5.3 Common Environment Variables and What They Contain

Variable	*Contains*
`FCEDIT`	Editor to use with `fc`
`HISTFILE`	Name and location of file used to save history
`HISTFILESIZE`	Maximum size of the history file

Variable	Contains
HISTSIZE	Maximum number of commands to save to the history file
HOME	Location of home directory
INPUTRC	Value of command-line editor
PATH	Directories to search for commands
PS1	Definition of what to display as your prompt
SHELL	Name of shell you are using
USER	Logged on user's username

You are able to alter bash's environment in several different ways. When the shell starts it scans its own environment, marking each parameter to be exported to any child processes it might spawn.

To see the present value of any one variable, execute the following command:

```
echo $VARIABLE_NAME
```

The `echo` command causes its argument to be printed to the screen. The value of a variable can be accessed by placing a `$` sign before the variable's name. In this way the value is printed and not the variable's name. If you leave off the `$`, the variable's name will be printed instead.

Environment variable names are in all caps, so `echo $path` would fail while `echo $PATH` would not. The `export` built-in command enables you to change any environment variable and then make it available for any child processes. Built-in commands are part of the shell and do not cause a new child process to be started.

If an environmental variable is modified, it replaces the old value. Executed commands inherit the environment that consists of the shell's initial environment along with any changes that might have been made.

PATH

One of the most important environment variables is `PATH`. This tells your operating system where to look for files and commands. To add another directory to your path you can issue the command

```
PATH=$PATH:/sbin ; export PATH
```

Here you are adding the new directory, `/sbin`, to the present path. Then the `export` command is used to make this new path available to the shell and any future child processes.

This, however, is a temporary fix because it lasts only until you log out. If you want this change to become a permanent part of your environment, you need to edit the `.bash_profile` file in your home directory. Look for the line

```
PATH=$PATH:$HOME/bin
```

and add the new directory.

```
PATH=$PATH:$HOME/bin:/sbin
```

To make a change in the path available for all users, edit the `/etc/profile` file as root and look for the line

```
PATH="$PATH:/usr/X11R6/bin;/usr/games:/usr/lib/games"
```

and make the appropriate changes.

Sometimes you try to run a command and get the error message `command not found`. When you enter a command, the shell searches the directories defined in the `PATH` environment variable. If the command you typed is not found, you get the error message.

There are two ways to run a command that is not located on your path. The first is to give either a relative or absolute path to the command, such as

```
/home/hadden/bin/MyCommand
```

If the command is in your working directory, you can type

```
./MyCommand
```

or, if the command is in your home directory, you can use

```
~/MyCommand
```

The tilde (~) is interpreted by the shell and expanded to be the absolute path to your home directory.

If you have a command that you frequently run, you can either move the command to a directory that is on your path or add the path to the directory to your `PATH` variable.

When adding directories to your path, you need to consider the ramifications. You could add your home directory by adding `.` at the end of your path. However, this is a potential security problem. Someone could place a script in a world writeable directory that you might inadvertently run while logged in as root. This script could run without you ever knowing and perhaps give someone root privileges without you ever being aware that it happened.

Prompt

The *prompt* is what you see printed out by the shell at the beginning of each command line. It is an indication that the shell is ready to receive new input.

The value of the variable `PS1` is examined just before bash prints each primary prompt. As long as it has been set, the value is executed just as if it had been typed on the command line.

In the earlier example of the output from `printenv`, notice the cryptic line

```
PS1=[\u@\h \W]\$
```

This line defines what is listed as your prompt. The prompt listed in the previous line displays as

```
[username@hostname directory] $
```

Table 5.4 describes the special characters that can be used when setting the prompt variable.

Table 5.4 Available Prompt Characters

Character	*Output Produced*
`\a`	A bell character
`\d`	The date, in Weekday Month Date format (for example, `Tue May 26`)
`\e`	An escape character
`\h`	The hostname, up to the first `.`
`\H`	The entire hostname
`\n`	A new line
`\r`	A carriage return
`\s`	The name of the shell
`\t`	The time, in 24-hour HH:MM:SS format
`\T`	The time, in 12-hour HH:MM:SS format
`\@`	The time, in 12-hour a.m./p.m. format
`\u`	The username of the current user
`\v`	The version of Bash
`\V`	The release of Bash, version + patch level
`\w`	The current working directory
`\W`	The base name of the present working directory
`\!`	The history number of this command
`\#`	The command number of this command
`\$`	If the effective uid is 0, `#`, otherwise `$`
`\`*nnn*	The character whose ASCII code is the octal value `nnn`
`\\`	A backslash
`\[`	Begin a sequence of nonprinting characters such as a terminal control sequence
`\]`	End a sequence of nonprinting characters

You can try out different prompts temporarily using the export command. To show date and time, use

```
PS1='Date: \d Time: \t-> '; export PS1
```

To show the shell and current directory, type

```
PS1='\s:\w' ; export PS1
```

To make the change in your prompt permanent, edit your .bash_profile or .profile file and add the line PS1=<*value*>. To change the prompt for all users, change the entry in the /etc/profile file.

HOME

The HOME variable contains the location of the logged on user's home directory. This information is determined by the entry in the passwd file for your account. See Chapter 7, for more information on the passwd file.

When you log in, your home directory as defined by the HOME variable is set as your pwd. If you issue a cd command without an argument, the shell consults the HOME variable and changes to that directory.

This variable also is used by the shell to expand pathnames if the tilde (~) is included. The following two examples would both result in making the /home/hadden/test directory the pwd:

```
cd /home/hadden/test
cd ~/test
```

Key Concept

Many environment variables can be set. PATH defines where the shell searches for commands. PROMPT defines how the prompt is displayed. HOME contains the path to your home directory. Use set, printenv, or env to see your environment variables.

History List

The bash shell maintains a list of the most recently used commands called *events.* You may reuse, edit, or review them to see why something went wrong. Each event is given a number by the shell.

When the shell starts up, the history is initialized from the file named by the HISTFILE environment variable, which by default is the .bash_history file in your home directory.

The size of the HISTFILE is determined by the HISTFILESIZE variable, which defines the number of lines to be contained in the HISTFILE.

When you exit an interactive shell, the number of lines defined by the HISTSIZE variable is copied from the history list to HISTFILE. If the histappend shell option is set, the lines are appended to the history file; otherwise, the history file is overwritten.

If HISTFILE is unset, or if the history file is unwriteable, the history is not saved. The file is truncated to contain no more than the number of lines defined by HISTFILESIZE unless HISTFILESIZE is not set, in which case no truncation is performed. The number of saved events and whether they are carried over to the next login session is configurable. Table 5.5 shows the three variables that configure your history and what each variable does.

Table 5.5 Environment Variables That Define Your History

Variable	*Defines*
HISTSIZE	The number of events to preserve during a session
HISTFILE	Filename in which to save your history list; used to initialize the history list at next login
HISTFILESIZE	Number of events to save for the next login session

You can review your history list by typing

```
history
```

which displays all the events in your history, or

```
history 10
```

which would display the last 10 events.

When displaying your history, which can be quite long, pipe it through a pager such as less, or, if you are using the bash shell, use the Shift+PgUp and Shift+PgDn keys to scroll through it. You also can use the up and down cursor (arrow) keys to review the most recent events, edit the command, and reexecute it by pressing the Enter key.

You can control which commands are saved in your history list with the HISTCONTROL and HISTIGNORE variables.

Key Concept

A record or history of what commands you type is kept by the shell. Three environment variables that define how much of this history is kept and where it is kept are HISTSIZE, HISTFILE, and HISTFILESIZE.

fc

Use the fc built-in command to display your history file. It also can be used to edit and rerun commands. Use the -l option to display a portion of your history list. To display the last 16 events you would type

```
fc -l
```

or to display the events numbered 100–103 you would type

```
fc -l 100 103
```

You also can define patterns to use to search your history. To display a range of events starting with the first event that begins with cat until the first event starting with man, type

```
fc - l cat man
```

You also can search for and display events by specifying the event number. To see event number 111, type

```
fc 111
```

The fc command can be used to edit commands in your history list by using the -e option. fc uses vi as its editor. You can change that by assigning a new value to the FCEDIT environment variable:

```
export FCEDIT=/usr/bin/joe
```

fc also enables you to reexecute a command by using the -s option. To reexecute the previous command, type

```
fc -s
```

or to reexecute a specific command, type

```
fc -s 111
```

Command Substitution

Command substitution refers to replacing a command with its output. Typing pwd will result in the shell displaying your working directory. You can get the same results using command substitution. Two examples are

```
echo $(pwd)
/home/hadden

echo `pwd`
/home/hadden
```

The first example uses the newer syntax of the bash shell. The second shows the older Bourne Shell syntax using the backward apostrophe. Both of these commands will output the `pwd`. Either one works with the bash shell.

The advantage of using command substitution is most obvious in scripting. It enables you to use a command's output as input to another command. So, if you wanted to use the name of the working directory as an argument to another command, you could use command substitution.

Processes

A shell is both a command interpreter and a programming language. Commands may be run either one at a time (synchronously) or more than one at a time (asynchronously).

When running synchronous commands the shell waits for the command to finish before accepting additional input. This is also referred to as *running the process in the foreground.*

An asynchronous command executes at the same time the shell executes other commands. This is known as *running a process in the background.*

All shells provide control of input and output of commands through redirection. Usually, redirection of input is initiated by the `<` character, whereas redirection of output is accomplished through the use of the `>` character.

For example, the command `ls -ltr > MyFile` sends the output of the command `ls -ltr` to a file called `MyFile`. The command `cat < MyFile` passes the file `MyFile` as input to the `cat` command.

As you remember from Chapter 4, "File Management," `cat` is used for displaying the contents of a file. The previous example would produce the same results as typing `cat MyFile`.

The use of redirection provides control of input and output to commands. Each shell also provides built-in commands referred to as *built-ins.*

Shells can be used either interactively or non-interactively. An *interactive* shell is one in which you can provide input to the shell and the output from commands is sent back to you. On the other hand, a *non-interactive* shell runs a process without input from the user.

Any request sent to the shell is called a *process.* Whenever a request is made, the shell assigns a unique process identifier (`PID`) to that process. To see a list of currently running processes, use the `ps` command.

```
ps
PID TTY STAT TIME COMMAND
. . .
```

```
396   1 S    0:00 bash
416   1 T    0:00 pine
418   1 R    0:00 ps
. . .
```

Additional information on each job can be obtained using

```
ps l
```

where the `l` option will cause `ps` to generate a long listing. Or, you could type

```
ps ef
```

to show the relationship of parent and child processes. Notice that the hyphen (`-`) is not needed. Table 5.6 shows the most common options to use with `ps`.

Table 5.6 Options That Can Be Used with the ps Command

Option	*Action*
`e`	Shows the environment
`f`	Causes the output to be formatted as a tree
`a`	Shows all processes
`l`	Results in a detailed listing
`u`	Includes username and start time in the output
`x`	Shows processes without an associated terminal

top

Another way to look at running processes is to use the utility `top`. Information provided by `top` includes statistics on memory, swap file, and processes. It also shows how long the system has been running, CPU status, and each process's size. Figure 5.1 shows sample output from `top`.

There are a number of interactive commands to use with `top`. Type `?` or `h` to display a help screen with available commands as shown in Figure 5.2.

Because `top` continuously updates the information it displays and it fills the screen, you will want to run it on a spare console or in a separate X-terminal window. You also can use `top` to interactively kill a process or change a process's priority.

```
  4:46pm  up 4 days,  9:15,  4 users,  load average: 0.15, 0.06, 0.01
49 processes: 47 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  4.3% user,  2.3% system,  0.0% nice, 93.6% idle
Mem:  127716K av, 120932K used,   6784K free,  49432K shrd,  27972K buff
Swap: 128484K av,       0K used, 128484K free                 59488K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
 7981 root      16   0 13476  13M  1636 R       0  4.5 10.5   0:09 X
 8025 root       6   0   728  728   564 R       0  1.1  0.5   0:00 top
 7986 root       2   0  4180 4180  2804 S       0  0.7  3.2   0:00 kwm
 8001 root       1   0  3628 3628  2456 S       0  0.1  2.8   0:00 krootwm
    1 root       0   0   404  404   336 S       0  0.0  0.3   0:03 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root     -12 -12     0    0     0 SW<     0  0.0  0.0   0:00 kswapd
    4 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 md_thread
    5 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 md_thread
  936 root       0   0   896  896   556 S       0  0.0  0.7   0:00 su
  346 hadden     0   0   844  844   656 S       0  0.0  0.6   0:00 bash
  305 root       0   0   344  344   296 S       0  0.0  0.2   0:00 gpm
   43 root       0   0   364  364   312 S       0  0.0  0.2   0:00 kerneld
  212 bin        0   0   320  320   256 S       0  0.0  0.2   0:00 portmap
  223 root       0   0   428  428   352 S       0  0.0  0.3   0:00 syslogd
  232 root       0   0   564  564   324 S       0  0.0  0.4   0:00 klogd
  243 daemon     0   0   404  404   328 S       0  0.0  0.3   0:00 atd
  254 root       0   0   480  480   396 S       0  0.0  0.3   0:00 crond
  265 root       0   0   400  400   332 S       0  0.0  0.3   0:00 inetd
  276 root       0   0   408  408   332 S       0  0.0  0.3   0:00 lpd
  293 root       0   0   876  876   648 S       0  0.0  0.6   0:00 sendmail
  347 root       0   0   296  296   248 S       0  0.0  0.2   0:00 mingetty
  321 root       0   0   728  728   532 S       0  0.0  0.5   0:00 smbd
  330 root       0   0   736  736   584 S       0  0.0  0.5   0:03 nmbd
```

Figure 5.1
Using `top` to check on what processes are running.

```
Proc-Top Revision 1.2
Secure mode off; cumulative mode off; noidle mode off

Interactive commands are:

space   Update display
^L      Redraw the screen
fF      add and remove fields
oO      Change order of displayed fields
h or ?  Print this list
S       Toggle cumulative mode
i       Toggle display of idle proceses
c       Toggle display of command name/line
l       Toggle display of load average
m       Toggle display of memory information
t       Toggle display of summary information
k       Kill a task (with any signal)
r       Renice a task
P       Sort by CPU usage
M       Sort by resident memory usage
T       Sort by time / cumulative time
u       Show only a specific user
n or #  Set the number of process to show
s       Set the delay in seconds between updates
W       Write configuration file ~/.toprc
q       Quit

Press any key to continue
```

Figure 5.2
The help screen from `top` shows the interactive commands used to change the display.

Key Concept

The `ps` command is used to list the current processes on your system. To get more detailed information, use the `l` option. The `ef` option will format the output of `ps` as a tree showing the relationships of parent and child processes.

The `top` utility dynamically displays processes, unlike `ps`, which shows a static view. `top` can be used to stop processes or repriortize them.

nice

The nice command can be used to lower the priority of a running process. The superuser also can use nice to raise the priority of a process. The syntax is

```
nice [option] command-line
```

The `command-line` argument refers to the command you want to execute. This command can be a single command or a complex one including arguments and redirection.

If no option is given, `nice` uses an adjustment of 10. To use a different value, use the `-n` *`value`* option. The range of acceptable values is -20 (highest priority) to 19 (the lowest priority). A positive value decreases priority, whereas a negative one increases priority.

Only the superuser can use negative values. Running a process at an increased priority can have significant impact on the system and any other processes, including the operating system itself.

Background and Foreground Processes

When you issue a command, the shell creates a child process and the parent process (the shell) goes to sleep until that process completes. The exception is when using built-in commands, which do not create a child process.

By default, every time you issue a command it runs in the foreground. You can make it run in the background by using the `&`. For example,

```
pine &
```

will cause a process (`pine`) to run in the background.

Under the bash shell, Ctrl+z will suspend a running program and put it in the background. Use `fg` to bring the program to the foreground where you can use Ctrl+c to kill the process if necessary.

Jobs

A *job* is a series of one or more commands. Whenever you give Linux a command, even when it consists of multiple commands connected by one or more pipes, you are creating a new job. The shell provides the capability to control jobs and assigns a number to each job.

Job control refers to the capability to suspend a process and to resume its execution at a later point. This capability is controlled by the shell—bash supports job control.

You can run only one job in the foreground, but you can run multiple jobs in the background. To get a list of commands running in the background as well as stopped and suspended programs, use the `jobs` command.

If a job has been suspended, it can be brought to the foreground using `%`, `fg`, or `fg %`. If more than one job has been suspended or is running in the background, it can be brought to the foreground using `fg %n` (where `n` is the job number) or `fg %name` (where `name` is the name of the process).

For example, if an editing session with `vi` has been suspended, `fg %vi` would bring it back to the foreground.

The shell keeps a list of currently running jobs. When bash starts a job asynchronously (in the background), it prints a line that looks like

```
[2] 4356
```

which shows that this is job number 2 and that the `PID` of the last process in the pipeline is `4356`. A *pipeline* is a command line that can contain one or more pipes, such as

```
cat MyFile | sort | lpt
```

All the processes in a single pipeline are members of the same job.

After a program is suspended, it can be restarted in the background by using the `bg` command. If only one process is suspended then `bg`, `bg %name`, or `bg %n` will restart the process in the background. If more than one process is suspended then either `bg %n` or `bg %name` is required to restart the process in the background.

If you attempt to exit bash while jobs are stopped, the shell warns you that there are stopped jobs. You can then use the `jobs` command to see what jobs are stopped. If, however, you again attempt to exit without entering a new command, bash allows you to exit without a warning and the stopped jobs are terminated.

The syntax for the `jobs` command is

```
jobs [options] [jobspec]
```

Issuing the `jobs` command without any options will show a list of all running, stopped, and suspended jobs.

To also see the `PIDs`, use the `-l` or `-p` option to list only the `PIDs` of the job's group leader. You can restrict the display to running jobs by using the `-r` option, or you can show only stopped jobs by using the `-s` option.

Key Concept

Processes can run in the foreground or background. Type *`command`* `&` or use `bg` to place a process in the background; `fg` will place a process in the foreground.

A job is one or more processes started from a single command line. Use the `jobs` command to see what jobs are running.

Signals

A *signal* is a communication sent to a process. It can be an interrupt, an illegal instruction, or other condition. You can use the `kill` command to send a signal to a process. These signals are usually used to stop a process.

The `kill` command by default sends a signal to terminate the specified process or job. Only root or a process's owner can kill it. The syntax for `kill` is

```
kill [options] ID
```

The ID can be the `PID`, `%` (which is the current job), `%n` (where `n` is the job number), or `%name` (which can be the initial string or a matching string).

If a process refuses to die, use the signal `9`, as in

```
kill -9 PID
```

You also can kill a process by name using the `killall` command. Its syntax is

```
killall [options] process_name
```

Use the `-v` option to have `killall` notify you when the process has been successfully killed. The `-i` option will cause `killall` to ask for confirmation before killing the process.

Summary

This chapter has covered a great deal of information about the bash shell and how it works. We have introduced how the shell and operating system interact to process what you type on the command line.

Some of the environment variables that you will see on the exam that have been covered include `PATH`, `PROMPT`, `HOME`, `SHELL`, and `USER`. In addition, commands used to manipulate your command history, such as `HISTFILE` and `HISTFILESIZE`, were discussed. Aliases are user variables that enable you to use a short name to enter a complex command.

We also covered how the shell handles command substitution and command completion. The Esc and Tab keys are used to invoke command completion.

You now understand what processes and jobs are, how they work, and how to manipulate them (including background and foreground execution). Setting the priority of a command can be accomplished with the `nice` command.

The `top` utility can be used to change the priority of a running process. It also provides information on all running processes. Unlike the `ps` command, which displays a static view of running processes, `top` provides a dynamic view.

You also know how to discover the ID of processes (using `ps` or `top`) and jobs (using the `job` command). Use this information in conjunction with the `kill` or `killall` command to stop them.

QUESTIONS AND ANSWERS

1. What command is used to change your shell to another one?

A. You would use the chsh command.

2. How do you prevent error messages from appearing on your screen when running a command?

A. Redirect the error messages to a location other than the display. There are two ways to do this. If you never want to see them, use *`command`* `2> /dev/null`. However, if you want to review them at a later time, save them to a file instead by using *`command`*` 2> `*`filename`*.

3. You have a file containing the names of your child's teachers. How could you display them in alphabetical order?

A. Of course, there are lots of what-ifs here pertaining to how the file is organized. Let's assume that the file contains one name per line. Therefore, you could use

```
cat file | sort | lpr
```

...continues

CH 5

...continued

4. How can you review the command-line syntax you used to find the file `AcctData`?

A. Because the `bash` shell keeps a history of the commands you run, it is easy to display exactly what you typed. If you ran the command recently, try the up arrow to scroll through the most recent commands in reverse order.

5. What is the difference between a process and a job?

A. A process is created whenever you run a command. A job is one or more processes started from a single command line. Jobs can be running in the background, suspended, or stopped.

6. How can you change the priority of a process?

A. There are two ways to do this. You can use the `top` utility or start the process using the `nice` command. Users can decrease a process's priority; the superuser can either increase or decrease it.

7. How can you change your path?

A. At a command line, you can issue the command `PATH=$PATH;/`*`newdir`*` ; export PATH`. However, this is only temporary. To make this change effective every time you log in, change the path statement in your `.bash_profile` file.

8. What is meant by command-line completion?

A. The bash shell attempts to complete either a command name (if it is the first word on the command line) or a filename when you press the Tab key.

PRACTICE TEST

1. Which of the following commands directs both standard input and standard output to the file `MyOut`?

a. `cat MyFile >> MyOut`
b. `cat MyFile 2> MyOut`
c. `cat MyFile 1> MyOut`
d. `cat MyFile &> MyOut`

Answer a is incorrect; this would append the contents of `MyFile` (`stdout`) to the file `MyOut`. Answer b is incorrect; this would send `stderr` to the file `MyOut`, but the `stdout` would be sent to the display. Answer c is incorrect; this would send the contents of `MyFile` (`stdout`) to the file `MyOut`, but the `stderr` would be sent to the display. **Answer d is correct; the `&>` sends both the `stdout` and `stderr` to the file `MyOut`.**

2. What would the following command do?

```
cat MyFile | sort | tee | lpr
```

a. Print the contents of `MyFile` in alphabetical order.
b. Print the contents of `MyFile` in alphabetical order and display the contents of `MyFile`.
c. It would not work because it contains too many pipes.
d. Print the contents of `MyFile` in alphabetical order and display the contents of `MyFile` in sorted order.

Answer a is incorrect; although the output of `sort` would be sent to the printer, that is not the complete explanation. Answer b is incorrect; the contents of `MyFile` would be displayed in alphabetical order, not the way they are contained in the file. Answer c is incorrect; there is no such limit on the number of pipes that can be contained on a command line. **Answer d is correct; the sorted contents of `MyFile` would be sent to both the printer and the display.**

3. Which of the following commands would discontinue a nonresponsive process that has a `PID` of 343?

a. `kill -9 343`
b. `stop -n 343`
c. `top 343`
d. `fork 343`

Answer a is correct; you use the `kill` command to terminate a process; the `-9` terminates it no matter what. Answer b is incorrect; there is no `stop` command. Answer c is incorrect; the `top` command can be used to terminate a process; however, it does not take the `PID` as an argument. Answer d is incorrect; `fork` would begin a child process.

4. The command `cat MyFile | sort | lpr &` would start

a. One job and one process.
b. Three jobs and three processes.
c. One job and three processes.
d. Three jobs and one process.

Answer a is incorrect; each command in the pipeline would start another process. Answer b is incorrect; all commands entered as a single pipeline create a single job. **Answer c is correct; all commands entered on one command line are considered as one job but each command contained therein would start a separate process.** Answer d is incorrect; this pipeline would start only one job.

5. After entering the command in question 4, [3] 3321 appears on your display. What does it mean?
   - **a.** The job number is 3 and the PID of the first process is 3321.
   - **b.** The job number is 3 and the PID of the last process is 3321.
   - **c.** The job contains 3 processes and the job number is 3321.
   - **d.** The job number is 3 and the PID of the parent process is 3321.

Answer a is incorrect; the PID is not that of the first process. **Answer b is correct; the job number is 3 and the PID (3321) is that of the last process.** Answer c is incorrect; the shell does not report the number of processes contained in the job. Answer d is incorrect; the shell does not report the PID of the parent process.

6. You want to type a very long string of commands on one command line; however, the line is longer than the display. How can you continue typing on another line without the operating system sending the enter to the shell?
   - **a.** commands<Esc>more_commands
   - **b.** commands/<Esc>more_commands
   - **c.** commands<Enter>more_commands
   - **d.** commands/<Enter>more_commands

Answer a is incorrect; the Esc key does not enable you to wrap the lines. Answer b is incorrect; combining the / and Esc keys will not enable you to wrap the lines. Answer c is incorrect; the Enter key would be interpreted by the operating system as indicating the end of the command line and everything would be sent to the shell for processing. **Answer d is correct; the / tells the operating system to ignore the next key so that when passing your commands to the shell, the operating system would ignore the newline character**.

7. You type PS1=[\H \w]\> at the command line. What would this affect?
   - **a.** Your prompt would be [host.home.com hadden]$.
   - **b.** Your prompt would be [host \home\hadden]>.
   - **c.** Your prompt would be [host.home.com hadden]>.
   - **d.** Your prompt would be [host.home.com \home\hadden]>.

Answer a is incorrect; \w prints the pwd and the $ should be >. Answer b is incorrect; \H prints the entire hostname. Answer c is incorrect; \w shows the pwd. **Answer d is correct; \H returns the full hostname, \w returns the pwd, and the > is defined for the end.**

8. You have a directory `/home/user/bin` that contains the scripts you wrote. What file would you edit to make sure this directory is on your path whenever you log on?

   a. `.inputrc`
   b. `.profile`
   c. `.bashrc`
   d. `.path`

Answer a is incorrect; the `.inputrc` file configures the command-line editor. **Answer b is correct; you can change your path at every login by defining it in your `.profile` file.** Answer c is incorrect; the `.bashrc` is used by nonlogin shells. Answer d is incorrect; there is not a configuration file called `.path`.

9. You are frequently deleting the wrong files. What could you do to prevent this?

   a. Alias `rm` to `rm -i`
   b. Change all your files to hidden files
   c. Use `cp` instead of `rm`
   d. Use `mv` instead of `rm`

Answer a is correct; by re-creating an alias so that the command `rm` is always called with the `-i` option, `rm` will ask you to verify each file before it is deleted. Answer b is incorrect; hidden files also can be deleted. Answer c is incorrect; this would make a copy of every file you want to delete. Answer d is incorrect; this would just change the location of the file you want to delete.

10. You have been working on a new script trying out each line at the command line. You finally got the command to sort a file right but now you have forgotten exactly what you typed. What would you type to find this information?

   a. `sort ?`
   b. `cat .bash_history`
   c. `history`
   d. `source history`

Answer a is incorrect; the `sort` command takes a filename as an argument. Answer b is incorrect; this would display what was saved in `.bash_history` from your last login session. **Answer c is correct; the `history` command will display all the commands you have recently typed.** Answer d is incorrect; you `source` configuration files to load environment variables you have changed.

CHAPTER PREREQUISITE

By this time, you should know how to use the Linux documentation as well as other resources to help you find more information when needed. You also should know how the Linux filesystem is organized and the basis of managing files.

CHAPTER

6

Maintaining the Filesystem

WHILE YOU READ

1. You have just received a CD containing a new application you want to install; however, when you insert the CD in your computer, you cannot access it. Why is this?
2. Several of your users have come to you asking why they cannot save a file to a floppy disk. What is causing this and how can you allow your users to use the floppy?
3. When you look at the contents of the `/proc` directory you see several directories with names that are numbers. What are these?
4. You want to limit the amount of disk space each of your users can use. What should you do?
5. The marketing department has created a report covering a recent survey. Each of the marketing employees wants a copy in their home directories to review the data. What should you do?
6. One of your users comes to you and says she thinks she has lost a document that she created yesterday. Even though she does not remember what name she gave the document, how could you locate this file?
7. You want all your users' documents to be referenced in the `locatedb` database. How can you make this happen?
8. You install a new application to `/usr/bin`; however, it does not behave as you expect. What could be the problem?

Mounting Filesystems

To access files on your system or run applications, you need to make these files available. For the exam you will need to understand how this process takes place. In addition, you will need to know how to manipulate this process.

To place your filesystems into the Linux hierarchy, you use the `mount` command. The filesystems you mount can be on a device such as a disk partition, floppy disk, or CD-ROM. You also can mount virtual filesystems such as the `/proc` filesystem or filesystems created for other operating systems. In fact, you can even mount filesystems that reside on a computer different from the one you are presently using.

The syntax for the `mount` command is

```
mount [options] [device] mount-point
```

The options that you can use with the `mount` command are listed in Table 6.1.

Table 6.1 Options That Can Be Used with the `mount` Command

Option	*Action*
`-a`	Mount all filesystems listed in `/etc/fstab`.
`-f`	Check to see whether filesystem is mountable, but do not actually mount it.
`-n`	Do not write mount information in `/etc/mtab`.
`-o` *`option`*	Modify the mount. See Table 6.2 for possible options.
`-r`	Mount as read-only.
`-t` *`fs-type`*	Designate the type of filesystem being mounted.
`-v`	Display mounting information.
`-w`	Mount as read-write; default action.

Table 6.2 Special Modifiers Used with `mount -o`

Option	*Action*
`exec`	Binaries can be executed.
`noauto`	Do not mount automatically.
`nosuid`	Do not process the `suid` or `sgid` bit.
`nouser`	Users can't mount the filesystem.
`ro`	Mount as read-only.
`rw`	Mount as read-write.
`user`	Allow users to mount the filesystem.

If you call `mount` without any arguments, it will display all the filesystems that are presently mounted, as in

```
/dev/hda1 on / type ext2 (rw)
none on /proc type proc (rw)
/dev/hda9 on /home type ext2 (rw)
/dev/hda7 on /tmp type ext2 (rw)
/dev/hda5 on /usr type ext2 (rw)
/dev/hda6 on /var type ext2 (rw)
/dev/fd0 on /mnt/floppy type vfat (rw)
/dev/cdrom on /mnt/cdrom type iso9660 (ro)
```

The information returned includes the device, mount point, type of filesystem, and any options such as read-only.

Key Concept

Use the **`mount`** command without any arguments to see what filesystems are presently mounted. To add or remove a filesystem, use the **`mount`** and **`umount`** commands.

If you use the `mount` command to mount a filesystem that is defined in the `/etc/fstab` file, the `mount` command looks at any options that have been defined and applies them when mounting the filesystem. To override these options you can use the `-o` option with the `mount` command.

In addition, you can use the `-o` option to apply options when mounting a filesystem that is not defined in the `/etc/fstab` file. The `/etc/fstab` file is discussed later in this chapter.

To mount a device, such as your CD-ROM, you could type

```
mount /mnt/cdrom
```

You do not need to specify additional information such as making it read-only or the filesystem type. Instead, the `mount` command consults the `/etc/fstab` file to see whether this mount point is defined.

The */etc/fstab* File

The `/etc/fstab` file defines each filesystem that the `fsck` utility checks by default. This file is also used by the `mount` and `umount` commands. A sample `/etc/fstab` is

```
/dev/hda1          /                  ext2     defaults        1 1
/dev/hda9          /home              ext2     defaults        1 2
/dev/hda7          /tmp               ext2     defaults        1 2
```

```
/dev/hda5          /usr              ext2     defaults        1 2
/dev/hda6          /var              ext2     defaults        1 2
/dev/hda8          swap              swap     defaults        0 0
/dev/fd0           /mnt/floppy       msdos    noauto,user     0 0
/dev/cdrom         /mnt/cdrom        iso9660  noauto,ro,user  0 0
none               /proc             proc     defaults        0 0
```

The fields of each line are

- The physical location of the filesystem or block device.
- The mount point. This is the place in the directory structure where the root of the filesystem is located.
- The filesystem type.
- Options to use when mounting the filesystem. If you want to use multiple options, separate them with commas and no white space.
- A number that determines whether to back up the filesystem if the `dump` utility is used.
- A number that tells `fsck` in what order to check the filesystems.

Although a filesystem can be mounted that is not listed in the `/etc/fstab` file, the filesystem must be listed to be mounted by the `mount -a` command or for a user to be able to mount a filesystem.

If you want to mount a filesystem that is not defined in the `/etc/fstab` file, you have to provide all the same information that is listed in the `/etc/fstab` file. So, if you wanted to mount a floppy with the vfat filesystem rather than the msdos filesystem defined in the previous `/etc/fstab` file, you would type

```
mount -t vfat /dev/fd0 /mnt/floppy
```

If you want to mount this floppy at a location other than `/mnt/floppy`, first make sure that an empty directory has been defined on your directory tree to use as its mount point. It is conventional to mount removable filesystems such as floppy disks under the `/mnt` directory; however, you can mount it anywhere on the directory tree you want.

The `/proc` Filesystem

Notice on the last line of the `/etc/fstab` file that the device is designated as none and the filesystem is `/proc`. This is a virtual filesystem that does not exist on a physical device but rather is a way to present information that is stored in memory. Each process is shown as a subdirectory of the `/proc` directory.

System information appears as a file. For example, the /proc/kcore file is just a method for addressing your computer's memory. If you type ls -l /proc/kcore you will see the amount of memory in your system as bytes. Only root can access the proc directory. A sample output from ls -l /proc is

```
dr-xr-xr-x   3 root     root            0 Nov 10 16:24 1
dr-xr-xr-x   3 root     root            0 Nov 10 16:24 2
dr-xr-xr-x   3 bin      root            0 Nov 10 16:24 212
. . .
-r--r--r--   1 root     root            0 Nov 10 16:24 cmdline
-r--r--r--   1 root     root            0 Nov 10 16:24 cpuinfo
-r--r--r--   1 root     root            0 Nov 10 16:24 devices
-r--r--r--   1 root     root            0 Nov 10 16:24 dma
-r--r--r--   1 root     root            0 Nov 10 16:24 filesystems
-r--r--r--   1 root     root            0 Nov 10 16:24 interrupts
-r--r--r--   1 root     root            0 Nov 10 16:24 ioports
-r--------   1 root     root    134156288 Nov 10 16:24 kcore
-r--------   1 root     root            0 Nov  5 18:23 kmsg
-r--r--r--   1 root     root            0 Nov 10 16:24 ksyms
-r--r--r--   1 root     root            0 Nov 10 04:23 loadavg
-r--r--r--   1 root     root            0 Nov 10 16:24 locks
-r--r--r--   1 root     root            0 Nov 10 16:24 mdstat
-r--r--r--   1 root     root            0 Nov 10 16:24 meminfo
```

A directory is created for each of the processes that is running named with the process's ID. This system information is helpful when diagnosing system problems. Say you want to see the interrupts on your system. You could issue the command cat /proc/interrupts, which would result in an output similar to the following:

```
 0:   42516998   timer
 1:        539   keyboard
 2:          0   cascade
 5:          0 + es1370
 6:         11 + floppy
 8:          1 + rtc
10:      69455   eth0
12:        132   PS/2 Mouse
13:          1   math error
14:     277135 + ide0
15:          0 + ide1
```

This file lists all your system's interrupts that are being used and what is using each one.

CH 6

The */etc/mtab* File

The mount table, `/etc/mtab`, is a listing of presently mounted filesystems. This file is displayed when the `mount` command is issued without any arguments.

It is created at the time the system boots. Using either the `mount` or `umount` command keeps the `/etc/mtab` file updated. This behavior can be overridden by using the `-n` option with either `mount` or `umount`. The `umount` command is discussed in the next section.

umount

If you no longer want to have access to a particular filesystem, unmount it using the `umount` command. Its syntax is

```
umount [options] [device]
```

The `umount` command is generally used to unmount removable devices prior to ejecting them from the system, such as floppy disks and CD-ROMs. Always unmount all removable devices before removing them from the system. Failure to do so can result in the loss of data.

The options that are used with the `umount` command are listed in Table 6.3.

Table 6.3 The Options That Can Be Used with the **umount** Command

Option	*Action*
`-a`	Unmount all filesystems listed in the `/etc/mtab`
`-n`	Unmount the specified filesystem but do not change the `/etc/mtab` file
`-t` *`fs-type`*	Unmount the *`fs-type`* filesystems

You can specify either the device or the mount point as an argument to the `umount` command. After the command has been issued, all pending read/writes are completed, and then the filesystem is unmounted.

Disk Quotas

Setting disk quotas that limit the amount of hard disk space that can be used by each user or group will help you better manage your system.

For the exam, you will need to know how to set disk quotas, including setting soft and hard limits. You also will be tested on how to monitor these quotas.

Disk quotas are set on a partition-by-partition basis. You can elect to set quotas on only one partition or to have different quota values on each partition.

Quotas can be managed on a per-user basis or on a group basis. These quotas are independent from each other; that is, the total of the disk quotas for the group's members is unrelated to the group's quota. So, if there are 10 users who are members of the `wp` group and each user has an individual quota of 3MB, you could still set the group's quota to 10MB.

Whenever a user creates a file, it is owned by that user and his group. See Chapter 9, "Permissions," for further details on file ownership. If quotas are in place, the records for both the user quotas and the group quotas are updated.

If a user attempts to create a file and his group's quota has been exceeded, the attempt to create the file will fail even if he has available space based on his individual quota. To create a file, he must then change his group. See Chapter 7, "Users and Groups," for more information on changing groups.

Key Concept

When creating a new file, the creator's group's quota is checked first. If a quota for that group exists, the size of the new file is counted toward that group's quota. If no quota exists for the group, the owner's quota is checked before the file is created.

By default, the ability to set quotas is turned off. To enable the ability to set quotas, you must first edit the `fstab` file and add the `usrquota` option to the line defining the filesystem where you want quotas to be applied:

```
/dev/hda1      /     ext2     defaults,usrquota 1 1
```

As root, you would next need to create an empty file called `quota.user` in the root of the partition where you are defining the quotas. You will need to set its permissions to read and write by root only (see Chapter 9), so you could type

```
touch /quota.user
chmod 600 /quota.user
```

You now will have to reboot your system to make these changes and turn on the quotas using the `quotaon` command. If you should ever decide that you do not want to enforce quotas on a particular partition, you can turn off the quotas by using the `quotaoff` command. Both of these take the filesystem's name as an argument. Table 6.4 shows the options that apply to the `quotaon` command and Table 6.5 shows the options for the `quotaoff` command.

Table 6.4 Options That Can Be Used with the `quotaon` Command

Option	*Action*
`-a`	Quotas are turned on for all read/write filesystems in `/etc/fstab` file (only if `usrquota` is defined for filesystem). This also occurs at boot to enable quotas.
`-v`	Displays information on each filesystem where quotas are turned on.
`-u`	Turns on named user's quotas; default action.
`-g`	Turns on named group's quotas.

Table 6.5 Options That Can Be Used with the `quotaoff` Command

Option	*Action*
`-a`	Disables quotas on all filesystems listed in the `/etc/fstab` file (only if `usrquota` is defined for filesystem)
`-v`	Displays information on each filesystem where quotas are disabled
`-u`	Turns off named user's quotas; default action
`-g`	Turns off named group's quotas

After your system has come back up and you have logged in as root, use the `edquota` command to open the `quota.`*`user`* file in your default editor:

```
edquota -u hadden
```

The options for the `edquota` command are listed in Table 6.6.

Table 6.6 Options to Use with the `edquota` Command

Option	*Action*
`-u` *`username`*	Edit this user's quotas
`-g` *`groupname`*	Edit this group's quotas
`-p` *`username`*	Duplicate this user's quota when used with the `-u` option; duplicate this group's quota when used with the `-g` option
`-t`	Edit soft limits

The file will read something like the following:

```
Quotas for user hadden:
/dev/hda5: blocks in use: 52, limits (soft = 0, hard = 0)
     inodes in use: 32, limits (soft = 0, hard = 0)
```

You can set quotas based on the number of files and directories and the total disk space that the user can fill up. You can set either or both of these limits.

Both the number of files and the amount of space are modified by three parameters:

- **Soft limit**—The user or group for whom the quota is defined can exceed this amount for a limited amount of time.
- **Hard limit**—After a hard limit is reached, no more files can be created. There is no grace period.
- **Grace period**—How long the user or group can exceed the soft limit. It is expressed as `days:hours:minutes:seconds` and defaults to seven days.

To continue with our example, if you want to limit hadden to 3–5MB of disk space and between 500 and 1,000 files, use the `edquota` command to change the parameters to read

```
Quotas for user edwina:
/dev/hda5: blocks in use: 52, limits (soft = 3000, hard = 5000)
     inodes in use: 32, limits (soft = 500, hard = 1000)
```

Each limit can have one of three values:

- `0`, which means no limit.
- `-1`, which means use the default values.
- Any integer greater than zero. This represents kilobytes.

The default limits for your system are defined in `quota.h`. After saving the file, you will next want to set a grace period so that the user is notified when she has filled the amount allocated. This is accomplished by using the `eduquota` command with the `-t` option.

```
Time units may be: days, hours, minutes, or seconds
Grace period before enforcing soft limits for users:
/dev/hda5: block grace period: 1 days, file grace period: 1 days
```

Now you can use the `quota` command to verify that your quotas have been set. The syntax for the `quota` command is

```
quota [option] [who]
```

Only root can look at quotas for other users and groups. Users can use the `quota` command to examine their own quotas. Users also can use the `-g` option to see the quotas for groups of which they are members.

The `who` can be a username or groupname. Use the `-u` option to specify a user. If you also use the `-g` option, the quotas for any groups of which the user is a member will be displayed as well.

By default, `quota` reports on all filesystems listed in the `/etc/fstab`. If you use the `-q` option, `quota` will return information only on filesystems where the quotas have been exceeded.

One record per partition provides the base defaults. Often this will be the only record modified with all users inheriting these values.

You can use the `repquota` command to check the status of quotas on your system. Its syntax is

```
repquota [options] [filesystem]
```

The `repquota` command displays a summary of the amount of disk usage and quotas for the specified filesystems. It displays the number of files and disk space used, as well as any defined quotas for each user in kilobytes. The options used with `repquota` are shown in Table 6.7.

Table 6.7 Options Used with the `repquota` Command

Option	*Action*
`-a`	Check the `/etc/fstab` file for filesystems with quotas and reports on them; does not take a filesystem as an argument.
`-v`	Report on all quotas even if the user is not using any space on that partition.
`-g`	Report quotas for named group.
`-u`	Report quotas for named user; default action. Only root can see others.

Only root can see quotas for other users. Users can see quotas for themselves and for any group for which they are a member.

The `quotacheck` command goes through the partition and updates the quota status, which is how much space is being used by each user or group that has a quota defined. It normally runs after an `fsck` at boot time. Its syntax is

```
quotacheck [options] [filesystem]
```

The `quotacheck` command outputs a `quota.user` file if a user scan is performed and outputs a `quota.group` file if a group scan is performed. Table 6.8 shows the options used with `quotacheck`.

Table 6.8 Options Used with the `quotacheck` Command

Option	*Action*
`-v`	Display information on its progress
`-d`	Debug the program
`-u`	Perform user scan; take UID as argument
`-g`	Perform group scan; take GID as argument
`-a`	Check quotas on all filesystems in `/etc/fstab` that have quotas set
`-R`	Use with `-a` to check quotas on all filesystems except root

Only root should run the `quotacheck` command.

The `quotastats` command calculates the amount of used space and number of files for every user and group. The results are stored in the `quota.user` and `quota.group` files located in the root of each filesystem. The `quotastats` command takes no arguments.

Links

The use of links is an important concept for the exam. The process of creating and deleting them will be covered. You also can expect to see questions that are scenario-based in which you have to assess how to provide the appropriate access. Often, this will be through the use of links.

You can use a *link* to give another user access to one of your files. First, you would give the other user access rights to your file. You also might need to change the permissions on the directory containing the file to allow the other user to change or execute the file.

You will want to consider the ramifications of changing the permissions on the directory before doing this. Review Chapter 9 for more information on allowing others access to your files and directories.

After the permissions have been appropriately set, the other user can create a link to your file but locate the link in her home directory. This simplifies sharing information.

Linking can be used to avoid creating multiple copies of a file. This saves both disk space and the administrative headaches associated with trying to keep multiple copies up to date.

In addition, links can be helpful in managing large directory hierarchies. You could save all your memos in a directory called `memo`. You could then create links to individual memos under the `business` and `personal` directories, making it easier to locate a specific file.

When you create a link, you are creating a new entry in the directory. If you create a link in the same directory as the original file, you must give it a different name. If the link is created in a different directory, they can both have the same name.

Key Concept

A link is not the same as a copy of a file. Instead, the link is another name for that file that points to the original file. Changes made to that file are saved whether you access the file via its original name or via one of its links.

When you change a file, those changes are seen whether the file is accessed directly or through one of its links. Linux supports two types of links: hard and symbolic (also called soft). The syntax is

```
ln [option] existing_file new_link
```

One difference between a soft and hard link is their inode numbers. As discussed in Chapter 3, an *inode* is a data structure containing information on a file. The directory structure provides a way to associate the file's name with its inode. The inode contains information on file attributes as well as the physical location where the file contents are located.

Hard Links

Whenever you create a file you are actually creating a link. If you use the `rm` command to delete that file, you are removing the link.

A *hard link* creates a new pointer to a file and not a new copy of the file. Hard links can be created only to an existing file. All the attributes of the link are identical to those of the original file: permissions, owner, and modification time. Just the filename and its place on the directory tree are different.

Any changes made to any link are reflected in the contents of all the links. For example, say you create a hard link called `NewLink` to a document called `MyFile` and make changes to `NewLink`. When you open `MyFile`, you will see those changes.

Another characteristic of a hard link is that you can delete any link and the file will still be there. So, in the preceding example, if you delete `MyFile`, `NewLink` is still there with all its contents. The same applies if you delete `NewLink`; `MyFile` will still be there. However, if you also delete `NewLink`, the file is gone for good. If you use the `ls` command with the `-i` option, you will see that both files have the same inode number:

```
ls -i MyFile Newlink
3798 MyFile     3798 NewLink
```

You can see the number of links a file has by using `ls` with the `-l` option:

```
ls -l MyFile NewLink
-rw-r--r-- 2 hadden hadden 23 Aug 13 11:35 MyFile
-rw-r--r-- 2 hadden hadden 23 Aug 13 11:35 NewLink
```

Remember, as far as the operating system is concerned, these two pointers are equal, and it does not care which one was created first.

Although the existing file and new link can be in the same directory, links are usually created in a different directory. To create a hard link, type

```
ln MyFile /NewDirectory/NewLink
```

If only the new directory location is specified then the new link will have the same name as the original file.

Some limitations of hard links are that users can't make hard links to directories and the links must be on the same filesystem. root, however, can create a hard link to a directory by using either the `-d` or `-F` option.

Symbolic Links

A *symbolic* or *soft link* is an indirect pointer to the file. It is a new directory entry that contains the path to the existing file. Symbolic links were created to overcome the limitations of hard links. They can be created to a

- Directory by a user
- Nonexistent file
- File on another filesystem

Symbolic links are created using the `ln` command with the `-s` option:

```
ln -s MyFile /NewDirectory/SymLink
```

When using the `ls -l` command to look at a symbolic link, you will notice a few differences. The type of file is listed as `l` (the first character), the path to the original file is listed, and the file attributes are different. For example, if both the original file and the symbolic link are located in the `pwd`, the `ls -l` command would display the following:

```
ls -l MyFile SymLink
-rw-r--r-- 1 hadden hadden 23 Aug 13 11:35 MyFile
lrw-rw-rw- 1 hadden hadden  4 Aug 13 12:13 SymLink -> MyFile
```

You can delete the original file without removing any symbolic links; however, any symbolic links will no longer work. Because a symbolic link contains only the path to the original file, when the original file is deleted the link is pointing to a nonexistent file.

To create a link to a directory, use the `-F` option. However, symbolic links to directories can't work exactly like you expect when using the `cd` command. Different shells will act differently.

You can use a symbolic link to a directory as the argument to the `cd` command. Under the bash shell, if you then issued the `pwd` command, it would return the symbolic name of the directory. However, if you issue the command `cd ..` to move up one level in the directory structure, you are placed in the directory containing the symbolic link, rather than the parent directory of the directory referenced by the symbolic link.

In other shells, issuing the `pwd` command returns the name of the original directory rather than the symbolic link. When you issue the `cd ..` command, you are placed in the parent directory of the link itself.

Key Concept

A link provides access to a file from different locations. Remember that all hard links to a file have the same inode number as the original file. This means that each link (filename) is associated with the same physical location on the disk. A soft link, however, contains only the path to the original file.

Any one of the hard links can be deleted without preventing access via other hard links to that file. If the original file is deleted then all soft links to that file will no longer work.

Managing System Files

In Chapter 3 we discussed the hierarchical nature of the Linux filesystem. How this directory hierarchy is organized is important in maintaining your system, installing software, and managing users. You will need to understand it for the exam.

Each distribution has some differences in where files are stored and how the directory structure should be organized. Over time, these differences have resulted in problems with software installation or going from one distribution to another, especially regarding where configuration files are stored.

As a result, an effort toward standardization has been undertaken. It is defined by the Filesystem Hierarchy Standard, which can be viewed at `http://linuxpowered.com/html/tutorials/fhs/fhs-toc.html`. Although this standard was first defined to cover the Linux filesystem, it has been expanded to apply to other UNIX systems.

For the exam, you will need to know the standard locations for files. On the distribution-specific exam, you will need to know the locations specific to that distribution.

Hierarchy

There are some directories that always should be present and that contain specific types of files. A good place to start is root, which is represented by `/`. This is the top or beginning of your directory structure.

The `/` is the container for various other directories. Think of it as the glue that holds your system together. As a rule, this directory should not contain other files. Table 6.9 shows the major directories, what they contain, and their purposes. These are the directories that you should be familiar with for the exam.

Table 6.9 The Contents of the Root Directory and the Purpose of Each

Directory	*Contents and Use*
`bin`	Contains binaries that are required for system startup.
`boot`	Contains the kernel and any configuration files necessary for the boot loader.
`dev`	Contains files necessary for accessing all types of peripheral devices, not just those present on your system.
`etc`	Contains the system configuration files and scripts such as shell startup scripts.
`etc/skel`	Contains skeleton user files to be placed in user's home directory when his account is created. See Chapter 7 for more information on this directory.
`etc/X11`	Contains X Window configuration files.
`etc/rc.d`	Contains startup scripts and the rc directories. See Chapter 2, "Booting Linux," for more information on the use and structure of this directory.
`home`	Contains subdirectories for each user except root to be used as home directories; every regular user needs a home directory to be able to log in. Additional subdirectories can be used to further classify users.

…continues

Table 6.9 continued

Directory	*Contents and Use*
`lib`	Shared libraries and kernel modules.
`Lost+found`	Directory used for recovered files.
`mnt`	Temporary point to which to connect devices.
`proc`	This is a virtual directory that contains system information on the kernel, running processes, and how system resources are being used.
`root`	Home directory for the root user; can use the `/` directory itself in some distributions.
`sbin`	Contains administrative binaries and tools to be used only by the system administrator, root.
`tmp`	Contains temporary files; read, write execute access for everyone.
`usr`	Contains binaries, libraries, applications, and packages such as X Windows that should be accessible to regular users. Often contains links to files stored in `var`.
`usr/bin`	Contains most user applications and should be in the user's path.
`usr/sbin`	Contains non-essential administration utilities.
`usr/local`	Contains software that is not part of the operating system itself. Simplifies backups.
`usr/local/bin`	Contains binary files for software installed after installation of the operating system and should be on user's path.
`usr/local/sbin`	Contains locally installed administration utilities.
`usr/include`	Contains standard C/C++ header files.
`usr/lib`	Contains static libraries as well as subdirectories for libraries for non-C/C++ languages; configuration files for `ldconfig`. Contains links to X Windows files.
`usr/src`	Contains source files for most software installed on the system with a separate subdirectory for each package.
`usr/src/linux`	Linux source directory.
`usr/X11R6`	Usually a link to another directory containing X Windows files. The organization of this directory might be different depending on the window manager being used.
`var`	Contains variable data such as logs and spooling directories for mail and news. Often contains links to files stored in `var`.
`var/log`	Contains log files. Can grow very large and should be monitored.
`var/spool`	Used to store temporary files for printing, mail, and news. Can grow very large and should be monitored.

Finding Lost Files

As you can see from Table 6.9, there might be several places where a certain file is stored, so you could spend considerable time searching multiple directories for a specific file. To prevent this waste of time, some utilities are available on your system to assist you in locating that long-lost command.

For the exam you will need to be comfortable using these commands. You might be asked a question where you will need to select which command would be the most appropriate to use in a given situation.

find

This utility can be used to find files starting at the specified directory and searching all its subdirectories to find filenames that match the specified pattern. The syntax for `find` is

```
find [path] [condition]
```

If no modifying condition is passed to the `find` command, it will produce a listing of all files and directories starting with the specified directory recursively through the entire directory tree. If no directory is specified, `find` searches the present working directory and all its subdirectories. Table 6.10 lists the possible conditions that can be used with the `find` command to narrow the results.

Table 6.10 Conditions That Can Be Used to Modify the Results from Running the `find` Command

Condition	*Action*
`-atime ±n`	Find files based on number of days since last accessed
`-ctime ±n`	Find files based on number of days since directory entry last changed
`-group groupname`	Find files belonging to *groupname*; can be name of GID
`-inum n`	Find files with inode *n*
`-links n`	Find files with *n* links
`-mount`	Find files only on same filesystem
`-mtime ±n`	Find files based on number of days since last modified
`-name pattern`	Find files matching specified pattern
`-perm nnn`	Find files with specified permission block
`-print`	Display matching files and their full paths; default action
`-size n[c]`	Find files *n* blocks in size or *n* characters
`-type c`	Find files of type `c`, which can be `b` for block, `c` for character device, `d` for directory, `l` for link, or `f` for plain file
`-user username`	Find files belonging to *username*; can be name of UID

The expression most commonly used with `find` is `-name`, which specifies the filename it is to find. This filename can contain wildcards, as listed in Table 6.10, or parentheses. However, they must be enclosed in quotes to prevent the shell from interpreting them. So, if you would like to find a file named `mydogma` but do not know where you saved it or exactly what you named it, but you are sure the name contains the letters `dog`, you could search for it by typing

```
find /home/hadden -name '*dog*' -print
```

The `-print` expression causes `find` to print all files that meet the criteria. It is the default action and therefore can be omitted. If, however, you had typed

```
find -print /home/hadden -name '*dog*'
```

`find` would have printed all the files contained in the directory `/home/hadden` and all its subdirectories whether or not they met the criteria. `find` sees the `-print` expression and assumes that there is no criteria. If you use the `-print` expression, make sure it is the last one on your command line.

`find` also can be used to find a file based on when it was accessed last.

```
find /home/hadden -atime ±n [-print]
```

where `n` is the number of days relative to today. When you use this, `find` changes the access time of every directory it searches. In addition, you can use `find` to locate a file based on when it was last modified.

```
find /home/hadden -mtime ±n [-print]
```

When using multiple criteria, a space is considered to mean and. The `-a` can be used instead of a space. Use either `-or` or `-o` to mean or. Use the `!` in front of an expression to designate not. The following examples search the `pwd`.

```
find . –name '*dog*' –a '*cat*'
```

The previous code finds all files that contain both “dog” *and* “cat” in their names.

```
find . –name '*dog*' –o '*cat*'
```

The preceding code finds all files that contain either “dog” *or* “cat” in their names.

```
find . –name !'*dog*'
```

The previous code finds all files that do *not* contain “dog” in their names.

locate

Instead of searching the directory tree as `find` does, the `locate` command searches a database called `locatedb`, which is located in the `/var/lib` directory. The syntax for `locate` is

```
locate [filename]
```

Typing

```
locate *.ps
```

would cause `locate` to search the `locatedb` database and return all files that end in `.ps`.

`locatedb` contains a listing of all the files on the filesystem. To keep this database up to date, rebuild the database with the command `updatedb`.

The syntax for the `updatedb` command is

```
updatedb [options]
```

You can tell `updatedb` what paths to use or exclude and the name of the database to use for the results. By default, the `updatedb` command saves its results to `locatedb`. Table 6.11 lists the options for `updatedb`.

Table 6.11 Options to Modify the **`updatedb`** Command

`--localpaths`	Local directories to be included
`--netpaths`	Network directories to be included
`--prunepaths`	Directories to exclude
`--output`	Name of new database

Many distributions create a cron job upon installation that keeps your `locatedb` up to date. You can edit this file to change which directories are indexed or which are excluded. In addition, you can change how frequently this script is run. If you want, you also can use an `updatedb.conf` file to configure which directories you want to index.

which

Use the `which` command to find what version of a command is being executed. It searches your path until a match is found. If a match is not found, `which` displays an error message.

```
which command
```

The `which` command does not have any options and returns the full path to the command that would be executed if that command were typed at the command line. Multiple commands can be passed on the same command line.

The `which` command is helpful when a command is not responding the way you expect. For example, if you want to see which `ls` and which `cat` is being executed, you could use the `which` command, as in

```
$ which ls cat
/bin/ls
/bin/cat
```

Occasionally, there can be more that one command with the same name in different locations. Using the `which` command verifies that you are running the command you thought you were. After a match is found, it quits the search and displays the full path to it.

Key Concept

The **`find`** command searches the specified directory tree. The **`locate`** command searches the **`locatedb`** database. The **`which`** command identifies the command that is being run. This is helpful if there are multiple commands with the same name.

Summary

Before you can access any files on your system, they must be made available to you. This is accomplished by mounting the volume containing the files you want to see. When mounting a filesystem, you can make it available as read-only, as well as for read and write access.

Understanding the `/etc/fstab` and `/etc/mtab` files and how the `mount` and `umount` commands use them will definitely be on the exam. Issuing the `mount` command without any arguments will display which filesystems are presently mounted by examining the `/etc/mtab` file. The `/proc` directory is actually a way to look at what is present in RAM. It contains subdirectories for various running processes. As part of maintaining your filesystems, you might want to enforce quotas. The exam will cover the implementation of quotas as well as the various parameters you can define. Quotas can be set as to number of files or amount of space that is used.

Links provide a way to make a file available under different names or from separate locations without wasting disk space with multiple copies of it. There are two types of links: hard and symbolic. Changes made to a file will be seen whether it is accessed via a hard or symbolic link. Hard links have the same inode number, but symbolic links do not. You can delete any hard link and the other links will still work. However, if you delete the referenced file, symbolic links will no longer work. Users can create symbolic links to directories but not hard links. Be sure you understand the purpose of the `user.quota` file as well as how to set both soft and hard limits.

You should be familiar with the Linux hierarchy and understand the appropriate placement of files. Executable files are generally contained in a `bin` directory. There is both a `/bin` and a `/usr/bin` directory. Files are placed according to who will need to use them. The ability to find files is important to users as well as administrators. You can expect to see the `find`, `locate`, and `which` commands on the exam. Maintaining the `locatedb` file to locate new files is also important, as is using the `find`, `locate`, and `which` commands.

QUESTIONS AND ANSWERS

1. You have just received a CD containing a new application you want to install; however, when you insert the CD in your computer, you cannot access it. Why is this?

A. To access removable media, you will need to use the **`mount`** command to place it on the Linux hierarchy. If the CD-ROM filesystem is defined in the **`/etc/fstab`** file, you can mount it by referencing the mount point. Otherwise, you will need to specify the device, mount point, and filesystem.

2. Several of your users have come to you asking why they cannot save a file to a floppy disk. What is causing this and how can you allow your users to use the floppy?

A. Just like a CD-ROM, floppies must be mounted. If your users know how to use the **`mount`** command but are still getting errors, you have not given them the right to mount it. Edit the **`/etc/fstab`** file and add the **`user`** option to the line defining the floppy filesystem.

3. When you look at the contents of the **`/proc`** directory you see several directories with names that are numbers. What are these?

A. The **`/proc`** directory is actually a virtual directory that provides access to the contents of your system's memory. These numbered directories are the various processes currently running on your computer.

4. You want to limit the amount of disk space each of your users can use. What should you do?

A. You can implement quotas for each of your users. You can set these for the number of files and directories they can have and the amount of disk space they can use.

…continues

...continued

5. The marketing department has created a report covering a recent survey. Each of the marketing employees wants a copy in their home directories to review the data. What should you do?

A. Rather than wasting space with multiple copies, create a link in each user's home directory to the original file. If the original file and the home directories are on separate partitions, use a symbolic link.

6. One of your users comes to you and says she thinks she has lost a document that she created yesterday. Even though she does not remember what name she gave the document, how could you locate this file?

A. The first thing to do is to use the **`find`** command to look for any files modified yesterday in that user's home directory. Chances are the file is there, she just does not recognize the name.

7. You want all users' documents to be referenced in the **`locatedb`** database. How can you make this happen?

A. You can designate which directories should be included by the **`updatedb`** command, which creates the **`locatedb`** database.

8. You install a new application to **`/usr/bin`**; however, it does not behave as you expect. What could be the problem?

A. It could be that you have two commands with the same name. Use the **`which`** command to see which one is being executed.

PRACTICE TEST

1. You mount a DOS floppy but the filenames are truncated. When you look at the floppy from Windows 95, long filenames are displayed. Which of the following commands will correct this problem?

- **a.** `mount -t vfat /dev/fd0 /mnt/floppy`
- **b.** `mount -lf /dev/fd0 /mnt/floppy`
- **c.** `mount /mnt/floppy --long_filenames`
- **d.** `mount -o ro /dev/fd0 /mnt/floppy`

Answer a is correct; this command specifies the vfat filesystem, which supports long filenames and specifies both the device and the mount point. Answer b is incorrect; there is no `-l` option for the `mount` command and the `-f` option would only check to see whether the filesystem was mountable without mounting it. Answer c is incorrect; there is no `--long_filenames` option. Answer d is incorrect; this would mount the floppy with the filesystem defined in the `/etc/fstab` file, but as read-only.

2. After installing a new modem you want to see what interrupt it is using. What command will show this?
   a. `ls /proc/interrupts`
   b. `cat /proc/interrupts`
   c. `which interrupts`
   d. `mount interrupts`

Answer a is incorrect; the `ls` command would return `/proc/interrupts`. **Answer b is correct; the `cat` command will list the contents of the `/proc/interrupts` file.** Answer c is incorrect; the `which` command returns the complete path to a specific command. Answer d is incorrect; the `mount` command is used to make a filesystem available via the Linux hierarchy.

3. How can you get a listing of all the filesystems that are presently mounted?
   a. `mount -a`
   b. `mount -f` *`filesystem`*
   c. `cat /etc/mtab`
   d. `cat /etc/fstab`

Answer a is incorrect; this would mount all the filesystems defined in `/etc/fstab`. Answer b is incorrect; this would check the specified filesystem to see whether it is mountable. **Answer c is correct; `/etc/mtab` contains a list of the mounted filesystems.** Answer d is incorrect; the `/etc/fstab` lists the defined filesystems.

4. Which of the following steps is not required to set up disk quotas for your users?
   a. Reboot the computer
   b. Create the `quota.user` file in the root of the filesystem
   c. Edit the `/etc/fstab` file
   d. Use the `eduquota` command to set the limits

Answer a is incorrect; you will have to reboot the system to activate quotas. Answer b is incorrect; you will need a `quota.user` in the root of each filesystem where you want to enforce quotas. **Answer c is correct; you need to edit the `/etc/fstab` to add the quota option to the desired filesystem.** Answer d is incorrect; you use the `eduquota` command to set the quota limits.

5. The graphic arts department has a large project underway and has exceeded the group's soft limits. Because the project is expected to continue for another six months, what should you do?

- **a.** Instruct each artist to change his group before saving any files.
- **b.** Extend the time limit on the soft limit for the group.
- **c.** Increase the hard limit for each user.
- **d.** Increase both the soft and hard limits for the group.

Answer a is incorrect; if each artist changes his group, then the other artists will not be able to access the new files. Answer b is incorrect; this will only allow the users to exceed the soft limit for a greater period of time, but because the hard limit is the same the total amount of disk space available does not change. Answer c is incorrect; this will affect only the user's individual quota and not the group's quota. **Answer d is correct; this will allow the group to use additional space.**

6. The HR department has an employee database located on `/dev/hda2` that needs to be accessed by several people who have their home directories on `/dev/hda1`. Which of the following will provide this access?

- **a.** Place a copy of the database in each user's home directory.
- **b.** Create a link to the file using the `ln -d` command.
- **c.** Create a link to the file using the `ln -s` command.
- **d.** Create a link to the directory using the `ln -s` command.

Answer a is incorrect; this is a waste of disk space and the copies would not stay up to date. Answer b is incorrect; the `ln -d` command is used to create a hard link to a directory. **Answer c is correct; because the file and home directories reside on different partitions, you need to create a symbolic link.** Answer d is incorrect; although this would work, they need a link only to the file and not the directory.

7. When you issue the command `ls -l *memo` you get the following output:

```
-rw-r--r-- 2 hadden hadden 23 Aug 13 11:35 bobmemo
lrw-rw-rw- 1 hadden hadden  4 Aug 13 12:13 billmemo -> bobmemo
-rw-r--r-- 2 hadden hadden 23 Aug 13 11:35 marymemo
```

What are you looking at?

- **a.** The file `marymemo` is a symbolic link to `bobmemo`.
- **b.** The file `bobmemo` is a symbolic link to `billmemo`.
- **c.** The file `marymemo` is a hard link to `bobmemo`.
- **d.** The file `bobmemo` is a hard link to `billmemo`.

Answer a is incorrect; the file `marymemo` is a hard link to `bobmemo`. Answer b is incorrect; the file `bobmemo` is the original and the file `billmemo` is a symbolic link to it. **Answer c is correct; the file `marymemo` is a hard link to `bobmemo`; to verify this you could use the `ls -i`**

command to see the inode number for each of these files. Answer d is incorrect; the file `bobmemo` is the original and the file `billmemo` is a symbolic link to it.

8. You are installing a new application for your users. Where should you put it?

- **a.** `/etc`
- **b.** `/usr`
- **c.** `/var`
- **d.** `/home`

Answer a is incorrect; the `/etc` directory contains system configuration files. **Answer b is correct; the `/usr` directory should contain user applications; these should be placed in the `/usr/bin` directory.** Answer c is incorrect; the `/var` directory contains files such as log files. Answer d is incorrect; the `/home` directory should contain your users' home directories.

9. You want a list of all the hard links that have been created to the employee database. Which of the following commands would you use?

- **a.** `find -inum`
- **b.** `locate`
- **c.** `which`
- **d.** `find -links`

Answer a is correct; by specifying the inode number of a file, you can locate all files with the same inode number that are its hard links. Answer b is incorrect; the `locate` command is used to find a file by name. Answer c is incorrect; the `which` command is used to see which command would be executed. Answer d is incorrect; this would find files with a specific number of links.

10. You install a new version of an application but it does not work correctly. What should be the first thing you do to troubleshoot this problem?

- **a.** Use the `locate` command.
- **b.** Examine the source code.
- **c.** Use the `which` command.
- **d.** Reinstall the application.

Answer a is incorrect; this would display the path to the command. Answer b is incorrect; first you need to make sure that something is not wrong with your installation. **Answer c is correct; this will tell you if you are really executing the new application or another command with the same name.** Answer d is incorrect; reinstalling without identifying what the exact problem is would be a waste of time.

CHAPTER PREREQUISITE

You should be comfortable with typing at the command line, as discussed in Chapter 5, "GNU and UNIX Commands," and have a good understanding of the Linux filesystem, which was covered in Chapter 3, "Linux Filesystem."

CHAPTER

7

Users and Groups

WHILE YOU READ

1. When checking out the various files on your system, you look at the contents of the file `/etc/passwd`. What does each line mean?
2. How can you change the password for multiple users at one time?
3. You are creating new users by editing the `/etc/passwd` file but do not have each user's full name. What should you do?
4. Your boss tells you to implement shadow passwords on the system you are administering. How do you do it?
5. You are planning on using the `useradd` command to create new accounts, but first you want to know what the default actions will be. How can you find this out?
6. You need to limit access to the payroll records to only three people. What is the best way to do this?
7. You want each new user to have his own `.profile` file. What is the best way to make sure this happens?
8. You have installed a new application in its own directory and want to make it available for all your users. What should you do?

Lab

SEE APPENDIX F

Manage Users and Groups

When Linux is installed, root is the only user account that is created. Because access to a Linux system requires a user account, the administrator will have to create accounts for all other users.

root

The root account is known also as the superuser because this account can do anything it wants, including accessing any file or device. The root account also can delete anything on the system, possibly resulting in a corrupted system. For this reason, you should be very careful when doing anything while logged on as root.

The root account should be used only when necessary, such as for system maintenance. To further protect your system, assign to the root account a strong password that is not easily guessed. The password should be changed frequently. Selecting passwords is discussed later in this chapter.

The root account is the account that has an ID number of zero and is defined in the file `/etc/passwd`. There is no rule that the root account cannot be known by any other name because the system identifies all users by UID and not username. Although the superuser account is given the name “root” during installation, you can change it after the system is installed—as long as the UID remains zero (0).

Key Concept

The superuser account is known as root by default. However, you can change its name to anything else you want. The only requirement is that the UID must be zero (0).

Another way to protect your system is by creating special login accounts to be used when performing certain system tasks that do not need extensive privileges. By creating logins that have been granted limited administrative privileges, you protect your system from inadvertent mistakes as well as malicious mischief.

The *su* Command

The `su` command provides you with the capability to perform a task with credentials different from those of your present logged-in identity. It is used in two different contexts:

- When you are logged in as a user and need to do something as root
- When you are logged in as root and want to impersonate another user

You should not perform day-to-day tasks while logged in as root because it puts your system at risk. If you need to perform an administrative task, you could log out and then log in as root. While logged in as root, you can perform the task and then log out and log back in as a user.

This is time-consuming, and using the `su` command provides a shortcut. If you type `su` at a command prompt, you will be prompted for root's password. If you enter the correct password, you will be logged in as root and your working directory will be the same as before you executed the command.

If you want to log in as another user, issue the command

```
su username
```

This will prompt you for that user's name. If you supply the correct password, you will be logged in as that user. If you run this command while logged in as root, you will not be prompted for a password.

By default, the `HOME` and `SHELL` variables are reset to that of the user. If you also want configuration files to be read, you can use the `-` option. What this means is, if you type

```
su Mary
```

and supply Mary's correct password, your working directory will remain the same; however, you will be logged in as Mary. If you also want to have Mary's home directory as your working directory, type

```
su - Mary
```

Now you are logged in as Mary and her home directory is your working directory.

If no username is provided when using the `su` command, root is assumed. Any user can use the `su` command to log in as root or any other user, as long as she knows the correct password. However, root can log in as any other user without needing to know the password.

Creating User Accounts

Even if you are the only user on your system, the first task you should address after installing Linux is to create a user account for yourself. You also will need to create user accounts for everyone else who might need to access your system.

By having every user use his own account, you increase the security of your system. You can define how much access any user has to any resource. This includes setting permissions on any file or directory on a user-by-user basis. (See Chapter 9, "Permissions," for more information on managing permissions.) In addition, this will enable you to track a user's activity.

The passwd File

All information about user accounts is included in the file /etc/passwd in a specific format. For the exam, you need to know what each field is used for and how other programs use this information. This file contains the user's login name, user ID, password, full name, and other information as shown in Table 7.1.

Table 7.1 Fields Contained in the /etc/passwd File

Field	*Contents*
username	Must uniquely identify the user
password	User's encrypted password
user ID (UID)	Unique number that the operating system uses to identify the user
group ID (GID)	Unique number that identifies the user's group
comment	User's full name, phone number, location, or other information
user's home directory	Directory in which the user is placed when she logs on
login command	Command that is executed upon logon, usually a shell

Chapter 9 discusses how the permissions are set on the /etc/passwd file. This file is owned by root with root having rw- permissions. Everyone else, including the group, can only read this file.

A sample /etc/passwd file might look like this:

```
root:Mrczx.kSgtYoQ:0:0:root:/root:/bin/bash
hadden:h4cSUQw27lGJs:503:500:Theresa Hadden,Home Office,555-5555,222-2222:/home/
hadden:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
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:/:
```

Even if a field is blank, it still needs to be defined by a colon (:). For example, the record for the user account mail ends in a colon because no login command is defined for this user. This same format is used on almost all UNIX systems.

Key Concept

Each line in the `/etc/passwd` file must contain seven fields. Although some of these fields might be blank, they still need to be defined by colons.

Username Field

The first field is the username. It must be unique to the system. It is usually eight characters or less and can contain letters, numbers, underscores, periods, and some special characters. However, the use of non-alphanumeric characters can lead to problems with some applications. In addition, usernames are case sensitive, so the usernames thadden and Thadden are two different users.

Because usernames are used for many purposes including mail and must be unique to your system, you need to consider how you are going to assign usernames. If you have only a few users on your system, it is not as important to be consistent with usernames. However, when a system is larger, it becomes too difficult to figure out just who a user is if there is not a logical method of assigning usernames.

One method of naming users is by using a combination of the first initial and part or all of the last name. So, a user by the name of Theresa Hadden would be given a username of thadden. Another combination could be first name and last initial, such as theresah. Be consistent in whatever method you decide to use.

Password Field

The next field in each record of the `/etc/passwd` file is the password. These passwords are stored in an encrypted fashion by the operating system. If this field becomes damaged in any way, that user will not be able to log in to the system until the administrator assigns a new password to this account.

You can use this field to prevent a user from logging in to the system by placing an asterisk (*) in this field. Although the user will be unable to log in, she might still gain access remotely and run commands, so don't depend on this method as your only line of defense.

If you need to temporarily restrict a user's access, place an asterisk at the beginning of the password field for her account. Be careful not to make any other changes to this field. Although this is not 100 percent foolproof, it is usually adequate. You can use this method for users who are on vacation or away for an extended period of time.

If this field is left empty, only the username is required to log in. The user will not be prompted for a password. Although this is sometimes helpful, if you want to have a guest account, it is also a significant security risk. Make sure that if you have an account with a blank password, you lock the account's access down as much as possible.

User ID Field

The third field in each record is the user ID (UID). The system tracks users by UID rather than by name. For example, when a user starts a process, his UID is associated with that process and is displayed when using the `ps` command.

Generally, ranges of UIDs are used to define the type of account. UIDs less than 100 are usually reserved for system accounts, root always has a UID of 0, and users are given UIDs greater than 100. The starting number for user IDs varies with different distributions but the principle is the same. The beginning user ID is defined in the `/etc/login.defs` file, an example of which might look like the following:

```
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7
```

```
#
UID_MIN                    500
UID_MAX                  60000

# Min/max values for automatic gid selection in groupadd
#
GID_MIN                    500
GID_MAX                  60000

# Require password before chfn/chsh can make any changes.
#
CHFN_AUTH                yes

# Don't allow users to change their "real name" using chfn.
#
CHFN_RESTRICT            yes

# If defined, this command is run when removing a user.
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD    /usr/sbin/userdel_local

# If useradd should create home directories for users by default
# This option is ORed with the -m flag on useradd command line.
#
CREATE_HOME     yes
```

Group ID Field

The group ID (GID) is used to define what group the user is a member of when she logs in. This is referred to as the user's default login group. Groups are used to organize users. The GID is used by the operating system to track file permissions and is defined in the `/etc/group` file, which is discussed later in this chapter.

System groups are usually numbered 0–49, and the numbering of user groups starts at 50 or more depending on the distribution. The beginning user group ID is defined in the `/etc/login.defs` file.

On many systems, a default group called group or users gets the first GID available for use by a user group. All users are then made members of this group.

Comments Field

This is not a required field but can be used to provide useful information. If the field is left blank, be sure to include the colon to designate that it is blank.

Most commonly, this field includes the user's full name, but it also can include phone numbers, departments, and other information. The `finger` command uses this field when returning user information. Email systems also can use this field to show who is sending mail.

Home Directory Field

When a user logs on to the system, the login process uses this field to determine where to locate the user in the directory structure after a successful login. For users, this field usually contains the path to the user's home directory. If this field contains an invalid directory name, the user's login attempt will fail.

Login Command Field

This field defines what command to execute when the login process terminates. Most of the time this will be a command that will start a shell program. This command can be used to restrict what a user can perform, such as to run a single application. If this field is left blank, the default shell is usually run.

This entry can be changed by the user by using the command `chsh`. The command `chsh` consults the `/etc/shells` file to determine whether the user's entry is valid.

The *passwd* Command

Passwords are changed using the `passwd` command. Any user can change his own password, and root can change any user's password. The syntax for `passwd` is

```
passwd [user]
```

When a user changes his password, he does not need to specify his username; however, he will be prompted for his old password before entering the new one. When root changes a user's password, it specifies the username but does not have to know the user's present password.

The `passwd` command requires that the new password be entered twice before it is changed. This helps prevent users locking themselves out by making a typographical mistake.

Key Concept

The `passwd` command is used to change passwords. Users can change only their own passwords, but root can change anyone's password.

Many passwords also can be changed at once as a batch by using the `chpasswd` command by root. Its syntax is

```
chpasswd [option]
```

The `chpasswd` command reads a file containing the username and password combinations in the format username:password with one user's entry per line.

By default, the `chpasswd` command encrypts the password before storing it. However, you can provide the passwords in an encrypted format and use the `-e` option, which will prevent `chpasswd` from performing any encryption.

Password characteristics are defined in the `/etc/login.defs` file. Use this file to define how long a password must be, when the password expires, and how long before expiration the user will be prompted to change her password. It also defines the directory where mailboxes are.

Passwords are your first line of defense in protecting your system. You should periodically educate your users about the need to not share their passwords, to change their passwords on a regular basis, and to choose a good password.

Passwords should be at least six characters in length. Using a dictionary word, username, pet or spouse name, or other easily guessed password should be discouraged. The use of mixed case, numbers, and symbols strengthens your passwords. Because the strongest passwords are also those that are the hardest to remember, they are often written down and stored on the monitor, under the keyboard, or in a desk drawer.

One way to prevent this behavior is to encourage your users to use two easily remembered words joined by a symbol, such as `dog$cat`. This password can be further strengthened by either adding numbers and/or uppercase characters, such as `1dog$Cat`. If users construct their passwords in this manner, they create stronger passwords that they can remember easily.

Shadow Passwords

If the password field in your /ect/passwd file is x, your system is using a shadow password file. This file contains the username and the user's encrypted password. It also can contain the password aging information, such as when a password expires. The available fields in the /etc/shadow file are

- The username
- The encrypted password
- The number of days that have elapsed between Jan. 1, 1970, and the date when the password was last changed
- The minimum password age or how long a user must wait before changing his password again
- The maximum password age or how many days a user can go without changing his password
- How much warning a user gets before his password expires (in days)
- How many days to wait after a password expires before an account is disabled
- Expiration date for the account in number of days since Jan. 1, 1970

When using shadow passwords, the password field cannot be blank. To prevent a user from changing his password, make the minimum password age greater than the maximum password age. Implementing shadow passwords for users and groups is covered later in this chapter.

Creating Users

To log on to a Linux system a user must have at a minimum a username and a home directory. Everything else is created for other reasons, such as using a password to enhance the security of the system.

Several different ways exist to add user accounts to your system. Each method has its own advantages and disadvantages. In addition, each distribution provides its own tools that can be used to accomplish this task. For the exam, you need to understand two methods: manually editing the /etc/passwd file and using the useradd command.

Editing the */etc/passwd* File

One way to add users to your system is by directly editing the /etc/passwd file. This can be done only when you are logged in as root. Before making any changes, be sure to make a copy of the original file. If you corrupt the file, you will not be able to log in, even as root.

To add a user, simply add a new line at the end of the file that contains the appropriate information for each field. Be sure to use an editor that saves the information in ASCII, such as `vi`. Do not enter anything in the password field because it needs to be encrypted.

You will need to enter a unique UID for each new user you add to the file. Keep a record of assigned UIDs and assign new ones sequentially. To create a new account for me, you might add the following line to your `/etc/passwd` file:

```
hadden::523:100:Theresa Hadden:/home/hadden:/bin/bash
```

After entering the user information and saving the file, you will need to use the `passwd` command to assign a password to the new account.

Next, you will need to create the user's home directory and manually copy the appropriate configuration files. These configuration files are discussed later in this chapter.

If a new group is also needed for this user, you will need to edit the `/etc/group` file to define the group before the user can log in. Groups are discussed later in this chapter.

If you manually create a user's home directory, you will need to change both the owner and group assigned to that directory or the user will be unable to log in. Use the `chown` command to change both the user and associated group. See Chapter 9 for more information on using this command.

Key Concept

New user accounts can be created by editing the `/etc/passwd` file and then using the `passwd` command to assign a password to the account. You also will need to create the user's home directory and assign the appropriate permissions to this directory before the user will be able to log in.

Using the *useradd* Command

Another, more convenient way to add user accounts to your system is by using the `useradd` command. By default, `useradd` uses the next available UID and assigns the default shell as the login command for that account. To display the system's default values, type `useradd -D` without a username. Its output might look like the following:

```
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

If you issue the command

```
useradd mary
```

and then examine the `/etc/passwd` file, the last line might look like the following:

```
mary:!!:505:506::/home/mary:/bin/bash
```

The `!!` in the password field is entered by the system to indicate that a password has not been assigned. You would need to use the `passwd` command before Mary could log in.

In the previous example, the `useradd` command assigns a UID of `505` to the account. This is the next available UID and makes the group with a GID of `100` Mary's default login group. You can specify a UID by using the `-u` *`n`* option. To specify a GID to assign to this account, use `-g` *`n`*. Be sure that you use a valid UID and GID.

If the ID you specify is not unique (user or group), the command will fail and the account will not be created. Use the `-o` option if you want to use an ID that is not unique.

Mary's home directory is named, but it is not created by default. Using the `-m` option causes `useradd` to also create the user's home directory when creating the account. The default directory is `/home/`*`<username>`*.

To explicitly specify a home directory use `-d <path>/<directory>`. If you do not have the `useradd` command, you will need to create the home directory and change its ownership and associated group before the user will be able to log in.

You also can have specific files copied to the new home directory by using `-k <skeleton directory>` in conjunction with the `-m` option. If no directory is specified, the files located in `/etc/skel` will be copied to the newly created home directory.

If directories are present in either the skeleton directory or `/etc/skel`, those directories will be created as well. In addition, the `useradd` command will set the appropriate permissions to these files and directories.

Use this when you have files or directories you want to be available for most of your users. You must create the files and directory structure in either the `/etc/skel` directory or another directory of your choosing. Next, create the user account using the `useradd -m` command and the files will be copied to the new user's home directory.

The comment field is left blank unless you specify its contents by using `-c "information"`. You must enclose in double quotes the information you want in the comment field.

The system's default shell was assigned to Mary as her login command. You can use the `-s` option to specify a different shell. However, the designated shell must be defined in the `/etc/shell` file or the user's login attempt will fail.

In addition, you can specify when you want a user's account to expire by using the -e date option. The date must be in the format MM/DD/YY. After this date, the user will no longer be able to log in to the system.

You can specify to have an account disabled after the password has expired by using the -f_days option. If you enter a value of 0, the account will be disabled as soon as the password expires. A value of -1 prevents the account from being disabled when the password expires. This is the default action.

Key Concept

You can use the useradd command to create a new user account. The system defaults for the home directory and login command will be used unless you specify otherwise. Use the -d option to specify the home directory and -s to change the login command. Use -m to have useradd create the home directory and copy the default skeleton files. The -k option enables you to specify the location of the skeleton files. Use -c to assign a value to the comment field and -e to specify an expiration date.

Using the *usermod* Command

After you create a user account you can modify any of the fields in the /etc/passwd file by using the usermod command. Table 7.2 shows the options used to modify each of these fields.

Table 7.2 Options Used with **usermod** to Modify Fields in the **/etc/passwd** File

Option	*Field Modified*
-l *<username>*	Username
-u *n*	UID; use -o if not unique
-g *n*	Login group
-c "*information*"	Comment
-d *<path/dir>*	Home directory; use -m to copy files from the old to the new home directory
-s *<path/shell>*	Login shell

You can't change the username if the user is logged in. In addition, you should change the name of the user's home directory to match her new username. Changing the name of the directory is not required as long as the home directory that is defined for that user exists.

When changing a user's ID, it must be unique unless you use the `-o` option. Be sure that the user is not logged in and verify that there are no processes running with that user's old ID.

Any files located in the user's home directory will have the UID associated with them changed; however, any files belonging to the user that are located elsewhere will have to be changed manually. Any `crontab` or `at` jobs also will need to be changed.

If you use the `-m` option when using the `-d` option to designate a new home directory, the files contained in the present home directory will be copied to the new location with the permissions intact.

Deleting Users

When a user no longer requires access to your system, you should remove his account, files, and any `crontab` or `at` jobs he might have running. To manually remove a user, follow these steps:

1. Remove that user's line from the `/etc/passwd` file.
2. Delete the user's home directory.
3. Search for and delete files owned by that user located outside his home directory.
4. Delete the user's mail and any mail aliases.
5. Delete any jobs he might have scheduled.

An easier way to perform this same task is to use the `userdel` command. Its syntax is

```
userdel [option] username
```

Using the `userdel` command without an option will delete the user from the `/etc/passwd` and other system files but will not delete his home directory. Use the `-r` option to also have his home directory deleted. You will still need to delete any files he owns that are located outside his home directory.

If you want to disable a user's account rather than delete it, add an asterisk at the beginning of the password field in `/etc/passwd`. Do not change the encrypted password itself—this will prevent the user from logging in. When you want to reactivate the account, merely remove the asterisk and the user can log in.

Special User Accounts

If you look at your `/etc/passwd` file or the example at the beginning of this chapter, you will see several user accounts that are system accounts. Each of these has a special purpose, some of which are listed in Table 7.3.

Table 7.3 System Accounts and Their Uses

Username	*Purpose*
root	Superuser of the system
daemon	Owns and sets permissions on system processes
bin	Owns executables
sys	Owns executables
adm	Owns log files
uucp	Used for UUCP access

Other system accounts control processes such as mail, ftp, gopher, news, and so on. Their purpose is usually obvious from their name. Never change the entries for these accounts because this can cause your system to fail. You also will notice that these system accounts have an asterisk in the password field to prevent anyone from logging in and impersonating one of these accounts.

Groups

Groups are used to organize your users and grant permissions to files. You might want to create groups based on the need to access certain files such as accounting information. Or, you might create groups to manage access to devices such as modems or printers. Groups can help you simplify your job.

The */etc/group* File

The `/etc/group` file defines each group and its members. The `/etc/group` file has a format similar to that of the `/etc/passwd` file, such as in

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root
lp::7:daemon,lp
mem::8:
kmem::9:
wheel::10:root
mail::12:mail
news::13:news
uucp::14:uucp
```

Each line in the `/etc/group` file has four fields, each separated by a colon. If a field is blank, the colon is still required. Table 7.4 lists the fields and their contents.

Table 7.4 The Fields Contained in Each Record in the **/etc/group** File and Their Contents

Field	*Contents*
Groupname	The group name must be unique and can be up to eight characters in length
Password	A password can be entered but usually it is either an asterisk or left blank
Group ID	The GID is used by the system and must be unique
Members	List of users who are members of a group separated by commas

When a user attempts to access a file, the operating system first checks to see whether that user is the owner of the file. If she is not the owner, the OS checks to see whether the user is a member of the associated group.

If the user belongs to the appropriate group, she gets the access defined for the group. If she is not a member of this group, she gets whatever permissions have been assigned to all others. Access to resources is discussed in more detail in Chapter 9.

Identifying Your Group

Every user on your system must belong to at least one group. If only one group exists, make all your users members of that group. Users can belong to many groups; however, only one GID can be associated with a user at any point in time. The default group for each user is defined in the `/etc/passwd` file.

To find out what your present default group is, use the `id` command. It returns your UID, the GID, and the name of your default group. In addition, it lists all your group memberships. Its output might look like the following:

```
uid=503(hadden) gid=500(hadden) groups=500(hadden),520(wp)
```

You also can determine to which groups you belong by using the `groups` command. It outputs the groups to which you belong but does not indicate your present default group.

When a user creates a new file, his present group is associated with that file. You can change which group your default group is by using the command `newgrp` and specifying the desired group. For example, typing

```
newgrp wp
```

will make your present group the `wp` group. Now when you type `ID`, you will see

```
uid=503(hadden) gid=520(wp) groups=500(hadden),520(wp
```

If you are a member of the new group as defined in the `/etc/group` file, your default group is changed and any files you create while that group is your default group will be associated with that group. You also can use the `newgrp` command without any arguments to change to your default login group as defined in the `/etc/passwd` file.

Working with Groups

You create new groups either by editing the `/etc/group` file directly or by using a utility that will create the new entry for you. When editing the `/etc/group` file, add one line for each new group. Be sure to follow the syntax exactly. If you added the line

```
wp::520:mary,john,theresa
```

to the `/etc/group` file, you would create a new group called `wp` with a GID of `520`. This group would have Mary, John, and Theresa as members.

You also can use the utility `groupadd` to create new groups. Its syntax is

```
groupadd [options] groupname
```

To add a new group called `acctng`, you would issue the following command:

```
groupadd acctng
```

This would add a line to the `/etc/group` file defining the group and then assign it the next available user GID. You can specify a GID by using the `-g` *`gid`*, but it must be unique. If you want to use a GID that's not unique, you must use the `-o` option. If you want to create a system group instead, use the `-r` option. System groups are discussed later in this chapter.

The `groupadd` command creates the group but does not add any users to that group. You still will need to populate your group with the appropriate users. You can do this either by editing the `/etc/group` file or by using the `gpasswd` command with the `-a` option. The syntax for the `gpasswd` command is

```
gpasswd [options] <user> group
```

To add Joe to the `wp` group, you would type

```
gpasswd -a Joe wp
```

The `gpasswd` command also is used to change the password on a group by passing the group's name as an argument. If a password is set on a group, nonmembers can join the group with the appropriate password. Use the `-r` option to remove a password from a group. Table 7.5 lists options that can be used with the `gpasswd` command.

Table 7.5 Options to Be Used with the **gpasswd** Command and Their Actions

Option	*Action*
`-R`	Prevents the use of `newgrp` to join the group
`-a` *username*	Adds named user to the group
`-d` *username*	Removes named user from the group
`-A` *username*	Makes named user an administrator for that group
`-r`	Removes password from the group

Modifying and Deleting Groups

You can change the name of a group or its GID either by editing the `/etc/group` file or by using the `groupmod` command. The syntax for the `groupmod` command is

```
groupmod [options] groupname
```

Use the `-n` *name* option to change the group's name or `-g` *GID* to change the GID. If you are changing to a GID that's not unique, you must use the `-o` command. If you change a group's GID, be sure to check the `/etc/passwd` file to see whether any user has that group listed as her login group. If so, be sure to change the GID here as well.

To delete a group, you can either remove its line from the `/etc/group` or use the `groupdel` command. If you delete the group by editing the `/etc/group` file, you will need to do a couple of other things.

1. Check the `/etc/passwd` file and verify that none of the users have the deleted group as their login group. If they do and you do not change it, they will not be able to log in.
2. Look for any files and directories that have this group as its owner. If you do not correct this, it can make the files inaccessible.

The syntax for the `groupdel` command is

```
groupdel groupname
```

This will delete the group's line from the `/etc/group` file. If you try to use this command on a group that is designated as any user's login group, it will fail.

You will need to review the `/etc/passwd` file to make sure that no user has this group designated as her login group. If any users are using this group as their default login group, you will need to change this before you can delete the group. You also will still need to check the filesystem for resources owned by this group.

Key Concept

Groups are defined in the `/etc/groups` file. Commands that can be used to manage groups are `groupadd`, `groupdel`, `groupmod`, and `gpasswd`.

System Groups

The system creates a number of groups such as bin, mail, sys, adm, and so on. Do not add users to any of these groups because this would give them wide-ranging rights on your system. System groups should have only system accounts as members. Some of the system groups and their functions are listed in Table 7.6. You should be familiar with these groups and their functions for the exam.

Table 7.6 System Groups and Their Functions

Group	*Function*
root	Owns most system files
daemon	Owns mail, printer, and other system software and directories
kmem	Manages direct access to kernel memory
sys	Owns some system files, swap files, and memory images; can work the same as the kmem group
nobody	Owns software that does not need special permissions
tty	Owns device files that control terminals

Implementing Shadow Passwords

As you remember, the passwords stored in the `/etc/passwd` file are encrypted by the system. An additional level of security can be invoked by using shadow passwords. When using shadow passwords, the password field in the `/etc/passwd` file is replaced by an `x` and passwords are actually stored in the `/etc/shadow` file.

The `/etc/shadow` file should be readable only by root in an effort to keep outsiders from trying to crack your passwords. See Chapter 9 for more information on restricting access to files. You can provide this same level of security for the passwords assigned to groups.

Group shadow passwords are stored in the `/etc/gpasswd` file, and the password in the `/etc/group` file is replaced with an `x`. This file should be restricted to root-only access. No one else should even be able to read it.

You should never directly edit the /etc/shadow or /etc/gshadow files. When you need to change a user's or group's password, use a utility such as useradd, passwd, usermod, or gpasswd to make changes to it.

To begin using shadow passwords, use the pwconv command, which takes no arguments. When you run this command, it creates a /etc/shadow file from your /etc/passwd file. You always can convert back by using the pwunconv command. Using the pwunconv command will update your /etc/passwd file and then delete the /etc/shadow file.

To use shadow passwords for groups, use the grpconv command. This will create a /etc/gshadow file and change all the passwords in the /etc/group file to x. To revert to saving group passwords in the /etc/group file, run the command grpunconv.

If you must edit the /etc/passwd and /etc/group files by hand, you can update your shadow passwords by rerunning the appropriate commands—pwconv for /etc/passwd and grpconv for /etc/group. A better way is to first use pwunconv and grpunconv to stop using shadow passwords, make your changes, and then convert back to using shadow passwords.

Key Concept

Shadow passwords increase the security of your system. Use pwconv to implement shadow passwords for your users and grpconv to use shadow passwords for your groups.

User Startup Files

The look and function of your shell results from the use of environment variables. These parameters are saved in various configuration files located either in /etc or /home/<username>. The default environment variables for bash when started as an interactive login shell are located in the /etc/profile file. The following is an example of a profile file:

```
# /etc/profile

# System wide environment and startup programs
# Functions and aliases go in /etc/bashrc

PATH="$PATH:/usr/X11R6/bin"
PATH="$PATH:/usr/local/bin/blender"
PS1="[\u@\h \W]\\$ "
BLENDERDIR=/usr/local/bin/blender
export BLENDERDIR
```

```
JAZZ=/usr/local/jazz
export JAZZ

ulimit -c 1000000
if [ `id -gn` = `id -un` -a `id -u` -gt 14 ]; then
        umask 002
else
        umask 022
fi

USER=`id -un`

ulimit -c 1000000
if [ `id -gn` = `id -un` -a `id -u` -gt 14 ]; then
        umask 002
else
        umask 022
fi

USER=`id -un`
MAIL="/var/spool/mail/$USER"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000
HISTFILESIZE=1000
export PATH PS1 HOSTNAME HISTSIZE HISTFILESIZE USER LOGNAME MAIL

for i in /etc/profile.d/*.sh ; do
        if [ -x $i ]; then
                . $i
        fi
done
```

Notice how the environment variables (such as the system `umask`), path, hostname, and history size are identified. See Chapter 5 for additional information on environment variables.

To make systemwide changes in your user's environment, make the changes in this file. For example, you might want to add a directory to the path or change the default system prompt. To prevent bash from executing the commands in the profile file, use the `--noprofile` option.

If the user wants some of the default variables to be different, startup files can be placed in the user's home directory. Copy the `/etc/profile` file to `.profile` in the user's home directory and change the ownership of the new file to the user. Now, the user can change any startup parameters, such as her prompt, and they will apply only to that user.

When you create a new user, the defaults located in the `/etc/profile` file are read and executed to create the user's environment. Next, the shell looks for the files `.bash_profile`, `.bash_login`, and `.profile` in the user's home directory.

If one or more of these files is found, they are read and executed in the order listed. If the same environment variable is set in all these files, the variable is set to the last value read.

So, if you want to change your path to something different from the system path as defined in the `/etc/profile` file, you can make that change in the `.profile` file in your home directory. Because this file is the last one read, whatever value it sets will be the value applied to that user's environment. Chapter 5 discusses environmental variables in more depth. When exiting an interactive login shell, bash will read and execute the commands in `/home/`*`<username>`*`/.bash_logout` if it exists.

When a bash shell starts an interactive shell that is not a login shell, such as when you type `bash <enter>` at a command prompt, it reads and executes the commands in the file `/etc/bashrc`. This file also can be called from within the profile file. An example of a `bashrc` file is

```
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile

# For some unknown reason bash refuses to inherit
# PS1 in some circumstances that I can't figure out.
# Putting PS1 here ensures that it gets loaded every time.
PS1="[\u@\h \W]\\$ "

alias lsa="ls -aF --color | more"
alias lsl="ls -alF --color | more"
alias rm="rm -i"
alias cp="cp -i"
alias mv="mv -i"
alias cl="clear"
alias dir="dir --color | more"
alias dira="dir -aF --color | more"
```

When starting bash as an interactive non-login shell, you can inhibit this behavior by using the `--norc` option. You also can specify another file to read instead of `bashrc` by using the `--rcfile` *`<filename>`* option, which will cause bash to execute the commands from the specified file instead.

When bash is called as a non-interactive shell, such as when running a shell script, it looks for the environment variable `BASH_ENV`. This variable can contain the name of a file other than the `bashrc` file to execute. The entire path as well as the file's name is required when setting this variable.

Bash can be invoked using the name `sh` instead of `bash`. When invoking bash in this manner, bash examines the value of the `ENV` variable to identify what file to execute. No other startup files are processed when calling bash in this manner.

An easy way to make sure your new users receive copies of these files in their home directories is to create the appropriate files and save them to `/etc/skel`. Whenever you create a new user with the `useradd` command, these files are copied to the new user's directory and the permissions are appropriately set.

Key Concept

If you choose to create startup files and manually copy them to the user's home directory, be sure you make the user the owner of these files and set the group ownership appropriately. Failure to do this can result in the user being unable to log on or environment variables being inappropriately set.

Summary

Managing users and groups is a significant part of any administrator's job. All user accounts are contained in the `/etc/passwd` file. Each user is defined in a single line and the fields are

- Username
- Password
- User ID
- Group ID
- Comment
- Home directory
- Login command

Each of these fields, which can be blank, is required.

You can create users by editing the `/etc/passwd` file or using the `useradd` command. Other commands used to manage users are `usermod` and `userdel`.

The `su` command is used when you want to impersonate another user. If you are root when you issue the `su` command to impersonate a user, you do not need to provide a password. If you are a user, however, you will need to supply the correct password for the command to complete successfully.

Groups are defined in the `/etc/groups` file. Each group is listed in a separate line. Groups can be managed by editing the `/etc/group` file or using the `groupadd`, `groupmod`, `groupdel`, or `gpasswd` commands.

Several different configuration files are read by the system in different circumstances. The default files for an interactive, login shell are `/etc/bash_profile`, `/etc/bash_login`, and `/etc/profile`. These files can be copied to the user's home directory to make changes that will apply to that user only.

Other configuration files are `.bash_logout` and `/etc/bashrc`. Two environment variables that affect the behavior of your shell are `BASH_ENV` and `ENV`.

- Group name
- Password
- Group ID
- Members

Additional security can be implemented by using shadow passwords for users and groups.

QUESTIONS AND ANSWERS

1. When checking out the various files on your system, you look at the contents of the file `/etc/passwd`. What does each line mean?

A. Each line of the `/etc/passwd` file contains information on one user account. The fields contained in this line (in order) are username, password, UID, GID, comment, home directory, and login command.

2. How can you change the password for multiple users at one time?

A. You can change the passwords of multiple users by giving the `chpasswd` a filename as an argument. This file contains one username/password combination per line. The username and password must be separated by a colon.

3. You are creating new users by editing the `/etc/passwd` file but do not have each user's full name. What should you do?

A. Normally, you would place the user's full name in the comment field. If you do not know the name for one of the new accounts you are creating, you can leave it blank. However, be sure that the colon is there as a placeholder for that field.

4. Your boss tells you to implement shadow passwords on the system you are administering. How do you do it?

A. All you will need to do is issue the command `pwconv`. This will read the `/etc/passwd` file, change all the password fields to `x`, and create a `/etc/shadow` file containing the newly encrypted passwords. Make sure that this file is readable only by root.

5. You are planning on using the `useradd` command to create new accounts, but first you want to know what the default actions will be. How can you find this out?

A. To see what system defaults the `useradd` command will use, simply type `useradd -D` at a command prompt. It will then display the values that have been set as system defaults.

...continues

…continued

6. You need to limit access to the payroll records to only three people. What is the best way to do this?

A. The first thing you will want to do is to create a new group and make these three people members. Next, set the appropriate access rights on the payroll records so that only the members of this group have rights to access them.

7. You want each new user to have his own `.profile` file. What is the best way to make sure this happens?

A. You could copy the file to each new user's home directory and then set the appropriate permissions. However, an easier way is to let the `useradd` command copy the file and set the permissions. Just copy the `.profile` file to `/etc/skel` and use the `-m` option when using the `useradd` command to create the new user account.

8. You have installed a new application in its own directory and want to make it available for all your users. What should you do?

A. First, grant the appropriate permissions. However, users keep calling you and complaining that when they try to run the application, they get an error message that the command could not be found. Edit the `/etc/profile` file and change the path to include the directory of the new application. Now when your users attempt to run the application, the system will be able to find it.

PRACTICE TEST

1. You spend your afternoon updating the comment field for each of your users. Later you get calls from two of your users that they cannot log in. What could the problem be and how should you fix it?

- **a.** They forgot their passwords. Look up their passwords.
- **b.** The password field for their accounts got corrupted. Change their passwords using the `passwd` command.
- **c.** You deleted their accounts; now re-create them.
- **d.** The password field for their accounts got corrupted. Change their passwords by editing the `/etc/passwd` file.

Answer a is incorrect; although they might have forgotten their passwords, it is unlikely that is the problem. **Answer b is correct; if you change the password field when editing the `/etc/passwd` file, the user will not be able to log in; correct this problem by using the `passwd` command to change their passwords so they can log in; they can then change their passwords themselves.** Answer c is incorrect; it is unlikely that you deleted an entire line by mistake. Answer d is incorrect; you should never edit the password field in the `/etc/passwd` file.

2. How does the operating system identify the superuser account?
   - **a.** By its name, root.
   - **b.** By its creation date and time because it is the first account created.
   - **c.** By its UID, which is always zero.
   - **d.** By its password.

Answer a is incorrect; the superuser account can have any name. Answer b is incorrect; the system does not track the creation time of user accounts; **Answer c is correct; the superuser account always has a UID of zero.** Answer d is incorrect; passwords are associated with a username for authentication.

3. You add the following line to the `/etc/passwd` file

   ```
   mary::122:50:Mary Jones:/home/mary:/bin/bash
   ```

   and use the `passwd` command to change her password. You also create her home directory. However, when Mary tries to log in, the login fails. What is the problem?
   - **a.** You did not create her home directory.
   - **b.** You did not assign a valid password to Mary's account.
   - **c.** You did not set the appropriate permissions to her home directory.
   - **d.** You cannot create a new user account by manually editing the `/etc/passwd` file.

Answer a is incorrect; you did create her home directory. Answer b is incorrect; you used the `passwd` command to assign a password to her account. **Answer c is correct; if you did not assign the correct permissions to Mary's home directory, she will not be able to log in.** Answer d is incorrect; editing the `/etc/passwd` file is a valid method to create a new user account.

4. What will the following command do?

```
useradd -m mary
```

   a. Create a new user by the name of mary. Now you need to create her home directory.
   b. Create a new user by the name of mary and create her home directory.
   c. Create a new user by the name of mary with a blank password.
   d. Create a new user by the name of mary, but the account will be disabled.

Answer a is incorrect; you will not need to create her home directory. **Answer b is correct; the `-m` option will cause the `useradd` command to create her home directory as well as create the account.** Answer c is incorrect; you will need to assign a password to her account using the `passwd` command before she can log in. Answer d is incorrect; to disable the account, change the password field in the `/etc/passwd` file to an asterisk.

5. You want to change the name of Mary's account to `mjones`. Which of the following commands will accomplish this?

   a. `usermod -u mjones`
   b. `usermod -l mjones`
   c. `usermod -c mjones`
   d. `usermod -s mjones`

Answer a is incorrect; the `-u` option is used to change the user's ID. **Answer b is correct; the `-l` option is used to change the username.** Answer c is incorrect; the `-c` option is used to update the comment field. Answer d is incorrect; the `-s` option is used to change the user's login shell.

6. Mary has quit her job and you need to delete her account, so you issue the command `userdel mjones`. What else do you need to do?

   a. Delete Mary's home directory and its files.
   b. Delete Mary's mail.
   c. Delete any jobs Mary might have scheduled.
   d. Delete the files that Mary owns that are not in her home directory.

Answer a is incorrect; the `userdel` command will delete Mary's home directory as well as her account. Answer b is incorrect; the `userdel` command will delete Mary's mail and any mail aliases she might have. Answer c is incorrect; the `userdel` command will delete Mary's scheduled jobs. **Answer d is correct; the `userdel` command will not delete any files Mary owns that are not located in her home directory.**

7. The sales department is working on a new promotion and wants to keep its quotes in the same directory. It wants to set the permissions on this directory so only the salespeople can read the files. You are given the job of creating a new group called `promo`. Which of the following commands will work?
   a. `groupadd promo; gpasswd -a bill, joe, karen promo`
   b. `groupadd promo; gpasswd -r bill, joe, karen promo`
   c. `gpasswd -a bill, joe, karen promo`
   d. `gpasswd -r bill, joe, karen promo`

Answer a is correct; first you create the group and then add the group members. Answer b is incorrect; the `-r` option to the `gpasswd` command removes the password from a group. Answer c is incorrect; this command will fail unless the `promo` group already exists. Answer d is incorrect; this command will remove the group's password.

8. You think that someone has been attempting to hack into your system by attacking your `/etc/passwd` file. What can you do to increase the security of your system?
   a. Implement shadow passwords using the `spasswd` command.
   b. Change all the passwords in the `/etc/passwd` file to `*`.
   c. Implement shadow passwords using the `pwconv` command.
   d. Change all the passwords in the `/etc/passwd` file to `x`.

Answer a is incorrect; there is no `spasswd` command. Answer b is incorrect; changing all the passwords in the `/etc/passwd` file to an asterisk will disable all the accounts. **Answer c is correct; the `pwconv` command will read the `/etc/passwd` file, create a `/etc/shadow` file containing the encrypted passwords, and change the passwords in the `/etc/passwd` file to `x`.** Answer d is incorrect; you run the risk of preventing any user from logging in when you edit the password field manually.

9. You have some users who want to have their personal scripts run without having to type the directory name. What could you do?
   a. Change the path in the `/etc/profile` file to include the directories containing the scripts.
   b. Tell your users that it cannot be done.
   c. Create a directory for these scripts and include it in the path in the `/etc/profile` file.
   d. Copy the `/etc/profile` file to each of these user's home directories as `.profile` and make the user the owner of the new file.

Answer a is incorrect; this would make the scripts available to all users and could cause problems depending on the names assigned to the scripts. Answer b is incorrect; you can allow your users to have individual paths. Answer c is incorrect; this also would make the scripts available to all users. **Answer d is correct; by giving each user her own `.profile` file, she can change the path to include her own script directory without giving access to it to other users.**

10. One of your users comes to you and says he is still having problems with his scripts not being on his path. He tells you that he has added a path statement to include the script directory in his `.bash_login` file. What is the problem?
    - **a.** He also has a path statement in his `.profile` file.
    - **b.** You cannot set your path in the `.bash_login` file.
    - **c.** The path statement in the `/etc/profile` file is overriding his changes.
    - **d.** He should have placed the path statement in his `.bash_profile` file.

Answer a is correct; if he has a path statement in both his `.bash_login` and `.profile` files, the one in his `.profile` file is read last and that will be the value that is set. Answer b is incorrect; you can add a path statement in the `.bash_login` file. Answer c is incorrect; the `/etc/profile` file is read first, so any path statement in the other startup files will overwrite it. Answer d is incorrect; even if he added a path statement to his `.bash_profile` file, it would be overwritten by the path statement in the `.profile` file.

CHAPTER PREREQUISITE

Typing at the command line; understanding the different file types; and using grep to search; all topics from Chapter 4, "File Management," and Chapter 5, "GNU and UNIX Commands."

CHAPTER

8

Text Streams

WHILE YOU READ

1. How do you prevent error messages from showing on your screen when running a command?
2. What command could you use to read a file called `MyFile`, replace each instance of "while" with "during", and save the changes to a new file?
3. How can you sort your list of telephone numbers?
4. You have a file with each telephone number and name on a single line. The telephone number and name are separated by a comma. You want to create a list of all your contacts, but want the name first, followed by the phone number. And you want the list to appear in alphabetical order. How could you do this?
5. You need to print a file but want to have only 50 lines printed per page. How would you do it?
6. You want to monitor the system log as entries are made. What should you do?
7. You are writing a report on last month's sales but it must not be longer than 500 words. How can you check the length of your document?
8. What does the `tee` utility do?

Lab

SEE APPENDIX F

Finding Your Files

Before you can begin editing a file, you need to know where it is. Two utilities that can help you locate the file you are looking for are `find` and `locate`, which were covered in Chapter 6, "Maintaining the Filesystem." To help you further, you can use special characters to represent other characters. These are called *wildcards* or *metacharacters*.

Wildcards

Three types of wildcards or metacharacters can be used with the `find` and `locate` utilities to make it easier to find what you are looking for.

Character	*Function*
`?`	Used to match a single character
`*`	Used to return zero or more characters at the beginning or end of an expression
`[]`	Used to return one of the characters defined within the brackets

See Chapter 4 for more information on using metacharacters and searching the content of files.

Text Filters

One of the most important tasks that you will be involved in is creating and modifying text files, such as configuration files. There are many utilities that you can use to automate repetitive tasks when editing text files. You can expect to see these on the exam, so you need to be comfortable with using them. Text filters perform a single job well and are extremely useful in this regard. The tasks that you might want to perform on a text include pagination, search and replace operations, and formatting.

You can use many utilities, or text filters, to automate repetitive tasks when editing text files. The ones that you need to be comfortable with for the exam are `sed`, `sort`, `cut`, `expand`, `fmt`, `head`, `join`, `nl`, `od`, `paste`, `pr`, `split`, `tac`, `tail`, `tr`, and `wc`.

sed

`sed` is a non-interactive text editor. You pass a series of commands to `sed` and it processes your file one line at a time. Remember, however, that `sed` does not change the original file. Rather, it copies the original file, changes the copy, and sends it to either standard output or to a file that you designate.

The syntax for sed is

```
sed [-n] [-e command] [-f command_file] filename [>newfile]
```

We will look at each of the options in this section.

Chapter 4 discussed using sed to search for text in a file; however, it can do much more than that. sed commands can be entered at the command line, as in the following:

```
sed s/term1/term2/ FileName
```

More commonly, however, you will use a file containing the commands in conjunction with sed, as in the following:

```
sed -f ComFile FileName
```

where ComFile contains the command

```
s/term1/term2
```

One of the simplest sed commands is substitution. In the previous example, sed replaces the first occurrence of the first term in each line with the second term. If term1 occurs more than once in a line, only the first occurrence is replaced. To replace every occurrence of term1 with term2, you would use g at the end, as in

```
s/term1/term2/g
```

The g tells sed to perform a global search and replace. You can limit the replacement to a specific occurrence by specifying which occurrence you want to act upon, as in the following:

```
s3/term1/term2/
```

This would replace only the third occurrence of term1 with term2. You also can designate specific lines to process by referring to the line by number:

```
1s/term1/term2/
1,3s/term1/term2/
```

The first example will process only line 1, whereas the second processes lines 1–3. Be sure to note the placement of the numbers. If the number is first, it denotes line number, but if it is after the s (command) then it denotes the number of its occurrence. The following is an example of this:

```
s2/term1/term2/
```

This would replace the second occurrence of term1 with term2.

You also can use the $ instead of a line number to specify the last line. If you do not know the line number, you can use patterns instead to locate the line. The command

```
/StartWord/./StopWord/s/term1/term2/
```

processes the file starting with the first line containing StartWord and ending with the first line containing StopWord. After StopWord has been found, sed looks for another occurrence of StartWord. If it is found, sed will process all commands on all lines until StopWord is found again. This process continues until the end of the file is reached.

If you want to process all lines except those specified, follow the address with !. You also can use a \n to specify a newline character within the pattern. However, it will not match a newline at the end of a pattern.

By default, sed prints the each line of the file after it has finished processing the line. However, if you want to see only the lines that have been changed, use a p. This causes sed to print only each line that has been changed. Of course, sed will print the same line for each change it made. So if it makes three changes to one line, it will print the same line three times (once after making each change).

```
s/term1/term2/gp
```

If you want to do the same thing from the command line, use the -n option with sed, as in

```
sed -n s/term1/term2/gp FileName
```

Both of the previous two examples would change every instance of term1 to term2 and print the changed line(s) to standard output. If you want to use both -n and -f, be sure that -n precedes the -f. You also could use -nf or -fn.

If you want to redirect the changed lines to a file, you can use the w option (if you are saving your commands to a script file). The following is an example of this:

```
s/term1/term2/w OutFile
```

When typing the commands on the command line you can redirect the output to a file by using a pipe, such as >.

In addition, you can use sed to add, delete, or change lines of a text file. Table 8.1 shows the commands you can use when you want to work on an entire line.

Table 8.1 Use These sed Options to Add, Delete, or Replace Text

Option	*Action*
a	Adds after the specified line
b	Adds before the specified line
d	Deletes the specified line
c	Replaces the specified line with the following

The d and c options enable you to specify a range of lines. The c option replaces lines by first inserting the text before the lines and then deleting the lines. Because these commands are used to manipulate larger sections of text, they are usually used only in sed scripts.

You also can use sed to insert another file into the original file by using r, as in

```
/term/r IncludeFile
```

This would place the contents of IncludeFile immediately following the line containing term. You always must specify an address when using r.

sort

The sort command is used to put lists in alphabetic or numeric order. In addition, it can merge one or more text files. The syntax for sort is

```
sort [options] [filelist]
```

sort takes its input from the files you specify or the standard input. Its output is sent to the standard output unless you use the -o option followed by a filename, which writes the output to that file.

The default action is to sort a file in alphabetic order. To sort a file in numeric order, use the -n option. This means that periods as well as plus and minus signs are interpreted in arithmetic order.

sort considers any leading blanks to be part of the field. Use the -b option to cause sort to ignore white space. Other options that you should know are listed in Table 8.2.

Table 8.2 `sort` Options and What They Do

Option	Action
-c	Checks to see that the file is sorted correctly
-d	Ignores all non-alphanumeric characters
-f	Considers all lowercase characters as uppercase
-i	Ignores non-printing characters
-m	Merges listed files
-r	Performs a reverse sort

If you use `-c` with `sort`, it checks to see only whether the file is already sorted. If it is not, an error message is returned and `sort` exits. When using the `-m` with `sort`, the files to be merged must first be individually sorted.

`sort` does not change the original file. The results of using `sort` either go to the standard output or are saved to a file you designate. You can use `-o` with `sort` to designate the file to be used for the out file. This enables you to sort a file and save it to the same name.

cut

The `cut` utility selects characters from the input and sends them to the standard output. It can be used to select tab delimited fields, too. Each character or field is referred to by the number of its position.

If the output of the `ls -l` command is

```
drwxrwxr-x   2 root     root         1024 Sep  9 07:41 graphics
-rw-r--r--   1 root     root        58575 May 20 13:40 install.txt
drwx------   2 hadden   hadden       1024 May  1 04:36 mail
-rw-------   1 hadden   hadden       1633 May 16 09:07 mbox
-rw-rw-r--   1 hadden   hadden         40 May 20 22:22 mywords
drwx------   2 hadden   hadden       1024 May  8 15:18 nsmail
-rw-rw-r--   1 hadden   hadden          0 Sep  4 16:53 script
drwxr-xr-x   2 hadden   hadden       1024 May  7 20:55 temp
```

and if you wanted to see only characters 2[nd]10 (permissions), you could issue the following command:

```
ls -l | cut -c2-10
```

The `-c` option lists the characters to be displayed, and the `|` (pipe) tells `cut` to get its input from the `ls -l` command. The output would be

```
rwxrwxr-x
rw-r--r--
```

```
rwx------
rw-------
rw-rw-r--
rwx------
rw-rw-r--
rwxr-xr-x
```

You can use the `-d` option to define which character should be used as a delimiter when counting fields and the `-f` option to let `cut` know which fields to return. If you had a comma-delimited file and want to see all the names listed in the sixth field, you would type

```
cut -d, -f6 MyFile
```

`cut` does not change the original file, but instead outputs a section of that file based on character or field count. You can redirect the output to another file using the > pipe and specifying a filename.

expand

The `expand` utility is used to change tabs to a specified number of spaces. The syntax for `expand` is

```
expand [options] file
```

By default, each tab is considered to be eight spaces. Use the `-t` to designate another value or provide a list of integers to define where each tab is to be placed. You can use the `-i` option to convert only tabs at the beginning of the line.

fmt

The `fmt` utility is used to perform simple text editing. `fmt`'s main purpose it to try to make all the lines the same length by joining or splitting lines. Its syntax is

```
fmt [option] filename
```

`fmt` does not change the original file, but instead sends its results to the standard output. If you want to format a file so that each line is 70 characters long, issue the following command:

```
fmt -70 MyFile
```

This prints the file to the screen with each line approximately 70 characters long. If a line length is not specified, `fmt` defaults to 75 characters.

`fmt` attempts to split lines at the end of a sentence if possible. Also, it will not break lines after the first word or before the last word of a sentence.

You can prevent `fmt` from joining lines by using the `-s` option. If you want to have only one space between words and two spaces between sentences, use the `-u` option. If you want to format only specific lines, use the `-p <prefix>` option. This will cause `fmt` to format only those lines beginning with *`prefix`*.

Blank lines, indentions, and spacing are preserved. You can redirect the output to a file just like other commands.

head

The `head` command lets you look at the beginning of one or more files. Use the `-n` option to designate how many lines to display. If the number of lines to display is not specified, `head` displays the first 10 lines. Its syntax is

```
head [option] filename(s)
```

You also can use `c`, `b`, or `l` after the number to designate characters, blocks, or lines. If nothing is designated after the number, `l` is assumed. To see the first 25 characters of a file you would type

```
head -25c FileName
```

In addition, you can use the `-c` option to print the designated beginning bytes. This can be in 512-byte blocks (`b`), 1KB blocks (`k`), or 1MB blocks (`m`).

If you pass a list of files to `head` to display, it will precede the display with the name of the file. To prevent this behavior use the `-q` option. If you always want to see the name of the file, even when displaying only one file, use the `-v` option.

join

The `join` utility displays one line for each pair of input lines. The output is the `join` field followed by the remaining fields from `file1` and then the remaining fields from `file2`. Its syntax is

```
join [options] file1 file2
```

Each of the two files to be joined should first be sorted alphabetically unless the `-t` option is used. By default, the `join` field is the first field in each line.

Table 8.3 lists the options that are used with the `join` command.

Table 8.3 Options Used with the `join` Command

Option	Action
`-a`	Also prints a line for each unpairable line
`-e` *`string`*	Replaces missing fields with *`string`*
`-1, -j1 field` `-2, -j2 field`	Joins on field of `file2`
`-t` *`character`*	Uses *`character`* as field separator
`--help`	Prints help and exits
`--version`	Prints version and exits

nl

The `nl` utility adds line numbers to a file and displays it to standard output. Its syntax is

```
nl [options] filename
```

`nl` divides its input into logical pages, each of which consists of a header, body, and footer. Any one of these sections can be empty and each can be numbered differently. Line numbering is reset to 1 at the beginning of each logical page. You can override this behavior by using the `-p` option or using the `-v` option to designate what number to start with on each page.

If multiple files are sent to `nl`, it sends its output as a single file to standard output, unless you specify otherwise. Line numbering is not reset at the beginning of each file.

The input page must contain a delimiter on a separate line to indicate the beginning of each logical page. The customary delimiters are

`'\:\:\:'`	start of header
`'\:\:'`	start of body
`'\:'`	start of footer

You can change the delimiters by using the `-c` option. The line containing a section delimiter is displayed as a blank line.

od

The `od` utility is used to dump the contents of a file and is useful for viewing files with non-printing characters such as binaries. By default, its output is in 2-byte octal numbers. Its syntax is

```
od [options] file
```

The output from a file might look like the following:

```
0000000 066101 064160 043141 066151 066145 071551 005164 064506
0000020 062554 064554 072163 061412 072141 027064 074164 005164
0000040 060544 031171 062012 074541 005163 064544 072143 063012
0000060 066151 071545 066412 073571 064550 064143 071412 063157
0000100 062164 074541 000012
0000105
```

In this example, the first column is the offset of each line and then each two bytes are represented by a three-digit, octal number.

paste

The `paste` utility joins the corresponding lines from the named files and sends the output to the display. The output separates the joined lines with a tab. Its syntax is

```
paste [options] filename
```

Use the `-d` option to have the joined lines separated by another character, rather than separating them with a tab.

You also can use the `-s` option. This causes `paste` to output one file followed by a newline character and then output the next file. With this option, the lines are not joined together but the files are pasted one after the other.

The `paste` utility often is used in conjunction with `cut` to rearrange columns. First, you would use `cut` to get each column in a separate file, and then use `paste` to join them in the order you want.

pr

The `pr` utility is used to break files into pages. Its syntax is

```
pr [options] file
```

The `pr` utility is most frequently used to send a file to a printer. The standard page length is 66 lines but can be changed with the `-l` option. You can use the `-f` option to generate a form feed at the end of each page, instead of filling the page with newline characters.

You also can have the output formatted in columns by using the *-columns* option to specify the number of columns to use. Use the -m option to print the listed files in parallel—each in a separate column.

The file name is printed as the header unless you use the -h *headertext* option. This will cause the *headertext* to be printed centered at the top of the page.

To prevent pr from changing the layout of your file, use the -t option. This will suppress printing of headers and footers. The -T option does the same as -t, but it also removes any form feeds contained in the file.

split

The split utility is used to divide files into equally sized pieces. Its syntax is

```
split [options] [input_file] [output_file]
```

The default size of each output file is 1,000 lines. You can change this by using the -l option to define how many lines to put in each output file.

output_file gives the syntax for prefixing each piece of the original file. If none is given then split uses x. This is added to the filenames, which are aa, ab, ac, and so on, as in

```
split FileName dog
```

If FileName is 3,780 lines long, split would divide the file into four files, which would be called aadog, abdog, acdog, and addog. If no output file is given, the files would be aax, abx, acx, and adx.

Files also can be divided by the number of bytes in each output file by using the option -b. split multiplies the number by 512KB. You can us -k, which multiplies by 1,024 or use -m, which multiples by 1,048,576. For example

```
split -b 2 FileName
```

would divide FileName into 1024-byte blocks.

tac

tac displays a file line-by-line, starting with the last line and ending with the first one. Think of tac as being a reverse cat—just like its spelling. Its syntax is

```
tac filename
```

If multiple files are passed to tac, each one is separated by a newline. You can specify the separator to use by using the -s option. By default, the separator appears after the file. Use the -b option to place the separator before the file.

tail

`tail` displays the last lines of a file and is especially helpful in reading long log files in which the newest messages are at the end. Its syntax is

```
tail [options] file
```

You can designate how many lines to display by using the `-n` option. If the number of lines is not specified, `tail` defaults to 10. `tail` also enables you to designate where to start displaying the file by using the `+n` option, which starts displaying a file at line number *n* and then displays to the end of the file.

You can tell `tail` which item to count instead of which lines. Use `b` for blocks and `c` for characters. To show the last 30 characters of a file, you would type

```
tail -30c filename
```

You can display by blocks, too. Use `b` for 512-byte blocks, `k` for 1KB blocks, and `m` for 1MB blocks.

The `-f` option keeps a file open and displays each new line that is added. This is helpful when you are troubleshooting and want to watch the error logs. Press Ctrl-C to end the display.

When you pass multiple files to `tail`, it heads each display with the filename. To suppress this, use the `-q` option.

tr

The `tr` utility can be used to replace one string with another or to delete a string in a file. `tr` send its output to standard display. Its syntax is

```
tr [options] string1 [string2]
```

The most common use of `tr` is to perform a search and replace operation in which you specify the string to search for and the string to use as a replacement—for example, changing lowercase to uppercase. To change the word `monday` to `Monday` in the file `FileName`, type the following:

```
cat FileName | tr monday Monday
```

If you specify only one string and use the `-d` option, `tr` deletes the specified string. You can use the `-s` option with only one string and `tr` will replace multiple occurrences with only one. For example, if you type

```
cat MyFile | tr -s l
```

tr changes the word Halloween to Haloween when displaying MyFile to your screen. tr also can be used to encrypt your file using the rotate 13 method. This replaces the first letter of the alphabet with the 13th, the second with the 14th, and so on. An example is

```
tr '[A-M][N-Z][a-m][n-z]' '[N-Z][A-M][n-z][a-m]
```

As you can see, you can use ranges of characters enclosed in square brackets as part of your string. Notice that single quotes are needed because of the special characters.

wc

The wc utility will tell you the number of words, lines, or characters in a file. Its syntax is

```
wc [options] [filename(s)]
```

If you specify multiple files, the number of words, lines, and characters will be displayed for each file followed by the sum total of words, lines, and characters.

You can use the -c option to display only the number of characters -w to display the number of words, and -l to display the number of lines.

So, if you called wc with days and mywhich listed as filenames, the output would be

```
      6       8      68 days
      5      17     101 mywhich
     11      25     169 total
```

To display the number of lines in days without showing the file's name, you could type

```
wc -l < days
```

Input and Output

Almost every command can get information passed to it by way of standard input (stdin). This usually is via the terminal, such as when you type a command. However, input also can come from a file.

In addition, each command has two types of output: standard output (stdout) and standard error (stderr). stdout is the results of the command, whereas stderr is where any error messages are sent. By default, both of these are directed to the display, but either or both of these streams can be redirected to a file.

The shell manages how standard input and standard output are handled. By default, your shell directs standard output and standard error from the command to the device file, which is associated with your terminal and causes it to appear on your display. In the same manner, the shell directs standard input from your terminal, or keyboard, to the command.

Consider the command `cat`. If you type only `cat <enter>`, the standard input comes from the keyboard and the standard output goes to the display. In other words, what you type appears on your display.

After you press Ctrl-d, you signal to the `cat` command that the input is finished. `cat` then returns control to the shell and your prompt returns. If, however, you type `cat MyFile`, the input is the file `MyFile` and the output again goes to your display.

Redirection and Pipes

Redirection is the method of altering the source for standard input or where to send output. The `>` can be used to define where to send standard output and `<` can be used to redirect standard input.

What if you type `cat file1 > file2`? This is an example of using redirection. The input comes from `file1`, the output goes to `file2`, and nothing appears on your display.

This affects only standard output. If any errors had occurred, those messages would have been sent to your display and not to `file2`. To redirect both the `stdout` and `stderr`, use the greater than sign (`>`) followed by the ampersand (`&`), as in the following:

```
cat file1 file2 >& error-msg
```

This example will send both the standard output and any error messages to the file called `error-msg`. If you are using the bash shell, you also can use

```
cat file1 file2 &> error-msg
```

to do the same thing.

The shell assigns the number 0 to the standard input, the number 1 to the standard output, and the number 2 to the standard error output. You can use these numbers when using redirection. To save the error messages and discard the regular output, type

```
cat file1 file2 2> error-msg
```

If you wanted to redirect both standard output and standard error, you could type

```
cat file1 file2 1> MyFile 2> error-msg
```

When redirecting output to a file, any existing file with the same name will be overwritten. If you do not want to overwrite the existing file, use `>>` to append the new information to the existing file.

Sometimes you might not want to save the output of a command, whether it be standard output or standard error. This can be accomplished by redirecting the output to the `/dev/null` device. The output then disappears without a trace, as in the following:

```
cat file1 file2 >& /dev/null
```

Another use of `/dev/null` is to zero out an existing file. In other words, you could type

```
cat /dev/null > MyFile
```

This replaces the contents of `MyFile` with a null string but keeps the file's permissions intact.

Another useful device is the pipe (`|`). The shell uses the pipe to send the standard output of one command as standard input to another command. Using a pipe enables you to be more efficient by eliminating intermediate steps.

Say you want to create an alphabetized listing (that is not case sensitive) of all the files in your directory. You could first create a file containing a list of all your files by typing

```
ls -A * > filelist
```

Then, use the `sort` command to alphabetize it:

```
sort -i filelist > AlphaFilelist
```

And finally, delete the file `filelist`. A quicker, more efficient method would be to use a pipe, thus eliminating the need for a temporary file altogether.

```
ls -A * | sort -i > AlphaFilelist
```

Here you perform a directory listing of all files except `.` and `..` and send that output as input to the `sort` command. The `sort` command ignores case and alphabetizes the list. Finally, the `sort` command sends its output to the file called `AlphaFilelist`.

What if you wanted to print the file list instead of saving it to a file?. You could use a filter instead of the pipe used previously. A filter takes the output from one command and sends it to another command. The command

```
ls -A * | sort -i | lpr
```

uses `sort` as a filter by processing the output of the first command, `ls`, and then sending its output as input to the second command, `lpr`.

Key Concept

`stdin`, `stdout`, and `stderr` can be redirected by using the < and > pipes. The pipe (`|`) is used to define a filter by taking the `stdout` from one command and sending it as `stdin` to another.

tee Utility

Another way to redirect output is by using the `tee` utility. `tee` simultaneously sends the output of a command to a file and to standard output, which can be either the display or a pipe to another command. The syntax for `tee` is

```
tee [options] files
```

So, if you wanted to view the output as well as save it to a file you could type

```
ls -A * | sort -i | tee AlphaFileList
```

This command would display the sorted file list as well as save it to `AlphaFileList`. If you want multiple copies of the file, `tee` will accept multiple filenames as arguments. You also can append the output to an existing file instead of replacing that file by using the `-a` option with `tee`.

xargs

The `xargs` command passes arguments from one command to another command. It enables the receiving command to handle more arguments than it normally could process. Most frequently, the arguments are numerous, such as a list of filenames generated by `ls`.

You can cause `xargs` to run interactively by using the `-p` option. This causes `xargs` to ask for confirmation before executing each command.

Pipes are very powerful, and you will see them used throughout the exam. They can be either simple or complex as shown in the following example:

```
find *.doc | xargs cat | tr ' ' '\n' | sort | uniq | tee dict | less
```

This command finds all the files in your directory that end with `.doc` and pipes those names to `xargs`. `xargs` then passes the filenames to `cat`, whose output is sent to `tr`. `tr` then changes each space to a carriage return. Next, the output is sorted (`sort`) and duplicates discarded (`uniq`). `tee` saves the stream to a file called `dict`, which is displayed using the `less` pager.

Summary

This chapter has covered an important group of tools referred to as text filters. Each of these is important to you when studying for the exam. `sed` is especially useful when writing scripts, so know it well. `cut` and `paste` are useful for rearranging text files by columns. Both `head` and `tail` are useful in obtaining a quick look at files.

For formatting text, be familiar with the `pr`, `fmt`, `nl`, and `expand` commands. You also can be sure you will see a question or two involving the `sort` command.

The use of redirection and pipes enables you to build complex commands to increase your efficiency. Be sure you understand the use of the `tee` and `xargs` commands.

QUESTIONS AND ANSWERS

1. How do you prevent error messages from showing on your screen when running a command?

A. You can pipe the standard error to a file by using the `>2` redirector. If you do not want to keep them, you also can send the error messages into the netherworld by piping standard error to `/dev/null`.

2. What command could you use to read a file called `MyFile`, replace each instance of `while` with `during`, and save the changes to a new file?

A.
```
cat MyFile | tr while during > NewFile
```

This command uses `cat` to read the file and then pipes it to `tr`. `tr` then changes `while` to `during` and the output from `tr` (the changed file) is saved to a new file called `NewFile`.

3. How can you sort your list of telephone numbers?

A. By using the `sort` command with the `-n` option to sort numerically, such as in

```
sort -n PhoneList
```

4. You have a file with each telephone number and name on a single line. The telephone number and name are separated by a comma. You want to create a list of all your contacts, but want the name first followed by the phone number. And you want the list to appear in alphabetical order. How could you do this?

A. First, use the `cut` command to divide the file into two sections, one containing the phone numbers and one containing the names. Then, use the `paste` command to put them together with the names first. Then, use the `sort` command to alphabetize them.

5. You need to print a file but want to have only 50 lines printed per page. How would you do it?

A. You can use the `pr` command to specify the number of lines and send it to the printer, as in

```
pr -l 50 MyFile | lpr
```

...continues

...continued

6. You want to monitor the system log as entries are made. What should you do?

A. Use the `tail` command to display the last few messages with the `-f` option to keep it open. When you are finished, use `Ctrl-C` to close the file:

```
tail -f MyLog
```

7. You are writing a report on last month's sales, but it must not be longer than 500 words. How can you check the length of your document?

A. Use the `wc` command with the `-w` option to display the total number of words in your document, as in the following:

```
wc -w Sales
```

8. What does the `tee` utility do?

A. The `tee` utility takes input and sends it to two different places. The input is usually the output from another command. The output goes to a file and to either the standard output device or another command.

PRACTICE TEST

1. Which of the following commands would change the word `yep` to `yes` in the file `MyFile` and display only the lines that were changed?

- **a.** `sed s/yep/yes/ MyFile`
- **b.** `tr yep yes MyFile`
- **c.** `sed -n s/yep/yes/gp MyFile`
- **d.** `cat MyFile | tr yep yes`

Answer a is incorrect; this command would replace the first instance of `yep` with `yes` and output each line of the file once for reading it and once for each change that was made. Answer b is incorrect; this command would result in an error message. **Answer c is correct; this command would replace every instance of `yep` with `yes` and display only the lines in which changes were made.** Answer d is incorrect; the `tr` command would display the entire file after making changes.

2. You used tabs when preparing your report on last month's sales figures. How can you change all the tabs to three spaces?

   a. `fmt -3 sales`
   b. `expand -t3 sales`
   c. `tr ' ' ' ' sales`
   d. `sort -t sales`

Answer a is incorrect; this is an invalid command. **Answer b is correct; the `expand` command is used to change tabs into spaces.** Answer c is incorrect; this is incorrect syntax for the `tr` command. Answer d is incorrect; `sort` does not convert tabs to spaces.

3. When printing your sales report, you want to format it so that the lines are all 70 characters in length. What command should you use?

   a. `fmt -70 sales`
   b. `head -70 sales`
   c. `pr -l 70 sales`
   d. `od sales`

Answer a is correct; `fmt` will format the `sales` file to 70 characters per line. Answer b is incorrect; this would display the first 70 lines of the `sales` file. Answer c is incorrect; this command would format the `sales` file with 70 lines per page. Answer d is incorrect; the `od` command would display the `sales` file in octal format.

4. You are sending the `sales` file to a co-worker to proofread. Before sending it, however, you want to save it to a file called `newsales` with each line numbered. What should you do?

   a. `cat sales > newsales`
   b. `wc sales newsales`
   c. `nl sales > newsales`
   d. `fmt -n sales newsales`

Answer a is incorrect; this would copy the `sales` file to `newsales`. Answer b is incorrect; this would count the characters, words, and lines contained in `sales` and `newsales`. **Answer c is correct; the `nl` command numbers the lines in a file and then the output is redirected to a new file called `newsales`.** Answer d is incorrect; the `fmt` command evens out the line length, and this syntax is incorrect.

5. You have three files called `letter1`, `letter2`, and `letter3`, which are all letters. How can you quickly determine which one is addressed to your boss?
   a. `tail letter?`
   b. `nl letter?`
   c. `expand letter?`
   d. `head -5 letter?`

Answer a is incorrect; this would display the last 10 lines of each letter. Answer b is incorrect; the `nl` command numbers each line. Answer c is incorrect; `expand` converts tabs to spaces. **Answer d is correct; this would display the first five lines of each file so that you could look for your boss's name.**

6. The file containing your telephone list is approximately 2MB in size. How could you divide it into two equally sized pieces?
   a. `split -k 1 phone`
   b. `pr -2 phone`
   c. `od phone`
   d. `split -k 2 phone`

Answer a is correct; the `split` command would divide the file `phone` into two sections of approximately 1MB each. Answer b is incorrect; the `pr` command formats the file into pages. Answer c is incorrect; the `od` command would display the file in octal format. Answer d is incorrect; here `split` is directed to divide the file into 2MB pieces.

7. What does the `paste` command do by default?
   a. Combines two or more files sequentially.
   b. Combines two or more files side by side.
   c. Combines two or more files by interleaving the lines.
   d. Combines two or more files and sends them to the printer.

Answer a is incorrect; the `paste` command can combine files sequentially, but it requires the `-d` option. **Answer b is correct; the `paste` command combines files on a line-by-line basis with the lines being separated by a tab and displayed side-by-side.** Answer c is incorrect; `paste` does not mix the files. Answer d is incorrect; `paste` does not print the file.

8. You need to change the word `losses` to `profits` in your sales report and print it out. What would be the quickest way to do this?

- **a.** Edit the file in your favorite text editor and print it.
- **b.** `cat sales | sed s/losses/profits/gp | lpr`
- **c.** `cat sales | tr 'losses' 'profits' | lpr`
- **d.** `pr sales | lpr`

Answer a is incorrect; although this would work, it is not the quickest way to accomplish the task. Answer b is incorrect; this would print only the lines in which changes were made. **Answer c is correct; the `tr` command will perform the search and replace and then send the output to the printer.** Answer d is incorrect; this would paginate the file and send it to the printer without making any changes.

9. You want to save your sales report after changing `losses` to `profits`, as well as print it. How could you do this in an efficient manner?

- **a.** Edit the file using a text editor and then print it.
- **b.** `cat sales | tr 'losses' 'profits' | tee sales2 | lpr`
- **c.** `cat sales | tee sales2 | lpr`
- **d.** `cat sales | split sales2 lpr`

Answer a is incorrect; although it will work it is not the most efficient way. **Answer b is correct; the `tr` command will make the changes and then send the output to the `tee` command, which will save the changed file as `sales2` and send it to the printer.** Answer c is incorrect; this would save the uncorrected file as `sales1` and print it. Answer d is incorrect; this command is incorrect syntax.

10. What is the function of the `xargs` command in the following?

```
ls *.doc | xargs cat | tr '[A-Z]' '[a-z]' | lpr
```

- **a.** It is an intermediary between the output of the `ls` command and the `cat` command.
- **b.** It is necessary in order for the first pipe to function correctly.
- **c.** Without using `xargs`, the output sent to the printer would be garbled.
- **d.** You could write the command with or without `xargs`.

Answer a is correct; `xargs` enables `cat` to handle a larger input. Answer b is incorrect; pipes can be used without using `xargs`. Answer c is incorrect; the `xargs` command does not affect the output that is sent to the printer. Answer d is incorrect; if you didn't use the `xargs` command, the command line would fail.

CHAPTER PREREQUISITE

Working with files and directories and typing at the command line; Chapter 4, "File Management," and Chapter 5, "GNU and UNIX Commands." The Linux filesystem and how it is organized, Chapter 3, "Linux Filesystem."

CHAPTER

9

Permissions

WHILE YOU READ

1. How can a file owner change ownership of files?
2. You have a directory of files containing payroll information that you want to share with other employees in the payroll department. As the system administrator, what would be the best way to do this?
3. You have a new employee, Mary, who replaced Bill. How can you give Bill's files and directories in the `/data/bill` directory to Mary?
4. One of the accounting department's directories contain information that should be changed only by the head accountant, but should be readable by all accounting employees. How do you set the permissions?
5. How do you set the permission block on the employee evaluation files so that only the HR director can read or change them?
6. All the salespeople save their quotes to a directory called `/sales/quotes`. Some of them have complained to you that files are being lost. How could you address this problem?
7. You have a user who tells you that every time he creates a new file it has its permissions set as `-rw-rw-r--`. He wants the new permissions to be `-rw-r--r--`. How can you configure his settings to make this happen?

Lab

SEE APPENDIX F

Owners and Groups

The management of permissions is a complex subject that is a large part of any system administrator's job. You can count on seeing questions that address access issues both from a straightforward approach as to how to set permissions and from a troubleshooting approach. The question might not specifically state it is about permissions, but when you analyze the problem, you will see that setting appropriate permissions is the answer.

Every file has both an owner and a group whether it is a text file, directory, or device. To see the owner and group associated with a specific file, use the `ls -l` command, which produces an output like the following:

```
drw-rw-r--   1 hadden   hadden      17249 Sep  7 10:08 dict
-rw-rw-r--   1 hadden   hadden         69 Sep  7 10:25 files
```

Usually, the owner is the individual who created the file and the group is whatever group was the owner's default group when the file was created. Both the group and owner can be changed.

Key Concept

Every file, whether it is a directory, device, binary, or text file, has an owner and a group associated with it. A *user* is anyone who is not the owner or a group member.

Changing the Owner

To change the owner of a file, use the `chown` command along with the name of the new owner. If you know the user's ID, you can use that instead of the name. Only root can change the ownership of a file. The syntax for `chown` is

```
chown [options] newowner file(s)
```

So, to change the ownership of all the files that begin with `memo` to martinez, while logged on as root, you would type

```
chown martinez memo*
```

Changing the owner does not change the owner's permissions. Instead, it changes only who can exercise those permissions defined for the owner.

When you run the `chown` command, it first checks to verify that you have provided a valid username by searching the `/etc/passwd` file. If this check succeeds, the ownership of the file is changed. Table 9.1 shows the options that can be used with the `chown` command.

Table 9.1 Options That Can Be Used with the `chown` Command

Option	Action
`-c`	Displays information on all changed files
`-v`	Displays information on all files whether or not the ownership change was successful
`-f`	Does not display error messages
`-R`	Applies changes recursively down the directory tree

Changing the Group

Groups are merely a way to organize users. Because every file has a group associated with it and permissions can be applied to groups, groups are an excellent tool for managing access to files and directories. Groups and group management are covered in Chapter 7, "Users and Groups."

You can change the group for a file using either the `chown` or `chgrp` command. You must be the owner or root to change a file's group.

As the owner of a file, if you want to change a file from one group to another, you must be a member of the new group. Root, however, can change the group to any group, regardless of whether root is a member of that group.

Performing an `ls -l` on the file `MyFile` shows you the current owner and group, as in

```
-rw-rw-r--   1 hadden     user           370 Sep  5 14:42 MyFile
```

To use the `chown` command to change the owner and group, you must separate the owner and group names by a colon, as in the following:

```
chown martinez:wp MyFile
```

This will change the owner of the file `MyFile` to martinez, as well as change its group from user to wp. Now, when you perform an `ls -l` on `MyFile` you see

```
-rw-rw-r--   1 martinez   wp             370 Sep  5 14:42 MyFile
```

You could have changed only the group on `MyFile` by issuing the following command:

```
chown :wp MyFile
```

This would have changed the group to wp without changing the owner of the file. An `ls-l` command would now show

```
-rw-rw-r--   1 hadden     wp             370 Sep  5 14:42 MyFile
```

Notice that the group's name must be prefaced with a colon (:) when using the `chown` command to change only the group.

The other command you can use to change the group is the `chgrp` command. The syntax of the `chgrp` command is

```
chgrp newgroup file
```

To use the `chgrp` command to change the group ownership on the files beginning with `memo`, you would type

```
chgrp wp memo*
```

Key Concept

Remember, only the owner of a file or root can change the file's group. If the owner attempts to change the group, she must also be a member of the new group.

When changing a group, the `chgrp` command verifies that you have entered a valid group by searching the `/etc/group` file. `chgrp` has the same options as `chown`. Refer to Table 9.1.

You can use the `chown` and `chgrp` commands to change the group on multiple files by using wildcards:

```
chown :wp memo*
```

However, it will fail on any files that you do not own unless this command is executed by root.

Accessing Files and Directories

Permissions to files are managed using the permission block, which is displayed when displaying a long listing by using the `ls -l` command. You will need to be able to interpret the permission block to answer questions pertaining to permissions. The permission block is the first 10 characters that are displayed:

```
-rw-rw-r--   1 hadden   hadden         39 Oct 19 18:30 ABill
-rw-rw-r--   1 hadden   hadden         35 Sep  6 19:47 AlphaFilelist
-rw-rw-r--   1 hadden   hadden         69 Sep  7 10:25 Filelist
-rw-rw-r--   1 hadden   hadden         39 Oct 19 18:28 phone
-rw-rw-r--   1 hadden   hadden      10240 Sep  6 12:46 catalog
-rw-rw-r--   1 hadden   hadden      17249 Sep  7 10:08 dict
-rw-rw-r--   1 hadden   hadden         69 Sep  7 10:25 files
-rw-rw-r--   1 hadden   hadden        169 Oct 19 20:58 filex
```

```
drwxrwxr-x   2 hadden   hadden       1024 Oct 20 10:55 letters
-rw-rw-r--   1 hadden   hadden         69 Oct 19 18:28 myfile
-rwxrwxr-x   1 hadden   hadden        101 Sep  6 12:46 mywhich
lrwxrwxrwx   1 hadden   hadden          4 Sep  6 16:44 softday -> days
-rw-rw-r--   1 hadden   hadden       4354 Sep  7 13:48 x
```

The first character represents the file type. As discussed in Chapter 3, there are several different types (see Table 9.2).

Table 9.2 Linux File Types

Character	*File Type*
`-`	Ordinary file
`b`	Block device
`c`	Character device
`d`	Directory
`l`	Link

Permissions are interpreted differently depending on the file type, which is why this character is important when interpreting what permissions will be applied.

The next nine characters of the permission block define the actual permissions being applied. The characters are interpreted in three groups of three. Each group consists of read, write, and execute permissions, which are either granted or denied. The three groups are the permissions for owner, group, and all others or users.

Three basic permissions can be assigned to any file or directory, each of which is represented by a single letter:

- `r` (read)
- `w` (write)
- `x` (execute)

Access to files of all types is determined by assigning one or more of these permissions to that file.

Nine permissions are listed for each file immediately following the file's type. Figure 9.1 shows the permission block and its associated user(s).

The first block represents the three permissions of the owner of that file. The second block of permissions contains the permissions of the group that owns the file and anyone who is a member of that group. The third set of permissions refers to anyone who is not the owner or a member of the specified group.

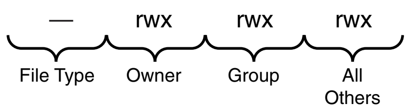


Figure 9.1
The permission block and its associated user(s).

The people who are referred to by the last set of permissions (All Others) are called user, world, or other. Figure 9.2 shows how access permissions are determined.

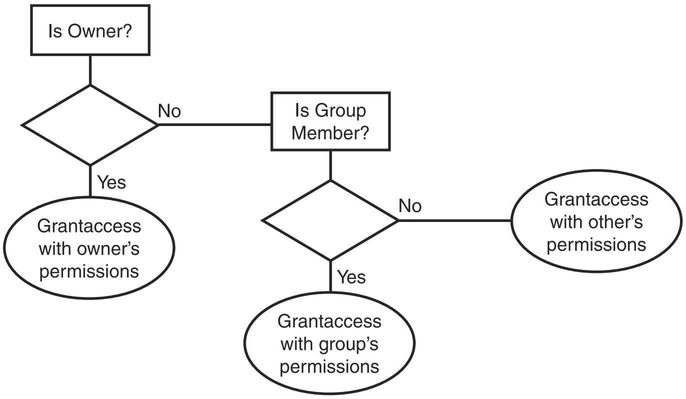


Figure 9.2
The operating system evaluates who is requesting access to determine what access permissions will be granted.

It is important to remember that by restricting the access permissions of the user, you are not limiting access to anyone with a valid account but instead to anyone who manages to get into your system. This means that if someone should hack into your system, her access to any file is defined by the third set of permissions contained in the permission block.

Your system does not evaluate the permissions granted to others by looking at the valid accounts. Instead, the individual's permissions are evaluated relative to ownership and group membership. If this evaluation fails, the individual requesting access is considered a user and is granted those permissions. So, if a file has a permission block of

```
rw-rw-r--
```

then both the owner of the file and the members of the specified group have the permission to read and write to the file. Everyone else has only the permission to read the file.

File Access

The r (read) permission means you can see the contents of that file, including opening the file with an application such as vi if it is a text file.

If you do not have the w (write) permission to a file when opening it in vi, the file will be opened as read-only. If you make changes to it, you will have to save the file using a new name.

The write permission means that you can modify the file by changing its contents. However, the write permission does not give you the permission to delete that file.

If you have write permission to a file but do not also have read permission, you will not be able to open that file with a text editor such as vi. Instead, you will get an access denied error message.

The x (execute) permission means you can run that file by typing its name at a command prompt, providing it is a binary file or script. If you try to execute a text file, you will get an error message.

Your effective access permissions to a file are determined by summing the individual permissions that have been granted to you. These are represented by the three positions that are always in the order read, write, and execute. If you do not have a particular permission, it will be represented by a hyphen (-). If the permission has been granted, the appropriate letter will appear in its correct position, as in the following:

```
-rw-rw-r--   1 hadden   hadden         39 Oct 19 18:30 ABill
-rw-rw-r--   1 hadden   hadden         35 Sep  6 19:47 AlphaFilelist
```

The permission block r-- means that you can read that file but you cannot change or delete it. Nor will you be allowed to execute it if it is a binary or script. The hyphens in the second and third positions indicate that the write permission (second position) and the execute permission (third position) are not granted.

The permission block -w- gives you the permission to change the file, but you can't read the file. For example, if you had only the permission to write to a file called File1, you would not be able to open that file in an editor; however, you could change its attributes by using the touch command. You also could overwrite the file, as in

```
cat file2 > File1
```

The permission block `--x` gives you the permission to execute a file if it is an executable or script file but not to examine its contents or change it.

Access to a file is the result of summing all the granted permissions. So, the `r-x` permission block means that you can read that file and execute it but you cannot make changes to it.

Key Concept

When evaluating permissions applied to a file, you need the `r` (read) permission to examine the file's contents; the `w` (write) permission to change the file; and `x` (execute) permission to run the file if it is a script or binary file.

Directory Access

The permissions to access a directory are applied a little differently from the permissions to access a file. The read permission to a directory means that you can list its contents. However, this does not give you the ability to make this directory your working directory.

The write permission gives you the permission to create files and directories within that directory. It also means that you can delete files contained in that directory even if you do not have any permissions to that file.

```
[eddie@scotty permdir]$ ls -l fill
-rw-rw-r--   1 hadden   hadden          0 Oct 27 09:46 fill
[eddie@scotty permdir]$ rm fill
rm: remove `fil1', overriding mode 0664?
```

In this listing, the user `eddie` has only the read permission on the file. The directory permissions are `rwx` for others. Before deleting the file, the `rm` command prompts the user to confirm that he wants to override the read-only permission; however, the delete operation succeeds after he responds `yes`.

The execute permission (`--x`) gives you the permission to make that directory your working directory. So, you can use the `cd` command to change to that directory. However, if you don't also have the read permission, you will not be able to obtain a listing of the directory's contents.

Therefore, if you have `r-x` permissions to a directory, you can list the contents of that directory and make that directory your working directory.

When determining access to a directory, permissions are a little different. If the permission block on a directory is

```
rwxr-xr-x
```

then the owner of the directory can read the contents of this directory, create new files and directories in this directory, and make this directory his working directory.

The group members and everyone else can read the directory's contents and make the directory their working directory. However, these people cannot create new files or directories within this directory. Their ability to change an existing file are determined by the permissions defined on that file.

Key Concept

When evaluating permissions applied to a directory, you need the read permission to list the contents of the directory, the write permission to create or delete files in that directory, and the execute permission to make that directory your working directory.

Changing Permissions

To change permissions on a file or directory, use the `chmod` command. Only the owner or superuser (root) can change the permissions on a file or directory. Two ways to use this command are by symbols and by numbers. You will see questions that cover changing permissions using both methods, so be familiar with them.

Changing Permissions with Symbols

The syntax to use symbols is

```
chmod <who> <change> <right> [files]
```

The `who` refers to whether you want to change the permissions for the owner, the group, or everyone. These are represented by the letters `u` (owner), `g` (group), `o` (others), and `a` (all). More than one `who` can be specified. If the `who` is omitted, `a` (all) is assumed and the changes are applied to the owner, the group, and everyone else.

The `change` refers to whether you want to grant or deny a permission—use `+` to grant it or `-` to deny it. The permissions are `r` (read), `w` (write), and `x` (execute). You also must specify the file or directory that will get the new permissions.

Any permissions that already might have been granted will not be changed. If you are granting the group the write permission, as in

```
chmod g+w filename
```

and they already have the read permission, the group's permission block would be `rw-`, not just `-w-`.

You also can explicitly set permissions by using the `=` operator, which will replace any existing permissions. So, look again at the last example where the group has read permission but you want them to have write permission. If you type

```
chmod g=w filename
```

the new permission block would be `-w-`, and not `rw-`. When using the `=` change operator, all old permissions are gone and only those you specify will be granted.

The `who`, `change`, and permission arguments must not have any spaces between them. Also, multiple selections can be used in a single command. For example

```
chmod ug+x MyFile
```

would add the execute permission for the owner and the group. If the permission block on `MyFile` was `rwxrw-r--` before the change, after the change it would be `rwxrwxr--`.

However, if you typed

```
chmod ug=x MyFile
```

the new permissions would be `--x--xr--`. Notice that any `who` that is not specified is not changed. In this example, others had the read permission before and after the change because others were not mentioned when using the `chmod` command.

If you do not follow the `=` with a permission, all permissions for the `who` are removed. For example, if you have a file with a permission block of `-rwxrwxrwx` and enter

```
chmod go= filename
```

the new permission block would be `-rwx------`. Notice that the permissions for both group and others have been removed.

Because using the `=` operator revokes any previous permissions, it is important to be careful when using `=` to set permissions. On the other hand, setting explicit permissions gives you more power to ensure that permissions are set exactly like you want them to be.

For example, say you type the following:

```
chmod go-x MyFile
```

This removes the execute permission from both the group and others. If the group and others did not have the execute permission to begin with, no changes would be made. However, if the permission to execute `MyFile` had been previously granted to one or the other, it would be removed but the status of the read or write permission would be unchanged.

Key Concept

When using symbols to change permissions, only those specified permissions and those specified for the `who` are changed unless you use the equal sign (=).

You also can use wildcards when referencing files, such as

```
chmod go+rw letter*
```

This command grants read and write permissions to the owner and group to any file beginning with `letter`. If you want to grant full access to a file, type the following:

```
chmod +rwx memo*
```

This command gives read, write, and execute permissions to the owner, group, and others to any file beginning with `memo`. This will not work, however, if there are any files beginning with `memo` that you do not own. You will get an error message from `chmod`:

```
chmod: file3: Operation not permitted
```

This identifies the file in which the permissions were not changed.

A shortcut to granting permissions is to grant them based on another set of permissions. If you had a file with a permission block of `-rwxr-xr--` and you wanted anyone to be able to execute the file, you could issue the command

```
chmod o=g MyFile
```

which would set the others' permissions to the same as the group or `-rwxr-xr-x`. Remember, if you use this method you are setting the permissions explicitly. In other words, the existing permissions for others are erased and then set to mirror the permissions set for group.

You can use any of the three `who`s to set equal permissions by using `u` for owner, `g` for group, and `o` for all others.

If you want to change the permissions for all files and subdirectories, use `chmod` with the `-R` option. This will force `chmod` to recursive down through the directory structure and change all the permissions. Other options to use with the `chmod` command are listed in Table 9.3.

Table 9.3 Options to Use with `chmod`

Option	*Action*
`-c`	Displays the names of those files whose permissions are changed
`-f`	Does not display any error messages
`-v`	Displays a list of each file that is changed and any error messages

Changing Permissions with Numbers

You can change permissions by passing a three-digit octal number to the `chmod` command. This octal number represents the bit pattern for the new permissions. Each digit of this number represents the permissions for one of the entities—owner, group, and others, in that order. See Figure 9.3.

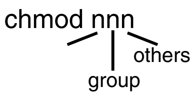


Figure 9.3
Setting permissions using an octal number with the `chmod` command.

The value of each permission is shown in Table 9.4.

Table 9.4 Numbers Used to Grant Permissions

Number	*Permission*
4	Read (r)
2	Write (w)
1	Execute (x)

You sum the permissions you want to grant and pass that number to `chmod`. If you enter the command

```
chmod 764 MyFile
```

you would be granting read, write, and execute to the owner, which is 4 (read) + 2 (write) + 1 (execute) = 7. The group is granted read and write, or 4 (read) + 2 (write) = 6. And all others would get read only, or 4.

When you use `chmod` with numbers you are setting explicit permissions. Any existing permissions on the file or directory are erased and the newly defined permissions are applied—so be careful.

Special Permissions

Now that you know how to change permissions, you need to know how to use them in some special circumstances. You should give careful consideration to your decision to use these special permissions.

These permissions can be set either by using symbols or by adding a fourth digit when using numbers. This digit goes before the three digits used to set permissions. You can expect to see questions covering these permissions just like setting the read, write, and execute permissions.

SUID and GUID

The first permission is set user ID (SUID) or set group ID (SGID). This permission changes the user's ID to that of the `who` parameter when setting the permission during file execution. This permission is represented as `s`, which appears instead of `x`.

If you have a file called `MyFile` with a permission block of `-rwxr-xr-x` and you issue the command

```
chmod o+s MyScript
```

then whenever anyone executes the file `MyScript` they will have the same privileges as if they were the owner running that file. Now, if you examined the permission block for `MyScript`, you would see `-rwsr-xr-x`. You could accomplish the same thing by using numbers with the `chmod` command:

```
chmod 4755
```

Table 9.5 shows the numbers that can be used before the three digits used to set permissions and their effects.

Table 9.5 Interpreting the First Digit When Using a Four-Digit Octal Number to Set Permissions

Digit	*Meaning*
4	Sets user ID when executing program
2	Sets group ID when executing program
1	Sticky bit

You also can have anyone running the script inherit the privileges of the group instead of the owner by issuing the command

```
chmod g+s MyScript
```

or

```
chmod 2755 MyScript
```

The permission block, after making this change, would be `-rwxr-sr-x`.

An example of using this permission is when a user changes her password. For a password change to become effective, the new password must be written to the `/etc/passwd` file. However, the permissions on the `/etc/passwd` file are

```
-rw-r--r--   1 root     root          876 Oct 20 11:32 passwd
```

Only the owner, root, has the permission to write to this file. Neither the group nor others have been granted the write permission to this file.

Users can change their passwords because the SUID bit has been set on the `/usr/bin/passwd` command as

```
-r-sr-xr-x   1 root     bin         15613 Apr 27  1998 /usr/bin/passwd
```

Notice the `s` where the bit is set for execute by owner. This means that when the file is executed, the user gets the same privileges as root—the capability to write to the `/etc/passwd` file. This is considered a significant security risk and should be used only if absolutely necessary.

Key Concept

Setting SUID and GUID enables a user to run a command or application with the permissions of another. Therefore, the use of these two special permissions can be a significant security risk.

Sticky Bit

The sticky bit enables you to manage shared directories more efficiently. The most common use of the sticky bit is on world writable directories where you want only the owner to be able to delete a file.

If the permission block on a directory called `MyDir` is set to `drwxrwxrwx`, anyone can create new files, list the contents of the directory, make the directory their working directory, or delete any files contained in that directory.

This type of permission block might be necessary when a directory is shared by many people. The drawback of this is that anyone can delete anyone else's files. This is prevented by setting the sticky bit on the directory itself.

Only root can set the sticky bit. To set it on `MyDir`, you would issue the command

```
chmod u+t MyDir
```

or

```
chmod 1777 MyDir
```

Both of these would change the permission block on `MyDir` to `drwxrwxrwt`.

You also can set a sticky bit on an executable. This would cause the program to stay in the swap area, which in turn results in faster program loading.

Default Permissions

During the exam, you can expect to see questions on default permissions for newly created files. You need to be familiar with the use of the `umask` command.

When you create a new file, the permissions that are applied to that new file are set by the system based on that individual's file creation mask. This creation mask is called `umask` (user's creation mask) and is defined either by the user's startup files or the system's default settings.

To see the value of your `umask`, type `umask` at a command line. The value is returned by the system:

```
$umask
002
```

The `umask` is represented as a three-digit octal number with each digit corresponding to one of the three-character blocks in the permissions block.

So, the first digit defines the default permissions for the owner, the second digit for the group, and the third digit for the world. Some systems return a four-digit number in which the first one is always a zero. The meaning of these numbers appear in Table 9.6.

Table 9.6 Permissions for `umask`

Value	*Permission*
0	read, write for files; read, write, execute for directory
1	read, write for files and directories
2	read for files; read, execute for directories
3	read for files and directories
4	write for files; write, execute for directories
5	write for files and directories
6	execute for files and directories
7	no permissions

Notice that the numbers used for setting permissions and the numbers used for defining a `umask` do not have the same meaning.

So, if the `umask` is `002`, the permissions on all the new files would be owner, with group having read and write permissions (00) and everyone else having read (2). If you performed an `ls -l` command to examine the permission block on the newly created file, it would be `rw-rw-r--`.

For new directories, the permissions would be owner and group read, write, and execute (00), with all others read and execute (2). The permission block on a newly created directory would be `rwxrwxr-x`.

Key Concept

You can see that the numbers used for setting permissions using the `chmod` command and for setting permissions using `umask` are not the same. A `umask` of `002` is the same as using `chmod 662` for a file or `chmod 775` for a directory.

If you want to change your `umask` value, issue the `umask` command followed by the permission level you want to define. So, if you want the permissions to be `rw-r-----` on files and `rwxr-x---` on directories, enter the following:

```
umask 027
```

This will temporarily change your mask. After you log off and log on again, the prior setting will return. To make this change permanent, add the previous command to your startup file. Only root can change the system `umask` value. See Chapter 7 for more information on using your startup files to manage your environment.

Summary

Permissions are an essential part of managing your system. They enable you to control who gets to access what. The possible permissions are read, write, and execute. For the exam, you need to understand what each permission allows the user to do whether applied to a file or a directory.

The permission block is the first 10 characters that are displayed when issuing the `ls -l` command. The first character designates the file type; the next three are the permissions for the owner; the next three after that are the permissions for the group; and the last three are the permissions for user. These permissions are then applied to one of three entities: owner, group, or user. You can change these permissions through the use of the `chmod` command. Permissions can be changed using either symbols or numbers.

To change the owner of a file or directory, use the `chown` command. Use either `chown` or `chgrp` to change the group associated with a file. The set UID (SUID) and set GID (SGID) are two special permissions that also can be applied. SUID allows an executable to be executed with another user's permissions instead of those of the user running the executable. The SGID works the same way, except it allows the permissions of another group rather than another user.

Another special permission is the sticky bit. This assists in the management of world-write directories. It also can be used to improve application execution. In addition, you need to understand what the `umask` is and how to correctly set it. Default permissions for all newly created files are defined by the system `umask` value. A user-specific `umask` also can be defined.

QUESTIONS AND ANSWERS

1. How can a file owner change ownership of files?

   A. A user cannot change the ownership of her files. Only root can do this. Using `chown` while logged in as root, you will need to change the ownership of the files.

2. You have a directory of files containing payroll information that you want to share with other employees in the payroll department. As the system administrator, what would be the best way to do this?

   A. Use the `chgrp` command and change the group for all the files in the directory to the payroll group. If there are subdirectories, you should use the `-R` option to recursively change the group on all the files. Then, set the permissions for the payroll group appropriately using the `chmod` command.

3. You have a new employee, Mary, who replaced Bill. How can you give Bill's files and directories in the `/data/bill` directory to Mary?

   A. You will need to change the ownership of the files and directories using the `chown` command with the `-R` option. Then, make sure that Mary is a member of the appropriate groups.

4. One of the accounting department's directories contains information that should be changed only by the head accountant but should be readable by all accounting employees. How do you set the permissions?

   A. You will want to make the head accountant the owner of the directory and its files. Put all the accounting employees in an accounting group. Set the permissions on the directory for the owner read, write, and execute; the group read and execute; and all others no permissions (`rwxr-x---`). Then, set the permissions on the files for the owner read and write; the group read; and all others no permissions (`rw-r---`).

5. How do you set the permission block on the employee evaluation files so that only the HR director can read or change them?

A. Put all the employee evaluation files in a single directory. Make the director of the Human Resources Department the owner of the directory and its files. Then, set the permissions so that the director has read, write, and execute on the directory and read and write on the files. Deny all access to everyone else, including the group.

6. All the salespeople save their quotes to a directory called `/sales/quotes`. Some of them have complained to you that files are being lost. How could you address this problem?

A. Set the permissions on the `/sales/quotes` directory so that the owner and the sales group have read, write, and execute permissions. The permissions on the files for the owner and sales group should be read and write. Deny any access to everyone else. This will keep the files accessible to only the sales group; however, anyone will be able to delete any file. You also should set the sticky bit so that only the owner of a file can delete it.

7. You have a user who tells you that every time he creates a new file it has its permissions set as `-rw-rw-r--`. He wants the new permissions to be `-rw-r--r--`. How can you configure his settings to make this happen?

A. The user's `umask` is `002`, which sets the permissions on all newly created files for owner and group to have read and write and all others to read. You will need to change his `umask` to 022, which will give the owner read and write, and group and others read-only permissions on newly created files. To make this change permanent, this setting will have to be added to the user's startup files.

PRACTICE TEST

1. The directory `/home/bill/letters` has a permission block of `drwxrw-r--`, the owner is Bill, and the group is wp. Mary is a member of the accounting and payroll groups. What would Mary be able to do in this directory?
   - **a.** Make the `/home/bill/letters` directory her working directory.
   - **b.** List the contents of the `/home/bill/letters` directory.
   - **c.** Save a file to the `/home/bill/letters` directory.
   - **d.** Delete files contained in the `/home/bill/letters` directory.

Answer a is incorrect; Mary would need the execute permission to make `/home/bill/letters` her working directory. **Answer b is correct; the read permission gives Mary permission to list the contents of the `/home/bill/letters` directory.** Answer c is incorrect; Mary would need the write permission to save a file to this directory. Answer d is incorrect; to delete files in the `/home/bill/letters` directory, Mary would need the write permission.

2. To give Mary permission to save a file to the `/home/bill/letters` directory, you would need to
   - **a.** Make Mary the owner of the directory.
   - **b.** Make Mary a member of the wp group.
   - **c.** Change the permissions on the directory to `drwxrw-rw-`.
   - **d.** Explicitly give Mary read and write permission to the `/home/bill/letters` directory.

Answer a is incorrect; although this would allow Mary to save a file to the `/home/bill/letters` directory, it would take away Bill's permissions as owner. **Answer b is correct; by making Mary a member of the wp group, she would be granted the permissions defined for the group—which are read and write, allowing her to save a file to the directory.** Answer c is incorrect; this would give Mary the needed permissions but would also grant these permissions to everyone else. Answer d is incorrect; you cannot explicitly give permissions to a single individual, only to the owner, the group, or others.

3. Bill has retired and has been replaced by Arthur. How can you transfer Bill's files to Arthur?
   - **a.** `chgrp - R arthur /home/bill`
   - **b.** `chmod o+rwx /home/bill`
   - **c.** `chown arthur /home/bill`
   - **d.** `chown -R arthur /home/bill`

Answer a is incorrect; this would change the group on all Bill's files and directories to a group called Arthur. Answer b is incorrect; this sets the permissions for the owner to read, write, and execute on all the files and directories in `/home/bill`; however, bill would still be the owner. Answer c is incorrect; this would change only the ownership of the `/home/bill` directory and its files. **Answer d is correct; this would change the ownership on the `/home/bill` directory, all its files, and all its subdirectories to Arthur.**

4. You have a directory containing technical documents and want to give the technical writers the permission to use this directory to create and change various manual pages for your products. How should you do this?
   - **a.** Put all the technical writers in a group and give this group read, write, and execute permissions for the directory.
   - **b.** Put all the technical writers in a group and give this group read and write permissions for the directory.
   - **c.** Put all the technical writers in a group and give this group read and write permissions for all the files in the directory.
   - **d.** Make the head technical writer the owner of the directory and its files. He can then grant access to the various writers.

Answer a is correct; this will give the technical writers access to the directory and its files. Answer b is incorrect; the writers would not be able to make the directory their working directory. Answer c is incorrect; although the writers could change existing files, they could not create new files unless they have the write permission on the directory. Answer d is incorrect; without having a group defined, the head technical writer could grant the permissions only to all users instead of just the technical writers.

5. If you issued the command `chgrp wp /home/letters` but forgot to first create the group wp, what would happen?
   - **a.** You would get an error message but the command would succeed.
   - **b.** No one would be able to access the directory until after you created the group.
   - **c.** Only the owner would be able to access the directory.
   - **d.** You would get an error message and the command would fail.

Answer a is incorrect; if you give an invalid group name, the command will fail. Answer b is incorrect; the group would not have been changed, so access would have remained the same. Answer c is incorrect; the owner and the group would have remained the same and the access permissions would be the same. **Answer d is correct; the `chgrp` command would first check the `/etc/group` file to verify that you specified a valid group, and because you didn't, the command would fail.**

6. The permission block on `MyDir` is `drw-r-xr--` and you issue the command `chmod o+x MyDir`. What would be another way to do this?
    - **a.** `chmod 777 MyDir`
    - **b.** `chmod 754 MyDir`
    - **c.** `chmod +x MyDir`
    - **d.** `chmod 661 MyDir`

Answer a is incorrect; this would change the permissions to `drwxrwxrwx`. **Answer b is correct; the `7` gives the owner read, write, and execute permissions; the `5` gives the group read and execute and others read; and the `4` gives all others read.** Answer c is incorrect; this grants the execute permission to all (owner, group, and others). Answer d is incorrect; this gives the owner and group read and write and others execute.

7. You have a file with a permissions block of `-rw-rw-r--` and issue the command `chmod go= file`. What would be the new permission block?
    - **a.** `-rw-rw-rw-`
    - **b.** `-rwxrw-r--`
    - **c.** `-rw-------`
    - **d.** `-rw-rw-r--`

Answer a is incorrect; to have this result, you would need to issue the command `chmod o+w file` or `chmod 666 file`. Answer b is incorrect; this permission block would result from either `chmod o+x file` or `chmod 764 file`. **Answer c is correct; the command uses the `=` symbol to explicitly remove all permissions from the group and others.** Answer d is incorrect; this shows no changes in permissions.

8. You inadvertently change the permission block on `/usr/bin/passwd` command to `-r-xr-xr-x`. What would be the effect of this change?
    - **a.** Users could no longer change their passwords because they do not have write permissions to `/etc/passwd`.
    - **b.** No changes in how the command runs would occur.
    - **c.** Users' passwords would not expire.
    - **d.** Shadow passwords would quit working.

Answer a is correct; by removing the SUID on the command, users would run the command in their own context and attempts to write to the `/etc/passwd` file would fail. Answer b is incorrect; users would not be able to change their passwords, although you might not realize this immediately. Answer c is incorrect; this would not affect any policy you might have established on password age. Answer d is incorrect; this would not disable shadow passwords.

9. You have a new script with a permission block of `-r-xr-xr-x` and issue the command `chmod 2555`. What would the new permission block be?

- **a.** `-r-sr-xr-x`
- **b.** `-r-xr-sr-x`
- **c.** `sr-xr-xr-x`
- **d.** `-r-xr-xr-x`

Answer a is incorrect; this permission block indicates that the SUID is set, but the command set the SGID. **Answer b is correct; the command sets the SGID so that anyone running the script runs it with the same permissions as the group.** Answer c is incorrect; the first character refers to the file's type. Answer d is incorrect; these permissions are the same as the ones you started with.

10. One of your users says that she changed her `umask` to 664, but now her new files all have a permission block of `---x--x-w-`. If she wanted her new files to have a permission block of `-rw-rw-r--`, what should she have used as a `umask`?

- **a.** `002`
- **b.** `022`
- **c.** `222`
- **d.** `220`

Answer a is correct; this would result in a permission block of `-rw-rw-r--` on all newly created files. Answer b is incorrect; this would result in a permission block of `-rw-r--r--`. Answer c is incorrect; this would result in a permission block of `-r--r--r--`. Answer d is incorrect; this would result in a permission block of `-r--r--rw-`.

CHAPTER PREREQUISITE

Before beginning this chapter, you should understand the sequence of events that occur when you boot Linux as covered in Chapter 2, "Booting Linux." You also need to understand how the filesystem is organized and where to locate specific types of files. Review the section on processes in Chapter 5, "GNU and UNIX Commands," to ensure that you understand how to stop and start a process.

CHAPTER

10

Administrative Tasks

WHILE YOU READ

1. You have an important meeting with your boss this afternoon and do not want to forget it. What should you do?
2. You need to re-index your database but want to do it when the system's load is low to prevent adversely affecting your users. What should you do?
3. You have a user who every morning schedules a job to run at 8 a.m., which slows down your system. What can you do to prevent him from scheduling a job using `at`?
4. Your database administrator needs to run a script every day for system maintenance. You do not allow users to run cron jobs. How could you make her job easier?
5. You want to save all messages from mail to the maillog file. What should you do to make this happen?
6. You made the changes as to where the messages from mail are to be saved. When you look at the maillog file, it is empty. What happened?
7. You want to find out when was the last time each of your users logged in. How can you find this information?
8. You want to start a new mail log once a month. What should you do?

Scheduling Jobs

A large part of a system administrator's job entails performing repetitive tasks. The ability to schedule these tasks becomes a great timesaver.

You might want to schedule an event to remind you of something or to run a program unattended. You might have events that occur only once and others that repeat at a regular interval. Linux has the capability to do both automatically.

at

When you have a job that you want to run only once at a specific time, you can use the `at` command. The syntax for the `at` command is

```
at [options] time
```

The options that can be used with the `at` command are listed in Table 10.1.

Table 10.1 Options That Can Be Used with the `at` Command

Option	*Action*
`-b`	Schedules job when system load is low; see the section on `batch` later in this chapter
`-d`	Deletes `at` job; same as `atrm`
`-f` *`filename`*	Reads job from specified file
`-l`	Lists all jobs for that user; same as `atq`
`-m`	Mails user when job completes

Using the `at` command is very simple and consists of three simple steps:

1. Type `at` with any options and the time of execution.
2. Enter the command you want to run.
3. Press Ctrl-D to save the job.

After `at` runs a job, it can send you an email with any error messages or to notify you that the job completed. `at` can run a job only once, and after a job runs, it is deleted from the queue. You can use the `at` command to set a quick reminder:

```
$ at 17:00
at> xmessage -display:0.0 "Time to go home!"
at> <EOT>
warning: commands will be executed using /bin/sh
job 11 at 1999-05-11 17:00
```

You also can use at to execute more than one command by entering multiple commands:

```
$ at 17:00
at> xsetroot -display :0.0 -solid blue
at> xmessage -display :0.0 "Time to go home!"
at> <EOT>
warning: commands will be executed using /bin/sh
job 12 at 1999-05-11 17:00
```

This will result in the commands being executed sequentially as though you had typed them on the command line separated by a semi-colon (;). If you want to have this job available to run at other times, save it to a file and then use the -f option to schedule it.

```
at 17:00 -f gohome
```

You can set the time for at to run in one of three ways:

- By time, in the format HH:MM
- The day of the week by number or name
- By using words such as midnight, noon, teatime (4 p.m.), today, tomorrow, and now

When specifying a time of day, at assumes you are using a 24-hour clock, so 4 p.m. should be entered as 16:00. You also can schedule the event to occur relative to the present time, such as the number of minutes, hours, days, weeks, or years from now. Using now as the time will also require a date or increment. The increment must be followed by minute(s), hour(s), day(s), or week(s). Some examples of ways to enter times are

```
at 23:15 -f mybackup
at +3 hours -f gohome
at now sat reboot
```

The jobs submitted to at are actually run by the crond daemon using atrun. Periodically, the crond daemon runs atrun to check whether any at jobs are to be run. This usually occurs every five minutes. See the section "crontab" later in this chapter for details on how to change this configuration.

To see what jobs have been scheduled, use the atq command, which is the same as at -l:

```
$ atq
12      1999-05-11 12:25 a
13      1999-05-11 12:35 a
```

This will list any pending jobs owned by the user running the atq command. The root user will see everyone's jobs or can specify to see only those jobs scheduled by a specific user. To remove one or more scheduled jobs, use either `at -d` or the `atrm` command and the job number, as in

```
atrm 12 13
```

To determine a jobs number, use `at -l` or `atq`.

Key Concept

Remember, the `at` command will schedule a job to occur only one time. Even though you can schedule the job for the future, after it runs, it is removed from the queue.

When you submit an `at` job, the shell saves your present environment including your `pwd`. This environment and directory will then be used when the job is executed. However, if you are scheduling a script to run, use its complete path to prevent possible errors from occurring.

Managing *at* Jobs

Running `at` jobs uses system resources. If many users run long, complex jobs at the same time, this can adversely affect your system's performance.

There are two files that control a user's ability to run `at` jobs—the `/etc/at.allow` and `/etc/at.deny` files. When a user attempts to run `at`, it first checks to see if the `/etc/at.allow` file exists and that that user's name is contained within it. If the user's name is there, he can use `at` to schedule a job.

If the `/etc/at.allow` file does not exist, `/etc/at.deny` is checked. To prevent a user from being able to schedule any jobs, add the user's name to the `/etc/at.deny` file. Any user who is not mentioned in the `/etc/at.deny` file is then allowed to use the `at` command.

If both files exist, `at` first checks the `/etc/at.allow` file. If the user who is requesting to schedule an `at` job is listed, he will be successful. `at` will not check the `/etc/at.deny` file if its check of the `/etc/at.allow` file is successful.

If you want to prevent all users from being able to run any `at` jobs, simply delete the `/etc/at.deny` file. If the `/etc/at.deny` file exists but is empty, all users can run `at` jobs. If neither file exists, only the superuser or root is allowed to use the `at` command. Figure 10.1 shows the process that `at` goes through when a user attempts to schedule `at` jobs.

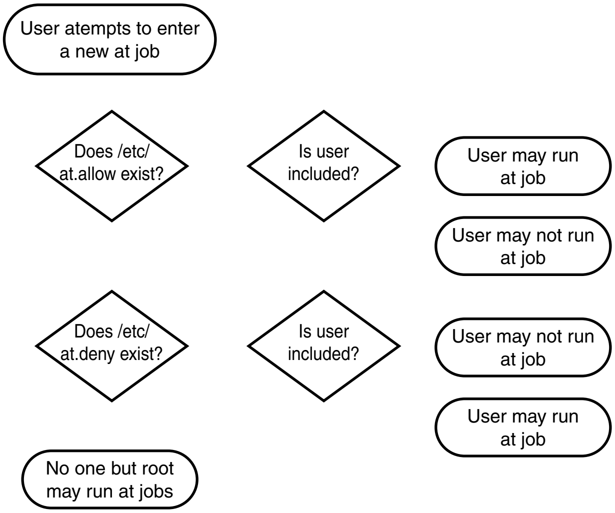


Figure 10.1
The files `/etc/at.allow` and `/etc/at.deny` are used to control each user's ability to schedule `at` jobs.

If the users on your system are allowed to schedule jobs using the `at` command, you will need to periodically track the load that is being put on your system. You can use the `atq` command to see who has scheduled jobs, when they will run, and what kind of load they might put on your system. You also can use the `atrm` command to delete users' jobs when indicated.

Key Concept

In order to evaluate the system resources being used by `at`, use `atq` or `at -l` to see what jobs are scheduled. Use `atrm` or `at 0d` to remove any scheduled job.

batch Command

The `batch` command is the same as the `at -b` command and schedules a job to run one time. Its syntax is

```
batch [options] [time]
```

You use the same options with `batch` as you use with `at` (see Table 10.1).

If a time is not specified with `batch`, the job will run as soon as the system load is low. If a time is specified, the job will run as soon as the system load falls after the specified time.

`batch` examines the `/proc/loadavg` to check the system load. It executes when the average load falls below 1.5.

Commands that are run using `batch` or `at -b` run with a lower priority than when run in the background. Unlike background tasks, which are killed when you log off the system, `batch` commands continue until the system is shut down. In addition, `batch` sends you a mail message when the job is completed or if any errors occur.

crontab

Often you will have a job that needs to be run on a regular basis. The `at` command allows you to schedule a reminder or job only once. To schedule recurring events, you can use the `crontab` command. Your scheduled events are then saved in a personal file.

The syntax for the `crontab` command is

```
crontab [options] [file] [username]
```

To use `crontab`, you first create a text file (with any name other than `crontab`) containing six fields, with a space between each field. The first five fields are `minute`, `hour`, `day of the month`, `month`, and `day of the week`; the sixth field is the command to be run. Table 10.2 shows the allowable values for these fields.

Table 10.2 Allowable Values for Fields When Creating a cron Job

Field	*Values*
`minute`	0–59
`hour`	0–23
`day of month`	0–31
`month`	0–12 (or names)
`day of week`	0–7 (0 or 7 is Sun, or use names)

You can use an asterisk (`*`) to indicate commands are to be run in every instance of that field. Also note that days of the week can be defined by a number—with 0 or 7 for Sunday—or by name. In addition, times are specified in a 24-hour format. If you wanted to create a file containing who is logged on to your system every day at 10 a.m., you would type

```
00 10 * * * who >> userlist
```

You can specify ranges instead of specific days and times by using a hyphen (-). To specify non-inclusive values, use commas (,) to separate each value. If you wanted to run the script called `myscript` every Tuesday and Thursday at 1 a.m., you would enter the following in your file:

```
00 1 * * 2,4 myscript
```

After creating the file, you would need to install it by using the `crontab` command. After a job has been scheduled, it is referred to as a *cron job*. The syntax for the `crontab` command is

```
crontab [options] [filename]
```

The options that can be used with the `crontab` command are listed in Table 10.3.

Table 10.3 Options to Be Used with the `crontab` Command

Option	*Action*
`-e`	Creates or edits user's `crontab` file
`-l`	Displays user's `crontab` file
`-r`	Deletes user's `crontab` file
`-u` *`username`*	Specifies which user's `crontab` file to act on; can be used only by root

The `crontab` command will create and install the cron file and save a copy of the original under the user's name in `/usr/lib/crontab`.

If you want to modify your cron configuration, edit this copy of the original file, and then install it using the `crontab` command as before. If you want to remove the file, use `crontab -r`.

Key Concept

Scheduling a job to run periodically entails first creating a file containing the commands and then scheduling it with the `crontab` command. Use `crontab -e` to edit the cron job or `crontab -r` to delete it.

The `crond` daemon checks several files to see whether there are jobs to be run. The first of these files is the `/etc/crontab` file, which looks like the following:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
```

```
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
```

These are directories containing jobs to be run on a regular basis. `run-parts` is a script that executes each script in the designated directory. Some of these are

```
# ls -l /etc/cron.daily
-rwxr-xr-x  1 root root  51 Jun 29  1998 logrotate
-rwxr-xr-x  1 root root 102 Oct 11  1998 tetex.cron
-rwxr-xr-x  1 root root  54 Sep 10  1998 tmpwatch
-rwxr-xr-x  1 root root 557 Jul 29  1998 updatedb.cron
```

Each of these files is a script that performs periodic routine tasks such as updating the `locatedb` file used by the `locate` command (refer to Chapter 6, "Maintaining the Filesystem") and rotating system logs (see the section "Rotating Logs" later in this chapter).

Another cron job that runs is the one that specifies when `atrun` is executed. This is necessary for `at` jobs to run. By default, `atrun` is executed every five minutes. You can change this to run more or less frequently or not at all.

Next, `crond` looks for schedules by username in the `/var/spool/cron` directory.

The `crond` daemon is responsible for running system and user cron jobs. It starts when Linux boots and wakes up every minute to see whether any jobs need to be run.

When constructing your cron jobs, you can include comments by starting a line with a pound sign (`#`). Comments can't be on the same line as a command. A sample comment might look like

```
# mail any output to 'hadden'
```

Users can edit only their own cron jobs. Root can edit any user's cron jobs by using the `-u` option.

Key Concept

The `crond` daemon checks every minute to see whether there are any jobs that need to be run. First it checks the `/etc/crontab` file and then it checks for any user scheduled jobs in the `/var/spool/cron` directory.

Managing *cron* Jobs

If you have many users running cron jobs, your system could get overloaded. Therefore, you need to decide if and when you will allow users to schedule jobs. You can control which users are allowed to run cron jobs through the use of the `/etc/cron.allow` and `/etc/cron.deny` files.

If the `/etc/cron.allow` file exists, only the users listed in this file are allowed to run cron jobs. If `cron` finds the user's name in the `/etc/cron.allow` file, it does not examine the `/etc/cron.deny` file.

If the `/etc/cron.allow` file does not exist but the `/etc/cron.deny` file does, any user not listed in the `/etc/cron.deny` file will be allowed to run cron jobs. If neither of these files exists, any user can run cron jobs. Figure 10.2 shows how the `crontab` uses the `/etc/cron.allow` and `/etc/cron.deny` files to determine whether a user can schedule cron jobs.

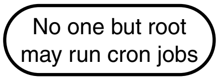


Figure 10.2
How `crontab` evaluates a user's ability to schedule a cron job.

Any user can view or edit her own cron jobs by using `crontab -e`. Users can delete their own jobs by using the `crontab -r` command.

As root, you can list any user's cron jobs by using the `crontab` command with the `-u` *`username`* option. You also can delete any user's cron jobs by using `crontab` with the `-r` option.

System Logs

System information is stored in a variety of logs. These logs might be specific to an application or messages from several sources might be written to a single log file. It is also possible to log messages based on their importance as well as their source.

`syslogd` is the daemon that is responsible for logging system activity. It is usually started as part of one of the `rc` files when the system boots. See Chapter 2 for more information on `rc` files and the boot process.

Upon startup, the `syslogd` daemon reads the `/etc/syslog.conf` file that specifies the options to be used when logging system information.

Another configuration file can be specified instead of the `/etc/syslog.conf` file by using the `-f` option. The options that can be used with `syslogd` are listed in Table 10.4.

Table 10.4 Options to Use with `syslogd`

Option	*Action*
`-f` *`filename`*	Specifies a configuration file other than `/etc/syslog.conf`
`-h`	Causes `syslogd` to forward messages received from remote hosts
`-l` *`hostnames`*	Causes *`hostname`* to be logged with simple hostname and not fully qualified domain name; separate multiple names with a colon (`:`)
`-m` *`interval`*	Time between two marked lines; default is 20 minutes
`-r`	Allows receipt of network messages

Key Concept

How and when system messages are written to a log file are controlled by the `syslogd` daemon. The configuration information usually is stored in the `/etc/syslog.conf` file, although another file containing configuration information can be used.

You can manage `syslogd` through the use of signals, too. A *signal* is a message that is sent to a process. Most often, this is used to tell a process to terminate. See Chapter 5 for additional information on processes and sending signals.

The use of signals enables you to stop and start `syslogd`, as well as update the `/etc/syslog.conf` file and have those changes processed by `syslogd`. The syntax for sending signals to `syslogd` is

```
kill -SIGNAL 'cat /var/run/syslogd.pid'
```

The file `/var/run/syslogd.pid` contains the PID of the `syslogd` process. By enclosing the `cat` command in back quotes (`), the actual contents of the file `syslogd.pid`—the PID of `syslogd`—is passed to the `kill` command. The signals that can be sent to `syslogd` are shown in Table 10.5.

Table 10.5 Signals That Can Be Sent to `syslogd`

Signal	*Action*
`SIGHUP`	Reinitializes `syslogd` by stopping the daemon, reading the `/etc/syslog.conf` file, and starting `syslogd`
`SIGTERM`, `SIGQUIT`, `SIGINT`	Stops `syslogd`
`SIGUSR1`	Toggles debugging if started with the `-d` switch
`SIGALRM`	Logs the mark line

After you kill `syslogd`, you can restart it using the options listed in Table 10.4. The best way is for you to specify a configuration file containing the instructions to be sent to `syslogd`.

Key Concept

You can manage the `syslogd` daemon through the use of signals. This enables you to make changes to the configuration file and then have them take effect.

Configuring *syslogd*

The `/etc/syslog.conf` file controls where information is written. Any line that begins with a `#` contains comments and is ignored by `syslogd`. Also, any blank lines are ignored. A typical `/etc/syslog.conf` file might look like the following:

```
# Log all kernel messages to the console.
kern.*                                                    /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none                           /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                                /var/log/secure

# Log all the mail messages in one place.
mail.*                                                    /var/log/maillog
```

```
# Everybody gets emergency messages, plus logs them on another
# machine.
*.emerg                                                     *

# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit                                              /var/log/spooler
```

Each line consists of three items:

- Message source
- Message type
- Log file location and name

Multiple sources can be named on one line by separating them with a comma.

When examining the previous file, notice the use of the asterisk (*). This specifies that all messages for the specified facility will be logged to the named destination. For example, in the sample `syslog.conf` file, all messages created by mail are logged to the same location.

The asterisk also can be used to specify all messages of a certain type, such as in the line starting with `*.emerg`. This line defines that any messages of type `emerg` are sent to all users (also indicated by an asterisk).

The equal sign (=) can be used to explicitly log the named class. For example, if you entered the line

```
*.=crit          /var/log/critical
```

in your `syslog.conf` file, all messages of the type `crit` from all sources would be written to the `/var/log/critical` log. Other messages of different types would be logged based on how you configured them.

Perhaps you want to send all kernel messages to a log file named `klog`. If you also had a line to send `crit` messages to another log, any kernel messages that were critical would be sent to this log instead of the `klog`.

The bang (!) is used to negate or exclude logging of the specified type. So, if you did not want to see messages of type `info` from mail, you could enter the following line:

```
mail.*;mail.!=info                                  /var/log/maillog
```

This line logs all messages from mail except those of type `info` to the `/var/log/maillog`. Using `mail.!*` or `mail.none` prevents logging any messages from mail.

The minus sign (-) before the name of the log will prevent the file from being synced every time it changes. Normally, when a message is written to a log it is written to RAM and then added to the file on the disk immediately. Using the - before the log's name results in the message being saved to RAM but not written to the disk until the next routine synchronization of RAM to disk.

Key Concept

When specifying the types of messages to log, the `*`, `!`, and `=` characters are extremely helpful. Use the `*` to specify either all message sources or all message types. The `!`, on the other hand, means *not*. Specifying `!*` before a message type would mean none of the following message types.

Managing Logs

Each message that is written to a log includes the date and source, as well as the message, on a single line. By default, you have the main system log, which is `/var/log/messages`. Depending on the services you have running and how you have configured `syslogd`, you might have many more.

Other system logs that you might have are `/var/log/wtmp`, `/var/log/utmp`, and `/var/log/lastlog`. Each of these contains information on users, which is used by various utilities. Table 10.6 shows how these logs are used.

Table 10.6 Log Files That Record User Logins and the Utilities That Use This Information

Log File	*Contents*
`/var/log/wtmp`	Contains login times and duration for each user; used by the `last` command
`/var/log/utmp`	Contains information on currently logged in users; used by the `who`, `w`, and `finger` commands
`/var/log/lastlog`	Contains login times for users; used by the `lastlog` command

The `last` command shows all logins listed in the `/var/log/wtmp` file. Its output can list users more than once. The `who` and `w` commands are used to determine who is presently logged in to the system. The `w` command also shows the time the user logged in. The `finger` command can be used to see the information contained in the comments field of the `/etc/passwd` file for logged on users.

The lastlog command is used to display the contents of /var/log/lastlog. If no options are used, lastlog displays the lastlog entries ordered by the UID. Using the -t option will cause lastlog to display all logins during the last specified number of days. You also can use the -u option to display the specified user's last login.

The lastlog output also will indicate whether a user has never logged in to the system. Sample output of the lastlog command looks like

```
Username         Port     From             Latest
root             tty1                       Fri Nov  5 19:21:04 1999
hadden           ttyp0    192.168.0.123     Fri Nov 12 07:49:45 1999
peanut           tty1                      Wed May 19 10:33:53 1999
eddie            tty3                       Sat Aug 21 19:11:05 1999
ronmart          tty2                      Sun May 16 16:10:07 1999
mary                                        **Never logged in**
```

Rotating Logs

Because logs can get very large, you will want to monitor them. Periodically you will want to clear the log. When you do this, you will need to decide whether you want to save the old information. This is referred to as *rotating the logs.*

One way to handle this is to periodically copy or move each log that you want to save to another location. Next, you would either need to recreate the log file (if you moved the original) or empty the original log (if you copied it).

This enables you to save older logs while also improving system performance. As the number of old logs increases, you will need to decide whether you want to archive them or just delete the older logs. This will depend on what the role of your computer is and your organization's policies. If it is a standalone computer, chances are you will not want to keep older logs.

To simplify this process, you can use the logrotate command. The logrotate command gives you the ability to automatically rotate your logs, compress or remove old logs, and even mail the log files. This activity can be based on time (such as daily) or on log size. The syntax for logrotate is

```
logrotate [options] config_file
```

By default, logrotate saves its status information to /var/lib/logrotate.status, but this behavior can be overridden by using the -s option and the name of another file.

More than one *config file* can be passed to the logrotate command; however, instructions contained in the last one named can override instructions in an earlier named file. Table 10.7 lists the commands that can be used in the config files.

Table 10.7 Commands Used in Config Files to Be Used with the `logrotate` Command

Command	*Action*
`compress`	Uses `gzip` to compress old logs
`copytruncate`	Copies log then truncates old log
`create`	Applies named permissions to new log; if no permissions are specified, permissions of old log are used
`daily`	Rotates logs daily
`delaycompress`	Compresses on next rotation
`errors` *`mailto`*	Mails errors to specified address
`ifempty`	Rotates empty log
`include` *`filename`*	Reads *`filename`* into config file
`mail` *`mailto`*	Mails logs to specified address when deleting
`monthly`	Rotates logs monthly
`nocompress`	Does not compress
`nocopytrucate`	Does not copy and truncate
`nocreate`	Does not use permissions specified by `create`
`nodelaycompress`	Compresses now
`noolddir`	Does not move to another directory
`notifempty`	Does not compress empty log
`olddir` *`directory`*	Moves old logs to specified directory
`postrotate`	Runs script after log is rotated
`prerotate`	Runs script before log is rotated
`rotate` *`n`*	Specifies number of old logs to retain
`size` *`n`*	Rotates when log reaches *`n`* bytes; adds *`k`* to specify kilobytes or *`M`* to specify megabytes

A configuration file can have both global and local options. *Global* options are applied to all log files, whereas *local* options are applied only to a specific file. Any local options will override global ones. The following is a sample configuration file:

```
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4
```

```
# mail errors to root
errors root

# create new (empty) log files after rotating old ones
create

# compress old log files
compress

# use configuration files contained in /etc/logrotate.d directory
include /etc/logrotate.d

# will only rotate wtmp monthly instead of weekly
/var/log/wtmp {
    monthly
}
```

In the preceding sample configuration file, the following global options were set:

- frequency of rotation
- number of old logs to keep
- compress old logs

The last section sets local options for the `wtmp` log only by specifying to rotate it monthly instead of weekly. Using local options enables you to configure specific logs with different parameters from the default ones you want to use for the majority of your logs.

Key Concept

The `logrotate` command automates the process of saving and removing the contents of your log files. One or more configuration files can be used to define how `logrotate` handles each log.

Archiving Logs

Now that you have configured `logrotate` to handle logs the way you want, you need to decide how long you will keep old logs. In some instances, you might want to keep some logs for a year or so.

In the earlier configuration file, `logrotate` was configured to keep logs for four weeks. After you have four weeks worth of logs, the oldest log is discarded and log.3 becomes log.4, log.2 becomes log.3, log.1 becomes log.2, and the current log is log. You now have four old logs (log.1, log.2, log.3, and log.4) as well as the current log.

At any one time, you have the current log and the prior four weeks' logs. What if you want to keep information for six months?

When running a Web server, you might want to retain your logs for six months to document attempts to breach your system. Your organization might want to keep records of when users log in and log out for several years. How many logs and how long you keep them will be different based on your organization's policies.

One way to keep logs for a long time would be to increase the number of logs that are kept. However, this could soon fill up your disk, especially if you want to keep logs for several months or years.

A better plan would be to move the older logs to another location. You might even want to move them offline to a tape or other media. Chapter 11, "Backup and Restore," covers managing backups in more detail.

You can use the `prerotate` command in your configuration file to run a script that will move the oldest log before rotating the logs. In this manner, you can keep logs for as long as you want and have the space to store them.

Key Concept

The `logrotate` command can be configured to keep as many logs for as long as you want. However, in order to save space, consider moving older logs to another location. The `prerotate` command can be used to move older logs before rotating. This saves space while still saving logs for an extended period of time.

Using Logs to Troubleshoot

One of the most frequent uses for log files is troubleshooting your system. To diagnose problems when booting your system, you can use the `dmesg` command to display system messages from the kernel ring buffer. The syntax for `dmesg` is

```
dmesg [options]
```

If you want to review a specific level of message, you can use the `-n` *`level`* option. If you want to see only panic messages, then use the following command:

```
dmesg -n 1
```

Although other messages will not be displayed, they will still be recorded and `syslogd` will continue to save them to the appropriate logs. Each level of message is defined in the `kernel.h` file. Check the one on your system to identify what the values are for your system.

Use the `-c` option to clear the buffer after displaying its contents. If you use both the `-n` and `-c` options, the first one will be ignored no matter what order you specify them in.

You also can use log files when you suspect an intruder has invaded your system. The log files `/var/log/wtmp`, `/var/log/utmp`, and `/var/log/lastlog` all contain information on users who have logged on to your system. These logs can be read directly or through the use of commands such as the `lastlog` command.

Summary

This chapter has covered how to schedule a job to run once or automate repetitive tasks through the use of the `at` and `crontab` commands. `at` runs a job only once. If called as `batch`, it waits for the system load to fall before running.

The `crontab` command is used to run the same job on a regularly scheduled basis based on time. The fields are minute, hour, day of month, month, and day of week followed by the command to be run. Be sure you know the fields and their order for creating cron jobs.

The `syslogd` daemon is responsible for writing messages to the specified log files. This information is contained in the `/etc/syslog.conf` file. When changing the configuration, `syslogd` must be stopped and restarted so that it will read the new configuration information. You can send a signal to `syslogd` to stop and then restart it.

Information on your users' logins is contained in the `/var/log/wtmp`, `/var/log/utmp`, and `/var/log/lastlog` files. Access these logs either directly or through the use of the `last`, `who`, `w`, `finger`, and `lastlog` commands.

How many logs you keep and for how long is a function of your organization's policies. Plan for how to configure them and where to store them to prevent running out of disk space.

QUESTIONS AND ANSWERS

1. You have an important meeting with your boss this afternoon and do not want to forget it. What should you do?

A. You could try putting Post-it notes on your monitor; however, a better solution is to use the `at` command to display a reminder of the meeting a short time before it begins.

2. You need to reindex your database but want to do it when the system's load is low to prevent adversely affecting your users. What should you do?

A. You can use the `batch` command to run the indexing job. That way it will not start until the system load is low.

3. You have a user who every morning schedules a job to run at 8 a.m. that slows down your system. What can you do to prevent him from scheduling a job using `at`?

A. You can add that user's name to the `/etc/at.deny` file to prevent him from running any more `at` jobs. If he really needs to run this, educate him about the `batch` command.

4. Your database administrator needs to run a script every day for system maintenance. You do not allow users to run cron jobs. How could you make her job easier?

A. You could schedule the job to run yourself. However, mail messages from `crond` about this job would be delivered to you rather than the database administrator, who should get them. You could either construct the job to mail the messages to her or add her name to the `/etc/cron.allow` file, which would allow her to schedule the job herself.

5. You want to save all the messages from mail to the `/var/log/maillog` file. What should you do to make this happen?

A. You can edit the `/etc/syslog.conf` file by adding the following line:

```
mail.*/var/log/maillog
```

This directs any messages from mail to the `/var/log/maillog` file.

6. You made the changes as to where the messages from mail are to be saved. When you look at the `maillog` file, it is empty. What happened?

   A. `syslogd` reads the `/etc/syslog.conf` file when the system boots. You could reboot your system to make the changes take effect. A better solution would be to stop and restart the `syslog` daemon.

7. You want to find out when each of your users last logged in. How can you find this information?

   A. This information is recorded in the `/var/log/lastlog` file. You could read the file directly, but a better solution is to issue the `lastlog` command, which displays this information sorted by `UID`.

8. You want to start a new mail log once a month and keep the last four months worth of logs. What should you do?

   A. You can use the `logrotate` command and set local options to rotate the mail log only once a month and to keep four logs.

PRACTICE TEST

1. You have written a new backup script and want to test it out. What would be the best way to do this?
   - **a.** Run the script now and check for errors.
   - **b.** Use the `at` command to schedule the script to run tonight and check on it in the morning.
   - **c.** Use the `at -b` command to run the script now.
   - **d.** Use the `crontab` command to schedule it to run every night.

Answer a is incorrect; although this would let you know whether it works or not, it is possible that it would adversely affect your users because it would be running during production hours. Answer b is incorrect; this delays your efforts to test the script and can even interfere with your regularly scheduled backups. **Answer c is correct; the `-b` option tells `at` to wait until the system load is low. Although you will not get your results immediately, you should not have to wait long and you do not need to worry about overloading your server.** Answer d is incorrect; because you do not know for sure that the new script works correctly, you should not schedule it to run on a regular basis.

2. What would be the results of running the following command?

```
at +15 minutes ls > myfiles
```

a. After waiting 15 minutes, the `at` command would save a listing of the files in the `pwd` to a file called `myfiles`.
b. The `at` command would save a listing of the files in the `pwd` to a file called `myfiles` every 15 minutes.
c. The `at` command would save a listing of the files in the `pwd` to a file called `myfiles` at 15 minutes after the hour.
d. Nothing, because the `at` command needs an exact time as to when it is to run.

Answer a is correct; this command will cause the `at` command to run the `ls > myfiles` command 15 minutes after entering it. Answer b is incorrect; the `+15 minutes` does not mean every 15 minutes. Answer c is incorrect; the `+15 minutes` is relative to now, not relative to the hour. Answer d is incorrect; you can give `at` a relative as well as absolute time.

3. You notice an increased load on your server and suspect it is from users scheduling jobs. What two commands could you use to confirm your suspicion?

a. `at -b` and `crontab -u`
b. `at -l` and `crontab -l`
c. `at -b` and `crontab -l`
d. `at -l` and `crontab -u`

Answer a is incorrect; the `at -b` command runs an `at` job when the system is low and `crontab -u` is used by root to list a specific user's cron jobs. **Answer b is correct; the `at -l` command is the same as `atq`, which shows a list of all pending `at` jobs, and the `crontab -l` command lists pending cron jobs.** Answer c is incorrect; although the `crontab -l` command will list cron jobs, the `at -b` will run an `at` job in batch mode. Answer d is incorrect; although the `at -l` command will list pending `at` jobs, the `crontab -u` command is used to look at a user's cron jobs.

4. You want to use a different configuration file called `sysconf` for `syslogd`. Which of the following commands would be appropriate?

a.
```
kill SIGHUP 'cat /var/run/syslogd.pid'
syslogd -f sysconf
```
b.
```
syslogd -f sysconf
```
c.
```
kill SIGTERM 'cat /var/run/syslogd.pid'
syslogd -f sysconf
```
d.
```
syslogd -r
```

Answer a is incorrect; the signal `SIGHUP` reinitializes `syslogd` by stopping the daemon, reading the `/etc/syslog.conf` and then starting `syslogd`. Answer b is incorrect; first you must stop the `syslogd` daemon before it can read the new configuration file. **Answer c is correct; the `SIGTERM` signal stops the `syslogd` daemon using its PID and then `syslogd` is restarted with the new configuration file.** Answer d is incorrect; the `-r` option would cause `syslogd` to receive network messages.

5. Which of the following lines would log all messages from news to the file `/var/log/newslog`?
   a. `news.!*                     /var/log/newslog`
   b. `news.*;news.!=info          /var/log/newslog`
   c. `*news                       /var/log/newslog`
   d. `news.*                      /var/log/newslog`

Answer a is incorrect; this would prevent logging any messages from news. Answer b is incorrect; this would log all messages except those of type `info`. Answer c is incorrect; the source of the messages must come first. **Answer d is correct; this would log all messages from news.**

6. You want to know how long each user's session was the last time they logged on. Which of the following log files contains this information?
   a. `/var/log/wtmp`
   b. `/var/log/utmp`
   c. `/var/log/usrlog`
   d. `/var/log/lastlog`

Answer a is correct; the `wtmp` log contains the login times and duration for each user. Answer b is incorrect; the `utmp` log has information on currently logged in users. Answer c is incorrect; there is no `usrlog`. Answer d is incorrect; the `lastlog` file contains the last login times for users but not the duration of their sessions.

7. Which of the following pieces of information is not included in the `syslog.conf` file?
   a. The source of the message
   b. The type of message
   c. The name of the file and its location
   d. The length of time to keep the message

Answer a is incorrect; the message source is always listed. Answer b is incorrect; the type of message is always listed. Answer c is incorrect; the name and the location of the log file to write to is required. **Answer d is correct; you cannot specify the amount of time to keep a message in the `syslog.conf` file.**

8. Which of the following cannot be accomplished using `logrotate`?

- **a.** Mail logs to root.
- **b.** Compress old logs.
- **c.** Log messages on another computer.
- **d.** Move old logs to another directory.

Answer a is incorrect; `logrotate` can mail logs to any user. Answer b is incorrect; `logrotate` can compress the old logs. **Answer c is correct; the `syslogd` daemon is responsible for where messages are written.** Answer d is incorrect; `logrotate` can save old logs to any specified directory.

9. When configuring `logrotate`, which of the following is possible?

- **a.** Specifying where to write messages
- **b.** Keeping three logs locally and three older logs to another directory
- **c.** Specifying how frequently to rotate the mail log
- **d.** Specifying what time of day to rotate the logs

Answer a is incorrect; the location of where to write messages is handled by `syslogd`. Answer b is incorrect; you can send all old logs to another directory but not just some. **Answer c is correct; you specify the frequency of rotating the logs in the configuration file used with `logrotate`.** Answer d is incorrect; you use `cron` to specify the time of day to perform the rotation.

10. You caught a glance of a lot of messages when your system booted. Which of the following would allow you to look at these messages?

- **a.** `dmesg`
- **b.** `lastlog`
- **c.** `cat /var/log/syslog`
- **d.** `syslogd -c`

Answer a is correct; the `dmesg` command displays the messages in the kernel's buffer; if you run this after booting, the messages will be those written during boot. Answer b is incorrect; the `lastlog` command displays the last time users logged in. Answer c is incorrect; unless you configured `syslogd` to write to a log file called `/var/log/syslog` there is no such file. Answer d is incorrect; the `syslogd` daemon is not used to display system messages.

CHAPTER PREREQUISITE

Before beginning this chapter you should be comfortable with the commands and tasks discussed in Chapters 5, "GNU and UNIX Commands," and Chapter 10, "Administrative Tasks," respectively. You also should have a good understanding of the Linux filesystem hierarchy, as discussed in Chapter 3, "Linux Filesystem," in order to be able to develop an appropriate backup strategy.

CHAPTER

11

Backup and Restore

WHILE YOU READ

1. Your server must be available to your users 24 hours a day, 7 days a week, but you need to add more hard disk space. What can you do?
2. You have been given the job of planning the backup strategy for your Web server. What files should you back up, and how often should you back them up?
3. You need to back up the employee database and keep the backups for five years. What would be the best way to do this?
4. Your backup plan consists of doing a full backup every Monday and a differential backup nightly Tuesday through Friday. Your server's hard disk fails on Thursday morning. What do you do to get the server back up as soon as possible?
5. You want to back up your users' home directories. You decide to use the `tar` utility and archive them to tape. What command should you use?
6. You need to restore a file from the backup of the home directories. How can you do this?
7. You need to compress your archive to save space on your tape. How would you modify the command you created for question 5?
8. You have created `tar` files of the employee database and want to store the last six months on a CD-ROM but there is not enough space. How might you make them fit?

Planning Your Backup

So what and why do you back up? Backing up your system is one of the most important jobs of any system administrator. It also is one of the least favorite jobs of most system administrators.

You create backups for only one reason—to be able to recover from a disaster when it hits. Notice, the operative word is *when* not *if* it happens.

Having an up-to-date backup means the difference between spending days or weeks recreating your server and having the system up and running in an hour or two. (It probably also means you get to keep your job.)

The subject of disaster recovery can and does fill entire books. Designing an appropriate disaster plan and periodically testing it is an important part of your job.

A wide variety of strategies can be utilized when creating your disaster recovery plan. When evaluating which one is best for you, you should consider some basic concepts.

Cost of Downtime

Even when running Linux, there will be times when the system will be unavailable for users to log in. Recompiling the kernel, upgrading software packages, installing new hardware, and performing system maintenance are just a few reasons why your server might be unavailable.

If the system is not available for users, the users' downtime will need to be taken into account when calculating the cost of server downtime. Because this can become quite expensive, you will need to weigh this cost against the cost of implementing your plan.

Cost of Backup Strategy Implementation

You will find a wide selection of disaster recovery strategies from which to choose. A solution costing a few dollars, such as copying important files to floppy disks, might be the solution you choose to implement. However, if your system provides critical services 24 hours a day, 7 days a week, your backup strategy might include a clustering and cost thousands of dollars to implement and maintain. Your choice of solution will be affected by the cost of server downtime, which directly relates to how critical your server is to your organization.

Workload of System

The role of the system is a determining factor of what level of availability is required. If you are running a Web server for an e-commerce site, you need 24-hour availability every day. If, on the other hand, your system is used for file storage in an office, its needed availability might be only during working hours. The increased cost for a more complex solution for disaster recovery is often justified when a server fills a critical role or requires a high level of availability.

Types of Backup Strategies

You can choose from several types of strategies when deciding which is right for you. Some of your options include

- Clustering
- Standby server
- Duplicate or backup server
- Backup and restoration

Clustering

Clustering consists of two or more computers that share access to a central data repository, such as a RAID array. Each of these servers can run different applications. If one member of the cluster goes down, another member will take over the downed server's job. This change is invisible to users. If you require 24-hour availability, this solution will provide it. However, it is expensive both as far as initial setup costs and the expense of ongoing management.

Standby or Failover Server

With this solution, a server that is identical to the online server is configured but is not available for users. It is live and listens to the main server to ensure it is available. When the main server goes offline, the standby server steps forward to handle the workload. It presents itself to the users with the same IP address and name, so the users never know which server is handling the workload. This is less expensive than clustering, but still is primarily used for systems that need 24-hour availability.

Duplicate or Backup Server

The *duplicate* or *backup* server is installed with the same applications as the main server but is left offline until the original server goes down. The change over is not instantaneous as it is in the two previous solutions. You will have to bring up the new server and connect it to any volatile data, perhaps by moving hard disks from one computer to another or connecting it to a RAID array. This solution is less expensive but does result in some downtime if the main server dies.

Backup and Restoration

The most common basis for disaster recovery is periodic copying of files to another location, which can then be used to re-create your system. Even with the more sophisticated solutions previously discussed, backing up files is the backbone of any disaster recovery plan.

Whether or not you elect to implement one of the other plans, you still will need to maintain a strategy of backing up your files. You also could elect to make this your only disaster recovery strategy.

Key Concept

Backup strategies can be as simple as copying a file to a floppy disk or as complex as clustering. Other middle-of-the-road solutions are backup and standby servers.

Backups

No matter how complex your disaster recovery plan is, you will need to maintain up-to-date backups of all your data as well as your system and application files.

Planning

The first step to planning your backup strategy is to answer four simple questions:

- What are you going to back up?
- How often will you back it up?
- How much time do you have to run your backup?
- What media will you use for your backups?

Let's address each of these separately.

What Are You Going to Back Up?

When deciding what to back up, consider the amount of time it will take to re-create the file versus the amount of time it will take to back it up. If it would take longer to re-create then back it up.

First, classify your files into those that change and those that don't. As a rule, system and application files change only when you upgrade an application or your operating system. Therefore, it is a waste of resources to back these up on a daily basis.

However, it is a good idea to periodically back up system and application files. Having a recent backup will enable you to bring your system back up with all the latest patches already applied in a shorter period of time.

For data files, you can have databases or other files that seldom change but can be used to retrieve information on a regular basis. In addition, some files change on an almost daily basis.

If you organize your files based on the frequency with which they change, it will be much easier to plan your backup strategy. See Chapter 6, "Maintaining the Filesystem," for more information on the Linux filesystem hierarchy and organizing your files.

Putting volatile data on separate partitions is another way to simplify your disaster recovery plans. If all your users' home directories are on a separate partition, you easily can back up this partition on a daily basis.

Backup Frequency

After you have decided on what files you want to back up—and chances are you decided on backing up everything—you will need to decide how frequently to back them up. You do not need to back up every file every day, although you could.

If files do not change, such as system and application files, why should you go to the expense of backing them up every day? Instead, you can elect to back them up weekly or monthly. Of course, you will want to back them up whenever you upgrade or make configuration changes. You also will want to perform a periodic backup of these files to ensure that you have a valid backup that can be used if you ever have to restore your system.

For files that change frequently, ask yourself the following question. What is the cost to my organization if that data is lost before a backup is performed? In most situations, daily backups of changing files is adequate. However, if you have mission-critical information that changes hourly, you might want to back up these files more frequently.

How Much Time Do You Have to Run Your Backup?

When planning what to back up, you will need to consider the amount of time you have for running the backup and how long it will take to back up the selected files.

For high availability systems, you might have a narrow window when certain files are available for backing up. Plan your backups appropriately. Even if you have to back up a few files at a time, you still should back up as frequently as you can.

Media Selection

Next, you will need to consider where to save your backups. Traditionally, this has been a tape drive. The advantages of using a tape drive include that they are relatively inexpensive—both the tape unit itself and the tapes. In addition, it is a technology that is well-known and very reliable.

The disadvantages of using tape drives primarily have to do with speed. The speed of writing to a tape is slower than writing to other media such as hard disks. A tape does not have a filesystem like what you have on random access media such as your hard disk.

In addition, because tape drives are read sequentially, it takes longer to retrieve information that has been stored to a tape. Other media, such as hard disks, enable random access, which speeds the retrieval process.

To use a tape for backing up files, you frequently use some utility or application that accesses the tape, creates a table of contents, and saves the data to the tape. This same utility or application is then used to examine the table of contents and read the tape sequentially from the beginning of the tape until the desired file is located.

Today, with the advent of cheaper hard drives and other media such as writable CD-ROMs and optical drives, you have several other choices. For the fastest access to your backups, you should keep the material online. This can be accomplished by saving the backup to another hard disk. However, this is a more expensive option.

The price of writable CD-ROMs has fallen to the point that they are a viable alternative. They provide fast and easy access when you need to retrieve data—if you clearly label them and maintain a log of what is stored where. They also have a long shelf life, making them a good choice when you have data that needs to be kept for an extended period of time.

If you need reusable storage media, you can consider one of the magneto-optical disks. These come in very large sizes, provide random access, and have a longer shelf life than tapes.

Key Concept

When planning your backup strategy, consider what files to back up, how often to back them up, how much time you have for your backup, and what media you will use to store your backups.

Types of Backups

There are four different types of backups:

- Copy
- Full backup
- Partial backup
- Incremental or differential backup

They are differentiated by what is backed up and how you go about restoring your files.

Copy

The easiest and most common method of backing up a file is to make a copy of that file. Although this method is frequently used, most of the time it is not considered a true backup.

When you copy a file to another location, you are then able to restore it if the original becomes corrupted or deleted by mistake. The media you copy it to is usually a floppy disk, but can be any of the other random access media available.

You should use this method when changing important system and configuration files. If you make a mistake, you only need to restore the original file by copying it back from its backup location to have your system back to its original state.

Full Backup

A *full* backup is exactly what its name implies. Every single file is backed up to your selected media. By having an up-to-date backup of your system, application, and data files, you can back up your system with a minimum of effort and in the shortest possible time.

The disadvantage of doing a full backup is that it can take a long time and will require more media to hold the backup. If your total system takes up a couple of gigabytes of disk space, you will need that much storage space. This might require that the media be changed, thereby preventing you from making this a completely automated process.

Partial Backup

When you keep files that change regularly on their own partition or at least in a separate area of the directory structure, you can do a full backup of just that partition or directory structure.

This is faster than backing up your entire system, but still ensures that the back ups you have are as up to date as possible. In addition, it requires less space on the storage media. If you do not need to change the backup media during the backup process, you can automate the backup. If you have critical data that changes frequently, you might want to perform a partial backup more frequently than daily.

Incremental or Differential Backup

When you do perform *incremental* or *differential* backup, you only back up those files that have changed since your last full backup. This is different from a partial backup in that the entire system is examined and only changed files are backed up. A partial backup is the same as a full backup, except only part of the filesystem is backed up instead of the entire system.

The advantage of doing a differential backup is that it is faster than doing a full backup. Also, because the total size of the files you are backing up is smaller, the backup is more likely to not require you to change the media.

The disadvantage of performing differential backups comes into play when you have to perform a restore. It will take you longer to restore when you use differential backups than if you use only full backups.

If you are restoring your entire system, you will need to restore the last full backup followed by any differential backups that have been performed since that time. These must be applied in the same order that they were made. Also, if you are restoring a file for a user, you might have to look through several differential backups before you find the desired file.

Key Concept

Four different types of backups exist. They are copy, full, partial, and differential or incremental.

How Long Should You Keep Backups?

Often you will be requested to restore a file that was deleted by mistake. Because these types of errors are not usually discovered immediately, the backup containing the file could be a week or more old.

When planning your backup strategy, you should consider how vital your information is and how frequently it is accessed. Use this information to decide how frequently you should recycle the reusable media you are using for backups.

For example, you might want to save your backups for a minimum of eight weeks. That means you should have enough media to hold eight weeks of backups. After eight weeks, you can recycle the oldest tape or disk. In addition, you might want to archive a full backup once a month and keep it for a year or more. Your organization's business model and the type of information you are backing up are integral to making this decision. One possible backup strategy might be

1. Full backup every Friday; reuse this tape in nine weeks
2. Differential backup every Monday through Thursday; reuse these tapes in nine weeks
3. Archive the first full backup of every month by placing it in the vault and keep it for one year

No matter what strategy you use, be sure to use and label your media appropriately. This is especially important when using tapes that are rotated. For example, you might have tapes that you rotate labeled *Monday, #1*, *Monday, #2*, *Monday, #3*, and so on.

Figure 11.1 shows one backup strategy in which a full backup is made every Friday and differential backups are created on Monday through Thursday. Each week you would use another set of backup tapes. So, the first week you would use a set labeled *number 1*, the next week you would use set *number 2*, and so on.

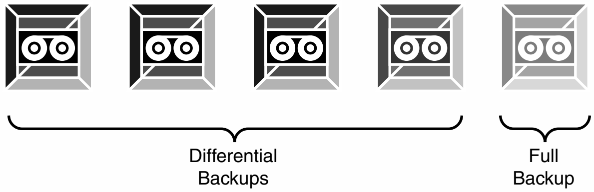


Figure 11.1
Although a backup is made daily, a full backup is made only on Friday. A differential backup is made Monday through Thursday nights.

Choosing a Storage Location

One other item you need to consider is the storage location of your backups. Remember, these backups contain the data from your system. Anyone who gets their hands on them can access your data. Do not leave backup media lying around where anyone can take it. To ensure the security of your data, lock up your backups.

Protecting Your Backups

Another consideration when storing your backups is the possibility of a fire. You can use fire-proof safes to store your media. However, these safes frequently reach internal temperatures high enough to destroy the media you have stored inside them. Consider storing your backups off-site to guarantee that you will be able to recreate your data no matter what the disaster.

Keeping a Log

In addition, you should keep a log recording what was backed up, when the backup was made, and where the backup media is stored. Update this log every time a backup is performed. This log should record such pieces of information as

- Date the backup was made
- Name of the backup tape

- Type of backup made
- What filesystem was backed up
- Where the backup media is stored

This log will assist you when you need to restore a file. Where you keep the log is up to you. You can keep the log with the media itself or next to the server. To further improve your ability to recover from a disaster, consider keeping a copy of your log in both locations.

Restoring Files from Backup

Even though you have a backup plan implemented, you still are not done. The backups you make are worthless unless you can retrieve that information. Don't wait until you have a system crash or a user comes to you needing a file restored to find out whether your restore works.

If you perform a full backup every day, restoring your system will consist of restoring the last backup you made. If you are using the backup plan shown in Figure 11.1, your restoration plan will be more complex than if you did a full backup every day.

If you have a file called `MyFile`, it will be backed up every Friday. As shown in Figure 11.2, the file `MyFile` was changed on Monday and Wednesday, so it also was backed up on those days.

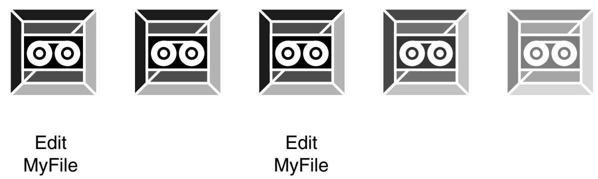


Figure 11.2
When doing differential backups, only files that were changed are backed up. When using differential backups to restore, you must restore the backups in the order that they were made.

If your system crashes on Thursday, to restore it you would first need to restore the full backup from the previous Friday. Then, you would need to apply each differential backup that had been made since then in the order they were made. So, you would need to restore Monday's tape, then Tuesday's tape, and finally Wednesday's tape. This would ensure that the latest version of all files would be restored.

A lot of things can go awry with your backups without you knowing about it. For example, your media might go bad. When you reuse your media, it suffers wear and tear.

Outside conditions also can adversely affect your media, such as rodents who like to eat tapes. With time, you can expect that parts of your media will become unusable.

You should routinely check your backups to ensure that you can restore these files. This includes verifying that you can restore your entire system.

Depending on what you use for your backups, the method you use to restore them will vary. If you use tapes, you will want to use the same utility when restoring that you used to create the backup. If you use read-only media such as CD-ROMS or reusable media such as magneto-optical disks, your indexing process will directly affect how difficult the restoration will be. This is where your log will save you time.

Next, we will discuss the most common utilities that are used with Linux to create backups. These are also the utilities that will be covered in the exam.

Key Concept

Backups are perishable. How long they last is dependent on the media used and the way they are stored. Keeping track of where they are stored is also important. To restore files, you must be able to find the correct backup—so keep good logs.

Backup Utilities

Many utilities and applications are available for you to use to perform your backups. Some of these are quite elaborate and simplify configuring your backup plan. Others are basic to all Linux systems. Choose the one that makes the most sense for your situation.

tar

The `tar` (tape archive) utility is used to combine multiple files into one large archive file while retaining the original directory structure. Although originally intended to create archives on a tape, `tar` files can be saved to any media. `tar` also has the capability to compress a file at the same time it creates it. Compression is covered in the section "Compressing and Uncompressing Files" later in this chapter.

Using *tar* to Create Backups

When creating a `tar` file, you should use only relative paths. In other words, construct the path to the files you want to archive relative to where you are in the filesystem hierarchy. If your location is `/home/hadden` and you want to archive all the files in the `/home/hadden/test` directory, you should enter the path as `test/*`.

If you try to use absolute pathnames with `tar`, the leading slash is stripped off. This prevents overwriting existing files when a `tar` file is unarchived. To override this behavior, use the `-P` option. The `tar` files end in `.tar`, and the syntax for `tar` is

```
tar [options] [tarfile] [files]
```

Options used with `tar` can be classified as either switches or modifiers. You can use only one switch at a time, but you can use multiple modifiers. Whichever switch you use must come before any modifier. Table 11.1 lists the switches that can be used with `tar`. Table 11.2 lists the modifiers used with `tar`.

Table 11.1 Switches to Be Used with `tar` (Only One Switch Can Be Used at a Time)

Switch	*Action*
`A`	Appends `tarfile2` to end of `tar` file
`c`	Creates a new `tar` file
`d`	Compares `tar` file's contents with other files and reports differences
`r`	Adds to the end of existing `tar` file
`t`	Displays name of files in `tar` file
`u`	Adds new or modified files to `tar` file
`x`	Extracts files from `tar` file

Table 11.2 Modifiers to Be Used with `tar`

Modifier	*Action*
`b`	Specifies block size
`e`	Prevents splitting files across volumes
`f`	`Tar` file's name with path or device
`F`	Filename containing `tar` arguments
`L`	Length of tape; *n* times 1024 bytes
`M`	Archive spans multiple volumes
`m`	Does not restore modification times
`n`	Device is not a tape
`p`	Keeps original permissions when extracting
`v`	Displays a list of files added
`W`	Verifies files after adding to `tar` file
`w`	Makes `tar` interactive
`z`	Uses `gzip` to compress archive

The command to create a `tar` file containing all the files in the `pwd` is

```
tar cf myarchive.tar .
```

This command causes the contents of the `pwd`, including any subdirectories, to be tarred into the file `myarchive.tar`. The `c` switch creates the `tar` file, and the `f` modifier provides the name for this new archive. The trailing dot (.) is used to specify the current working directory.

In addition, you can use the `f` option to specify the device name, as in

```
tar cf /dev/tape /
```

This command creates an archive of the entire root filesystem to the tape device. This overwrites any existing contents on the tape with the new archive, so be sure there is not anything on the tape you need to keep.

When creating an archive on a tape device, you usually have to specify the blocking factor by using the `b` modifier, as in the following example:

```
tar cvfb /dev/tape 20 /
```

The blocking factor times 512 bytes instructs `tar` as to how much data can be written at a time. A blocking factor of 20 can be used with most tape devices. When reading tapes, you do not need to specify the block size.

Key Concept

When a modifier requires an argument, the argument is listed after all modifiers are specified. These arguments must be in the same order as the modifiers are listed.

If your backup will span multiple tapes or disks, you will need to specify the media's length, as well as indicate that this will be a multi-volume archive. This can be a problem when backing up a large amount of data, especially if you also are compressing the backup.

When backing up to floppy disks, your backup probably will span multiple floppies. You will need to specify the size of the floppy and tell `tar` that the backup will span multiple disks, as in the following:

```
tar cnfM /dev/fd0 1200 /home/*
```

This command tells `tar` to create a `tar` file on the floppy device (`/dev/fd0`), which is 1200 bytes in size and contains all the files in the `/home` directory. The `M` tells `tar` that this archive will span multiple volumes. When the first one is full, you will have to manually change the backup media.

You can view the contents of your `tar` file using the following command:

```
tar -tf myarchive.tar
```

The `t` option causes `tar` to print the names of the files contained in the archive named `myarchive.tar`.

Restoring Files with *tar*

Not only will `tar` enable you to create an archive, it also enables you to restore the contents of an archive. To extract your `tar` file, type

```
tar xvf myarchive.tar
```

Here, the `x` option tells `tar` to extract the contents of the `tar` file called `myarchive.tar`. The `v`, or verbose, option causes `tar` to print which files are being extracted to the standard output. The `f` option names the `tar` file to be extracted.

It is important to examine your `tar` file before extracting it. You need to know how the `tar` file was built to ensure that files will be extracted to the correct location.

For example, if you created a backup of your users' home directories by typing

```
tar cf home.tar home/*
```

the directory name `home` is added to the beginning of each file added to the new archive. To extract this correctly, you must be in the `/` directory.

However, if you first made the `home` directory your `pwd` and then typed

```
tar cf home.tar .
```

the `home` directory would not be appended to the files' names. So, before extracting any files from this archive, you would want to make the `home` directory your `pwd`.

Key Concept

How you create a `tar` file makes a big difference in how files are stored. Using the command `tar tvf` *`tarfile`* will enable you to see the contents of the `tar` file and verify what directory names are stored in it. Always verify whether directory names are stored in a `tar` file before extracting any of its contents.

There are two different ways to restore a single file. First, you can choose which file to extract from a list of the contents of the `tar` file by typing

```
tar xvwf myarchive.tar
```

The `w` option places `tar` in interactive mode. You then are asked on a file-by-file basis whether to extract each file. If the `tar` file contains a large number of files, this can take a long time.

Remember that `tar` has the capability to accept a list of filenames as arguments. The second way to restore a file requires that you know its name. If you know the name of the file you want to extract, you can pass it directly to `tar`:

```
tar xf myarchive.tar myfile
```

Be careful when using this option. Although the `tar` command will not overwrite directories, it will overwrite any existing file with the same name. In order to prevent this behavior, use the `k` option, which will leave any existing files intact.

Many people use `tar` to manage their backups. Two options that make this so functional are the `u` and `r` options. The `u` option causes `tar` to update or append the specified file to an existing `tar` file. The `r` option appends a new file to an existing `tar` file. Both of these options require you to specify the existing `tar` file's name as well as the file you want to update or append.

cpio

The copy in and out (`cpio`) command also can be used to manage archives. It has three modes of operation:

- Create an archive containing multiple files
- Restore files from an archive
- Copy a directory hierarchy to a new location

The basic syntax for the `cpio` command is

```
cpio [options]
```

Table 11.3 shows the major options that can be used with the `cpio` command. You must use one and only one of these.

Table 11.3 Major Options to Be Used with the `cpio` Command

Option	*Action*
`-o`	Creates a new archive
`-i`	Extracts from an existing archive
`-p`	Copies a directory structure

Additional options available with `cpio` are listed in Table 11.4. Not all of these will work with each of the major options.

Table 11.4 Additional Options to Use with `cpio`

Option	*Action*
`-a`	Resets access times of files after copying them
`-d`	Creates directories as needed; used with `-i` or `-p`
`-e` *`filename`*	Specifies a file containing a pattern; used with `-i`
`-F` *`archivename`*	Names of archive to extract; used with `i`
`-m`	Preserves modification times of files; used with `-i`
`-r`	Prompts you for new name for file before copying the file; if no name is given, file is not copied
`-t`	Lists contents of archive; used with `-i`
`-u`	Overwrites existing files
`-v`	Lists all files as processed; used with all three modes

To create a new archive, you must pass the names of the files to `cpio`. To create an archive of all the files in the `/home` directory, type

```
find /home | cpio -o > home.cpio
```

This command uses the output of the `find` command to provide a list of filenames to be included in the archive. To verify the contents of the archive, you could then type

```
cpio -itF home.cpio
```

The `-i` option places `cpio` in extract mode, and the `-t` option causes `cpio` to display the names of the files but not to extract them. The `-F` is used to specify the name of the archive to examine.

To extract a specific file from the archive, you need to supply a pattern or filename to match. For example, to extract the file `memo.acct` from the `home.cpio` archive, type the following:

```
cpio -iF home.cpio memo.acct
```

Compressing and Uncompressing Files

Many utilities are available to compress and uncompress files. The most commonly used include `gzip`, `gunzip`, `tar`, and sometimes `compress`. You need to know how these work and how they are used because you will see them on the exam.

gzip

Files frequently are compressed to save storage space or transmission time when copying or moving from one computer to another. In addition, backups are compressed to reduce the amount of media that will be needed.

In the DOS/Windows world, this is accomplished by using the `pkzip/pkunzip` utilities. In the Linux world, the preferred format is `gzip/gunzip`, which creates a file with the `.gz` extension. Some differences exist between `gzip/gunzip` and `pkzip/pkunzip`, however:

- `gzip` does not compress more than one file
- `gzip` and `gunzip` automatically delete the original file

After using the `tar` command to create an archive, you might want to compress your file. You can use `tar` and `gzip` to create a compact backup, leaving you with the original directory structure and files intact inside a tarred and gzipped file.

The syntax for `gzip` is

```
gzip [options] [file-to-zip]
```

Table 11.5 lists options that are used with the `gzip` and `gunzip` commands.

Table 11.5 Options to Use with the `gzip` and `gunzip` Commands

Option	*Action*
`-c`	Displays file contents without deleting compressed file; used with `-d` or `gunzip`
`-d`	Uncompresses file; same as `gunzip`
`-n`	Does not save original date and timestamp
`-N`	Saves original name and timestamp; default action
`-q`	Suppresses warnings
`-r`	Recursively compresses files in specified directory; if used with `-d` or `gunzip` will recursively uncompress files
`-t`	Tests integrity of compressed file; used with `-d` or `gunzip`
`-v`	Prints name and percent compressed of each file; also can be used with `gunzip`

You can specify the amount of compression `gzip` will apply by using `gzip -`*`n`*. The accepted range is 1–9, with 1 being the fastest but with the least amount of compression. Using 9 causes the operation to take the most time but also results in the most compression.

You can use `gzip` to compress the `tar` file `myarchive.tar` by typing

```
gzip myarchive.tar
```

You also can combine these on the same command line using the pipe (|) character:

```
tar -cf myarchive.tar . | gzip myarchive.tar
```

This command first creates the `tar` file `myarchive`, which contains all the files in the `pwd`. Then, `gzip` compresses `myarchive.tar`, creating a new compressed file called `myarchive.tar.gz`. Lastly, `gzip` deletes the original file, `myarchive.tar`.

Because both the original and compressed versions of the file are present at the same time, you must have enough room for both the compressed file and the non-compressed file to exist at the same time or the compression operation will fail.

You can use another feature of the `tar` utility to `gzip` on-the-fly by using the `-z` option. The following example does the same thing, but requires fewer characters.

```
tar -cvzf myarchive.tar.gz
```

Key Concept

Remember, you will `tar` first, and then `gzip` to create an archive. `gzip` deletes the original file—in this case, `myarchive.tar`. The name of the file can end with `.tar.gz` or `.tgz`.

compress

The `compress` utility is one of the oldest compression utilities. The compressed file must end in `.Z` and its syntax is

```
compress filename
```

This utility is not used much anymore because its compression is not as good as newer utilities such as `gzip`. To uncompress the file, type the following:

```
uncompress filename.Z
```

gunzip

Okay, you have all these compressed files. What do you do if you need to extract files? First, you must uncompress the `tar` file using `gunzip`.

The syntax for `gunzip` is

```
gunzip filename
```

See Table 11.5 for options that can be used with `gunzip`. After `gunzip` has executed, you will have the original, uncompressed file. However, the compressed file is gone because gunzip deletes the compressed file after uncompressing it. In addition, you can use the gzip command to uncompress the file by using the -d option.

You can use tar to uncompress a compressed tar archive. Do this by using the z option. So, if you type

```
tar xvzf MyArchive.tar.gz
```

all the files contained in MyArchive.tar.gz will be restored, and the compressed file will still be there because tar does not erase the original file.

zcat

The zcat utility works like cat, except it enables you to see the contents of a compressed file. It works with files compressed using gzip (.gz extension) or compress (.Z extension). Its syntax is

```
zcat filename
```

After zcat is finished displaying the file's contents, the original file is not changed. It is still on your disk and it is still compressed.

Summary

Planning and maintaining up-to-date backups is an important part of your job. You should take the time to plan and implement a backup strategy that is appropriate for your organization and that takes into account the cost of the backup strategy versus the cost of data loss.

For the exam, you will need to know how to use tar to create archives and restore files contained in these archives. When creating a tar file, the original files are still left in place. Use the c option to create a new tar file and x to restore the tar file. Also, t can be used to display the contents of the tar file without extracting any of its files. It is important that you be aware of possible problems related to how path names are saved to a tar file.

Another command that can be used to make backups is the cpio command. It can be used to back up or restore files, as well as to copy an entire directory heirarchy. You also will need to know how to use the compression utilities gzip, gunzip, and compress and how to use tar to compress your archives. Remember that gzip and gunzip delete the original file, whereas compress does not. tar has the capability to compress while creating the new tar file.

QUESTIONS AND ANSWERS

1. Your server must be available to your users 24 hours a day, 7 days a week, but you need to add more hard disk space. What can you do?

A. To install another hard disk, you will have to take your server offline. However, your users still need to access the server. In this situation, you should consider having a backup server. This would enable you to work on the original server while still providing access to your users.

2. You have been given the job of planning the backup strategy for your Web server. What files should you back up and how often should you back them up?

A. You should perform a full backup including system files on a regular basis—as often as once a week. On the other days, back up only the information that changes, such as the Web pages.

3. You need to back up the employee database and keep the backups for five years. What would be the best way to do this?

A. Because this is data that needs to be kept for such a long time, you should select a medium with a shelf life greater than five years. Either CD-ROM or magneto-optical media would be good choices in this instance.

4. Your backup plan consists of performing a full backup every Monday and a differential backup nightly on Tuesday through Friday. Your server's hard disk fails on Thursday morning. What do you do to get the server back up as soon as possible?

A. First, you would restore the last full backup from Monday. Next, you would need to apply the differential backups in the same order they were made. So, after restoring Monday's full backup, you would restore Tuesday's differential backup, and then Wednesday's. The only potential data loss would be any files that were created or changed after Wednesday's backup but before the server crashed.

5. You want to back up your users' home directories. You decide to use the `tar` utility and archive them to tape. What command should you use?

A. You will need to tell `tar` what files to back up, the name of the `tar` file, and the device where the `tar` file is to be created. Therefore, you would type

```
tar cfb /dev/tape 20 /directory
```

6. You need to restore a file from the backup of the home directories. How can you do this?

A. If you know the name of the file you want to restore, you could type

```
tar xf /dev/tape filename
```

However, if you are not sure of the filename, you can have `tar` prompt you with each filename until you find the correct one by typing

```
tar xwf /dev/tape
```

7. You need to compress your archive to save space on your tape. How would you modify the command you created for question 5?

A. You could use `gzip` to compress the `tar` file after you create it. However, a more efficient way would be to tar and compress in one action by using `z` and typing

```
tar czfb /dev/tape 20 /directory
```

8. You have created `tar` files of the employee database and want to store the last six months on a CD-ROM, but there is not enough space. How might you make them fit?

A. You should compress these files before copying them to the CD-ROM. You could use the `compress` utility, but `gzip` would be a better choice because it uses a more efficient compression algorithm.

PRACTICE TEST

1. You have been instructed to devise a backup plan that will enable you to limit downtime to an hour while limiting the expense as much as possible. The data on this server changes rarely. Which of the following would be your best choice?

a. Clustering
b. Standby server
c. Backup server
d. Back up files to a tape drive

Answer a is incorrect; although this would limit your downtime, it is also expensive. Answer b is incorrect; although this is less expensive than clustering, this is also an expensive solution. **Answer c is correct; because the data seldom changes, this solution would enable you to be back online within the allowed time and cost less than either clustering or a standby server.** Answer d is incorrect; even if you have a good backup, if the server is down because of hardware, you probably will not be back up in the allotted period of time.

2. You have a server that is using 2GB of disk space and need to devise a backup plan. You want to have the backup finished in as short a period of time without you having to be there to change the tapes, but it takes two tapes to hold a full backup. What would be your best strategy to ensure up-to-date backups?

- **a.** Perform a full backup daily when you first get to work.
- **b.** Perform a differential backup each evening.
- **c.** Perform a full backup weekly and a differential backup on the other days.
- **d.** Perform a full backup weekly and have users copy any files they create or modify to a floppy disk.

Answer a is incorrect; although this will give you all the files and you will be there to change the media, it can interfere with your users' work. Answer b is incorrect; although this will get all new and changed files from that day, it would be difficult and time-consuming to perform a restore. **Answer c is correct; this strategy limits the requirement that you be present to change the media every day and reduces the time required for a restore.** Answer d is incorrect; leaving backup to your users will usually result in lost data.

3. You use CD-ROMs to back up your system by performing a differential backup on Monday through Thursday and a full backup on Friday. Your supervisor comes to you and tells you that the employee database is corrupt. She wants you to restore the version from last Thursday. What can you do?

- **a.** Use `grep` to examine each CD-ROM to find the version from last Thursday.
- **b.** Check your logbook to determine which CD-ROM has the correct file.
- **c.** Restore the employee database from the last full backup before last Thursday and then restore it from each of the differential backups since then.
- **d.** Ask the head of the Human Resources Department if she makes a copy of the database on a regular basis.

Answer a is incorrect; although you will eventually find the file you are looking for, it might take you several hours. **Answer b is correct; if you kept your logbook up to date, you will be able to find the correct CD-ROM in no time at all.** Answer c is incorrect; this will get you where you want to be but will take longer than checking the log to find the correct file. Answer d is incorrect; although she might have a copy, it is your responsibility to maintain the backups.

4. You are using `tar` to create a full backup of your system to a tape drive. Which of the following commands is your best choice?

   a. `tar cvf mybackup.tar /`

   b. `tar cfb /dev/tape 20 /`

   c. `tar cvzfb mybackup.tar /`

   d. `tar cvzfb /dev/tape 20 /`

Answer a is incorrect; this will create a `tar` file of your entire system but save it locally. Answer b is incorrect; this will write your `tar` file to the tape without compressing it. Answer c is incorrect; this would create the `tar` file and compress it but save it locally and not on the tape. **Answer d is correct; this creates a compressed `tar` file on the tape containing all the files and directories of your system.**

5. You want to create a backup of your home directory to floppies, the total size of which is 2.1MB. After making your home directory your `pwd`, which of the following commands would be correct?

   a. `tar cnfM /dev/fd0 1200 /home/*`

   b. `tar cnfM /dev/fd0 1200 .`

   c. `tar cnf /dev/fd0 1200 .`

   d. `tar cvf /def/fd0 .`

Answer a is incorrect; this would create a `tar` file of all the files in the `/home` directory, not just your files. **Answer b is correct; this would create a multi-volume backup to 1.2MB floppies of the files and directories in the current directory.** Answer c is incorrect; this would fail because your backup will not fit on a single floppy. Answer d is incorrect; you need to specify the size of the floppy and the fact that this is a multi-volume archive.

6. You backed up your `memos` directory to `mymemo.tar` and now you need to restore the `personnel.memo` file. Which of the following commands will work?

   a. `tar -xf mymemo.tar personnel.memo`

   b. `tar -xW mymemo.tar personnel.memo`

   c. `tar -xw mymemo.tar personnel.memo`

   d. `tar -xn mymemo.tar personnel.memo`

Answer a is correct; this will extract the file `personnel.memo` from the `tar` file `mymemo.tar`. Answer b is incorrect; the `-W` option verifies files after adding to a `tar` file. Answer c is incorrect; although this would place `tar` in interactive mode, it does not use `f` to name the `tar` file. Answer d is incorrect; the `n` option specifies that the device is not a tape.

7. You want to use `cpio` to create an archive of the users' home directories. Which of the following commands is correct?

   a. `cpio -o homebackup.cpio /home/*`

   b. `find /home | cpio -o homebackup.cpio`

   c. `find /home | cpio -o > homebackup.cpio`

   d. `cpio -o > homebackup.cpio`

Answer a is incorrect; `cpio` requires a list of all the files to be added to the archive. Answer b is incorrect; `cpio` outputs the archive to standard output unless it is redirected. **Answer c is correct; here you use the `find` command to get a list of files to add to the archive and then instruct `cpio` to save its output to the file `homebackup.cpio`.** Answer d is incorrect; although this command redirects the output, it does not supply a list of files to add to the archive.

8. You have created a full backup of your system called `myfull0913.tar` and plan on storing it on a CD-ROM. First you want to compress the archive as much as possible. Which of the following commands is correct?

   a. `gzip -d myfull0913.tar`

   b. `gzip myfull0913.tar`

   c. `gzip -v myfull0913.tar`

   d. `gzip -9 myfull0913.tar`

Answer a is incorrect; the `-d` option is used to list the contents of a compressed file. Answer b is incorrect; this would compress the file, balancing speed and amount of compression. Answer c is incorrect; the `-v` option would display the name and percent compressed of the file. **Answer d is correct; the `-9` tells `gzip` to use the maximum amount of compression at the cost of speed.**

9. When reviewing some backup scripts written by another system administrator, you see the following line:

   ```
   tar cf fullbackup.tar.Z / | compress
   ```

   What does this mean?

   a. A `tar` file is created of the entire system called `fullbackup.tar.Z` and then compressed. This is optimal.

   b. A `tar` file is created of the entire system called `fullbackup.tar.Z` and then compressed. Using `tar czf fullbackup.tar.gz` would result in greater compression.

   c. A `tar` file is created of the entire system called `fullbackup.tar.Z` and then compressed. A better command would be `tar cf fullbackup.tar.gz | gzip`.

   d. A `tar` file is created of the entire system called `fullbackup.tar.Z` and then saved to the `compress` directory.

Answer a is incorrect; the `compress` utility is an older compression utility that does not have the most optimum compression algorithm. **Answer b is correct; this command would use `gzip` instead of `compress` to perform the compression, resulting in a smaller file.** Answer c is incorrect; although this command line would achieve the same results as b, it is less efficient. Answer d is incorrect; the command pipes the `tar` file to the `compress` utility; it does not save it to the `compress` directory.

10. You used the command `gzip memo.bill` to compress the file `memo.bill`. You now want to review some figures that were included in this file. How could you do this?

- **a.** `zcat memo.bill.gz`
- **b.** `gunzip memo.bill.gz; less memo.bill`
- **c.** `cat memo.bill.gz`
- **d.** `less memo.bill.gz`

Answer a is correct; the `zcat` utility enables you to examine the contents of a compressed file without compressing it. Answer b is incorrect; this would uncompress the file and then display it using the `less` command, but you would then have to recompress it. Answer c is incorrect; the `cat` command cannot display compressed files. Answer d is incorrect; the `less` command cannot display a compressed file.

Glossary

access To connect to and utilize a device; to store and retrieve information stored on a peripheral device such as a hard disk.

account Relationship between user and operating system; consists of username, password, and permissions.

address A location in memory; identifier assigned to a device such as a sound card; to manage.

administer To control the operation and use of a computer or other device; the task of a system administrator.

algorithm Logical steps used to solve a problem; will generate output as a result that satisfies specific requirements.

alias An alternative name used to identify a program.

Alt The Alternative key on a keyboard.

analog Representation of an original signal as the telephone turns vibrations into electrical signals.

ANSI American National Standards Institute, which is responsible for setting standards for many protocols such as telecommunications protocols.

API Application Programming Interface used by an application to communicate with the operating system or another application.

application A specific use of a computer or program that runs on top of an operating system.

archive To copy data to another device for retention; a file that contains one or more files.

argument A value passed to a command, function, or program.

ASCII American National Standard Code for Information Interchange; binary code used for text.

background Method of running a non-interactive process.

backup Copy of data saved for insurance purposes; to make a copy of data.

binary Numbers consisting of 1s and 0s; used in computers as on/off switches; file containing compiled code.

BIOS Basic Input/Output System, which provides an interface between the operating system and hardware.

bit Binary digit; smallest measurement of computer storage; single binary digit (0 or 1).

block device A peripheral device that moves one group of bytes at a time.

boot The action whereby a computer starts to execute instructions.

boot loader An application used to start an operating system.

buffer Memory that temporarily holds data to be processed.

build Version of a program; to compile a program.

bus Path between multiple devices.

byte BinarY TablE. Measure of storage consisting of 8 bits; a single character.

cache Temporary data storage used to improve speed.

card Any printed circuit board that can be plugged into a computer expansion slot.

CD-ROM Compact Disc Read-Only Memory. A removable medium used to hold data, such as text, binaries, sound, and video; the device used to access this data.

character A letter, number, or symbol equivalent to a byte; command-driven interface.

character device A device that moves data one character or byte at a time.

child process A process called by another process.

client A computer that connects to another computer and receives information.

command-line interface A text-based interface used to enter commands.

compression The process of removing redundant information to reduce the size of a file.

computer A machine that processes data and its attached devices.

console A display terminal used to control and monitor a computer; user interface of a computer.

core dump The contents of memory written to a file, usually called core, when a program crashes.

corrupted Damaged file or data.

CPU Central Processing Unit. The processor or computing component of a computer.

crack To access a computer system without authorization, usually for destructive reasons.

cracker Person who cracks a computer system; a program used to crack a computer system.

crash An unexpected stoppage of a program or computer system; termination of software or hardware due to some failure or error.

Ctrl The Control key on a keyboard.

current working directory Present location in filesystem hierarchy.

cwd *See* current working directory.

cylinder All tracks located in the same location on every disk surface.

daemon A program that runs in the background and performs an operation when required.

data Facts and figures that can be processed to provide information.

database A collection of data stored in a logical and accessible manner; program used to access this data.

default The assumed value of a parameter if no value is explicitly specified.

Del The Delete key on a keyboard.

delete To remove or erase data.

device A piece of hardware that can be attached to a computer.

die To stop execution.

Direct Memory Access Circuitry that moves data from one memory block to another, bypassing the CPU.

directory A type of file on a disk used to organize other files.

disk A direct access storage device.

display A screen or monitor used to display information from a computer.

distribution A collection of software needed to operate a computer, including the Linux kernel and various utilities and applications.

DMA *See* Direct Memory Access.

documentation A written document that describes a system or application.

drive A device that provides the capability to store and retrieve data.

driver Software that provides access to a peripheral device.

dynamic Decisions or actions performed during instead of before processing.

editor Software used to create and edit text files.

emulator Software that acts like another type of hardware or software.

environment Configuration of hardware or software variables that are used by a process.

environment variables Parameters used to define the hardware or software configuration.

Esc The Escape key on a keyboard.

Ethernet A shared media type of LAN in which all nodes share the available bandwidth.

event A happening or occurrence.

executable A binary or script that can be run.

execute To cause a program to start.

execute permission Permission set on a file that allows the file to be run.

export To make a value available to another process.

FAT *See* File Allocation Table.

FHS *See* Filesystem Hierarchy Standard.

file A collection of bytes stored as a single entry with a name.

File Allocation Table A simple filesystem that tracks where data is stored.

filesystem The structure used to control the storage and retrieval of data.

Filesystem Hierarchy Standard A proposed standard for the location of files on a UNIX system.

Filesystem standard An older standard for the structure of the Linux filesystem, which has been replaced by the FHS.

filter Process to change the content of files or displayed data.

floppy disk Removable, reusable magnetic storage media.

floppy drive A device used to read and write floppy disks.

foreground To run an interactive process; the highest priority.

fork When an executing process creates a duplicate of itself.

format Structure of an item that specifies how data is stored; to apply the structure to storage media.

Free Software Foundation A tax-exempt charity that raises funds for work on the GNU project.

FSF *See* Free Software Foundation.

FSSTND *See* Filesystem standard.

GB *See* gigabyte.

GID *See* Group Identifier.

gigabyte One billion bytes.

global A variable available to other processes.

GNU GNU's Not UNIX. Free Software Foundation project to build UNIX-compatible utilities and programs based on free source code.

GPL (GNU General Public License) A license for distribution of free software that permits copying and modification.

group A way to organize users to grant the same privileges and access rights to them all.

Group Identifier Unique number the system uses to identify a group.

hack To write source code to solve a problem; often refers to accessing a computer system without authorization.

hacker One who hacks; similar to cracker.

hang To freeze or lock up without terminating the process.

hard disk A magnetic device used for storage of data.

hard link A pointer to data location; shares the inode of that file.

hardware Equipment that makes up a computer.

HOWTO Documents that explain the steps necessary to accomplish a task.

I/O *See* Input/Output.

I/O port Memory address peripheral devices use to communicate with the CPU.

ID Identifier; name.

idle Inactive; awaiting instructions.

information Summary of data.

inode Data structure containing information on stored data.

input Data that is passed to a process; the action of entering data.

Input/Output The transfer of data from a device to the CPU and back.

install The act of creating an operating system or adding an application to an operating system.

integer A whole number.

interactive A data exchange between user and computer.

interface A connection between hardware, software, and user.

interrupt A signal used by a device to get the attention of the CPU.

invoke To execute a program.

IRQ Interrupt ReQuest. Hardware interrupt.

ISO International Standards Organization. Organization that sets international standards of the classification of materials, manufacture of products, and provision of services through a network of organizations from over 130 countries.

job A unit of work composed of one or more processes that execute in the background.

KB *See* kilobyte.

kernel The basic part of the operating system that always resides in memory.

keyboard An input device composed of individual keys marked with letters, numbers, and symbols.

kilobyte A unit of measurement; 1024 bytes.

LDP *See* Linux Documentation Project.

LILO LInux LOader. A program that loads the kernel so Linux or other operating systems can boot.

Linux A UNIX-like operating system that runs on x86, PowerPC, or Alpha chips; first developed by Linus Torvalds and freely available under the General Public License.

Linux Documentation Project A voluntary effort to improve Linux documentation.

load To copy a program from storage to memory to execute it.

log A record of computer activity.

logfile *See* log.

login The process of accessing a computer system by being authenticated.

logoff The process of exiting or stopping access to a computer system.

logon *See* login.

logout *See* logoff.

LPI Linux Professional Institute. Nonprofit organization founded to create a certification program for Linux.

man page *See* manual page.

manual A document created that covers the structure and use of hardware or software; performing a task yourself.

manual page Documentation installed on UNIX systems that covers various software.

Master Boot Record Usually the first sector of the first partition of a disk, which points to the operating system and contains instructions for booting the operating system.

MB *See* megabyte.

MBR *See* Master Boot Record.

media Storage material that holds data.

megabyte A measure of storage; 1,048,576 bytes.

memory Location for storage of data accessible by CPU; can be RAM, ROM, cache, or virtual.

mini-HOWTO A smaller HOWTO.

monitor Display; watch computer activity; software used to watch a computer's activity.

offline Not connected to a computer or network; not installed.

online Connected to a computer or network; installed.

operating system A program that controls the computer; first program installed and executed.

output Data generated by a process.

owner Account associated with a file; usually the account that created the file.

parameter A value passed to a program.

parent process The process that was responsible for starting the present, or child, process.

Parent Process ID The process identifier of a process's parent process.

partition A section of a hard disk used to store data; the act of dividing a hard disk into one or more sections.

password A word associated with an account that is used to gain access.

path The road to a file composed of directories and subdirectories; the route searched by default when executing a command.

PC *See* Personal Computer.

peripheral A device connected to a computer.

permission A defined level of access.

Personal Computer A computer intended to be used by one individual at a time.

PID *See* process ID.

pipe A method of using the output from one process as the input of another process.

PPID *See* Parent Process ID.

priority The relative importance of a process that determines the amount of CPU time it can have.

process A running program that is consuming computer resources.

process ID The identifier of a process assigned by the kernel.

processor The workhorse of a computer; also known as *CPU*.

program A collection of instructions that accomplish a task.

prompt An indicator that the user can provide input to the program.

RAM *See* Random Access Memory.

Random Access Memory Memory that can be randomly accessed; requires power to maintain its contents.

Read-Only Memory Memory chip used to permanently store data and instructions; cannot be altered.

read permission The right to examine the contents of a file.

redundant Repeated or duplicate information.

regular expression A pattern that can be used for searching.

resource Any hardware, software, or datum that is part of a larger system.

restore The process of placing a copy of data back on the system.

Return The Enter key on a keyboard.

ROM *See* Read-Only Memory.

root The superuser account, has a UID of zero; the base of the filesystem.

run To execute a program.

runlevel The level or mode of operation.

scan Sequential searching of a file or computer devices.

script A program written in an interpretive language; not compiled.

sector The smallest unit of storage on a disk.

server A computer shared by multiple users providing a particular service.

service *See* daemon.

Set Group ID SGID. Special permission that sets a user's effective permissions to that of the group.

Set User ID SUID. Special permission that sets a user's effective permissions to that of the owner.

SGID *See* Set Group ID.

shadow A complex encryption method for the protection of passwords.

shell A program that enables the user to interact with the computer, conventionally applied to command-line driven interfaces with scripting capabilities such as Bash.

shutdown Command used to shut down or reboot the computer.

signal An interrupt sent to a process.

single-user mode Maintenance mode that is not multiuser.

soft link *See* Symbolic Link.

software A series of computer instructions used to perform a task.

source code The plain text instructions used to create a program or application.

spawn To create a child process.

static Unchanging.

stderr The output device for error messages.

stdin The default input device.

stdout The standard output device.

sticky bit A bit set on an executable that keeps the program in memory after execution has ended.

SUID *See* Set User ID.

superuser The root account.

swap space Referred to as *virtual memory*; location on disk where data can be stored when not needed immediately.

Symbolic Link A pointer to the location of a file.

symlink *See* Symbolic Link.

synchronize To make a copy of data the same as the original.

syntax The grammar that defines command use.

system The software and hardware that compose a computer.

terminal An input/output device for a computer; usually a display and keyboard.

terminate To stop or end.

text Characters that can be displayed on a terminal.

third party A vendor that supplies an add-on to the base system; can be hardware or software.

tool A program used to manipulate data.

troubleshoot The process of identifying a problem, diagnosing its cause, and correcting it.

tune The process of changing parameters to improve operation.

UID *See* User ID.

umask The default permissions applied to newly created files.

uninstall To remove hardware or software from a computer.

unload To remove a process or service from memory.

upgrade To change hardware or software to a newer version.

user An individual who is using a computer; an account on a computer.

User ID The unique ID used by the system to identify a user.

utility A piece of software used to accomplish a task.

vendor A company or individual that provides a service or product.

virtual A condition without boundaries; simulated feature or state.

virtual memory Hard disk space used to mimic RAM.

wildcard A character used to represent one or more other characters.

write permission The permission to make changes to a file.

APPENDIX

B

Certification Process

Test Development

The Linux Professional Institute (LPI) started out as a group of volunteers who had a common goal of developing a certification for Linux that would be a valid indicator of the candidate's abilities. This certification then could be used by employers as a way to determine a prospective employee's competence.

LPI has worked for over a year to develop a distribution-independent certification program for Linux. The debate over the need for a certification has been quite heated. (See the archives at `http://www.lpi.org` for further information.)

As one of the first steps in creating the certifications, LPI performed a job analysis. The purpose of this job analysis was to determine which tasks are common to all Linux system administrators.

Both the frequency and importance of each task was evaluated. See `http://www.lpi.org/p-jasreport.html` for the results of this survey. These results then were used to determine the various certification levels.

As a result of these efforts, LPI has defined three levels of certifications. They are

- LPIC Level 1
- LPIC Level 2
- LPIC Level 3

Each of these certifications builds on the one before. In other words, achieving LPIC-2 certification will require that LPIC-1 certification is first successfully completed.

The individual who has achieved certification as LPIC-1 will be expected to be able to install and manage a Linux workstation, while the LPIC-2 certified individual will be able to manage a network. The LPIC-3 certified individual will possess specialized knowledge and be able to function as administrator of a large, complex network.

The LPIC Level 1 is intended to certify that an individual can

- Install and configure a workstation, including connecting it to a LAN or configuring a modem to connect to the Internet
- Work at the UNIX command line
- Shut down and reboot the computer
- Perform helpdesk duties
- Manage user accounts
- Perform backups and restorations of files

In order to achieve LPIC Level 1 certification, the candidate must pass two separate exams, which are

- Exam 101, General Linux I
- Exam 102, General Linux II

To review the objectives for a specific exam, visit the LPI Web site at `http://www.lpi.org`.

Registering for an Exam

After you have studied and are ready to take Exam 101, General Linux I, you must register for it with VUE. To register, go to `http://www.vue.com/Linux`.

When you register for your first LPI exam, you will need to call VUE. This is necessary to link your LPI testing information to your VUE information. This process is required for only the first time you register to take one of the LPI exams. The appropriate telephone number can be obtained at `http://www.vue.com/student-services/vuephone.html`.

After your LPI and VUE information is linked, you can register online at `http://www.vue.com`, by calling VUE's main office directly, or by calling the VUE testing center where you want to take your exam.

You also can register by mail. The necessary form and mailing address are available at `http://www.vue.com/testing/mailorder.html`.

To register online, you will need a username and password. If you have not taken an exam through VUE before, you can obtain your username and password online. After receiving your username and password, you can log in and register.

The registration process consists first of selecting a testing center. Based on your location, you will be presented with a list of testing centers in your area. Next, you can specify a range of dates and times in order to find an available slot to select.

The exam is presently being offered only in English and costs $100. When registering online or by phone, you will need a credit card to pay for your exam. If you want to pay by check, you will need to send your registration by mail. An appointment slot cannot be reserved until VUE has received your payment.

After you have scheduled an exam, you can use the VUE Web site for other functions. For example, you can reschedule or cancel your exam after first logging into the Web site. In addition, after completing your exam, you can view your results online.

When you go to take your exam, you first must sign in at the testing center. This process will require that you provide two types of identification, one of which must be a picture ID. Types of identification that are acceptable include the following:

- Driver's license
- Passport
- Credit card
- School ID
- Birth certificate
- Baptismal certificate
- Any other government issued identification

You should arrive a few minutes before your scheduled appointment time so that you will have enough time to complete this process.

You will be allotted one and one-half hours (90 minutes) to take the exam, so be sure that you arrive early. See Appendix C, "Testing Tips," for some suggestions on how to improve your testing experience.

Certification Results

When you finish an exam, you will be given a printout of your test results, regardless of whether or not you pass. Be sure to put this document in a safe place because it cannot be replaced. If any question should arise at a later date as to whether or not you passed, this will be your proof.

After you have passed any of the LPI exams, your certification status can be verified to any current or prospective employer at your request. LPI will verify only which exams you passed, the level of the exam, and the date you passed. They will not provide information on any exams you failed or give out your exam scores. After you have passed the two exams necessary for LPIC Level 1 certification, you will receive a certificate from LPI acknowledging your accomplishment.

Examination Revision

LPI plans to review each examination and its objectives every two years. This review will include any new material as well as the security and validity of each exam.

If an exam is revised, LPI will not require certificate holders to recertify. However, they will provide information to the public as to any changes in the content or objectives of their exams. Also, if the certificate holder requests verification of his or her certification status, the information on the present version of the exam also will be provided.

APPENDIX

C

Testing Tips

Taking a certification exam is different from taking a test in school. In the first place, chances are you do not have an instructor to lead you through the material you need to know for the certification exam, as you did in school.

Certification exams are technically oriented. They are written with certain objectives in mind, and the questions are devised to determine whether you know how to perform certain tasks as well as whether you understand specific concepts.

In school, you are tested as a step to the next level of your education. Certification exams, however, are written with the aim that only those who really understand the concepts being tested will pass.

After you have studied and feel you know the material, you should register for the exam. Presently, the LPI exam is being given by the VUE. You can register through their Web site or by phone. See Appendix B, "Certification Process," for details on the registration process.

Scheduling the Exam

Know your own personal circadian rhythms. If you are a morning person, try to schedule your exam during morning hours. However, if you are the type of person who does not wake up until noon, you should schedule your exam for the afternoon.

You will be allotted an hour and a half to take the test. You want this time to coincide with the time of day that you are at your best.

Next, you must decide where you want to take your examination. There might be more than one testing center in your city. You can call the center you want to use and ask them for their ID code. Then, when you register for the exam, you will use this code to ensure you'll be registered at the correct location.

When you register, you will be given an ID number, which you'll use for every test you take. You'll also use it if you ever need to contact VUE to get copies of your transcript.

Be sure to obtain and write down the following information during the registration process:

- Date and time of the exam
- Number and title of the exam
- Address and phone number of the testing center
- Confirmation number
- Deadline by which to cancel or reschedule your exam

LPI will use its own method to identify you. This number will be used to provide information to prospective employers to verify your certification status.

The Day of the Exam

On the day of the exam, you should verify the time and location of the test you'll be taking. Call the testing center the morning of your test make sure you are registered to take the correct exam at the time you expect. Each center downloads the day's schedule first thing in the morning, so you can call anytime after the center opens to verify whether you're on that day's schedule.

If you are scheduled to take your examination when the testing center first opens, you will not have this option. The testing centers do not get their schedules until that day.

This might seem like overkill, but many prospective candidates have arrived at the testing center to learn they are not on the list or that their appointment time is different from what they expected. This has happened to individuals taking certification tests sponsored by various vendors. Be sure you confirm three important pieces of information:

- Where the testing center is located
- Exactly what time your exam begins
- How far in advance of the start time you should arrive

Get enough rest the night before the exam. On the day of your exam, be sure to schedule enough time so you aren't rushed. Don't forget to eat—your brain functions much better when it is well nourished. And you don't want to be distracted by your stomach growling at you because it is empty.

You also should wear comfortable clothes. There is nothing as distracting as trying to sit in jeans that are too tight. No dress code exists for sitting the test, so wear what you are most comfortable in.

In addition, you should be sure you leave in plenty of time to get to the testing center with 15 to 30 minutes to spare. Don't start off your test by being rushed. That feeling will carry over to the test and possibly affect your performance.

Take some time in the parking lot to review your notes. Use this time to verify that you know the facts you memorized. This is not the time to cram but rather the time to review what you already know one last time. Having time to verify that you know the material will help place you in the correct frame of mind to do well.

When you go in to take the test, you will be required to provide two forms of identification. At least one of these must have your picture on it.

After you have presented your identification and signed in, you will be taken to the cubicle or room in which you will take your test. You also will be given writing materials, which might be a pen and paper or a marker and dry erase board. Be sure that the pen or marker writes; get a spare to be on the safe side. (I will cover how to use these writing materials later in this appendix.)

All examinations are taken on a computer. Before the proctor leaves the room, verify that everything on your computer is in working order. Does the mouse work? Is the display easy to read? Does the lighting cause excessive glare? If there is a problem, speak up. You have the right to an environment suitable for a comfortable testing experience.

The time allotted to take the test does not start when you enter the testing room. Rather, the time starts when you actually begin the test itself.

After you have verified that everything is in working order, you will have 10 to 15 minutes to get oriented to the testing environment. A sample test is available for you to take just to get used to the experience of computer testing. This sample test is optional.

Remember those writing materials you were given? Now is the time to put them to use. Although you cannot take any notes into the exam with you, use the materials provided to jot down the facts you memorized and reviewed in the parking lot. You can use the time allotted for the sample exam to write down any notes or charts you feel might help you when taking the test.

Two reasons exist to write these things down: First, you can refer to these notes when answering questions. Second, and most important, the act of writing down this information has great psychological importance.

The fact that you can recall the information increases your self-confidence. If you can remember the facts to write them down then you know the material—so you can expect to do well on the exam.

The time you spend writing out your notes also helps calm those pretest jitters so that you will be able to perform better. Many test-takers become so nervous about the idea of taking a test that their performance suffers. Do everything you can to help yourself remain calm.

Taking the Test

At the time of this book's writing, the questions on the exams are all multiple-choice and short answer. For the multiple-choice questions, the question itself might be scenario-based, but you will have four or five answers from which to select. The short answer questions cover various commands, sometimes including necessary options.

The exams have a total of 60 questions, and you are given 90 minutes to finish them. At the end of each exam, a short survey is included. This survey is optional and the time spent completing it is not included in the 90 minutes.

When planning your strategy for the exam, you should determine whether the exam will be conventional or adaptive. The LPI exam is currently conventional, although there is talk of possibly making it adaptive in the future.

With an adaptive exam, you are given a question, with the next question being dependent on whether or not you got the first one right. If you are correct, the next question is more difficult. If you are incorrect, an easier question is presented. After you answer a question, you can't go back and change your answer.

With a conventional exam, you can go back and review your answers. The order in which you answer questions is not important. You should use a strategy of reading each question and choosing an answer. Do not take too much time deciding on an answer. If you are unsure of the correct answer, choose your best guess and mark the question for review.

Types of Questions to Expect

Each question on a technical exam usually falls into one of three general categories: enumeration of facts, understanding the concept, and applying the concept.

Enumeration of Facts

The easiest questions to answer are those that fall into the enumeration of facts category. These questions simply test your memorization. In this type of question, you might be asked to supply a specific term or define a term. You might be asked a question such as "What are the types of partitions?"

Extraneous material might even be supplied in the question itself. However, if you read each question and identify what information is wanted, it should be obvious what the correct answer is. For example, you might have a question that gives you a list of all the hardware on a computer as well as what is on the hard drive, but the question actually wants to know if another primary partition can be created.

Understanding the Concept

A more difficult type of question for most individuals to answer is the one that asks for a concept instead of discrete facts. Let's use our previous example of partitioning. A question testing the concept of partitioning might be looking for evidence that you understand the concept of the types of partitions that can be created on any single hard disk.

So, you might have a question that gives extensive information on the hardware configuration of your computer but is actually attempting to ascertain whether you understand the limitations on the number and types of partitions you can have on a single hard disk.

Applying the Concept

The third, and hardest, type of question tries to determine whether you can take a concept and apply it in real life. Instead of asking you what types of partitions you can create, the question might present a scenario in which you must identify that the problem relates to partition creation and then decide how to correct it.

Using the same example, your question might be that you have free space you want to use for a new database application, but the operation fails. Your answer would be based on identifying why you can't create the partition and how to correct it.

When reading each question, you will want to identify the type of question it is. This will go a long way toward being able to select the correct response.

You also should try to identify the objective that is being tested. If the question is intended to gauge your abilities in using text filters, the answer will not be one that has to do with the directory structure.

If the question is scenario-based, you will be presented with a situation and have to select the answer that best solves the problem. Some candidates become overwhelmed when faced with scenarios, but scenarios really are not that hard.

Once again, you want to use that writing material. As you read the scenario, take notes on the facts presented; you can expect to be given more information than is necessary to answer the question. Writing down each item makes it easier to pick out the salient points and ignore the rest.

Still having trouble with the scenario? Try approaching it from the other direction. Forget the answers given. Instead, think about what you would do in a similar situation. Oftentimes if you use this method, you will find that the question is really not that hard.

Reviewing Marked Questions

After you have read all the questions and answered them, go back and review those questions you have marked. When you go through the question for the second time, you can take more time. Analyze the question and pick out the most important points. What is the objective of the question? What information is provided that is not relevant to answering it?

When reading the questions, remember not to read anything into them. Rather, take each question at face value. Don't assume anything that is not explicitly mentioned in the question itself.

Remember, the people who write the questions are not trying to trick you; they're just trying to provide an objective measurement of your skills in relation to the exam's stated objectives.

APPENDIX

D

Alternate Resources

Hundreds of resources exist online for information about and help with Linux. A few are listed here to get you started—you'll surely discover more as you surf the Net.

Support Sites

CNET.com—Linux: `http://linux.cnet.com/linux`

Filesystem Hierarchy Standard: `http://www.pathname.com/fhs/`

Linux Fool Dot Com: `http://www.linuxfool.com/`

Partition types: `http://www.win.tue.nl/~aeb/partitions/partition_types-1.html`

Planet IT Linux Technology Center:
`http://www.planetit.com/techcenters/linux`

System Administrator's Guide:
`http://members.xoom.com/basiclinux/sag/book1.html`

TechRepublic: `http://techrepublic.com/trbbs/trbbs.jhtml`

Documentation

Linux Documentation Project: `http://metalab.unc.edu/LDP/`

Linux Knowledge Base Org.: `http://www.linuxkb.org/`

Linux.com—HOWTOs: `http://linux.com/howto/`

Software Sites

Bash prompts: `http://bash.current.nu/`

Freshmeat: `http://freshmeat.net`

FtpFind: `http://www.ftpfind.com/`

Linux RPM search: `http://www.whichrpm.com/`

Linux software: `http://linux.tucows.com`

Linux News

Linux Gazette: `http://www.linuxgazette.com/`

Slashdot: `http://www.slashdot.org/`

Humor

Humorix—All Linux Humor: `http://i-want-a-website.com/about-linux/`

User Friendly the Comic Strip: `http://www.userfriendly.org/`

Tutorials

Basic Linux: `http://basiclinux.hypermart.net`

Linux OpenContent Stuff: `http://w3.one.net/~johnb/linux/`

Miscellaneous

Linux User Groups WorldWide: `http://lugww.counter.li.org/`

LPI—Getting Certified: `http://www.lpi.org/`

Linus Torvalds: `http://www.cs.helsinki.fi/~torvalds/`

The EZLinux Command Card:
`http://www.csvtech.com/aff.pl?page=downloads/command/index.html&aff_id=0`

APPENDIX

E

Using the CD-ROM

The CD-ROM accompanying this book contains 250 questions that comprise the practice tests covering the material you have studied. These questions should be used when you have finished the book and think you are ready to take the exam.

All the questions presented here are multiple choice or short answers. When typing in your answers, use lowercase unless a command requires uppercase. These are the same types of questions you will see on the exam.

Equipment Requirements

To use the Self Tests, you must have an Intel-based computer running the Linux operating system. The distribution does not matter. The test is designed to run under the command-line interface; therefore, it is not guaranteed to run correctly in a terminal window under X.

Installing the Self-Test Software

Installation consists of creating a directory and copying the appropriate files from the CD-ROM to this directory. The steps to do this are as follows:

1. Create a directory to contain the testing files. Usually, this should be a subdirectory of your home directory.
2. Copy the following files into this directory:
   - `lpic`
   - `test`

3. While this directory is your working directory, first use the `su` command to become root and then change the ownership of these files by issuing the following command:

```
chown <your name> *
```

4. Type `exit` to quit being root. Change the `a.out` file to be executable by issuing the following command:

```
chmod a+x lpic
```

5. Verify that you have the permission to read the `test` file by using the `ls -l` command. If you need to change it then issue this command:

```
chmod a+r test
```

Running the Self-Test Software

To run the Self-Test Software, make the directory containing the files your working directory and then type

```
./lpic
```

When you start the test, you will be presented with a screen in which you can select which test you want to take. Tests 1–5 each randomly present 50 questions from a static pool of questions. After you finish all five tests, you will have seen all the questions. Test 6 presents you with 50 questions randomly selected from all the questions. This will be very similar to taking the real exam.

Make sure your Caps Lock key is *not* on while taking this test. All answers are case sensitive, so your answer might be scored as incorrect if you use capital letters.

During the test, you can quit at any time by typing the letter `q` and pressing Enter. You then will be given the chance to score your test or quit.

To answer a question, type your answer and then press Enter to proceed to the next question. If you want to come back to this question later, type the letter `M` and press Enter. (Notice that it must be a capital `M`.)

Scoring Your Test

When you finish a test, you will be presented with your overall score. In addition, your score will be broken down according to objectives. You should use this information as a guide to further study.

For any questions you answered incorrectly, you will be presented with the question as well as an explanation of the correct answer. The appropriate place in the book discussing this material also will be listed. If you miss a question, take the time to determine why you got the answer wrong.

Technical Support

Like any application, surprises occasionally can occur. Every effort has been made to ensure that this application will run on any distribution of Linux. If any problems should occur, please email a detailed description of what happened to `ronmart@ibm.net`. You also can use this method if you have any technical questions about this application.

We hope that this exam will be a satisfying experience and wish you the best of luck in your certification quest.

APPENDIX

F

Lab Exercises

Equipment Requirements

In order to complete these exercises you will need a computer running any version of Linux. You need to have an account and be able to log in to the computer. These exercises are designed to be run from a command line. If you are running X Window then perform these from a terminal window.

Lab 1: Using Local Documentation

In this lab, you will learn how to use the documentation that was installed on your computer when Linux was installed.

Using the man Pages

After logging in to the computer, perform a listing of the `/usr/man` directory. Notice that there are several subdirectories of the `/usr/man` directory. Each of these refers to a class of documentation. Refer to Chapter 1, "Finding and Creating Documentation," for more details on what is contained in each of these directories.

At the command prompt, type `man man`. This will display the manual page that covers how to use and configure your man pages. Notice the section that discusses how to change your configuration. This is controlled by the file `/etc/man.config`, which might also be called `/etc/man.conf`.

Review your configuration file and see what the `manpath` is. Depending on what applications are installed on your computer, more than one directory might be included on this path.

This file also contains the configuration for the pager used to display the documents and the order in which the man pages are searched. You can change either one of these if you want.

Try changing your pager. If it is defined as `less`, configure it to use `more`. One way of doing this is to comment out the line beginning with `PAGER` and then add an identical line under it with the exception of the command that is listed. After making your changes, issue the `man man` command again. Do you see any changes?

Compare the output from the following two commands:

```
man -k password
man -K password
```

What is the difference between the two? Notice that typing `man -k password` displays a list of commands or utilities and their short descriptions. Each of these contain the word `password` in the description.

The `man -K password` command displays the complete path and command name and prompts you to decide whether this is the page you want to view. Your choices are `y`, `n`, and `q`. Type `y` for yes, `n` for no, or `q` to quit. Then, press Enter.

Both the `-k` and `-K` options will cause the `man` command to search the description. If you use the `-a` option, it will display each page that matches that command. Type `man -a write`. Each page that covers `write` will be displayed sequentially.

Other Documentation

A method of documentation that is easier to understand is the `info` command. Type `info info <enter>` at a command prompt. Use the Tab key to highlight the `Getting Started` menu item and press Enter. Spend some time navigating around this screen to become more comfortable with this utility.

Make the `/usr/doc/HOWTO` directory your working directory. Examine the listing of files that are there. These documents are helpful when working on a specific task. Now type `less Tips-HOWTO` at a command line to review this document.

Now make the `/usr/doc/FAQ` your working directory. Look through the documents available here. They will come in handy in the future. You can use either the `less` or `more` pager to look at any of these documents, as well.

Next, you will want to look at the `/usr/doc` directory to see what application documentation has been installed. If you ever have a question about a program on your computer, check here first.

Each application that you install will install a man page and usually a directory under `/usr/doc`. This directory is named for the application and version that contains files you will use for configuration, instructions for using the application, and troubleshooting information.

Searching for Information on Commands

Several commands can be used for getting additional help on commands. Each command provides different information. Type the following commands and see the difference in output:

```
which sort
whatis sort
apropos sort
```

Lab 2: Booting

Understanding the boot process is important, especially when troubleshooting. You should know what happens and how to customize this process.

Boot Messages

You can examine the system messages by using the `dmesg` command. Type `dmesg` at a command prompt immediately after booting your system. Look at the display to see whether you have any error messages. Also look to see what devices were loaded at boot.

inittab

Look at the `/etc/inittab` file. What are the defined runlevels for your system? What is your default runlevel? Which line defines it? This file is fairly straightforward. If you have problems answering these questions, review Chapter 2, "Booting Linux."

Type `init S` at a command prompt. This changes the runlevel to single-user mode. Now use the `init` command to return to your default runlevel.

What are the default scripts that are run at boot? How did you identify them? Look at the `rc` scripts to identify which are run when you boot.

Edit the `/etc/inittab` file to change your runlevel. Be sure to make a copy of the original before making any changes. Now issue the `init q` command to force `init` to reread the configuration file.

Be sure to change back to your original runlevel unless you want the default runlevel to stay changed.

LILO

If your computer is using LILO as your boot loader, look at your configuration file. The default is `/etc/lilo.conf`. Do you have any other operating systems installed? If so, do you use LILO to start them?

Edit your `/etc/lilo.conf` file to change the label on your default operating system and reboot your computer. When the `BOOT:` prompt appears, press the Tab key to see your choices. Type the new label to start your system.

Shutting Down Your System

Issue the `shutdown -k` command. What happened? You should see a warning that the system is being shut down, but it will stay up.

Is your computer configured to trap the Ctrl+Alt+Del key combination? Where did you look to find this out? If your computer is configured to trap this combination, you should see it in the `/etc/inittab` file.

Lab 3: Disk Space

This exercise is designed to make you comfortable with your partitioning scheme. In addition, you will learn how to monitor your disk space use.

Use the `fdisk` command to see how your disk is partitioned. Type `fdisk` at a command prompt. When you get the `fdisk` prompt, type `p` to get a listing of your partitions. Do you understand the partition naming scheme? If not, review Chapter 3, "Linux Filesystem."

If you have available free space, create a new partition in this space using the `fdisk` command. Be very careful that you do not destroy any existing data.

Format a floppy disk by using the `fdformat /dev/fd0` command. Save a file to it and then read the file. You might need to be logged in as root depending on how your system is configured.

Perform a listing of the `/proc` directory. What do you see? Remember that this is a virtual directory and is a method for you to access the information in memory. You should be able to identify the amount of RAM in your system from this listing.

Type `du` at a command prompt. What information is displayed? Also try using the `df` command. What is the difference between the two? The `du` command shows occupied space, and the `df` command shows size and usage by partition.

Lab 4: Navigating Directories and Files

It is important to understand how to move around the directory structure. You also need to know how to examine the contents of files.

Navigation

Use the `cd` command to change to the `/usr/bin` directory. While here use the `ls` command to ascertain what type of files are contained in this directory. These are executable files that are available for any user to use.

Use the `ls -l`, `ls -a`, and `ls -R` commands. Notice the difference between each listing. The `ls -l` command shows the file attributes; `ls -a` shows all files, including hidden ones; and `ls -R` recursively lists the contents of all the subdirectories.

Next, type `ls -l new*`. What is displayed this time? This shows only the files that begin with the letters `new`. The asterisk (*) represents any other characters that might follow these letters.

Now, type just `cd` and `enter`. Where are you now? This command returns you to your home directory. You also can use the tilde (~) to return to your home directory by typing `cd ~`.

Examining File Contents

Now type `cat > myfile`. What happened to your prompt? You are now using the `cat` command to create a new file. Type your address and press Enter after each line. Then type `Ctrl-D` to close the file.

Type `cat myfile` to display the contents of this new file. Next, type the following:

```
cat > myfile
Ctrl-D
cat myfile
```

Now the previous contents of the file are gone because you overwrote the file. Re-create the file containing your address, and then type the following:

```
cat >> myfile
My new address file
Ctrl-D
cat myfile
```

This time you appended information to your file. You also can use `cat` to combine files. Type the following:

```
cat > file1
```

```
This is file1
Ctrl-D
cat > file2
This is file2
Ctrl-D
cat > file3
This is file3
Ctrl-D
cat file1 file2 file3 > file4
cat file4
```

When you display `file4` you will see the contents from the three original files.

Type the following, noting the differences in output:

```
wc -c file4
wc -w file4
wc -l file4
wc file4
```

This displays the number of characters, words, and lines contained in `file4`.

Next, type `tail -9 /var/log/messages`. This will display the last nine lines of the log file containing system messages. Compare this output to `tail -f /var/log/messages`. What is the difference? The second command keeps the file open and displays each new message that is written to the log.

Manipulating Directories

Type the following commands:

```
ls -l file4
touch -a file4
ls -l file4
```

Notice the timestamp on `file4` and how it changes.

Type the following commands:

```
mkdir filedir
cp fil* filedir
ls filedir
```

This creates the directory called `filedir` and then copies all the files starting with `fil` to this new directory. The last line shows that the files were copied to the new directory.

Now type

```
mv fil* filedir
```

How does this differ from the earlier command? Rather than making a copy, the files are relocated to the `filedir` directory. Were you prompted to verify that you wanted to move each file?

Now type

```
rmdir filedir
```

What happened? You should have received a message that the directory was not empty. To remove the directory, either delete all its contents first or type `rm -r filedir` and respond `y` to each prompt.

Searching Files

Copy the file `searching` from the CD-ROM to your home directory. At a command prompt type `grep house`. Compare this output to the output produced by the command `grep ^house`. The first output displays all the lines that contain `house`. The second displays only the lines that start with `house`. Use `grep` to display all the lines that do not contain house.

Compare the output of the commands `sed /house/p searching` and `sed -n /house/p searching`. The first will print the entire file with those lines containing `house` appearing twice. The second displays only the lines containing `house`.

With your home directory as your working directory, type `tar cf MyArchive.tar .` This creates a tarfile containing all the files in your home directory. Next, compress the file using `gzip` by typing `gzip MyArchive.tar`. After doing this, type `ls My*`. What do you see? The tarfile itself is no longer there. Instead, you have a compressed file called `MyArchive.tar.gz`.

Lab 5: Managing Your Environment

One of the biggest strengths of using Linux is the ability to customize everything. You will practice changing your environment to reflect your personal preferences.

Environment Variables

What shell are you using? Type `echo $SHELL` to determine what it is. What shells are available on your system? Type `cat /etc/shells` to see a listing of which ones are available.

The `inputrc` or `.inputrc` file can contain key mappings. Assign the `F1` key to insert your name by adding the following to your `.inputrc` file:

```
"\e[11~": "Your Name"
```

Use the `bind -v` command to verify that this was successful. Did you see it? What went wrong? You need to force the system to reread the `.inputrc` file. Issue the command `source .inputrc` or `. .inputrc`. Now check to see whether the key binding was successful.

At a command prompt, type `gr <Tab>`. What do you see? The system should display any available commands that start with `gr`. Now type `grep sear <Tab>`. What happened? The filename `searching` should now appear after `grep`.

Type either `env` or `printenv` at a command prompt. What is displayed? These are your environment variables and their values. What is the value for `PATH`? For `PS1`?

Change your prompt to include the date and time by issuing the following command:

```
PS1='Date: \d Time: \t-> '; export PS1
```

Did your prompt change? Why did you need to export the variable? This makes the variable available for all future processes.

Type `history 10`. What is displayed? You should see a listing of the last 10 commands that you have entered. Use the up arrow key to scroll through your history list. When you see the command to create the `filedir` directory, press the Enter key. Issue the `ls` command to verify that the directory was created.

Processes

Issue the commands `ps ef` and `ps aux`. What is the difference between the two displays? The first command displays your processes. The second displays all running processes along with the user's name.

Use the `top` command to display processes. Type `h` to display the help screen. Now sort the processes by CPU usage. Use `P` after exiting the help screen. Which process is listed first?

Type `pine &`. What does this command do? It starts pine in the background. Use the `jobs` command to verify this. Now bring `pine` to the foreground. What command did you use? You should have used the `fg` command with the job number.

Lab 6: Filesystems

Check to see what filesystems are defined in your `/etc/fstab` file. How did you do this? What filesystems are mounted by default? What filesystems are mountable by a user? You can use the `cat` command to display the `/etc/fstab` file. Look at the options to determine which filesystems are user mountable.

Use the cat command to display the contents of the /etc/mtab file. How is this file different from the /etc/fstab file? How is it different from the output of the mount command? The /etc/mtab file lists currently mounted filesystems and is used by the mount command, whereas the /etc/fstab defines mountable filesystems.

Use the mount command to mount a floppy disk. Check the /etc/fstab file for the mount point for your floppy. If it is not user mountable, log on as root.

Quotas

Copy the file searching to the floppy, and then use ls to verify that the copy operation succeeded. What one thing do you need to do before removing the floppy? Be sure you unmount it first.

Set up disk quotas for one of your users. The steps are

1. While logged in as root, create a file called quota.user in the root of the filesystem where you want the quotas to be enforced.
2. Issue the command chmod 600 /quota.user.
3. Edit the /etc/fstab file and add the quota option to the appropriate filesystem.
4. Reboot the computer.
5. Use the edquota command to set the appropriate limits:

```
Quotas for user username:
/dev/hda5: blocks in use: 52, limits (soft = 3000, hard = 5000)
      inodes in use: 32, limits (soft = 500, hard = 1000)
```

6. Set the grace period to one day by issuing the command eduquota -t *username*.
7. Use the repquota command to see the quota status.

Links

Enter the following commands:

```
touch files
ln files filelnk
ln -s files fileslnk
ls -i file*
ls -l file*
```

What did the preceding commands do? What do you notice about the output from the ls command?

First, you created a file called `files`. Next, you created a hard link to this file and then a symbolic link to it. When you look at the listing, both the original file and `filelnk` have the same inode, but `fileslnk` does not. Also, the attributes for `files` and `filelnk` are the same; that is not the case with `fileslnk`.

Enter the following two commands:

```
find . -name '*fil*' -a '*lnk*'
find . -name '*fil*' -o '*lnk*'
```

What is the difference between the two? The first command found the files `filelnk` and `fileslnk` because they contain both `fil` and `lnk`. The second command found `files`, `filelnk`, and `fileslnk` because the search criteria were `fil` or `lnk`.

Type the command `locate fil*`. Did you get any output? If not, why?

Because these files were recently created, they will not be in the `locate` database. Run the `updatedb` command and then try again. Was the operation successful this time?

Lab 7: Users and Groups

In this exercise you will create some new users and groups using different methods.

The */etc/passwd* and */etc/groups* Files

One way of creating new accounts is by manually editing the appropriate files. First, create a new user account for Kim by editing the `/etc/passwd` file. Next, set the password for Kim and create a home directory.

Make a backup copy of the `/etc/passwd` file before making any changes. Add a new line containing the username, leave the password field blank, UID, GID, Comments (may be blank), and login command (default shell). Your line might look like

```
kim::67:51::/home/kim:/bin/bash
```

The UID and GID for your system might be different. The UID should be unique. The GID must be for a group that exists on your system before Kim tries to log in. You will have to create the home directory. Use the shell of your choice for the login command.

Next, assign a password to this account by entering the command `passwd kim` while logged in as root.

You will need to create Kim's home directory by typing `mkdir /home/kim`. Copy any files that need to be available for Kim and change the ownership by entering `chown -R /home/kim kim`. This will change all existing files and directories.

Before Kim can log in, you need to be sure that the default group exists. If you make the default group one that exists, Kim can now log in. If not, then you will have to edit the `/etc/group` file to create the group, making sure you use the appropriate GID.

Create a new group to use as Kim's default group. Do this by adding a line to the `/etc/group` file, such as

```
kimgroup::51:kim
```

The first field is the name of the group and the next field is the group password, which you should leave blank. Be sure the GID is unique and matches the one you entered in the `/etc/passwd` file.

Now create a user account for Joe using the `useradd` or `adduser` command. Type

```
useradd -m joe
```

This will create the account and home directory. However, you will still need to assign a password.

To make your system more secure, you will want to use shadow passwords. Implement them on your system by using the `pwconv` command for user passwords and `grpconv` command for group passwords. Neither of these commands takes any arguments.

While logged in as root, look for the files `/etc/shadow` and `/etc/gshadow`. These indicate that the operation was successful. Also, type `cat /etc/passwd`. What do you see? The password field has been changed to `x`, indicating that shadow passwords are in effect.

Lab 8: Text Streams

Use `sed` to change `rabbit` to `hare` in the file called `searching`. So, how did you do it? If you use `cat` to look at the file now, which term is listed?

You can enter the command `sed s/rabbit/hare/g searching` to change the term. However, this change is only on the display; the original file is not changed.

Type the following at a command line:

```
cat > fruit
tomato
pear
apple
peach
cherry
Ctrl-D
```

Now alphabetize the file `fruit` that you just created. Save your output to another file called `fruit2`.

To sort the file in alphabetical order, use the `sort fruit` command. However, this outputs the results to the display. To save it to another file, you will need to pipe the output to the new file. So, the command you need is `sort fruit > fruit2`.

Display the second, third, and fourth characters of each line of the file `fruit` to the display. You can accomplish this using the `cut` command. Type `cut -c2-4 fruit`.

Now paste the two files `fruit` and `fruit2` together so that each line matches:

```
tomato apple
pear   cherry
apple  peach
peach  pear
cherry tomato
```

You can do this using the `paste` command:

```
paste fruit fruit2
```

This will send the output to the display. You also could redirect this output to a file if you wanted to save it.

Change the file `fruit` so that it is displayed as a single line with a hyphen between each word. This can be accomplished with the `tr` utility by typing `cat fruit | tr '/n' '-'`. This command changes the newline character to a hyphen.

Lab 9: Permissions

Issue the command `ls -l *fil*`. What are the permissions on the files? On the directory? Who is the owner of the files and the directory?

Because you created them, these files and the directory will have you listed as the owner. The permissions can vary depending on how your system is configured. Check your creation mask by issuing the `umask` command. Does it match with the permissions on the files?

Issue the command `chmod 4755 files`. How did this change the permissions on this file? What is the new permission block? This command gives the owner read, write, and execute permissions; the group and users get read and execute permissions. In addition, this sets the user ID.

While logged in as root, create a directory to be shared by all users. Make sure that each user can create files in this directory and delete only her own files. What permissions did you apply to this directory?

You will want to set the sticky bit on the directory to prevent users from deleting each other's files. This can be accomplished by entering the command `chmod 1777 DirName`.

Lab 10: System Administrative Tasks

You will perform many tasks as a system administrator. Many of these will involve repetitive tasks. These exercises will cover automating these tasks.

Scheduling Tasks

Use the `at` command to run a job. Type `at time` at a command prompt. Use a time that is about 30 minutes from now. After pressing the Enter key, you will be at the `at` prompt, where you should type a command. Try `ls -l > MyList`.

What message did you get? If the command was successful, great. However, if you got an error message what do you think was the problem? Check for the files `/etc/at.allow` and `/etc/at.deny`. If the latter does not exist, create it.

If the operation was successful, type either `atq` or `at -l` to see what jobs are scheduled. Did you see your job listed?

You also can use `crontab` to schedule jobs. First, create a file containing the commands you want to run. Next, you will use `crontab` to install the job. The format for each line is

```
min hr day-of-month month day-of-week commands
```

Next, install your cron job by issuing the command `crontab filename`. Did you get an error message? If so, check for the `/etc/cron.allow` and `/etc/cron.deny` files. If neither exists, create the second one.

To verify that your cron job was properly installed, type `crontab -l`. After the job has run you will receive a mail message notifying you.

Logging

Review the `/etc/syslog.conf` file. What logs are configured in this file? What type of messages are being logged to each one? Look at these logs. Is there any information there that you should be notified of when they are generated? How could you make this happen?

Edit the `/etc/syslog.conf` file to add the following line (if this line is not already there):

```
.emerg                        *
```

APP F

This will send messages to the console. However, you will need to notify the `syslogd` daemon of the changes. How can you do this?

You can reboot your computer in order to force the `syslogd` daemon to reread the configuration file. However, a better way would be to enter the command

```
kill -SIGHUP 'cat /var/run/syslogd.pid'
```

Now any emergency messages will be sent to the console.

Look at the contents of the directory `/var/log`. Do you see older log files being saved here? How is this being done? Chances are this is the result of the `logrotate` command.

Check out the `/etc/logrotate.conf` file. What logs are being rotated? How frequently? How many old logs are being kept? Many distributions also install a cron job to run `logrotate`. Look in the `/etc/cron.daily` directory for the `logrotate` job. You can view it using `cat` or a text editor. Just be careful not to make any changes.

Lab 11: Backup and Recovery

For this exercise, you will devise a backup strategy and implement it. You also will perform trial restores to ensure that your plan will work in case of a disaster.

First, answer the following three questions:

- What are you going to back up?
- How often will you back up?
- What media will you use for your backup?

The answers will vary from system to system. If you have a single-user system, you will probably back up the entire system whenever you make changes and back up your files on a daily basis. If it is a multiuser system, however, you will want to make complete backups on a regular basis as well as daily backups of user files.

For this exercise, you will create a backup of your home directory on a daily basis. The first step is to construct and test the commands necessary to create your backup.

For this you will use `tar` to create a compressed archive. You will save this to your home directory. Later you will want to modify this to save the archive to your backup device, such as a tape drive.

Type the following:

```
cd
tar czf MyBackup.tar.gz .
```

After you have the command line the way you want it, you need to create a file containing this command to be used to install it as a cron job. Using a text editor, create the file `backups` to contain the following line:

```
45 23 * * * tar czf MyBackup.tar.gz .
```

This will create an archive every night at 11:45 pm. If you want to save these, remember to move it before it is overwritten. You could add a command to do this in your cron job.

Now that you have your backup, you will want to test that you can restore files from it. Be sure that you are in your home directory when extracting files so they will be saved to the correct location. You also might want to use the `w` option to place `tar` in interactive mode so you can verify each file you want restored.

To select the file to extract from your archive, type the command `tar xzwf MyBackup.tar.gz`. This will prompt you for each file stored in the archive. To quit, press Ctrl-C.

APPENDIX

G

Objectives Index

Objective	*Subobjective*	*Page*
3. GNU and UNIX Commands	1. Work Effectively on UNIX Commands Interact with shells and commands using the command line. Includes typing valid commands and command sequences; defining, referencing, and exporting environment variables; using command history and editing facilities; invoking commands inside and outside the path; using command substitution; and applying commands recursively through a directory tree.	104-116
	2. Process Text Streams Using Text Processing Filters Send text files and output streams through text utility filters to modify the output in a useful way. Includes the use of standard UNIX commands found in the GNU textutils package such as `sed`, `sort`, `cut`, `expand`, `fmt`, `head`, `join`, `nl`, `od`, `paste`, `pr`, `split`, `tac`, `tail`, `tr`, and `wc`.	186-197
	3. Perform Basic FileManagement Use basic UNIX commands to copy and move files and directories. Perform advanced file management operations such as copying multiple files recursively and moving files that meet a wildcard pattern. Use simple and advanced wildcard specifications to refer to files.	74-97

	5. Use File Permissions to Control Access to Files Set permissions on files, directories, and special files; use special permission modes such as suid and sticky bit; use the group field to grant file access to workgroups; and change default file creation mode. Includes the `chmod` and `umask` commands. Requires an understanding of symbolic and numeric permissions.	210-215
	6. Manage File Ownership Change the owner or group for a file and control what group is assigned to new files created in a directory. Includes the `chown` and `chgrp` commands.	208-210
	7. Create and Change Hard and Symbolic Links Create hard and symbolic links, identify the hard links to a file, copy files by following or not following symbolic links, and use hard and symbolic links for efficient system administration.	139-142
	8. Find System Files and Place Files in the Correct Location Understand the filesystem hierarchy standard, know standard file locations, know the purpose of various system directories, and find commands and files. Involves using the `find`, `locate`, `which`, and `updatedb` commands. Involves editing the `/etc/updatedb.conf` file.	142-145
6. Boot, Initialization, Shutdown, Runlevels	1. Boot the System Guide the system through the booting process, including giving options to the kernel at boot time, and check the events in the log files. Involves using the `dmesg` (LILO) command. Involves reviewing the `/var/log/messages`, `/etc/lilo.conf`, `/etc/conf.modules`, and `/etc/modules.conf` files.	28-37
	2: Change Runlevels and Shut Down or Reboot System Securely change the runlevel of the system, specifically to single-user mode; halt (shut down) the system; or reboot the system. Make sure to alert users beforehand and properly terminate processes. Involves using the `shutdown` and `init` commands.	30-40

APP G

8. Documentation	1. Use and Manage Local System Documentation Use and administer the man facility and the material in `/usr/doc/`. Includes finding relevant man pages, s earching man page sections, finding commands and man pages related to the command, configuring access to man sources and the man system, using system documentation stored in `/usr/doc/` and related places, and determining what documentation to keep in `/usr/doc/`.	8-18
	2. Find Linux Documentation on the Internet Find and use Linux documentation at sources such as the Linux Documentation Project, vendor and third-party Web sites, newsgroups, newsgroup archives, and mailing lists.	16-18
	3. Write System Documentation Write documentation and maintain logs for local conventions, procedures, configurations, configuration changes, file locations, applications, and shell scripts.	20-21
	4. Provide User Support Provide technical assistance to users over the telephone, through email, and through personal contact.	21-22
11. Administrative Tasks	1. Manage Users and Group Accounts and Related System Files Add, remove, and suspend user accounts; add and remove groups; change user and group information in passwd and group databases; and create special-purpose and limited accounts. Includes using the `useradd`, `userdel`, `groupadd`, `gpasswd`, `passwd`, `filepasswd`, `group`, `shadow`, and `gshadow` commands.	156-164
	2. Tune the User Environment and System Environment Variables Modify global and user profiles to set environment variables, maintain skel directories for new user accounts, and place proper commands in the path. Involves editing `/etc/profile` and `/etc/skel/`.	174-177
	3. Configure and Use System Log Files to Meet Administrative and Security Needs Configure the type and level of information logged, manually scan log files for notable activity, arrange for automatic rotation and archiving of logs, and track down problems noted in logs. Involves editing `/etc/syslog.conf`.	239-243

	4. Automate System Administration Tasks by Scheduling Jobs to Run in the Future Use `cron` to run jobs at regular intervals, use `at` to run jobs at a specific time, manage `cron` and `at` jobs, and configure user access to `cron` and `at` services.	232-239
	5. Maintain an Effective Data Backup Strategy Plan a backup strategy, back up filesystems automatically to various media, perform partial and manual backups, verify the integrity of backup files, and partially or fully restore backups.	255-280

INDEX

Symbols

A

B

G

H

I

J-K

M

N-O

Q

R

S

U

NOTE: This CD-ROM uses long and mixed-case filenames requiring the use of a protected-mode CD-ROM driver.

Linux and UNIX Installation Instructions

These installation instructions assume that you have a passing familiarity with UNIX commands and the basic setup of your machine. Because UNIX has many flavors, only generic commands are used. If you have any problems with the commands, please consult the appropriate man page or your system administrator.

1. Insert CD-ROM in CD drive.
2. If you have a volume manager, mounting of the CD-ROM will be automatic. If you don't have a volume manager, you can mount the CD-ROM by typing

```
Mount  -tiso9660  /dev/cdrom  /mnt/cdrom
```

NOTE: `/mnt/cdrom` is just a mount point, but it must exist when you issue the mount command. You may also use any empty directory for a mount point if you don't want to use `/mnt/cdrom`.