Andrew Hudson
Paul Hudson
Matthew Helmke
Ryan Troy

Ubuntu

UNLEASHED

2010 Edition

Covering 9.10 and 10.4

SAMS

Ubuntu Unleashed 2010 Edition: Covering 9.10 and 10.4

ISBN-13: 978-0-67233-109-1
ISBN-10: 0-67233-109-8

Library of Congress Cataloging-in-Publication data is on file.

Printed in the United States of America

First Printing December 2009

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the CD or programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact:

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact:

International Sales
international@pearson.com

Acquisitions Editor
Debra Williams Cauley

Development Editor
Michael Thurston

Managing Editor
Patrick Kanouse

Project Editor
Jennifer Gallant

Copy Editor
Paula Lowell

Indexer
Tim Wright

Proofreader
Leslie Joseph

Technical Editor
Dallas Releford

Publishing Coordinator
Kim Boedigheimer

Multimedia Developer
Dan Scherf

Cover and Interior Designer
Gary Adair

Composition
Mark Shirar

Introduction

Welcome to *Ubuntu Unleashed, 2010 Edition!* This book covers the free Linux distribution named Ubuntu and includes a fully functional and complete operating system produced by the Ubuntu Community, sponsored by Canonical Software. This book covers Ubuntu version 9.10.

Ubuntu directly descends from one of the oldest and most revered Linux distributions ever: Debian. Those of you who know nothing about Linux will likely not have heard of Debian; it is enough to know that it is considered to be one of the most stable and secure Linux distributions currently available. Ubuntu benefits directly from many contributions from free software developers across the world.

If you are new to Linux, you have made a great decision by choosing this book. Sams Publishing's *Unleashed* books offer an in-depth look at their subject, taking in both beginner and advanced users and moving them to a new level of knowledge and expertise. Ubuntu is a fast-changing distribution that can be updated at least twice a year. We have tracked the development of Ubuntu from early on to make sure that the information in this book mirrors closely the development of the distribution. A full copy of Ubuntu is included on the enclosed disc, making it possible for you to install Linux in less than an hour! No longer an upstart, Linux now has an enviable position in today's modern computing world. It can be found on machines as diverse as mobile phones and wristwatches, all the way up to supercomputers—in fact, Linux currently runs on more than half of the world's top 500 supercomputers.

Do not let the reputation of Linux discourage you, however. Most people who have heard of Linux think that it is found only on servers, looking after websites and email. Nothing could be further from the truth because Linux is making huge inroads in to the desktop market, too. Corporations are realizing the benefits of running a stable and powerful operating system that is easy to maintain and easy to secure. Add to that the hundreds of improvements in usability, and Linux becomes an attractive proposition that tempts many CIOs. The best part is that as large Linux vendors improve Linux, the majority of those improvements make it into freely available distributions, allowing you to benefit from the additions and refinements made. You can put Ubuntu to work today and be assured of a great user experience.

This book provides all the information that you need to get up and running with Ubuntu. It even tells you how to keep Ubuntu running in top shape and how to adapt Ubuntu to changes in your own needs. You can use Ubuntu at home, in the workplace, or, with permission, at your school or college. In fact, you might want to poke around your school's computer rooms: You will probably find that someone has already beaten you to the punch—Linux is commonly found in academic institutions. Feel free to make as many copies of the software as you want; because Ubuntu is freely distributable all over the world, no copyright lawyers are going to pound on your door.

After an introduction to Linux and Ubuntu, you will find out how to get started with Ubuntu, including installation and initial configuration. We also take you through installing software, managing users, and other common administrative tasks. For the more technically minded, we also cover some starting steps in programming across several languages—why not pick one and try it out? Throughout this book, you will also find information about multimedia applications, digital graphics, and even gaming (for after-hours when you are finished tinkering). After you make it through this book, you will be well equipped with the knowledge needed to use Linux successfully. We do assume that you are at least familiar with an operating system already (even if it is not with Linux) and have some basic computer knowledge.

Licensing

Software licensing is an important issue for all computer users and can entail moral, legal, and financial considerations. Many consumers think that purchasing a copy of a commercial or proprietary operating system, productivity application, utility, or game conveys ownership, but this is not true. In the majority of cases, the *end user license agreement (EULA)* included with a commercial software package states that you have paid only for the right to use the software according to specific terms. This generally means you may not examine, make copies, share, resell, or transfer ownership of the software package. More onerous software licenses enforce terms that preclude you from distributing or publishing comparative performance reviews of the software. Even more insidious licensing schemes (and supporting legislation, especially in the United States) contain provisions allowing onsite auditing of the software's use!

This is not the case with the software included with this book. You are entirely free to make copies, share them with friends, and install the software on as many computers as you want—we encourage you to purchase additional copies of this book to give as gifts, however. Be sure to read the README file on the disc included with this book for important information regarding the included software and disk contents. After you install Ubuntu, go to http://www.gnu.org/licenses/gpl.html to find a copy of the GNU GPL. You will see that the GPL provides unrestricted freedom to use, duplicate, share, study, modify, improve, and even sell the software.

You can put your copy of Ubuntu to work right away in your home or at your place of business without worrying about software licensing, per-seat workstation or client licenses, software auditing, royalty payments, or any other type of payments to third parties. However, be aware that although much of the software included with Ubuntu is licensed under the GPL, some packages on this book's disc are licensed under other terms. There is a variety of related software licenses, and many software packages fall under a broad definition known as *open source*. Some of these include the Artistic License, the BSD License, the Mozilla Public License, and the Q Public License.

For additional information about the various GNU software licenses, browse to http://www.gnu.org/. For a definition of open-source and licensing guidelines, along with links to the terms of nearly three dozen open-source licenses, browse to http://www.opensource.org/.

Who This Book Is For

This book is for anyone searching for guidance on using Ubuntu and primarily focuses on Intel-based PC platforms. Although the contents are aimed at intermediate to advanced users, even new users with a bit of computer savvy will benefit from the advice, tips, tricks, traps, and techniques presented in each chapter. Pointers to more detailed or related information are also provided at the end of each chapter.

If you are new to Linux, you might need to learn some new computer skills, such as how to research your computer's hardware, how to partition a hard drive, and (occasionally) how to use a command line. This book helps you learn these skills and shows you how to learn more about your computer, Linux, and the software included with Ubuntu. System administrators with experience using other operating systems can use the information in this book to install, set up, and run common Linux software services, such as the *Network File System (NFS)*, a *File Transfer Protocol (FTP)* server, and a web server (using Apache, among others).

What This Book Contains

Ubuntu Unleashed is organized into seven parts, covering installation and configuration, Ubuntu on the desktop, system administration, programming and housekeeping, and a reference section. A disc containing the entire distribution is included so that you have everything you need to get started. This book starts by covering the initial and essential tasks required to get Ubuntu installed and running on a target system.

If you are new to Linux, and more specifically Ubuntu, first read the chapters in Part I, "Installation and Configuration." You will get valuable information on the following:

- Detailed steps that walk you through installation
- Critical advice on key configuration steps to fully install and configure Linux to work with your system's subsystems or peripherals, such as pointers, keyboards, modems, USB devices and power management
- Initial steps needed by new users transitioning from other computing environments
- Working with GNOME, the default desktop environment for Ubuntu

Part II, "Desktop Ubuntu," is aimed at users who want to get productive with Ubuntu and covers the following:

- Discovering the many productivity applications that come with Ubuntu
- Surfing the Internet and working with email and newsgroups
- Using Ubuntu to listen to music and watch video
- Using Ubuntu to download and manipulate images from digital cameras
- Setting up local printers for Ubuntu
- Understanding the current state of gaming for Linux

Moving beyond the productivity and desktop areas of Ubuntu, Part III, "System Administration," covers the following:

- Managing users and groups
- Automating tasks and using shell scripts
- Monitoring system resources and availability
- Backup strategies and software
- Network connectivity, including sharing folders and securing the network
- Internet connectivity via dial-up and broadband connections

Part IV, "Ubuntu as a Server" gives you the information you need to start building your own file, web and other servers for use in your home or office.

- Building and deploying web servers
- Database creation, management, and manipulation
- File and print servers
- Using FTP for serving files across the Internet and local networks
- Building and deploying email servers using Postfix and managing mailing lists
- Creating remote access gateways and services

- Configuring DNS for your network
- Using LDAP for storing information on users and security

Part V, "Programming Linux," provides a great introduction to how you can extend Ubuntu capabilities even further using the development tools supplied with it. This part covers the following:

- Programming in Perl, using variables and scripting
- An introduction to the Python language
- Writing PHP scripts and linking them to databases
- C and C++ programming tools available with Ubuntu and how to use the GNU C Compiler (gcc)

Part VI, "Ubuntu Housekeeping," looks at some of the more advanced skills you need to keep your system running in perfect condition, including the following:

- Securing your machine against attack from outsiders and viruses
- Performance tuning
- Command-line masterclass
- Advanced apt
- Kernel and module management and compilation

An extensive reference in Part VII, "Appendixes," gives you scope to explore in even more depth some of the topics covered in this book as well as providing historical context to Ubuntu and installation resources.

Conventions Used in This Book

A lot of documentation is included with every Linux distribution, and Ubuntu is certainly no exception. Although the intent of *Ubuntu Unleashed* is to be as complete as possible, it is impossible to cover every option of every command included in the distribution. However, this book offers numerous tables of various options, commands, and keystrokes to help condense, organize, and present information about a variety of subjects.

This edition is also packed full of screenshots to illustrate nearly all Ubuntu-specific graphical utilities—especially those related to system administration or the configuration and administration of various system and network services.

To help you better understand code listing examples and sample command lines, several formatting techniques are used to show input and ownership. For example, if the command or code listing example shows typed input, the input is formatted in boldface, as follows:

```
$ ls
```

If typed input is required, as in response to a prompt, the sample typed input also is in boldface, like so:

```
Delete files? [Y/n] y
```

All statements, variables, and text that should appear on your display use the same boldface formatting. In addition, command lines that require root or super user access are prefaced with the `sudo` command, as follows:

```
$ sudo printtool &
```

Command-line examples that any user can run are prefaced with a dollar sign ($), like so:

```
$ ls
```

The following elements provide you with useful tidbits of information that relate to the discussion of the text:

NOTE

A note provides additional information you might want to make note of as you are working; augments a discussion with ancillary details; or points you to an article, a whitepaper, or another online reference for more information about a specific topic.

TIP

A tip can contain special insight or a timesaving technique, as well as information about items of particular interest to you that you might not find elsewhere.

CAUTION

A caution warns you about pitfalls or problems before you run a command, edit a configuration file, or choose a setting when administering your system.

Sidebars Can Be Goldmines

Just because it is in a sidebar does not mean that you will not find something new here. Be sure to watch for these elements that bring in outside content that is an aside to the discussion in the text. You will read about other technologies, Linux-based hardware, and special procedures to make your system more robust and efficient.

Other formatting techniques used to increase readability include the use of italics for placeholders in computer command syntax. Computer terms or concepts are also italicized upon first introduction in text.

Finally, you should know that all text, sample code, and screenshots in *Ubuntu Unleashed* were developed using Ubuntu and open-source tools.

Read on to start learning about and using the latest version of Ubuntu. Experienced users will want to consider the new information in this edition when planning or considering upgrades. There are many different Linux distributions from different vendors, but many derive from, or closely mimic, the Debian distribution.

CHAPTER 4

Command Line Quickstart

IN THIS CHAPTER

The command line is one of the most powerful tools available for use with Ubuntu, and indeed Linux. Knowledge of the commands associated with it and also how to string them together will make your life with Ubuntu much easier.

This chapter looks at some of the basic commands that you need to know to be productive at the command line. You will find out how to get to the command line, and also get to grips with some of the commands used to navigate around the file system. Later on in this book is the Command Line Masterclass (Chapter 30), which explores the subject in more depth. The skills you learn in this chapter will give you confidence when you're called upon to work at the command line.

What Is the Command Line?

Hang around Linux users for any length of time and it will not be long before you hear them speak in hushed tones about the command line or the terminal. Quite rightly too, as the command line offers a unique and powerful way to interact with Linux. However, for the most part you may never need to access the command line because Ubuntu offers a slew of graphical tools that enable you to configure most things on your system.

But sometimes things go wrong and you may not have the luxury of a graphical interface to work with. It is in these situations that a fundamental understanding of the command line and its uses can be a real life saver.

NOTE

In Chapter 3, "Working with Gnome," you learned about BulletProofX, a project whose goal it is to always provide a fallback if your X server fails. Under Ubuntu 9.10, this has been further improved, although there will still be some instances where even BulletProofX won't save you.

Don't be tempted to skip over this chapter as irrelevant; rather, work through the chapter and ensure that you are comfortable with the command line before moving on.

It is tempting to think of the command line as the product of some sort of black and arcane art, and in some ways it can appear to be extremely difficult and complicated to use. However, perseverance is key and by the end of this chapter you should at least be comfortable with using the command line and ready to move onto Chapter 30, "Command Line Masterclass."

More importantly, though, you will be able to make your way around a command line–based system, which you are likely to encounter if you work within a server environment.

This chapter introduces you to a number of commands, including commands that enable you to do the following tasks:

- **Perform routine tasks**—Logging in and out, using the text console, changing passwords, listing and navigating directories
- **Implement basic file management**—Creating files and folders, copying or moving them around the file system, renaming and ultimately deleting them (if necessary)
- **Execute basic system management**—Shutting down or rebooting, reading man pages, and using text-based tools to edit system configuration files

The information in this chapter is valuable for individual users or system administrators who are new to Linux and are learning to use the command line for the first time.

TIP

Those of you who have used a computer for many years will probably have come into contact with MS-DOS, in which case being presented with a black screen will fill you with a sense of nostalgia. Don't get too comfy; the command line in Linux is far superior to its distant MS-DOS cousin. Whereas MS-DOS skills are transferable only to other MS-DOS environments, the skills that you learn at the Linux command line can be transferred easily to other Unix-like operating systems, such as Solaris, OpenBSD, FreeBSD, and even Mac OS X, which allows you access to the terminal.

User Accounts

One concept you will have to get used to is that of user-based security. By and large, only two types of users will access the system as actual users. The first type is the regular user, of which you created one when you started Ubuntu for the first time (see Chapter 1, "Installing Ubuntu"). These users can change anything that is specific to them, such as the wallpaper on the desktop, their personal preferences, and so on. Note that the emphasis should be on anything that is specific to them, as it prevents regular users from making system-wide changes that could affect other users.

To make system-wide changes, you need to use super-user privileges which you should have if your account was the first one specified (i.e. when you specified a user during the installation). With super-user privileges you basically have access to the entire system and can carry out any task, even destructive ones! In order to use your super-user privileges you need to prefix the command you wish to execute with the command `sudo`. When you hit enter (after typing the remaining command) you will be prompted for your password, which you should type in followed by the Enter key. Ubuntu will then carry out the command, but with super-user privileges.

An example of the destructive nature of working as the super-user can be found in the age-old example of $sudo `rm -rf /`, which erases all the data on your hard drive. You need to be especially careful when using your super-user privileges, otherwise you may make irreparable damage to your system.

Don't let this worry you, though, as the ability to work as the super-user is fundamental to a healthy Linux system. Without it you would not be able to install new software, edit system configuration files, or do any number of administration tasks. By the end of this chapter you will feel comfortable working with your super-user privileges and be able to adequately administer your system.

Ubuntu works slightly differently to other Linux distributions by giving users super-user privileges by default. If you work with any other Linux distro you will quickly come across the `root` user, which is a super-user account. So rather than having to type in `sudo` before every command, the root account can simply issue the command and not have to worry about entering a password. You can tell when you are working at a root prompt because you will see the pound sign (#). Within Ubuntu the `root` account is disabled by default in preference to giving super-user privileges to users. If you wish to enable the root account then issue the command `sudo passwd`. When prompted, enter your user password. You will then be asked for a new UNIX password; this will be the password for the root account, so make sure and remember it. You will also be prompted to repeat the password, in case you've made any mistakes. Once you've typed it in and pressed Enter, the `root` account will now be active. You'll find out how to switch to `root` later on.

An alternative way of getting a root prompt, without having to enable the root account, is to issue the command `sudo -i`. After entering your password you will find yourself at a root prompt (#). Do what you need to do and when you are finished, type `exit` and press Enter to return to your usual prompt.

As with most things, Ubuntu offers you a number of ways to access the command line. You can use the Terminal entry in Applications, Accessories, but by far the simplest way is to press Ctrl + Alt + F1. Ubuntu switches to a black screen and a traditional login prompt that resembles the following:

```
Ubuntu 9.10 karmic karmic-dev ttyl
karmic-dev login:
```

TIP

This is actually one of six virtual consoles that Ubuntu provides for your use. After you have accessed a virtual console, you can use the Alt key and F1 through F6 to switch to a different console. If you want to get back to the graphical interface, press Alt + F7. You can also switch between consoles by holding the Alt key and pressing either the left or the right cursor key to move down or up a console, such as tty1 to tty2.

Ubuntu is waiting for you to log in as a user, so go ahead and enter your username and press the return key. Ubuntu then prompts you for your password, which you should enter. Note that Ubuntu does not show any characters while you are typing your password in. This is a good thing because it prevents any shoulder surfers from seeing what you've typed or the length of the password.

Hitting the Return key drops you to a shell prompt, signified by the dollar sign:

```
andrew@karmic-dev ~]$
```

This particular prompt tells me that I am logged in as the user `andrew` on the system `karmic-dev` and I am currently in my home directory (Linux uses the tilde as shorthand for the home directory).

TIP

Navigating through the system at the command line can get confusing at times, especially when a directory name occurs in several different places. Fortunately, Linux includes a simple command that tells you exactly where you are in the file system. It's easy to remember because the command is just an abbreviation of present working directory, so type **`pwd`** at any point to get the full path of your location. For example, typing `pwd` after following these instructions shows `/home/yourusername`, meaning that you are currently in your home directory.

Using the `pwd` command can save you a lot of frustration when you have changed directory half a dozen times and have lost track.

Another way to quickly access the terminal is to go to Applications, Accessories and choose the Terminal entry. Ubuntu opens up `gnome-terminal`, which allows you to access

the terminal while remaining in Gnome. This time, the terminal appears as black text on a white background. Accessing the terminal this way, or by using the Ctrl + Alt + F1 method makes no difference because you are interacting directly with the terminal itself.

Navigating Through the File System

Use the `cd` command to navigate through the Ubuntu file system. This command is generally used with a specific directory location or pathname, like this:

```
$ cd /etc/apt/
```

Under Ubuntu, the cd command can also be used with several shortcuts. For example, to quickly move up to the *parent* (higher-level) directory, use the cd command like this:

```
$ cd ..
```


To return to one's home directory from anywhere in the Linux file system, use the `cd` command like this:

```
$ cd
```

You can also use the `$HOME` shell environment variable to accomplish the same thing. Type this command and press Enter to return to your home directory:

```
$ cd $HOME
```

You can accomplish the same thing by using the tilde (~) like this:

```
$ cd ~
```

Don't forget the `pwd` command to remind you where you are within the file system!

Another important command to use is the `ls` command, which lists the contents of the current directory. It's commonly used by itself, but a number of options (or switches) available for `ls` give you more information. For instance, the following command returns a listing of all the files and directories within the current directory, including any hidden files (denoted by a `.` prefix) as well as a full listing, so it will include details such as the permissions, owner and group, size and last modified time and date:

```
$ ls -al
```

You can also issue the command

```
$ ls -R
```

which scans and lists all the contents of the sub-directories of the current directory. This might be a lot of information, so you may want to redirect the output to a text file so you can browse through it at your leisure by using the following:

```
$ ls -alR > listing.txt
```

TIP

The previous command sends the output of `ls -alR` to a file called listing.txt, and demonstrates part of the power of the Linux command line. At the command line you are able to use files as inputs to commands, or generate files as outputs as shown. For more information about combining commands, see Chapter 30.

We've included a table showing some of the top-level directories that are part of a standard Linux distro in Table 4.1.

TABLE 4.1 Basic Linux Directories

Name	Description
`/`	The root directory
`/bin`	Essential commands
`/boot`	Boot loader files, Linux kernel
`/dev`	Device files
`/etc`	System configuration files
`/home`	User home directories
`/initrd`	Initial RAM disk boot support (used during boot time)
`/lib`	Shared libraries, kernel modules
`/lost+found`	Directory for recovered files (if found after a file system check)
`/media`	Mount point for removable media, such as DVDs and floppy disks
`/mnt`	Usual mount point for local, remote file systems
`/opt`	Add-on software packages
`/proc`	Kernel information, process control
`/root`	Super-user (root) home
`/sbin`	System commands (mostly root only)
`/srv`	Holds information relating to services that run on your system
`/sys`	Real-time information on devices used by the kernel
`/tmp`	Temporary files
`/usr`	Secondary software file hierarchy
`/var`	Variable data (such as logs); spooled files

Knowing these directories can aid you in partitioning in any future systems, letting you choose to put certain directories on their own distinct partition.

Some of the important directories in Table 4.1, such as those containing user and root commands or system configuration files, are discussed in the following sections. You use and edit files under these directories when you use Ubuntu.

Linux also includes a number of GNU commands you can use to search the file system. These include the following:

- `whereis` *`command`*—Returns the location of the command and its man page.
- `whatis` *`command`*—Returns a one-line synopsis from the command's man page.
- `locate` *`file`*—Returns locations of all matching file(s); an extremely fast method of searching your system because `locate` searches a database containing an index of all files on your system. However, this database (about 4MB in size and named `slocate.db`, under the `/var/lib/slocate` directory) is built daily at 4:20 a.m. by default, and does not contain pathnames to files created during the workday or in the evening. If you do not keep your machine on constantly, you can run the `updatedb` command either using `sudo` or by using the `root` account to manually start the building of the database.
- `apropos` *`subject`*—Returns a list of commands related to subject.

Managing Files with the Shell

Managing files in your home directory involves using one or more easily remembered commands. If you have any familiarity with the now-ancient DOS, you recognize some of these commands (although their names are different from those you remember). Basic file management operations include paging (reading), moving, renaming, copying, searching, and deleting files and directories. These commands include the following:

- `cat` *`filename`*—Outputs contents of filename to display
- `less` *`filename`*—Allows scrolling while reading contents of filename
- `mv` *`file1 file2`*—Renames `file1` to `file2`
- `mv` *`file dir`*—Moves file to specified directory
- `cp` *`file1 file2`*—Copies `file1` and creates `file2`
- `rm` *`file`*—Deletes file
- `rmdir` *`dir`*—Deletes directory (if empty)
- `grep` *`string file(s)`*—Searches through files(s) and displays lines containing matching string

Note that each of these commands can be used with pattern-matching strings known as *wildcards* or *expressions*. For example, to delete all files in the current directory beginning with the letters `abc`, you can use an expression beginning with the first three letters of the desired filenames. An asterisk (`*`) is then appended to match all these files. Use a command line with the `rm` command like this:

```
$ rm abc*
```

Linux shells recognize many types of filenaming wildcards, but this is different from the capabilities of Linux commands supporting the use of more complex expressions. You learn more about using wildcards in Chapter 11, "Automating Tasks."

NOTE

Learn more about using expressions by reading the `grep` manual pages (`man grep`).

Working with Compressed Files

Another file management operation is compression and decompression of files, or the creation, listing, and expansion of file and directory archives. Linux distributions usually include several compression utilities you can use to create, compress, expand, or list the contents of compressed files and archives. These commands include

- `bunzip2`—Expands a compressed file
- `bzip2`—Compresses or expands files and directories
- `gunzip`—Expands a compressed file
- `gzip`—Compresses or expands files and directories
- `tar`—Creates, expands, or lists the contents of compressed or uncompressed file or directory archives known as *tape archives* or *tarballs*

Most of these commands are easy to use. The `tar` command, however, has a somewhat complex (although capable) set of command-line options and syntax. Even so, you can quickly learn to use `tar` by remembering a few simple invocations on the command line. For example, to create a compressed archive of a directory, use `tar`'s `czf` options like this:

```
$ tar czf dirname.tgz dirname
```

The result is a compressed archive (a file ending in `.tgz`) of the specified directory (and all files and directories under it). Add the letter `v` to the preceding options to view the list of files added during compression and archiving. To list the contents of the compressed archive, substitute the `c` option with the letter `t`, like this:

```
$ tar tzf archive
```

Of course, if many files are in the archive, a better invocation (to easily read or scroll through the output) is

```
$ tar tzf archive ¦ less
```

TIP

In the previous code example, we used a pipe character (|). Each pipe sends the output of the first command to the next command. This is another of the benefits of the command line under Linux—you can string several commands together to get the desired results.

To expand the contents of a compressed archive, use `tar`'s `zxf` options, like so:

```
$ tar zxf archive
```

The `tar` utility decompresses the specified archive and extracts the contents in the current directory.

Use Essential Commands from the `/bin` and `/sbin` Directories

The `/bin` directory (about 5MB if you do a full install) contains essential commands used by the system for running and booting Linux. In general, only the root operator uses the commands in the `/sbin` directory. Many (though not all) these commands are *statically* linked which means that such commands do not depend on software libraries residing under the `/lib` or `/usr/lib` directories. Nearly all the other applications on your system are *dynamically* linked—meaning that they require external software libraries (also known as *shared* libraries) to run.

Use and Edit Files in the `/etc` Directory

More than 10MB of system configuration files and directories reside under the `/etc` directory if you install all the software included with this book. Some major software packages, such as Apache, OpenSSH, and `xinetd`, have directories of configuration files under `/etc`. Other important system-related configuration files in `/etc` are

- `fstab`—The file system table is a text file listing each hard drive, CD-ROM, floppy, or other storage device attached to your PC. The table indexes each device's partition information with a place in your Linux file system (directory layout) and lists other options for each device when used with Linux (see Chapter 32, "Kernel and Module Management"). Nearly all entries in `fstab` can be manipulated by root using the `mount` command.
- `modprobe.d/`—This folder holds all the instructions to load kernel modules that are required as part of the system startup, and replaces the historic modprobe.conf file.
- `passwd`—The list of users for the system, along with user account information. The contents of this file can be changed by various programs, such as `useradd` or `chsh`.
- `shells`—A list of approved shells (command-line interfaces).

Protect the Contents of User Directories—`/home`

The most important data on a Linux system resides in the user's directories, found under the `/home` directory. Segregating the system and user data can be helpful in preventing data loss and making the process of backing up easier. For example, having user data reside on a separate file system or mounted from a remote computer on the network might help shield users from data loss in the event of a system hardware failure.

Use the Contents of the `/proc` Directory to Interact with the Kernel

The content of the `/proc` directory is created from memory and exists only while Linux is running. This directory contains special "files" that either extract information from or send information to the kernel. Many Linux utilities extract information from dynamically created directories and files under this directory, also known as a *virtual file system*. For example, the `free` command obtains its information from a file named `meminfo`:

```
$ free
             total       used       free     shared    buffers     cached
Mem:       1026320     822112     204208          0      41232     481412
-/+ buffers/cache:     299468     726852
Swap:      2031608          0    2031608
```

This information constantly changes as the system is used. You can get the same information by using the `cat` command to see the contents of the `meminfo` file:

```
$ cat /proc/meminfo
MemTotal:      1026320 kB
MemFree:        204200 kB
Buffers:         41252 kB
Cached:         481412 kB
SwapCached:          0 kB
Active:         307232 kB
Inactive:       418224 kB
HighTotal:      122692 kB
HighFree:          244 kB
LowTotal:       903628 kB
LowFree:        203956 kB
SwapTotal:     2031608 kB
SwapFree:      2031608 kB
Dirty:               0 kB
Writeback:           0 kB
AnonPages:      202804 kB
Mapped:          87864 kB
Slab:            21736 kB
SReclaimable:    12484 kB
SUnreclaim:       9252 kB
PageTables:       5060 kB
NFS_Unstable:        0 kB
Bounce:              0 kB
CommitLimit:   2544768 kB
Committed_AS:   712024 kB
VmallocTotal:   114680 kB
VmallocUsed:      6016 kB
VmallocChunk:   108148 kB
HugePages_Total:     0
HugePages_Free:      0
HugePages_Rsvd:      0
Hugepagesize:     4096 kB
```

The `/proc` directory can also be used to dynamically alter the behavior of a running Linux kernel by "echoing" numerical values to specific files under the `/proc/sys` directory. For example, to "turn on" kernel protection against one type of denial of service (DOS) attack

known as *SYN flooding*, use the `echo` command to send the number `1` (one) to the following `/proc` path:

```
$ sudo echo 1 >/proc/sys/net/ipv4/tcp_syncookies
```

Other ways to use the `/proc` directory include

- Getting CPU information, such as the family, type, and speed from `/proc/cpuinfo`.
- Viewing important networking information under `/proc/net`, such as active interfaces information under `/proc/net/dev`, routing information in `/proc/net/route`, and network statistics in `/proc/net/netstat`.
- Retrieving file system information.
- Reporting media mount point information via USB; for example, the Linux kernel reports what device to use to access files (such as `/dev/sda`) if a USB camera or hard drive is detected on the system. You can use the `dmesg` command to see this information.
- Getting the kernel version in `/proc/version`, performance information such as uptime in `/proc/uptime`, or other statistics such as CPU load, swap file usage, and processes in `/proc/stat`.

Work with Shared Data in the `/usr` Directory

The `/usr` directory contains software applications, libraries, and other types of shared data for use by anyone on the system. Many Linux system administrators give `/usr` its own partition. A number of subdirectories under `/usr` contain manual pages (`/usr/share/man`), software package shared files (`/usr/share/`*`name_of_package`*, such as `/usr/share/emacs`), additional application or software package documentation (`/usr/share/doc`), and an entire subdirectory tree of locally built and installed software, `/usr/local`.

Temporary File Storage in the `/tmp` Directory

As its name implies, the `/tmp` directory is used for temporary file storage; as you use Linux, various programs create files in this directory.

Access Variable Data Files in the `/var` Directory

The `/var` directory contains subdirectories used by various system services for spooling and logging. Many of these variable data files, such as print spooler queues, are temporary, whereas others, such as system and kernel logs, are renamed and rotated in use. Incoming electronic mail is usually directed to files under `/var/spool/mail`.

Linux also uses `/var` for other important system services. These include the top-most File Transfer Protocol (FTP) directory under `/var/ftp` (see Chapter 18, "Remote File Serving with FTP"), and the Apache web server's initial home page directory for the system, `/var/www/html`. (See Chapter 17, "Apache Web Server Management," for more information on using Apache.)

Logging In to and Working with Linux

You can access and use a Linux system in a number of ways. One way is at the console with a monitor, keyboard, and mouse attached to the PC. Another way is via a serial console, either by dial-up via a modem or a PC running a terminal emulator and connected to the Linux PC via a null modem cable. You can also connect to your system through a wired or wireless network, using the `telnet` or `ssh` commands. The information in this section shows you how to access and use the Linux system, using physical and remote text-based logins.

NOTE

This chapter focuses on text-based logins and use of Linux. Graphical logins and using a graphical desktop are described in Chapter 3, "Working with Gnome."

Text-based Console Login

If you sit down at your PC and log in to a Linux system that has not been booted to a graphical login, you see a prompt similar to this one:

```
Ubuntu 9.10 karmic karmic-dev ttyl
karmic-dev login:
```

Your prompt might vary, depending on the version of Ubuntu you are using. In any event, at this prompt, type in your username and press Enter. When you are prompted for your password, type it in and press Enter.

NOTE

Note that your password is not echoed back to you, which is a good idea. Why is it a good idea? Well, people are prevented from looking over your shoulder and seeing your screen input. It is not difficult to guess that a five-letter password might correspond to the user's spouse's first name!

Logging Out

Use the `exit` or `logout` commands to exit your session. Type the command and press Enter. You are then returned to the login prompt. If you use virtual consoles, remember to exit each console before leaving your PC. (Otherwise, someone could easily sit down and use your account.)

Logging In and Out from a Remote Computer

Although you can happily log in on your computer, an act known as a *local* login, you can also log in to your computer via a network connection from a remote computer. Linux-based operating systems provide a number of remote access commands you can use to log in to other computers on your local area network (LAN), wide area network (WAN), or the Internet. Note that not only must you have an account on the remote computer, but the remote computer must be configured to support remote logins—otherwise, you won't be able to log in.

NOTE

See Chapter 14, "Networking" to see how to set up network interfaces with Linux to support remote network logins and Chapter 11 to see how to start remote access services (such as `sshd`).

The best and most secure way (barring future exploits) to log in to a remote Linux computer is to use the `ssh` or Secure Shell client. Your login and session are encrypted while you work on the remote computer. The `ssh` client features many different command-line options, but can be simply used with the name or IP address of the remote computer, like this:

```
[andrew@karmic-dev ~]$ ssh 192.168.0.41
The authenticity of host '192.168.0.41 (192.168.0.41)' can't be established.
RSA key fingerprint is e1:db:6c:da:3f:fc:56:1b:52:f9:94:e0:d1:1d:31:50.
Are you sure you want to continue connecting (yes/no)?

yes
```

The first time you connect with a remote computer using `ssh`, Linux displays the remote computer's encrypted identity key and asks you to verify the connection. After you type **`yes`** and press Enter, you are warned that the remote computer's identity (key) has been entered in a file named `known_hosts` under the `.ssh` directory in your home directory. You are also prompted to enter your password:

```
Warning: Permanently added '192.168.0.41' (RSA) \
to the list of known hosts.
andrew@192.168.0.41's password:
andrew~$
```

After entering your password, you can then work on the remote computer. Again, everything you enter on the keyboard in communication with the remote computer is encrypted. Use the `exit` or `logout` commands to exit your session and return to the shell on your computer.

Using Environment Variables

A number of in-memory variables are assigned and loaded by default when the user logs in. These variables are known as shell *environment variables*, which can be used by various commands to get information about your environment, such as the type of system you are running, your home directory, and the shell in use. Environment variables are used by Linux operating systems to help tailor the computing environment of your system, and include helpful specifications and setup, such as default locations of executable files and software libraries. If you begin writing shell scripts, you might use environment variables in your scripts. Until then, you only need to be aware of what environment variables are and do.

The following list includes a number of environment variables, along with descriptions of how the shell uses them:

- `PWD`—To provide the name of the current working directory, used by the `pwd` command (such as `/home/andrew/foo`)
- `USER`—To declare the user's name, such as `andrew`
- `LANG`—To set language defaults, such as English
- `SHELL`—To declare the name and location of the current shell, such as `/bin/bash`
- `PATH`—To set the default location of executable files, such as `/bin`, `/usr/bin`, and so on
- `TERM`—To set the type of terminal in use, such as `vt100`, which can be important when using screen-oriented programs, such as text editors
- `MACHINE`—To declare system type, system architecture, and so on

NOTE

Each shell can have its own feature set and language syntax, as well as a unique set of default environment variables. See Chapter 15, "Remote Access for SSH and Telnet," for more information about using the different shells included with Ubuntu.

At the command line, you can use the `env` or `printenv` commands to display these environment variables, like so:

```
$ env
SSH_AGENT_PID=5761

SHELL=/bin/bash

DESKTOP_STARTUP_ID=

TERM=xterm
```

```
GTK_RC_FILES=/etc/gtk/gtkrc:/home/andrew/.gtkrc-1.2-gnome2

WINDOWID=56623199

USER=andrew

...
USERNAME=andrew

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

DESKTOP_SESSION=default

GDM_XSERVER_LOCATION=local

PWD=/usr/local

LANG=en_GB.UTF-8

GNOME_KEYRING_PID=5714

GDM_LANG=en_GB.UTF-8

SHLVL=1

HOME=/home/andrew

LOGNAME=andrew

XDG_DATA_DIRS=/usr/local/share/:/usr/share/:/usr/share/gdm/

...
LESSOPEN=¦ /usr/bin/lesspipe %s

WINDOWPATH=7

DISPLAY=:0.0

LESSCLOSE=/usr/bin/lesspipe %s %s

COLORTERM=gnome-terminal

XAUTHORITY=/home/andrew/.Xauthority

_=/usr/bin/env
```

```
OLDPWD=/usr/share/locale
```

This abbreviated list shows a few common variables. These variables are set by configuration or *resource* files contained in the `/etc`, `/etc/skel`, or user `/home` directory. You can find default settings for `bash`, for example, in `/etc/profile`, `/etc/bashrc`, `.bashrc`, or `.bash_profile` files installed in your home directory. Read the man page for `bash` for details about using these configuration files.

One of the most important environment variables is `$PATH`, which defines the location of executable files. For example, if, as a regular user, you try to use a command that is not located in your `$PATH` (such as the imaginary `command` command), you will see something like this:

```
$ command
-bash: command: command not found
```

NOTE

If the command that you're trying to execute exists, but is not yet installed on your system, then Ubuntu will prompt you to install it, even giving you the correct command to do so.

However, you might know that `command` is definitely installed on your system, and you can verify this by using the `whereis` command, like so:

```
$ whereis command
command: /sbin/command
```

You can also run the command by typing its full pathname, or complete directory specification like this:

```
$ /sbin/command
```

As you can see in this example, the `command` command is indeed installed. What happened is that by default, the `/sbin` directory is not in your `$PATH`. One of the reasons for this is that commands under the `/sbin` directory are normally intended to be run only by root. You can add `/sbin` to your `$PATH` by editing the file `.bash_profile` in your home directory (if you use the `bash` shell by default, like most Linux users). Look for the following line:

```
PATH=$PATH:$HOME/bin
```

You can then edit this file, perhaps using the `vi` editor (discussed in this chapter), to add the `/sbin` directory like so:

```
PATH=$PATH:/sbin:$HOME/bin
```

Save the file. The next time you log in, the `/sbin` directory is in your `$PATH`. One way to use this change right away is to read in the new settings in `.bash_profile` by using the `bash` shell's `source` command like so:

```
$ source .bash_profile
```

You can now run commands located in the `/sbin` directory without the need to explicitly type the full pathname.

Some Linux commands also use environment variables, for example, to acquire configuration information (such as a communications program looking for a variable such as `BAUD_RATE`, which might denote a default modem speed).

To experiment with the environment variables, you can modify the `PS1` variable to manipulate the appearance of your shell prompt. If you are working with `bash`, you can use its built-in `export` command to change the shell prompt. For example, if your default shell prompt looks like

```
[andrew@laptop ~]$
```

You can change its appearance by using the `PS1` variable like this:

```
$ PS1='$OSTYPE r00lz ->'
```

After you press Enter, you see

```
linux-gnu r00lz ->
```

NOTE

See the `bash` man page for other variables you can use for prompt settings.

Using the Text Editors

Linux distributions include a number of applications known as *text editors* that you can use to create text files or edit system configuration files. Text editors are similar to word processing programs, but generally have fewer features, work only with text files, and might or might not support spell checking or formatting. The text editors range in features and ease of use, but are found on nearly every Linux distribution. The number of editors installed on your system depends on what software packages you've installed on the system.

Some of the console-based text editors are

- `emacs`—The comprehensive GNU `emacs` editing environment, which is much more than an editor; see the section "Working with `emacs`" later in this chapter
- `joe`—Joe's Own Editor, a text editor, which can be used to emulate other editors
- `nano`—A simple text editor similar to the `pico` text editor included with the `pine` email program
- `vim`—An improved, compatible version of the `vi` text editor (which we call `vi` in the rest of this chapter because it has a symbolic link named `vi` and a symbolically linked manual page)

Note that not all text editors described here are *screen oriented*. Some of the text editors for the X Window System, which provide a graphical interface, such as menu bars, buttons, scrollbars and so on, are

- `gedit`—A GUI text editor for GNOME
- `kate`—A simple KDE text editor
- `kedit`—Another simple KDE text editor

A good reason to learn how to use a text-based editor, such as `vi`, is that system maintenance and recovery operations generally never take place during X Window sessions (negating the use of a GUI editor). Many larger, more complex and capable editors do not work when Linux is booted to its single-user or maintenance mode. If anything does go wrong with your system, you probably won't be able to get into the X Window system, making knowledge and experience of using both the command line and text editors such as `vi` important. Make a point of opening some of the editors and playing around with them; you never know—you might just thank me someday!

Another reason to learn how to use a text-based editor under the Linux console mode is so that you can edit text files through dial-up or network shell sessions because many servers do not host graphical desktops.

Working with `vi`

The editor found on nearly every Unix and Linux system is, without a doubt, the `vi` editor, originally written by Bill Joy. This simple-to-use but incredibly capable editor features a somewhat cryptic command set, but you can put it to use with only a few commands. Although more experienced Unix and Linux users continue to use `vi` extensively during computing sessions, many newer users might prefer learning an easier-to-use text editor such as `pico` or GNU `nano`. Die-hard GNU fans and programmers definitely use `emacs`.

That said, learning how to use `vi` is a good idea. You might need to edit files on a Linux system with a minimal install, or a remote server without a more extensive offering of installed text editors. Chances are better than good that `vi` will be available.

You can start an editing session by using the `vi` command like this:

```
$ vi file.txt
```

The vi command works by using an insert (or editing) mode, and a viewing (or command) mode.

When you first start editing, you are in the viewing mode. You can use your cursor or other navigation keys (as shown later) to scroll through the text. To start editing, press the **i** key to insert text or the **a** key to append text. When finished, use the Esc key to toggle out of the insert or append modes and into the viewing (or command) mode. To enter a command, type a colon (**:**), followed by the command, such as **w** to write the file, and press Enter.

Although vi supports many complex editing operations and numerous commands, you can accomplish work by using a few basic commands. These basic vi commands are

- **Cursor movement**—h, j, k, l (left, down, up, and right)
- **Delete character**—x
- **Delete line**—dd
- **Mode toggle**—Esc, Insert (or i)
- **Quit**—:q
- **Quit without saving**—:q!
- **Run a shell command**—:sh (use 'exit' to return)
- **Save file**—:w
- **Text search**—/

NOTE

Use the vimtutor command to quickly learn how to use vi's keyboard commands. The tutorial takes less than 30 minutes, and it teaches new users how to start or stop the editor, navigate files, insert and delete text, and perform search, replace, and insert operations.

Working with emacs

Richard M. Stallman's GNU emacs editor, like vi, is included with Ubuntu and nearly every other Linux distribution. Unlike other Unix and Linux text editors, emacs is much more than a simple text editor—it is an editing environment and can be used to compile

and build programs, act as an electronic diary, appointment book and calendar, compose and send electronic mail, read Usenet news, and even play games. The reason for this capability is that `emacs` contains a built-in language interpreter that uses the Elisp (`emacs` LISP) programming language. `emacs` is not installed in Ubuntu by default; instead you'll need to install it using `apt-get` or `synaptic`. The package you need is simply `emacs`.

You can start an `emacs` editing session like this_FIRST:

```
$ emacs file.txt
```

TIP

If you start `emacs` when using `X11`, the editor launches in its own floating window. To force `emacs` to display inside a terminal window instead of its own window (which can be useful if the window is a login at a remote computer), use the `-nw` command-line option like this: `emacs -nw file.txt`.

The `emacs` editor uses an extensive set of keystroke and named commands, but you can work with it by using a basic command subset. Many of these basic commands require you to hold down the Ctrl key, or to first press a *meta* key (generally mapped to the Alt key). The basic commands are listed in Table 4.2.

TABLE 4.2 Emacs Editing Commands

Action	Command
Abort	Ctrl+g
Cursor left	Ctrl+b
Cursor down	Ctrl+n
Cursor right	Ctrl+f
Cursor up	Ctrl+p
Delete character	Ctrl+d
Delete line	Ctrl+k
Go to start of line	Ctrl+a
Go to end of line	Ctrl+e
Help	Ctrl+h
Quit	Ctrl+x, Ctrl+c
Save As	Ctrl+x, Ctrl+w
Save file	Ctrl+x, Ctrl+s
Search backward	Ctrl+r
Search forward	Ctrl+s
Start tutorial	Ctrl+h, t
Undo	Ctrl+x, u

TIP

One of the best reasons to learn how to use `emacs` is that you can use nearly all the same keystrokes to edit commands on the `bash` shell command line. Another reason is that like `vi`, `emacs` is universally available on nearly every Unix and Linux system, including Apple's Mac OS X.

Working with Permissions

Under Linux (and Unix), everything in the file system, including directories and devices, is a file. And every file on your system has an accompanying set of permissions based on ownership. These permissions form the basis for security under Linux, and designate each file's read, write, and execute permission for you, members of your group, and all others on the system.

You can examine the default permissions for a file you create by using the `umask` command, or as a practical example, by using the `touch` command and then the `ls` command's long-format listing like this:

```
$ touch file
$ ls -l file
-rw-r—r— 1 andrew andrew 0 Feb 1 20:54 file
```

In this example, the `touch` command is used to quickly create a file. The `ls` command then reports on the file, displaying information (from left to right) in the first field of output (such as `-rw-r—r—` previously):

- **The type of file created**—Common indicators of the type of file are a leading letter in the output. A blank (which is represented by a dash in the preceding example) designates a plain file, `d` designates a directory, `c` designates a character device (such as `/dev/ttyS0`), and `b` is used for a block device (such as `/dev/sda`).
- **Permissions**—Read, write, and execute permissions for the owner, group, and all others on the system. (You learn more about these permissions later in this section.)
- **Number of links to the file**—The number one (`1`) designates that there is only one file, whereas any other number indicates that there might be one or more hard-linked files. Links are created with the `ln` command. A hard-linked file is an exact copy of the file, but it might be located elsewhere on the system. Symbolic links of directories can also be created, but only the root operator can create a hard link of a directory.
- **The owner**—The account that created or owns the file; you can change this designation by using the `chown` command.
- **The group**—The group of users allowed to access the file; you can change this designation by using the `chgrp` command.
- **File size and creation/modification date**—The last two elements indicate the size of the file in bytes and the date the file was created or last modified.

Assigning Permissions

Under Linux, permissions are grouped by owner, group, and others, with read, write, and execute permission assigned to each, like so:

```
Owner    Group    Others
rwx      rwx      rxw
```

Permissions can be indicated by mnemonic or octal characters. Mnemonic characters are

- `r` indicates permission for an owner, member of the owner's group, or others to open and read the file.
- `w` indicates permission for an owner, member of the owner's group, or others to open and write to the file.
- `x` indicates permission for an owner, member of the owner's group, or others to execute the file (or read a directory).

In the previous example for the file named `file`, the owner, `andrew`, has read and write permission, as does any member of the group named `andrew`. All other users may only read the file. Also note that default permissions for files created by the root operator will be different because of `umask` settings assigned by the shell.

Many users prefer to use numeric codes, based on octal (base 8) values, to represent permissions. Here's what these values mean:

- `4` indicates read permission.
- `2` indicates write permission.
- `1` indicates execute permission.

In octal notation, the previous example file has a permission setting of `664` (read + write or `4 + 2`, read + write or `4 + 2`, read-only or `4`). Although you can use either form of permissions notation, octal is easy to use quickly after you visualize and understand how permissions are numbered.

NOTE

In Linux, you can create groups to assign a number of users access to common directories and files, based on permissions. You might assign everyone in accounting to a group named `accounting`, for example, and allow that group access to accounts payable files while disallowing access by other departments. Defined groups are maintained by the root operator, but you can use the `newgrp` command to temporarily join other groups to access files (as long as the root operator has added you to the other groups). You can also allow or deny other groups' access to your files by modifying the group permissions of your files.

Directory Permissions

Directories are also files under Linux. For example, again use the `ls` command to show permissions like this:

```
$ mkdir foo
$ ls -ld foo
drwxrwxr-x    2 andrew    andrew        4096 Jan 23 12:37 foo
```

In this example, the `mkdir` command is used to create a directory. The `ls` command and its `-ld` option is used to show the permissions and other information about the directory (not its contents). Here you can see that the directory has permission values of `775` (read + write + execute or `4 + 2 + 1`, read + write + execute or `4 + 2 + 1`, and read + execute or `4 + 1`).

This shows that the owner and group members can read and write to the directory and, because of execute permission, also list the directory's contents. All other users can only list the directory contents. Note that directories require execute permission for anyone to be able to view their contents.

You should also notice that the `ls` command's output shows a leading `d` in the permissions field. This letter specifies that this file is a directory; normal files have a blank field in its place. Other files, such as those specifying a block or character device, have a different letter.

For example, if you examine the device file for a Linux serial port, you will see

```
$ ls -l /dev/ttyS0
crw-rw— — 1 root dialout 4, 64 Feb 1 19:49 /dev/ttyS0
```

Here, `/dev/ttyS0` is a character device (such as a serial communications port and designated by a `c`) owned by root and available to anyone in the `dialout` group. The device has permissions of `660` (read + write, read + write, no permission).

On the other hand, if you examine the device file for an IDE hard drive, you see

```
$ ls -l /dev/sda
brw-rw— — 1 root disk 8, 0 Feb 1 19:49 /dev/sda
```

In this example, `b` designates a block device (a device that transfers and caches data in blocks) with similar permissions. Other device entries you will run across on your Linux system include symbolic links, designated by `s`.

You can use the `chmod` command to alter a file's permissions. This command uses various forms of command syntax, including octal or a mnemonic form (such as `u`, `g`, `o`, or `a` and `rwx`, and so on) to specify a desired change. The `chmod` command can be used to add, remove, or modify file or directory permissions to protect, hide, or open up access to a file by other users (except for root, which can access any file or directory on a Linux system).

The mnemonic forms of `chmod`'s options (when used with a plus character, `+`, to add, or a minus sign, `-`, to take away) designate the following:

- `u`—Adds or removes user (owner) read, write, or execute permission
- `g`—Adds or removes group read, write, or execute permission
- `o`—Adds or removes read, write, or execute permission for others not in a file's group
- `a`—Adds or removes read, write, or execute permission for all users
- `r`—Adds or removes read permission
- `w`—Adds or removes write permission
- `x`—Adds or removes execution permission

For example, if you create a file, such as a `readme.txt`, the file will have default permissions (set by the `umask` setting in `/etc/bashrc`) of

```
-rw-rw-r--    1 andrew    andrew          12 Jan 2 16:48 readme.txt
```

As you can see, you and members of your group can read and write the file. Anyone else can only read the file (and only if it is outside your home directory, which will have read, write, and execute permission set only for you, the owner). You can remove all write permission for anyone by using `chmod`, the minus sign, and `aw` like so:

```
$ chmod -aw readme.txt
$ ls -l readme.txt
-r--r--r--    1 andrew    andrew          12 Jan 2 16:48 readme.txt
```

Now, no one can write to the file (except you, if the file is in your home or `/tmp` directory because of directory permissions). To restore read and write permission for only you as the owner, use the plus sign and the `u` and `rw` options like so:

```
$ chmod u+rw readme.txt
$ ls -l readme.txt
-rw-------    1 andrew    andrew          12 Jan 2 16:48 readme.txt
```

You can also use the octal form of the `chmod` command, for example, to modify a file's permissions so that only you, the owner, can read and write a file. Use the `chmod` command and a file permission of `600`, like this:

```
$ chmod 600 readme.txt
```

If you take away execution permission for a directory, files might be hidden inside and may not be listed or accessed by anyone else (except the root operator, of course, who has access to any file on your system). By using various combinations of permission settings, you can quickly and easily set up a more secure environment, even as a normal user in your home directory.

Understanding Set User ID and Set Group ID Permissions

Another type of permission is "set user ID", known as *suid*, and "set group ID" (*sgid*) permissions. These settings, when used in a program, enable any user running that program to have program owner or group owner permissions for that program. These settings enable the program to be run effectively by anyone, without requiring that each user's permissions be altered to include specific permissions for that program.

One commonly used program with suid permissions is the `passwd` command:

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 29104 Nov 6 19:16 /usr/bin/passwd
```

This setting allows normal users to execute the command (as root) to make changes to a root-only accessible file, `/etc/passwd`.

You also can assign similar permission with the `chfn` command. This command allows users to update or change `finger` information in `/etc/passwd`. You accomplish this permission modification by using a leading `4` (or the mnemonic `s`) in front of the three octal values.

NOTE

Other files that might have suid or guid permissions include `at`, `rcp`, `rlogin`, `rsh`, `chage`, `chsh`, `ssh`, `crontab`, `sudo`, `sendmail`, `ping`, `mount`, and several Unix-to-Unix Copy (UUCP) utilities. Many programs (such as games) might also have this type of permission to access a sound device.

Files or programs that have suid or guid permissions can sometimes present security holes because they bypass normal permissions. This problem is compounded if the permission extends to an executable binary (a command) with an inherent security flaw because it could lead to any system user or intruder gaining root access. In past exploits, this typically happened when a user fed a vulnerable command with unexpected input (such as a long pathname or option); the command would fail, and the user would be presented a root prompt. Although Linux developers are constantly on the lookout for poor programming practices, new exploits are found all the time, and can crop up unexpectedly, especially in newer software packages that haven't had the benefit of peer developer review.

Savvy Linux system administrators keep the number of suid or guid files present on a system to a minimum. The `find` command can be used to display all such files on your system:

```
# find / -type f -perm +6000 -exec ls -l {} \;
```

NOTE

The `find` command is quite helpful and can be used for many purposes, such as before or during backup operations. See the section "Using Backup Software" in Chapter 13, "Backing Up."

Note that the programs do not necessarily have to be removed from your system. If your users really do not need to use the program, you can remove the program's execute permission for anyone. You have to decide, as the root operator, whether your users are allowed to, for example, mount and unmount CD-ROMs or other media on your system. Although Linux-based operating systems can be set up to accommodate ease of use and convenience, allowing programs such as `mount` to be suid might not be the best security policy. Other candidates for suid permission change could include the `chsh`, `at`, or `chage` commands.

Working as Root

The root, or super-user account, is a special account and user on Unix and Linux systems. Super-user permissions are required in part because of the restrictive file permissions assigned to important system configuration files. You must have root permission to edit these files or to access or modify certain devices (such as hard drives). When logged in as root, you have total control over your system, which can be dangerous.

When you work in root, you can destroy a running system with a simple invocation of the `rm` command like this:

```
# rm -fr /
```

This command line not only deletes files and directories, but also could wipe out file systems on other partitions and even remote computers. This alone is reason enough to take precautions when using root access.

The only time you should run Linux as the super-user is when you are configuring the file system, for example, or to repair or maintain the system. Logging in and using Linux as the root operator isn't a good idea because it defeats the entire concept of file permissions.

NOTE

The next couple of paragraphs assume that you have enabled the root account, as described at the start of this chapter.

Knowing how to run commands as root without logging in as root can help avoid serious missteps when configuring your system. Linux comes with a command named `su` that enables you to run one or more commands as root and then quickly returns you to normal user status. For example, if you would like to edit your system's file system table (a

simple text file that describes local or remote storage devices, their type, and location), you can use the `su` command like this:

```
$ su -c "nano -w /etc/fstab"
Password:
```

After you press Enter, you are prompted for a password that gives you access to root. This extra step can also help you "think before you leap" into the command. Enter the root password, and you are then editing `/etc/fstab`, using the `nano` editor with line wrapping disabled.

CAUTION

Before editing any important system or software service configuration file, make a backup copy. Then make sure to launch your text editor with line wrapping disabled. If you edit a configuration file without disabling line wrapping, you could insert spurious carriage returns and line feeds into its contents, causing the configured service to fail when restarting. By convention, nearly all configuration files are formatted for 80-character text width, but this is not always the case. By default, the `vi` and `emacs` editors don't use line wrap.

You can use `sudo` in the same way to allow you to execute one-off commands. The above example would look like this, using `sudo`:

```
$ sudo nano -w /etc/fstab
```

Creating Users

When a Linux system administrator creates a user, an entry in `/etc/passwd` for the user is created. The system also creates a directory, labeled with the user's username, in the `/home` directory. For example, if you create a user named `bernice`, the user's home directory is `/home/bernice`.

NOTE

In this chapter, you learn how to manage users from the command line. See Chapter 10, "Managing Users," for more information on user administration with Ubuntu using graphical administration utilities, such as the graphical `users-admin` client.

Use the `useradd` command, along with a user's name, to quickly create a user:

```
$ sudo useradd andrew
```

After creating the user, you must also create the user's initial password with the `passwd` command:

```
$ sudo passwd andrew
```

```
Changing password for user andrew.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Enter the new password twice. If you do not create an initial password for a new user, the user cannot log in.

You can view `useradd`'s default new user settings by using the command and its `-D` option, like this:

```
$ useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

These options display the default group ID, home directory, account and password policy (active forever with no password expiration), the default shell, and the directory containing defaults for the shell.

The `useradd` command has many different command-line options. The command can be used to set policies and dates for the new user's password, assign a login shell, assign group membership, and other aspects of a user's account.

Deleting Users

Use the `userdel` command to delete users from your system. This command removes a user's entry in the system's `/etc/passwd` file. You should also use the command's `-r` option to remove all the user's files and directories (such as the user's mail spool file under `/var/spool/mail`):

```
$ sudo userdel -r andrew
```

If you do not use the `-r` option, you have to manually delete the user's directory under `/home`, along with the user's `/var/spool/mail` queue.

Shutting Down the System

Use the `shutdown` command to shut down your system. The `shutdown` command has a number of different command-line options (such as shutting down at a predetermined time), but the fastest way to cleanly shut down Linux is to use the `-h` or halt option, followed by the word `now` or the numeral zero (`0`), like this:

```
$ sudo shutdown -h now
```

or

```
$ sudo shutdown -h 0
```

To incorporate a timed shutdown and a pertinent message to all active users, use `shutdown`'s time and message options, like so:

```
$ sudo shutdown -h 18:30 "System is going down for maintenance this evening"
```

This example shuts down your system and provides a warning to all active users 15 minutes before the shutdown (or reboot). Shutting down a running server can be considered drastic, especially if there are active users or exchanges of important data occurring (such as a backup in progress). One good approach is to warn users ahead of time. This can be done by editing the system Message of the Day (MOTD) `motd` file, which displays a message to users after login. To create your custom MOTD, use a text editor and change the contents of `/etc/motd`. You can also make downtimes part of a regular schedule, perhaps to coincide with security audits, software updates, or hardware maintenance.

You should shut down Ubuntu for only a few very specific reasons:

- You are not using the computer and want to conserve electrical power.
- You need to perform system maintenance that requires any or all system services to be stopped.
- You want to replace integral hardware.

TIP

Do not shut down your computer if you suspect that one or more intruders has infiltrated your system; instead, disconnect the machine from any or all networks and make a backup copy of your hard drives. You might want to also keep the machine running to examine the contents of memory and to examine system logs.

Rebooting the System

You should also use the `shutdown` command to reboot your system. The fastest way to cleanly reboot Linux is to use the `-r` option, and the word `now` or the numeral zero (`0`):

```
$ sudo shutdown -r now
```

or

```
$ sudo shutdown -r 0
```

Both rebooting and shutting down can have dire consequences if performed at the wrong time (such as during backups or critical file transfers, which arouses the ire of your system's users). However, Linux-based operating systems are designed to properly stop active system services in an orderly fashion. Other commands you can use to shut down and reboot Linux are the `halt` and `reboot` commands, but the `shutdown` command is more flexible.

Reading Documentation

Although you learn the basics of using Ubuntu in this book, you need time and practice to master and troubleshoot more complex aspects of the Linux operating system and your distribution. As with any operating system, you can expect to encounter some problems or perplexing questions as you continue to work with Linux. The first place to turn for help with these issues is the documentation included with your system; if you cannot find the information you need there, check Ubuntu's website.

Linux, like Unix, is a self-documenting system, with man pages accessible through the `man` command. Linux offers many other helpful commands for accessing its documentation. You can use the `apropos` command—for example, with a keyword such as `partition`—to find commands related to partitioning, like this:

```
$ apropos partition
diskdumpfmt          (8) - format a dump device or a partition
fdisk                (8) - Partition table manipulator for Linux
GNU Parted [parted]  (8) - a partition manipulation program
mpartition           (1) - partition an MSDOS hard disk
MPI_Cart_sub         (3) - Partitions a communicator into subgroups which form
                           lower-dimensional cartesian subgrids
partprobe            (8) - inform the OS of partition table changes
pvcreate             (8) - initialize a disk or partition for use by LVM
sfdisk               (8) - Partition table manipulator for Linux
```

To find a command and its documentation, you can use the `whereis` command. For example, if you are looking for the `fdisk` command, you can do this:

```
$ whereis fdisk
fdisk: /sbin/fdisk /usr/share/man/man8/fdisk.8.gz
```

Using Man Pages

To learn more about a command or program, use the `man` command, followed by the name of the command. Man pages for Linux and X Window commands are within the `/usr/share/man`, `/usr/local/share/man`, and `/usr/X11R6/man` directories; so, for example, to read the `rm` command's man page, use the `man` command like this:

```
$ man rm
```

After you press Enter, the `less` command (a Linux command known as a *pager*) displays the man page. The `less` command is a text browser you can use to scroll forward and backward (even sideways) through the document to learn more about the command. Type the letter **`h`** to get help, use the forward slash to enter a search string, or press **`q`** to quit.

NOTE

Although nearly all the hundreds of GNU commands included with Linux each have a man page, you must use the `info` command to read detailed information about using a GNU command. For example, to learn even more about `bash` (which has a rather extensive manual page), use the `info` command like this:

```
$ info bash
```

Press the **n** and **p** keys to navigate through the document, or scroll down to a menu item on the screen and press Enter to read about a specific feature. Press **q** to quit reading.

Related Ubuntu and Linux Commands

The following programs and built-in shell commands are commonly used when working at the command line. These commands are organized by category to help you understand the command's purpose. If you need to find full information for using the command, you can find that information under the command's man page.

- **Managing users and groups**—chage, chfn, chsh, edquota, gpasswd, groupadd, groupdel, groupmod, groups, mkpasswd, newgrp, newusers, passwd, umask, useradd, userdel, usermod
- **Managing files and file systems**—cat, cd, chattr, chmod, chown, compress, cp, dd, fdisk, find, gzip, ln, mkdir, mksfs, mount, mv, rm, rmdir, rpm, sort, swapon, swapoff, tar, touch, umount, uncompress, uniq, unzip, zip
- **Managing running programs**—bg, fg, kill, killall, nice, ps, pstree, renice, top, watch
- **Getting information**—apropos, cal, cat, cmp, date, diff, df, dir, dmesg, du, env, file, free, grep, head, info, last, less, locate, ls, lsattr, man, more, pinfo, ps, pwd, stat, strings, tac, tail, top, uname, uptime, vdir, vmstat, w, wc, whatis, whereis, which, who, whoami
- **Console text editors**—ed, jed, joe, mcedit, nano, red, sed, vim
- **Console Internet and network commands**—bing, elm, ftp, host, hostname, ifconfig, links, lynx, mail, mutt, ncftp, netconfig, netstat, pine, ping, pump, rdate, route, scp, sftp, ssh, tcpdump, traceroute, whois, wire-test

Reference

- http://www.winntmag.com/Articles/Index.cfm?ArticleID=7420—An article by a Windows NT user who, when experimenting with Linux, blithely confesses to rebooting the system after not knowing how to read a text file at the Linux console.
- http://standards.ieee.org/regauth/posix/—IEEE's POSIX information page.
- http://www.itworld.com/Comp/2362/lw-01-government/#sidebar—Discussion of Linux and POSIX compliance.
- http://www.pathname.com/fhs/—Home page for the Linux FHS, Linux Filesystem Hierarchy Standard.
- http://www.tldp.org/—Browse the HOWTO section to find and read The Linux Keyboard and Console HOWTO—Andries Brouwer's somewhat dated but eminently useful guide to using the Linux keyboard and console.
- http://www.gnu.org/software/emacs/emacs.html—Home page for the FSF's GNU `emacs` editing environment; you can find additional documentation and links to the source code for the latest version here.
- http://www.vim.org/—Home page for the `vim` (`vi` clone) editor included with Linux distributions. Check here for updates, bug fixes, and news about this editor.
- http://www.courtesan.com/sudo/—Home page for the `sudo` command. Check here for the latest updates, security features, and bug fixes.

Index

Symbols and Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X-Y-Z