

Andrew Hudson
Paul Hudson
Matthew Helmke
Ryan Troy

DVD

Ubuntu 9.10

on DVD

+

Free Upgrade to
Ubuntu 10.04

Ubuntu

UNLEASHED

SAMS

2010 Edition

Covering 9.10 and 10.4

Ubuntu Unleashed 2010 Edition: Covering 9.10 and 10.4

Copyright © 2010 by Sams Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

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 bold-face 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

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.

IN THIS CHAPTER

- ▶ What Is the Command Line?
- ▶ Logging Into and Working with Linux
- ▶ Getting to the Command Line
- ▶ Using the Text Editors
- ▶ Working with Permissions
- ▶ Working as Root
- ▶ Reading Documentation
- ▶ Reference

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 tty1
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 -a1R` 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 filenames 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 tty1
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`).

4

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 r001z ->'
```

After you press Enter, you see

```
linux-gnu r001z ->
```

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 `rx`, 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

& (ampersand), shell background processing, 252–253

***** (asterisk), 566

 shell pattern-matching searches, 250

 wildcards, 463

&& (double ampersand) number comparison operators, 275

== (double equal sign) string comparison operators, 272

" (double quotes), resolving variables in shell strings with embedded spaces, 264–265

= (equal sign)

 string expression operators, 267

 number comparison operators, 273

! (exclamation point)

 logical comparison operators, 271

 number comparison operators, 275

!= (is not equal) **symbol (SQL)**, 506

? (question mark)

 shell pattern-matching searches, 250

' (single quotes), maintaining shell strings with unexpanded variables, 265–266

!= string expression operators, 267, 272

run-parts lines (/etc/crontab files), 245

#! (shebang lines), 256–258, 539

+ operator (Python), 566–567

-a logical comparison operators, 271

-d file comparison operators, 270, 274

-e file comparison operators, 274

-eq number comparison operators, 268

-f file comparison operators, 270, 274

-le number comparison operators, 268

-lt number comparison operators, 269

- n string expression operators, 267
- ne number comparison operators, 269
- o file comparison operators, 274
- o logical comparison operators, 271
- qe number comparison operators, 268
- qt number comparison operators, 269
- r file comparison operators, 270, 274
- s file comparison operators, 270
- w file comparison operators, 270, 274
- x file comparison operators, 270, 274
- z file comparison operators, 274
- z string expression operators, 267
- .htaccess Apache Server configuration file
 - AllowOverrides configuration directive, 421
 - Options configuration directive, 420–421
- .NET
 - advantages of, 625–626
 - Mono, compared, 625
- /bin/netstat network configuration tool, 346–347
- /configs subdirectory (/usr/src/linux-2.6 directory), 702
- /etc/at.deny files, 244
- /etc/crontab files, 245
- /etc/cups directory, 186
- /etc/dhcp3/dhcpd.conf file, DHCP server configuration, 354
- /etc/dhcp3/dhcpd.leases files, 354
- /etc/fstab file, 230
- /etc/host.conf network configuration files, 349
- /etc/hosts network configuration files, 347
- /etc/init.d/apache2 script, starting/stopping Apache Server, 414–416
- /etc/inittab files, determining runlevels, 237
- /etc/modprobe.conf files, 340, 707
- /etc/nsswitch.conf network configuration files, 348
- /etc/printcap, manual editing, 187
- /etc/resolv.conf network configuration files, 348
- /etc/samba/smb.conf files, Samba manual configuration, 387–389

- /etc/services network configuration files, 348
- /etc/squid/squid.conf configuration file, 491
- /etc/vsftpd.banned emails configuration file, 446
- /etc/vsftpd.chroot list configuration file, 446
- /etc/vsftpd.user list configuration file, 446
- /sbin/ifconfig network configuration tool, 342–345
- /sbin/route network configuration tool, 345–346
- /usr directory, subdirectories, 56
- /usr/src/linux-2.6 directory
 - /configs subdirectory, 702
 - arch directory, 704
 - Documentation directory, 703
 - kernel source code, 704
 - kernel testing programs, 704
 - scripts directory, 704
- /usr/src/linux-2.6/ sound card documentation directory, 154, 177
- /usr/src/linux-2.6/configs directory, 713
- /usr/X11/man directory, 56
- /var/log/vsftpd.log configuration file, 446
- /var/log/xferlog files, 467–469
- /var/spool/cron directories, 246
- 00-INDEX files, 703
- 1000BASE-T NIC (network interface cards), 336
- 1000BASE-X NIC (network interface cards), 336
- 1000BASE-X, 336
- 100BASE-T NIC (network interface cards), 335
- 10BASE-T NIC (network interface cards), 335

A

- a2p filters, 557
- AbiWord (GNOME Office), 146–147
- abiword command, 151
- ac command, 219, 231
- accept command, 194

access

Apache Web server file systems, controlling, 422–425

Calc (OpenOffice.org), 141

environment variables, 92–95

login, 90

remote access, 91

text-based console, 90

Writer (OpenOffice.org), 138

access control, configuring wu-ftpd servers, 449–452

access control directives (ftppass configuration file)

anonymous user access, limiting, 449

host's server access, blocking, 450

invalid password entries, limiting number of, 451–452

number of users in classes, 451

permissions, restricting based on Group IDs, 450–451

user classes, defining, 449–450

access permissions (files), changing, 669

access points, wireless network security, 643

ACID (Atomicity Consistency Isolation Durability), database compliance comparisons, 509

ACLs (access control lists), Squid proxy server, 491–495

ACPI (Advanced Configuration and Power Interface), Ubuntu power management, 43

actions (PHP loops), 594

Adblock Plus, 115

add-on postfixes/prefixes, 461

Add/Remove Applications, software management, 689

adding

sudo command, 229

users, 218–219

address-based virtual hosts, 432

administration tools, LDAP, 532–533

Administrator Mode button (kdm), 73

ADSL modems, 363

adsl-stop command, 366

Advanced Linux Sound Architecture (ALSA) sound card drivers, 154–177

afio backup software, 316

alias command, 459

aliases, myenv shell script, 254

allow directive (Apache Web server), 422–423

AllowOverrides configuration directive (Apache Web server), 421

Amanda backup application, 315

amdump command, 322

American Registry for Internet Numbers website, 329

ampersand (&), shell background processing, 252–253

AND statements (SQL), 506–507

anon mkdir write enable setting (vstfpt server anonymous access control), 445

anon other write enable setting (vstfpt server anonymous access control), 445

anon upload enable setting (vstfpt server anonymous access control), 445

anonymous access, controlling vsftpd servers, 446

anonymous enable setting (vstfpt server anonymous access control), 445

anonymous FTP servers, 439–440

configuring, 448

Apache Group website, 408

Apache server

installing

file locations, 412

from APT, 409

from APT, 410

quick start guide, 412

runtime configuration

configuration directives, DirectoryIndex, 419

- configuration directives, DocumentRoot, 418
- configuration directives, Listen, 417
- configuration directives, ServerAdmin, 418
- configuration directives, ServerName, 418
- configuration directives, ServerRoot, 417
- configuration directives, User, 417–418
- configuration directives, UserDir, 419
- configuration directives, 416–417, 420
- httpd.conf configuration file, 416
- MPM, 419

- security report websites, 409

- source code website, 410

- source code, building

- via configure script, 411

- via ln command, 411–412

- via rpm command, 411–412

- starting/stopping

- manually starting, 413–414

- via /etc/init.d/apache2 script, 414–416

- upgrading file locations, 410

Apache Software Foundation, 408

Apache Web server

- development of, 407–408

- documentation websites, 408

- downloading, 409

- file system access control, 425

- via allow/deny directives, 422–423

- file system authentication

- AuthGroupFile directive, 425

- AuthName directive, 425

- AuthType directive, 425

- AuthUserFile directive, 425

- htpasswd command, 424

- Internet security, 422

- logging

- common log format, 434

- CustomLog directive, 435

- mod_access module, 427

- mod_alias module, 427

- mod_asis module, 427

- mod_auth module, 428

- mod_auth_anon module, 428

- mod_auth_dbm module, 428

- mod_auth_digest module, 428

- mod_autoindex module, 429

- mod_cgi module, 429

- mod_dir module, 429

- mod_env module, 429

- mod_expires module, 429

- mod_headers module, 429

- mod_include module, 429

- mod_info module, 430

- mod_log_config module, 430

- mod_mime module, 430

- mod_mime_magic module, 430

- mod_negotiation module, 430

- mod_proxy module, 430

- mod_rewrite module, 430

- mod_setenvif module, 430

- mod_sll module, 431

- mod_speling module, 431

- mod_status module, 431

- mod_unique_id module, 431

- mod_userdir module, 431

- mod_usertrack module, 431

- mod_vhost_alias module, 431

- mod_vhost_alias module, 432

- optimizing, 656–658

- runtime configuration

- configuration directives, AllowOverrides, 421

- configuration directives, Group, 417–418
- configuration directives, Options, 420–421
- usage statistics, 407
- version information, 409
- virtual hosting
 - address-based hosts, 432
 - intranet websites, 433
 - name-based hosts, 432–434
- apache2ctl command, 437**
- append operator, 251**
- Application.Run() method, 633**
- applications, system-config-display, 57**
- Applications menu, software management, 689**
- APT (Advanced Package Tool)**
 - Apache Server, installing, 409–410
 - deb file storage, 696
 - software management, 694
 - day-to-day usage, 694–697
 - finding software, 697–698
- apt-cache search tool, 697**
- apt-get (command-line package-management tool), 699**
- apt-get command, 760**
- apt-get dist-upgrade command, 695**
- apt-get remove command, 696**
- apt-get-install command, 696**
- apt-get-update command, 694–696**
- aptitude command, 30**
- ar command, 622**
- arch directory (/usr/src/linux-2.6 directory), 704**
- archives, 86**
- ark backup tool, 313**
- ark command, 322**
- array functions (PHP)**
 - array keys(), 600
 - array unique(), 600
 - array values(), 600
 - arsort(), 600
 - asort(), 600
 - extract(), 601
 - krsort(), 600
 - ksort(), 600
 - shuffle(), 600
- arrays, 582–583**
 - Perl(), 540–541
- as command, 622**
- assessing backup needs, 303**
- assigning values**
 - to shell script variables, 258
 - to strings in Python, 564
- asterisk (*), 85**
 - shell pattern-matching searches, 250
- at command, scheduling tasks, 242–244**
- ATAPI (AT Attachment Packet Interface), 46**
- atime files, disabling, 655**
- atomicity (ACID), database compliance comparisons, 509**
- atq command, viewing batch job numbers, 244**
- atrm command, deleting batch jobs, 244**
- Audio Format FAQ, 156, 177**
- authenticated FTP servers, 439–440**
- authentication, Apache Web server file systems, 423**
 - AuthGroupFile directive, 425
 - AuthName directive, 425
 - AuthType directive, 425
 - AuthUserFile directive, 425
 - htpasswd command, 424
- AuthGroupFile directive (Apache Web server), 425**
- AuthName directive (Apache Web server), 425**
- AuthType directive (Apache Web server), 425**
- AuthUserFile directive (Apache Web server), 425**
- autoconf command, 622**

autoconfig utility (C/C++ programming language), 617

autohacking, 640

automating tasks

scheduling tasks

at command, 242–244

batch command, 243–244

cron daemon, 245–247

shell scripts

#! (shebang lines), 256–258

built-in variables, 258, 263

commands as, 253

environment variables, 258

executing, 254–255

positional parameters, accessing/retrieving command line variables, 259–261

reasons for using, 253

special characters, list of, 263–264

storing for systemwide access, 255–256

testing, 262

trojan scripts, 255

user variables, 258

variables, accessing values, 259

variables, assigning values to, 258

variables, storing strings in, 259

writing, aliases, 254

writing, comments, 253

writing, text wrapping, 253

shells

/ (backslashes) as escape characters, 266

changing, 256

Fedora Core shells list, 247–248

job-control commands, 249

maintaining shell strings with unexpanded variables, 265–266

man pages, 248

resolving variables in strings with

embedded spaces, 264–265

shell command line, background processing, 252–253

shell command line, input/output redirection, 248, 251–252

shell command line, job-control commands, 249

shell command line, pattern-matching, 248, 250

shell command line, pipes, 248, 252

shell command line, positional parameters, 259–261

~ (backticks), replacing strings with output, 266

system services

changing runlevels, 240

manually starting/stopping, 241–242

troubleshooting runlevels, 240–241

system services operation at bootup

booting to default runlevel, 237

booting to nondefault runlevel, 237–238

booting to runlevel, 236

controlling services via administrative tools, 239

init scripts, 238–239

initiating boot loading process, 234–235

loading Linux kernel, 235–236

autoresponders (email), 485

avi files, 180

Axis Linux-based cameras website, 168

B

background processing, 252–253

backgrounds (desktop), changing, 34

backing up, full backups with incremental backup strategy, 307

backports, 31

backreferencing, 606–607

backslashes (/) as escape characters, 266

backticks (`), replacing strings with output, 266**backups**

- as security, 643
- configuration files, 302
- consultants, 303
- data loss, reasons for, 302
- frequency of, determining, 303
- full backups, tar command-line backup software, 311
- hardware, 307
 - CD-RW drives, 308
 - FireWire (IEEE-1394) drives, 308
 - NAS, 309
 - tape drives, 309
 - USB drives, 308
- incremental backups
 - full backup strategies with, 306
 - tar command-line backup software, 311
- kernel, 712
- levels of, 305–306
- MBR, 320
- needs assessments, 303
- resources, availability of, 303
- restoring files from, 311–312
- software
 - afio, 316
 - Amanda backup application, 315
 - ark, 313
 - cdbbackup, 316
 - File Roller, 312
 - flexbackup, 316
 - kdat, 313
 - tar, command-line options, 310
 - tar, find command, 311
 - tar, full backups, 311
 - tar, incremental backups, 311
 - tar, restoring files from backups, 311–312

sound practices, 304**strategies**

- choosing, 307
- evaluating, 304–306
- home user strategies, 305
- inheriting, 305
- large enterprise strategies, 305
- mirroring data, 307
- principles of, 304
- RAID arrays, 307
- small enterprise strategies, 305
- small office strategies, 305

badblocks command, file system optimization, 655**bak file extensions, 27****Balsa (GNOME Office), 147****Banshee, 161****bar operator (|), pipes, 252****Base (OpenOffice.org), 135****bash shells, 247–248**

- comparison of expression
 - file comparisons, 270–271
 - logical comparisons, 271
 - logical comparisons, 272
 - number comparisons, 268–270
 - string comparisons, 267–268
- comparison of expression, 267
- test command
 - file comparisons, 270–271
 - logical comparisons, 271–272
 - number comparisons, 268–270
 - string comparisons, 267–268
- test command, 267

batch command, scheduling tasks, 242–244**Beagle, searching Mono libraries, 631–634****beep codes, 235**

Behlendorf, Brian, Apache Web server development, 407

benchmarks, hard disk optimization, 652

BIOS (Basic Input Output System)

beep codes, 235

boot loading process, initiating, 234

hard disk optimization, 652

bmp graphics file format, 166

Boot Loader, 13

boot loaders

boot loading process, initiating, 235

GRUB

booting system services to nondefault
runlevel, 237-238

passwords, 238

system boots, 321

boot loading process

initiating, 234-235

via CD, 235

booting systems from generic floppy disks, 320-321

bootloaders, passwords, 643

Bootp protocol, 350

Brasero, 173-174

break keyword (Python), controlling loops, 572

break statements, 285

loops (PHP), 595

switch/case blocks (PHP), 593

Breezy Badger, 731

bridges, 339

network security, 645

broadcast addressing, 334

browsers, 114

brute-forcing, 375

Bsdfstp-ssl servers, 441

Bt series video chipsets (Brooktree), 178-179

BugTraq mailing list, 648

built-in shell commands, 109

built-in variables (shell scripts), 258, 263

viewing, 263

Bullet Proof X, 57

Bynari, 486

bzDisk directive, kernel compilation, 714

bzImage directive, kernel compilation, 714

bzip2 utility, 710

C

C programming language

C# programs

Mono, 628-629

website, 636

C/C++ programming language

development of, 613-614

gnc (GNU C compiler), 619-620

graphical development tools, 620-621

project management tools

autoconf utility, 617

debugging tools, 618-619

make command, 615-617

Subversion system, 617-618

project management tools, 614

cable (networking)

fiber optic, 338

UTP, 337

Calc (OpenOffice.org), 134

cancel command, 194

Canonical Software, 730

capturing screen images, 168

caricatures, 213

case statements, 283-284

cat command, 88

cat shell command, 666-667

CD drives

assignments, checking, 47

configuring, 46

cd shell command, 667–668

CD-RW drives, 308

assignments, checking, 47

configuring, 46

cdbackup backup software, 316

cdrecord command, 175

cdrecord—scanbus command, 171

CDs

boot loading process, 235

creating from command line, 175

uses for, 171

certification, course websites, 761–762

cervisia command, 622

changing

directories, cd shell command, 667–668

file access permissions, chmod shell command, 669

kernel symbolic links, 710

runlevels, 240

shells, 256

chattr command, file system optimization, 655

checklists, inventory, 749

chfn command, 103

chgrp command, 212

chmod command, 101, 212

chmod shell command, 669

choosing backup strategies, 307

chown command, 212

chsh command, 286

classes

defining in Python, 573

inheritance (Python), 575–577

instances, creating in Python, 573

methods, 573

object variables (Python), 574

classes (networking), netmasks, 329–330

CLI (command line interpreter)

basic commands list, 665–666

cat command, 666–667

cd command, 667–668

chmod command, 669

commands, combining, 684–685

cp command, 669

du command, 669–670

find command, 670–672

grep command, 673–674

less command, 674–676

ln command, 676–677

locate command, 677

ls command, 678–679

man command, 679

mkdir command, 680

mv command, 680

ps command, 680–681

reasons for using, 664–665

rm command, 681

screen command, 686–687

tail command, 682

top command, 682, 684

which command, 684

client/server database model, 500

clients

Glade client, 621

GNOME, 52

KDE monitoring, 298

KDevelop client, 620–621

CNs (common names), 526

multiple CN, 529

code, symbolic debugging, 619

codecs, 180

color, changing on desktops, 35

column types (SQL), 503

column-level privileges (MySQL), 512

combining commands, 684–685

command line, 79–82

- /bin and /sbin directories, 87

- /home directory, 87

- /proc directory, 87–89

- /tmp directory, 89

- /usr directory, 89

- /var directory, 89–90

- basic commands list, 665–666

- cat command, 666–667

- cd command, 667–668

- chmod command, 669

- commands, combining, 684–685

- compressing, 86–87

- cp command, 669

- database clients, 518

- du command, 669–670

- editing, 87

- find command, 670–672

- grep command, 673–674

- less command, 674–676

- ln command, 676–677

- locate command, 677

- ls command, 678–679

- man command, 679

- managing, 85–86

- mkdir command, 680

- Mono tool, 626–627

- mv command, 680

- navigating, 83–85

- ps command, 680–681

- Python scripting

- conditional statements, conditional checks, 570

- dictionaries, defining, 569

- dictionaries, indexing, 569

- dictionaries, mutability, 569

- dictionaries, 569

- for loops, 570

- functions, defining, 572

- functions, 573

- infinite loops, 571

- installing, 561

- interactive interpreter, prompts, 562

- interactive interpreter, variable-handling, 562

- lists, built-in methods, 568

- lists, copying, 568

- lists, mutability, 567

- lists, 567

- loop blocks, 571

- loops, break keyword, 572

- loops, continue keyword, 572

- multiline loops, 571

- nested lists, 567

- number-handling, floating-point values, 563

- number-handling, integers, 563

- number-handling, large numbers, 564

- number-handling, number type conversions, 564

- number-handling, numeric operators, 563

- operator overloading, 567

- script execution, 562

- strings as immutable sequences, 564

- strings, assigning value to, 564

- strings, built-in methods, 566

- strings, concatenating, 566

- strings, indexing, 565

- strings, repeating, 566

- unofficial scripts/add-ons websites, 577

- while loops, 571

- Python programming

- OOP, class definitions, 573

- OOP, class inheritance, 575– 577

- OOP, class methods, 573
- OOP, class object variables, 574
- OOP, constructor/destructor, 575
- OOP, creating class instances, 573
- standard library modules, 577
- reasons for using, 664–665
- rm command, 681
- screen command, 686–687
- software management, APT, 694– 698
- tail command, 682
- top command, 682–684
 - Ubuntu updates, 28–30
- which command, 684

command-line

- command-line processing, Perl coding example, 556

commands

- abiword, 151
- accept, 194
- apropos subject, 85
- aptitude, 30
- ar, 622–622
- autoconf, 622
- built-in shell, 109
- bunzip2, 86
- bzip2, 86
- cancel, 194
- cat filename, 85
- cervisia, 622
- change, 231
- chfn, 231
- chgrp, 231
- chmod, 231
- chown, 231
- chpasswd, 231
- chsh, 218, 231
- combining, 684–685

- cp file1 file2, 85
- cvs, 622
- designer, 622
- disable, 194
- edquota, 230
- enable, 194
- epiphany, 469
- for writing shell scripts, 286
- ftp, 469
- ftpcopy, 469
- ftpcount, 464
- ftpcp, 469
- ftprestart, 464
- ftpsht, 464, 467
- ftpwho, 464–465
- gcc, 622
- gdb, 622
- gftp, 469
- gimp, 151
- glade-3, 622
- gnnumeric, 151
- gpasswd, 214
- gprof, 619, 622
- greeting, 453
- grep string file(s), 85
- groupadd, 214
- groupdel, 214
- groupmod, 214
- groups, 231
- grpck, 214
- gunzip, 86
- gzip, 86
- kdevelop, 622
- konqueror, 469
- koshell, 151
- kspread, 151
- less filename, 85

- lftp, 469
- locate file, 85
- location, printing via top shell command, 684
- logname, 231
- lp, 194
- lpc, 194
- lpq, 194
- lprm, 194
- lpstat, 194
- make, 622
- mv file1 file2, 85
- nautilus, 470
- ncftp, 470
- newusers, 231
- oocalc, 151
- ooinpress, 151
- passive, 447
- passwd, 217, 231
- patch, 622
- planner, 151
- printenv, 92
- quotacheck, 230
- quotaoff, 230
- quotaon, 230
- repquota, 230
- rm file, 85
- rmdir dir, 85
- sftp, 470
- smbclient, 470
- splint, 622
- split, 618
- su, 231
- sudo, 231
- svn, 622
- tar, 86
- touch, 99

- useradd -D, 216
- useradd -G, 214
- useradd, 216, 231
- userdel, 217
- usermod -G, 214
- usermod, 217, 231
- vi, 97
- visudo, 227
- vsftpd, 470
- webcam, 470
- whatis, 85
- whereis, 85, 94

comments

- in shell scripts, 253
- PHP, 585

commercial support, websites, 762

common log format (Apache Web server), 434

Common Unix Printing System, 185

CommuniGate Pro, 486

comparison of expressions

- pdksh shells versus bash shells
 - file comparisons, 270-271
 - logical comparisons, 271-272
 - number comparisons, 268-270
 - string comparisons, 267-268
- tcsh shell
 - file comparisons, 274-275
 - logical comparisons, 275
 - number comparisons, 273
 - string comparisons, 272

comparison operators (Perl)

- numeric comparison operators list, 542-543
- string comparison operators list, 543

compiled languages, 614

compiling kernel, 712

- BzDisk directive, 714

- bzImage directive, 714
 - multiple kernel versions, 710
- retaining current kernel version, 713
- speeding up, 713
- troubleshooting, 721
- zImage directive, 714
- Compiz window manager, 53–54**
- components, SANE, 165**
- compound operators (Perl), 543**
- compressing directories, tar command-line backup software, 317**
- CONCAT() function, 506**
- concatenating strings in Python, 566**
- conditional checks (Python conditional statements), 570**
- conditional statements**
 - Perl
 - if/else, 546
 - overview of, 545
 - unless, 546
 - PHP, 589– 592
 - Python conditional checks, 570
- conditions (PHP loops), 594**
- configuration files, creating backups, 302**
- configure script, building Apache Server source code, 411**
- configuring**
 - anonymous FTP servers, 448
 - Apache Server
 - configuration directives, AllowOverrides, 421
 - configuration directives, DirectoryIndex, 419
 - configuration directives, DocumentRoot, 418
 - configuration directives, Group, 417–418
 - configuration directives, Listen, 417
 - configuration directives, Options, 420–421
 - configuration directives, ServerAdmin, 418
 - configuration directives, ServerName, 418
 - configuration directives, ServerRoot, 417
 - configuration directives, User, 417–418
 - configuration directives, UserDir, 419
 - configuration directives, 416–417, 420
 - httpd.conf configuration file, 416
- MPM, 419
- disk quotas, 230–231
- firewalls, 646
 - FTP servers, file-conversion actions, 460–462
- kernel
 - make config utility, 714
 - make menuconfig utility, 715–716
 - make xconfig utility, 716–717
 - RAM disk images, 720
 - selecting kernel type, 705
 - subsections of, 717–720
- MySQL
 - data directory ownership, 510
 - initializing data directories, 510
 - initializing grant tables, 510
 - root user passwords, 511
- PostgreSQL
 - initializing data directories, 514–515
- Squid proxy server
 - ACL, 491–495
 - client configuration, 490
 - examples of, 496–497
 - specifying client IP addresses, 495–496
- vsftpd servers, 445–448
- wu-ftp servers
 - ftppass file command, 448–460
 - xinetd daemons, 444
- Ubuntu
 - CD/DVD drive assignments, 46–47

- date/time resets, 44–46
- first updates, 28–30
- modems, 42–43
- power management, 43
- software repositories, 30, 33
- sudo command, 27
- troubleshooting, 26
- wireless networks, 48–49

connected user information, displaying FTP servers, 464–465

Connection Type dialog box, 190

connections (FTP servers), allowing/denying, 463

consistency (ACID), database compliance comparisons, 509

console

- basic commands list, 665–666
- cat command, 666–667
- cd command, 667–668
- chmod command, 669
- commands, combining, 684–685
- cp command, 669
- du command, 669–670
- find command, 670–672
- grep command, 673–674
- less command, 674–676
- ln command, 676–677
- locate command, 677
- ls command, 678–679
- man command, 679
- mkdir command, 680
- mv command, 680
- ps command, 680–681
- reasons for using, 664–665
- rm command, 681
- screen command, 686–687
- tail command, 682
- top command, 682–684
- which command, 684

Console.WriteLine() method, 629

constants (PHP), 584

constructor methods in Python, 575

continue keyword (Python), controlling loops, 572

control structures, 547

controllerless modems, troubleshooting, 746

convert utility, 167

converting number types in Python, 564

copying

- directory structures, tar command-line backup software, 317

files

- cp command, 318
- cp shell command, 669
- mc command-line file management software, 318–319
- tar command-line backup software, 316–317
- to remote servers, 374

lists, 568

MBR, 320

multiple files between servers, 374–375

remote files to remote servers, 374

cp command, 322

configuration file backups, 302

copying files, 318

cp shell command, 669

CPAN (Comprehensive Perl Archive Network), 551–552

modules, installing in Perl, 553–554

cpio command, 322

CPU (central processing unit), troubleshooting, 748

crackers, 640

CREATE DATABASE statement (PostgreSQL), 516

CREATE DATABASE statement (MySQL), 511

CREATE statements (SQL), 504

CREATE USER statement (PostgreSQL), 516

createdb command (PostgreSQL), 524

createuser command (PostgreSQL), 524

cron daemon

repeatedly running tasks, 245–247

scheduling tasks, 242

crontab files, editing, 246–247

CrossOver Office, 150

CUPS (Common UNIX Printing System), 185

network printing

enabling on LAN, 397–399

printer entries, creating, 400

SMB printing, 398

cupsd daemon, 187

current directory, listing files via ls shell command, 678–679

customizing Ubuntu

desktop backgrounds, 34

desktop color, 35

input devices

keyboard layouts, 40–41

keyboard shortcuts, 40

mouse configurations, 41

Preferred Applications, 37–38

Removable Drives and Media, 39

CustomLog directive (Apache Web server), 435

cvs command, 622

D

daemons

cupsd, 187

mail, 485

smbd

smbclient command, 391

smbstatus command, 390

xinetd daemons, configuring for wu-ftpd, 444

Dapper Drake, 731

data directories (MySQL)

initializing, 510

ownership of, 510

data directories (PostgreSQL), initializing, 514–515

data integrity, database comparisons, 509

data locking, database comparisons, 508

data loss, reasons for, 302

data mirroring, 307

data pilots, 143

data storage in RDBMS, 501

database administrators (DBAs), responsibilities of

data integrity, 500

database client installation/maintenance, 499

database security, 500

database clients

as middleware, 520

command-line clients, 518

local GUI database access, 520

MySQL

command-line client, 521–522

graphical clients, 523

PostgreSQL command-line client, 523

SSH database access, 518–520

web database access, 520–521

database-level privileges (MySQL), 512

databases

access/permission issues, 518

client/server model, 500

comparisons

ACID compliance, 509

data locking, 508

procedural languages, 510

speed, 507

SQL subqueries, 509

triggers, 510

comparisons, 507

flat file, 500

MySQL

adding user accounts, 512

creating, 511–512

data directory ownership, 510

granting/revoking privileges, 512–513

initializing data directories, 510

initializing grant tables, 510

installing, 510–511

root user passwords, 511

PostgreSQL

creating database users, 516

creating databases, 515

deleting database users, 517

exiting psql command-line client, 517

granting/revoking privileges, 517–518

initializing data directories, 514–515

installing, 513

RPM distribution, 513

starting postmaster program, 515

RDBMS

data storage, 501

SQL basics, 503–507

table relations, 501, 503

RDBMS, 501

date command, resetting date settings, 45

date/time resets, 44

date command, 45

hwclock command, 45

time-admin client, 46

DBAs (database administrators), responsibilities of, 499–500

dd command, 322

de Icaza, Miguel, 52

deb files, storing, 696

Debian Linux distribution, 728

debugging tools

C/C++ programming language, 618–619

symbolic debugging, 619

declarations (PHP loops), 594

defining

classes in Python, 573

dictionaries, 569

Python functions, 572

deleting

batch jobs, 244

directories via rm shell command, 681

files via rm shell command, 681

PostgreSQL database users, 517

denial of service (DOS) attacks, 88

deny directive (Apache Web server), 422–423

dependency checking, 615

deployment

applying inventories, 749–752

business applications, 738–741

checklists, 740–742

installations, 742–743

planning, 738

system considerations, 741–742

troubleshooting, 746–748

user considerations, 742

depmod command, modular kernel management, 706

description field (ftpconversions file), 462

designer command, 622

desktop environments

GNOME

File Roller, 312

optimizing, 656

KDE

ark backup tool, 313

kdat backup tool, 313

Konqueror web browser, configuring for Squid proxy server, 490

optimizing, 656

desktops

- background, changing, 34
- color, changing, 35
- GNOME, 53

destructor methods in Python, 575**development releases (kernel) versus stable kernels, 709****device drivers, 704****devices.txt files, 703****df command, 249****dhclient, configuration options, 353****dhclient command, 369****DHCP (Dynamic Host Configuration Protocol), 334**

- activating at boot time, 352
- advantages/disadvantages of, 352
- DHCP Server, 354
- dhcpd.conf files, 355–357
- identifying information pairs, 351
- IP address registration, 352
- LAN, 356
- leases, 351
- NT servers, 356
- server configuration, 354–356

DHCP Handbook website, The, 357**DHCP Server, 354****dhcpd.conf files, 355–357****Dia (OpenOffice.org), 135****dial-up Internet connections, configuring manually, 367****dictionaries in Python, 569****die function (Perl), 555****directives (configuration), Apache Server runtime configuration**

- AllowOverrides, 421
- DirectoryIndex, 419
- DocumentRoot, 418
- Group, 417–418

Listen, 417**ServerAdmin, 418****ServerName, 418****ServerRoot, 417****User, 417–418****UserDir, 419****directories, 85**

- cd shell command, changing, 667–668
- creating via mkdir shell command, 680
- deleting via rm shell command, 681
- files
- listing via ls shell command, 678–679
- structures, copying, 317
- tar command-line backup software, compressing, 317

DirectoryIndex configuration directive (Apache Server), 419**dirlist enable setting (vsftpd server default settings), 448****dirmesssage enable setting (vsftpd server default settings), 448****disable command, 194****disabling**

- atime (files), file system optimization, 655
- multiple login feature, 66
- SSH1 in SSH servers, 373

disaster recovery plans, 647–648**disk quotas, 229–231****disk usage, printing, 669–670****display banners, 455****Display Settings main screen (system-config-display client), 64****displaying**

- connected user information, 464–465
- kernel version information, 709

distribution systems, 359**DivX files, 180****dmesq command, troubleshooting Ubuntu configurations, 26**

DN (distinguished names), 526–526

do...until loops (Perl), 549

do...while loops (Perl), 549, 595

Documentation directory (/usr/src/linux-2.6 directory), 703

DocumentRoot configuration directive (Apache Server), 418

documents, HOWTO, 734

Don Becker's Linux Ethercard Status, Diagnostic and Setup Utilities website, 341

dotted decimal format, 329

double ampersand (&&) number comparison operators, 275

double equal sign (==) string comparison operators, 272

double pipe (||) number comparison operators, 275

double quotes ("), shell strings with embedded spaces, resolving variables in, 264–265

download enable setting (vsftpd server default settings), 448

downloading Apache Web server, 409

Draw (OpenOffice.org), 135

drives

CD

checking assignments, 47
configuring, 46

CD-RW

checking assignments, 47
configuring, 46

DVD

checking assignments, 47
configuring, 46

Removable Drives and Media, 39

dropdb command (PostgreSQL), 524

dropuser command (PostgreSQL), 517, 524

DSL (Digital Subscriber Line)

configuring access, 362–363

PPPoE, 363

configuring manual connections,
364–365

du shell command, 669–670

dumb gateways, 339

dummy interfaces, 327

durability (ACID), database compliance
comparisons, 509

DVD drives

assignments, checking, 47
configuring, 46
Ubuntu installations, 13

DVD+RW/-RW drives, 177, 308

DVDs

creating from command line, 176
formats of, 176

dynamic IP addresses, 361

E

e2fsck command, file system optimization, 655

echo command, 249

editing

/etc/modprobe.conf files, 340

crontab files, 247

FTP server files, 446

httpd.conf Apache Server configuration file

DirectoryIndex configuration directive,
419

DocumentRoot configuration directive,
418

Group configuration directive, 417–418

Listen configuration directive, 417

ServerAdmin configuration directive, 418

ServerName configuration directive, 418

ServerRoot configuration directive, 417

User configuration directive, 417–418

UserDir configuration directive, 419

httpd.conf Apache Server configuration file,
417

printer settings, 192

system jobs, 245–246

elinks, 437**emacs, 728****emacs, 96****email**

autoresponders, 485

daemons, 485

Evolution, 531

maildir email directory format, 473

mbox email directory format, 474

MDA, 474

choosing, 483

Fetchmail, 479–483

Procmail, 483–484

Spamassassin, 484

Squirrelmail, 484

virus scanners, 484

MTA

choosing, 474

Exim, 473

Postfix, 472, 475–478

Qmail, 473

Sendmail, 473

MUA, 474–475

pine directory format, 474

reading as root, 479

sending/receiving

overview of, 471

via Perl, 553–554

Thunderbird, 532

enable command, 194**encryption, wireless networks, 49****endless loops, 277–278****env command, 92****environment variables (shell scripts), 258****epiphany command, 469****equal sign (=), string expression operators, 267****error checking, Mono, 630–631****escape characters**

/ (backslashes) as, 266

' (single quotes) as, 265

escape sequences, PHP, 585–586**ethereal tool, 296****evaluating backup strategies, 304–305**

full backup strategies, on periodic basis, 306

full backup strategies, with incremental backups, 306

mirroring data, 307

RAID arrays, 307

simple backup strategies, 306

Evolution (GNOME Office), 147**Evolution email client, 531****Exchange Server (Microsoft), alternatives to, 485–487****exclamation points (!)**

logical comparison operators, 271

number comparison operators, 275

executing shell scripts, 254– 255**execution operators (PHP), 591****Exim MTA (mail transport agent), 473****exit statements, 285****exiting psql command-line client (PostgreSQL), 517****expressions, 85, 248**

comparison of expressions

pdksh shells versus bash shells, file comparisons, 270–271

pdksh shells versus bash shells, logical comparisons, 271–272

pdksh shells versus bash shells, number comparisons, 268–270

pdksh shells versus bash shells, string comparisons, 267–268

tcsh shells, file comparisons, 274–275

tcsh shells, logical comparisons, 275

tcsh shells, number comparisons, 273

tcsh shells, string comparisons, 272

external commands (ftpconversion file), 461
 external computer attacks, defining, 640
 extract() array function, 601
 extraversion level (kernel), 709

F

faxes, 403
 fclose() file function, 603–604
 FDDI (Fiber Distributed Data Interface)
 networks, 336
 Feisty Fawn, 731
 Fetchmail MDA (mail delivery agents), 479
 configuring
 global options, 480
 mail server options, 481
 user accounts, 481–483
 installing, 479
 fiber optic cable, 338
 fiber optic NICs (network interface cards), 336
 fields
 /var/log/xferlog, 468–469
 ftpwho command, 464
 file command, determining file compression
 methods, , 698
 file comparisons (comparison of expression),
 270–275
 file extensions, bak, 27
 file functions (PHP)
 fclose(), 603–604
 file get contents(), 602
 file handles, 603
 file put contents(), 602
 filesize(), 603
 fopen(), 602–603
 fread(), 603
 fwrite(), 603
 file get contents() file function, 602
 file handles, 603

file put contents() file function, 602
 File Roller, 312
 file sharing
 NFS
 client configuration, 384–385
 installing, 382
 server configuration, 383–384
 starting/stopping, 382
 Samba
 configuring via SWAT, 392–394
 connection status reports, 390
 installing, 386
 manual configuration via
 /etc/samba/smb.conf files, 387–389
 mounting shares, 391–392
 SMB protocol, 385
 starting smbd daemon, 390–391
 SWAT, 385
 testing via testparm command, 390

file systems
 optimizing
 badblocks command, 655
 disabling atime, 655
 e2fsck command, 655
 tune2fs command, 654
 sync command, 654

File Transfer Protocol (FTP), 439

file-conversion actions, configuring FTP servers,
 460–462

filenaming wildcards, 85

files
 /etc/fstab, 230
 access permissions, changing, 669
 assigning, 100
 contents, printing, 666–667
 copying
 cp command, 318
 cp shell command, 669

- mc command-line file management software, 318-319
- tar command-line backup software, 316-317
- current directory, listing in, 678-679
- deleting via rm shell command, 681
- directories, 101-102
- FTP server files, editing, 446
- header files, 614
- include file, 614
- linking in shell command, 676-677
- moving via mkdir shell command, 680
- multiple files, copying between servers, 374-375
- paging through output, less shell command, 674- 676
- permissions, 99
- printing last lines via tail shell command, 682
- remote files, copying to remote servers, 374
- restoring from backup, tar command-line backup software, 311- 312**
- searches
 - find shell command, 670-672
 - locate shell command, 677
 - suid/sgid, 103-104
- filesize() file function, 603**
- find command, tar command-line backup software, 311**
- find shell command, 670-672**
- find2per utility, 557**
- finger information fields, 220**
- Firefox web browser, Squid proxy server configuration, 490**
- firewalls**
 - configuring, 646
 - hosts as, 332
- FireWire (IEEE-1394) drives, 308**
- Flash plug-in (Macromedia), 181**
- flat file databases, 500**
- flexbackup backup software, 316**
- floating-point numeric values in Python, 563**
- floppy disks, system boots, 320-321**
- fopen() file function, 602- 603**
- for loops, 594**
 - Perl, 547
 - Python, 570-571
- for statements, 276-277**
- foreach loops, 594**
 - Perl, 547
- formatting spreadsheets (OpenOffice.org), 142**
- forwarding email with aliases, Postfix MTA, 478**
- fread() file function, 603**
- free command, 249**
- Free Software Foundation, 728**
- FROM clauses, SELECT statements, 505**
- FTP (File Transfer Protocol), 439**
 - administration via Wu-FTP servers
 - counting number of connections, 465
 - displaying connected user information, 464-465
 - scheduling server downtime, 466
 - viewing server transaction logs, 467
 - allowing/denying connections via ftphosts file commands, 463
 - commands list, 469
 - example of, 462-463
 - installing, 441-442
 - servers
 - administration commands, 464-469
 - allowing/denying connections, 463
 - anonymous servers, 439-440, 448
 - authenticated servers, 439-440
 - Bsdfp-ssl servers, 441
 - choosing, 439-440

- connected user information, 464–465
- editing files, 446
- file-conversion actions, 460–462
- NcFTPd servers, 440–441
- packages, 440
- vsftpd servers, 440, 445–448
- wu-ftp servers, 448–460
- users, 442–444
 - vsftpd servers, default settings, 448
 - xinetd daemons, configuring for wu-ftp servers, 444
- ftp command, 373, 469
- FTP sites, ftp.kernel.org, 709**
- ftppaccess file, 449**
 - configuring wu-ftp servers, 448–460
- ftpconversions Wu-FTPd server configuration files, 448**
- ftpcopy command, 469**
- ftpcount command, 464**
- ftpcp command, 469**
- ftphosts configuration file for allowing or denying users listing (20.3), 463**
- ftphosts file, allowing/denying FTP server connections, 463**
- ftphosts Wu-FTPd server configuration files, 448**
- ftprestart command, 464–467**
- ftpshut command, 464**
 - FTP servers, scheduling downtime, 466
 - magic cookies, 467
- ftpusers file, ftpaccess configuration file, 450**
- ftpwho -V Command Output listing (20.4), 465**
- ftpwho command, displaying connected user information, 464–465**
- full backups, 306–307**
 - tar command-line backup software, 311
- function time, tracking, 619**
- functions. See also functions (PHP)**
 - as class methods in Python, 573

- Perl
 - die, 555
 - use, 552
- Python, defining, 572–573
- shell scripts and, 286

functions (PHP)

- array functions, 600
- file functions
 - fclose(), 603–604
 - file_get_contents(), 602
 - file handles, 603
 - file_put_contents(), 602
 - filesize(), 603
 - fopen(), 602
 - fopen(), 603
 - fread(), 603
 - fwrite(), 603
- isset(), 604
- PCRE functions
 - preg_match_all(), 606
 - preg_match(), 605
 - preg_replace(), 606
- string functions
 - str_replace(), 597
 - strlen(), 596
 - strpos(), 599– 600
 - substr(), 598
 - trim(), 597
- unset(), 604
- var_dump(), 605
- var_dump(), 606
- fwrite() file function, 603**

G

games

- Battle for Wesnoth, 204
- DOOM 3, 200

- installing, 199
- overview of, 197–198
- Quake 4, 202
- Unreal Tournament 2004, 201–202
- video drivers, installing, 198–199
- Windows, playing with Cedega, 204–205
- Wolfenstein: Enemy Territory, 203
- gcc (GNU C compiler), 613, 619–722**
- gdb command (C/C++ programming language), 619, 622**
- gdm (GNOME display manager), configuring, 66–67, 71**
- gdmsetup client, 66**
- Gecko rendering engine, 115**
- GetCodecs, 180**
- gftp command, 469**
- Ghostscript, versions of, 189**
- gif graphics file format, 166**
- gigabit ethernet, 336**
- gimp command, 151**
- GKrellM tool, 296**
- Glade client, 621**
- glade-3 command, 622**
- Glade, 621**
 - global-level privileges (MySQL), 512
- GNOME (GNU Network Object Model Environment) desktop environment, 52**
 - configuring, 53–54
 - File Roller, 312
 - Glade client, 621
 - mailing lists, 767
 - optimizing, 656
- GNOME office, 146–149**
- gnome-app-install (GUI package manager), 699**
- gnome-nettool tool, 296**
- gnome-panel-screenshot utility, 168**
- gnome-system-monitor tool, 297**
- gnome-volume-properties, 39**

- GNU commands, 85**
- GNU Compiler Collection (gcc), 613**
- GNU General Public License, 727**
- Gnumeric (GNOME Office), 147**
- gnumeric command, 151**
- gocr optical character recognition client, 165**
- Google BrowserSync, 115**
- Google website, 761**
 - kernel errors, troubleshooting, 722
 - troubleshooting Ubuntu configurations, 26
- GPL (GNU General Public License), 727**
- gprof (profile) command (C/C++ programming language), 619, 622**
- Gracenote CDDB Music Recognition Service, 172**
- GRANT statement (PostgreSQL), 517**
- GRANT statement (SQL), 512– 513**
- grant tables (MySQL), initializing, 510**
- granting privileges**
 - MySQL, 512–513
 - PostgreSQL, 517–518
- Graph view, 298**
- graphical database clients (MySQL), 523**
- graphical development tools, C/C++ programming tools, 620–621**
- graphics**
 - converting, 167
 - digital cameras, 168
 - file formats, 167
 - F-Spot, 169– 170
 - GIMP (GNU Image Manipulation Program), 147
 - GTK (Gimp Tool Kit) widget set, 146
 - handheld, 168
 - man pages, 167
 - menu navigation, 164
 - scanners, 164–166
 - screen images, capturing, 168
 - websites, 183

graphics drivers, installing, 33

greater than (>) number comparison operators, 273

Green, Andy, Telnet servers, 372

greeting command, hiding FTP version information, 453

grep command, 249

grep shell command, 673–674

grep string file(s) command, 85

greplog shell script, 261–262

Group configuration directive (Apache Server), 417–418

growisofs command, 177

GRUB (Grand Unified Bootloader)

installing, 13

passwords, 238

system boots, 321

system services, booting to nondefault runlevel, 237–238

gs client, 186

gs command, 189

gThumb, 169

Gtk#, creating GUIs, 634–635

guides (Linux), 763

GUIs, creating, 634–635

Gutsy Gibbon, 731

gv command, 189

gzip command, 322

gzip packages, 710

H

hackers, 640

hard disks, optimizing

benchmarks, 652

BIOS adjustments, 652

file system adjustments, 654–655

hdparm command, 653

hard links, 676

hardware

applying inventories, 749–752

backups

CD-RW drives, 308

DVD+RW/-RW drives, 308

FireWire (IEEE-1394) drives, 308

NAS, 309

tape drives, 309

backups, 307

hard drive storage, 744

legacy hardware, 744

requirements, 743–744

testing compatibility, 745–746

troubleshooting, 746–748

Ubuntu installation requirements, 12

Hardware tab (system-config-display client), 64

hashes (Perl), 540–541

hdparm command, hard disk optimization, 653

hdx=ide-scsi command, hard disk optimization, 652

header files, 614

here operator, 251

hidden processes, viewing, 298

Hoary Hedgehog, 731

home user backup strategies, 305

Horde, 487

hosting file systems, 756–757

HOWTO documents, 734

DVD+RW/+R/-R[W], 176

LILO configuration, 722

Network Boot and Exotic Root, 643

HTML, PHP, 607–608

htpasswd command (Apache Web server), creating user files, 424

httpd, 437

httpd.conf Apache Server configuration file, editing, 416

DirectoryIndex configuration directive, 419

DocumentRoot configuration directive, 418

- Group configuration directive, 417–418
- Listen configuration directive, 417
- ServerAdmin configuration directive, 418
- ServerName configuration directive, 418
- ServerRoot configuration directive, 417
- User configuration directive, 417–418
- UserDir configuration directive, 419

hubs, uplink ports, 338

hwclock command, resetting time settings, 45

I

- IBM Linux commercial support website, 762
- IDE drives as SCSI drive emulators, 172
- ide.txt files, 703
- idebus=xx command, hard disk optimization, 652
- identifying information pairs (DHCP), 351
- index=autotune command, hard disk optimization, 652
- index=dma command, hard disk optimization, 652
- if statements, 281–282
- if/else conditional statements (Perl), 546
- ifconfig command
- loopback interface availability, 326
- ifconfig command, 329, 342–345, 358, 369
- lftp command, 469
- ImageMagick, 167
- immutable sequences, strings as (Python), 564
- implementing disk quotas, 230
- Impress (OpenOffice.org), 135
- in-line kernels, 705
- include file, 614
- include keyword (PHP), 596
- incremental backups, 307
 - full backup strategies with, 306
 - tar command-line backup software, 311
- indexing**
 - dictionaries, 569
 - strings in Python, 565
- infinite loops, 593**
 - in Python, 571
- information service/information technology (IS/IT) environments, 729**
- inheritance (classes) in Python, 575–577**
- inheriting backup strategies, 305**
- inheriting the environment, 225**
- init scripts, 238–239**
- initdb program, Postgre SQL data directory permissions, 515**
- initializing network hardware, 339**
 - editing /etc/modprobe.conf files, 340
 - manually loading kernel modules, 341
- initiating boot loading process, 234–235**
- input devices**
 - keyboards
 - layouts, 40–41
 - shortcuts, 40
 - mouse configurations, 41
- input strings, grep shell command, 673–674**
- input/output redirection, 248–252**
- InputDevice section (xorg.conf configuration file), 60**
- INSERT statements (SQL), 504–505**
- insmod command, modular kernel management, 706**
- installing**
 - Apache Server
 - file locations, 412
 - from APT, 409–410
 - Apache Web server, 409
 - CD-ROMs, 753–754
 - CPAN modules in Perl, 553–554
 - FTP software, 441–443
 - GIMP, 164

- graphics drivers, 33
- GRUB, 13
- MySQL, 510–511
- OpenOffice.org, 136
- partitioning, 754–757
- PostgreSQL, 513
- precompiled kernels, 708
- preparing for, 750–753
- Python, 561
- Squid proxy server, 490
- Tripwire security tool, 644

Ubuntu

- CD installations, 13
- distribution options, 12
- DVD installations, 13
- first updates, 20–22
- GRUB, 13
- hardware requirements, 12
- partition strategies, 12–13
- partitioning options, 17
- passwords, 18–19
- step-by-step guide, 14–20

integers in Python, 563

Intel-based Linux distribution websites, 764

interactive interpreter (Python), 562

interfaces

- GUIs, creating, 634–635
- Konqueror, 115

internal computer attacks, 639–640

internal network websites, Apache Web server virtual hosting, 433

Internet. See also Internet connectivity

- email, 115
- email, 120
- Evolution, 116–118
- Firefox, 114–115, 122
- Instant Messaging (with Pidgin), 123–125

- IRC, 124–126

- KMail, 120–121

- Liferea, 122

- overview of, 113

- Pan news client newsreaders, 128–130

- RSS readers, 120

- Thunderbird, 118

- Usenet newsgroups, 127–128

- videoconferencing (with Edge), 130–131

Internet connectivity

- AOL, Linux, 360

- dial-up connections, configuring manually, 367

DSL

- configuring access, 362–363

- PPPoE, 363–365

ISP

- assigning, 361

- dynamic IP addresses, 361

- Linux support, 360

- static IP addresses, 361

- troubleshooting, 368

Internet Relay Chat (IRC) clients, 225

intranet websites, Apache Web server virtual hosting, 433

IP (Internet Protocol), 328

IP addresses

- DHCP, 352

- dynamic, 361

- Squid proxy server, 495–496

- static, 361

IP masquerading, 331

IPP (Internet Printing Protocol), 185

IPv4 (Internet Protocol version 4) addressing, 329–331

IPv6 (Internet Protocol version 6) addressing, 331

IRC (Internet Relay Chat), 225, 767–768

IS/IT (information service/information technology) environments, 729

ISC (Internet Software Consortium) website, 354

isolation (ACID), database compliance comparisons, 509

ISPs (Internet service providers)

ADSL modems, 363

assigning, 361

dynamic IP addresses, 361

Linux support, 360

static IP addresses, 361

isset() function, 604

iteration statements

break, 285

case, 283–284

exit, 285

for, 276–277

if, 281–282

repeat, 280

select, 280

shift, 281

until, 279–280

while, 277– 279

iwconfig command, 369

wireless networks, 357–358

iwlist command, wireless network, 357

iwspy command, wireless network, 357

J

job numbers (batch tasks), viewing, 244

job-control commands, 249

joe, 96

joining strings in Python, 566

Joy, Bill, 96

jpg graphics file format, 166

K

kdat backup tool, 313

KDE desktop environment, 77

ark backup tool, 313

kdat backup tool, 313

Konqueror web browser, configuring for Squid proxy server, 490

mailing lists, 767

optimizing, 656

KDevelop client, 620–621

kdevelop command, 622

kdf tool, 299

kdm (KDE display manager), configuring, 72

kernel

#! (shebang lines), 256–258

backups, 712

C programming language, 614

compiling, 712

bzDisk directive, 714

bzImage directive, 714

retaining current kernel version, 713

speeding up, 713

troubleshooting, 721

zImage directive, 714

configuring

make config utility, 714

make menuconfig utility, 715–716

make xconfig utility, 716–717

RAM disk images, 720

subsections of, 717–720

development of, 702

developmental releases, 709

device drives, 704

in-line kernels, 705

Linux kernel, security, 639

managing modules, 707

modular kernels, 705

- /etc/modprobe.conf files, 707
 - loading modules, 708
 - managing, 706– 707
 - modules, loading manually, 341
- monolithic kernels, 705
- multiple versions, compiling, 710
- optimizing, 655–656
- overview of, 702
- patched kernels, 708–710
 - multiple kernel versions, 711
 - patching methods, 711
- PID, 235
- precompiled kernels, installing, 708
- recompiling, 708
- runtime errors, troubleshooting, 722
- SCSI disk drivers, 705
- source code in /usr/src/linux-2.6 directory, 704
- source tree makefiles, 702
- sources, obtaining
 - bzip2 utility, 710
 - gzip packages, 710
- sources, obtaining, 709
- stable kernels, 709
- symbolic links, changing, 710
- testing programs in /usr/src/linux-2.6 directory, 704
- type of, selecting, 705
- versions of, 708
 - displaying version information, 709
- numbering system, 709

kernel-parameters.txt files, 703

key-based logins, 375–376

keyboards

- layouts, 40–41
- shortcuts, 40

keys function, Perl hashes, 541

keywords, reserved SQL keywords, 504

kibitz command, 286

kill-SIGHUP command, 495

KOffice, 149

konqueror command, 437, 469

Konqueror web browser (KDE), configuring for Squid proxy server, 490

KOrganizer scheduling program (KOffice), 150

koshell command, 151

ksh shell, 247

ksort() array function, 600

KSpread (KOffice), 149

kspread command, 151

ksysguard tool, 299

KWord (KOffice), 149

L

LANs

- DHCP, 356
- network printing, enabling, 397–398
- security, 332

laptops

- Linux informational websites, 765
- WinModem configurations, 42–43

large enterprise backup strategies, 305

last statments (Perl), 548

LDAP (Lightweight Directory Access Protocol)

- adding people to, 528– 530
- CN, 526
 - multiple CN, 529
- DN, 526–527
- email clients
 - Evolution, 531
 - managing, 532–533
 - Thunderbird, 532
- functions of, 525
- ldapadd command, 529
- ldapsearch command, 530

- LDIF, 528
- nonspecific searches, 527
- OpenLDAP, 525
 - file permissions, 530
 - management tools, 532–533
- phpLDAPadmin administration tool, 533
- root users
 - assigning passwords to, 527
 - defining, 526
- server configuration, 526–527
- slappasswd tool, 527
- ldapadd command, 529**
- ldapsearch command, 530**
- LDIF (LDAP Data Interchange Format), 528**
- leases, 351**
- legacy printers, troubleshooting, 403–404**
- less command, 108**
- less shell command, 674– 676**
- less than () number comparison operators, 273**
- lftp command, 469**
- libraries, Mono, 631–635**
- licq, 124**
- LILO configuration, HOWTO document websites, 722**
- linkers, 614**
- linking files with shell command, 676–677**
- links**
 - hard links, 676
 - symlinks, 676–677
- Linux**
 - distribution websites, 764**
 - guides, 763
 - kernel
 - #! (shebang lines), 256–258
 - C programming language, 614
 - optimizing, 655–656
 - security, sysctl command, 639

- overview of, 727–728
- reasons to use, 729–730
- software modem support, 367
- versions, 728
- viruses, security, 646

Linux Documentation Project website, 763

Linux Foundation website, 727

Linux Laptop website, 765

Linux Zaurus PDA websites, 765

Listen configuration directive (Apache Server), 417

listing

- files in current directory, 678–679
- processes, 680–681

listings

- 20.1 (Shadow password file ftp user entry), 442
- 20.3 (ftphosts Configuration File for Allowing or Denying Users), 463
- 20.4 (ftpwho -V Command Output), 465
- 20.5 (Sample /var/log/xferlog File with Inbound and Outbound Logging), 469
- purging log files, 554

lists

- built-in methods, 568
- copying, 568
- in Python, 567
- mutability, 567
- nested lists, 567
- operator overloading, 567

In command, 676– 677

- Apache Server source code, building, 411–412

loading modules, 708

localhost interfaces

- checking availability of, 326
- manual configuration, 326–327

localhost interfaces, 326

locate shell command, 677

locking data, database comparisons, 508

**LogFormat statement (Apache Web server),
variables of, 434–435**

logging

in Apache Web server

common log format, 434

CustomLog directive, 435

Perl system logging, 538

**logical comparisons (comparison of expression),
271–275**

login retry delay, configuring, 71

logs, purging (Perl coding example), 554–555

lokkit command, firewall configuration, 646

long integers in Python, 564

Long Term Support (LTS) badge, 731

loop blocks, in Python, 571

loopback addressing, 330

loopback interfaces, 326

looping in Python

break keyword, 572

continue keyword, 572

for loops, 570

infinite loops, 571

loop blocks, 571

multiline loops, 571

while loops, 571

looping constructs (Perl)

do...until loops, 549

do...while loops, 549

for loops, 547

foreach loops, 547

last statements, 548

next statements, 548

redo statements, 548–549

until loops, 548

while loops, 548

loops (endless), 277–278

loops (PHP)

actions, 594

break statements, 595

conditions, 594

declarations, 594

do...while loops, 595

for loops, 594

foreach loops, 594

infinite loops, 593

while loops, 593

losing data, reasons for, 302

lp command, 194

lpc command, 194

lpq command, 194

lprm command, 194

lpstat command, 194

ls command, 101

ls shell command, 678–679

**lsmod command, modular kernel management,
706–707**

LTS (Long Term Support) badge, 731

LUG (Linux Users Groups), 762

lun (logical unit number), 172

M

magic cookies, 455

ftpsht command, 467

list of, 453–454

mail, sending in Perl, 552–553

maildir directory format (email), 473

mailing lists

GNOME, 767

KDE, 767

Ubuntu Project, 767

mailing lists, 735, 767

Mailscanner virus scanner website, 484

Mail::Sendmail module (Perl), 553–554

Main() method, 628

major version section (kernel), 709

make bzimage command, kernel compilation, 714

make clean command, kernel compilation, 714

make command (C/C++ programming language), 615–617

make command, 622, 722

make config utility, kernel configuration, 714

make dep command, kernel compilation, 713

make install command, kernel compilation, 714

make menuconfig utility, kernel configuration, 715–716

make modules command, kernel compilation, 714

make modules_install command, kernel compilation, 714

make utility, documentation website, 704

make xconfig utility, kernel configuration, 716–720

makefiles, 617, 702

targets, creating, 615–616

malicious code, trojan scripts, 255

man shell command, 679

managing

/etc/passwd file, 220–221

creating user accounts, 209–213

groups, 213–214

LDAP, 532–533

modifying batches, 224

modular kernels, 706–708

passwords, 220

power, ACPI, 43

shadow, 221–223

software

Add/Remove Applications, 689

APT, 694–698

Synaptic, 691–692

tarballs, 698

Update Manager, 693

system policies, 220

tools, 214–218

users, 216–224

manual pages, reading via man shell command, 679

manual system service, starting/stopping, 241–242

manually starting Apache Server, 413–414

MARC (Mailing listARChives) website, troubleshooting kernel errors, 722

masquerading Postfix MTA, 477

mastering (DVDs), 176

Math (OpenOffice.org), 135

math operators (Perl), 544

max clients setting (vsftpd server default settings), 448

max per ip setting (vsftpd server default settings), 448

mbox directory format (email), 474

MBR (Master Boot Record)

backups, 320

boot loading process, initiating, 235

copying, 320

restoring, 320

mc (Midnight Commander) command-line file management software, copying files, 318–319

mc command, 286

MDA (Mail Delivery Agents) , 474

choosing, 483

Fetchmail

configuring, global options, 480

configuring, mail server options, 481

configuring, user accounts, 481–483

installing, 479

Procmail, 483–484

Spamassassin, 484

Squirrelmail, 484

virus scanners, 484

methods

- Application.Run(), 633
- class methods in Python, 573
- Console.WriteLine(), 629
- constructor methods in Python, 575
- destructor methods in Python, 575
- in lists, 568
- Main(), 628
- Run(), 633
- SendAsync(), 633
- string methods in Python, 566
- WriteLine(), 629

Microsoft Exchange Server, alternatives to, 485–487

Microsoft Exchange Server/Outlook Client, 486

middleware, database clients as, 520

mini-CD Linux distribution websites, 764

minor version section (kernel), 709

mirroring data, 307

mkbootdisk command, 722

mkdir command, 101

mkdir shell command, 680

mkinitrd command, 722

mkisofs command, 174–175

modems, detecting/configuring

- serial-port modems, 42

- WinModem laptop configurations, 42–43

modifying

- su command, 225–226

- system fonts, 36

- visual effects, 37

modinfo command, modular kernel management, 706

modprobe command

- manually loading kernel modules, 341

- modular kernel management, 706

modular kernels, 705

- /etc/modprobe.conf files, 707

- loading modules, 708

- managing, 706–707

modules

- loading, 708

Perl

- installing from CPAN, 553–554

- Mail::Sendmail, 553–554

- standard modules list, accessing, 552

- Python standard library modules, 577

mod_access module (Apache Web server), 427

mod_alias module (Apache Web server), 427

mod_asis module (Apache Web server), 427

mod_auth module (Apache Web server), 428

mod_auth_anon module (Apache Web server), 428

mod_auth_dbm module (Apache Web server), 428

mod_auth_digest module (Apache Web server), 428–429

mod_cgi module (Apache Web server), 429

mod_dir module (Apache Web server), 429

mod_env module (Apache Web server), 429

mod_expires module (Apache Web server), 429

mod_headers module (Apache Web server), 429

mod_include module (Apache Web server), 429

mod_info module (Apache Web server), 430

mod_log_config module (Apache Web server), 430

mod_mime module (Apache Web server), 430

mod_mime_magic module (Apache Web server), 430

mod_negotiation module (Apache Web server), 430

mod_proxy module (Apache Web server), 430

mod_rewrite module (Apache Web server), 430

mod_setenvif module (Apache Web server), 430

mod_speling module (Apache Web server), 431
mod_ssl module (Apache Web server), 431
mod_status module (Apache Web server), 431
mod_unique_id module (Apache Web server), 431
mod_userdir module (Apache Web server), 431
mod_usertrack module (Apache Web server), 431
mod_vhost_alias module (Apache Web server), 431-432
monitoring users, 219-220
monitors, modifying, 62
Mono
 advantages of, 625-626
 C# programs, 628-629
 command-line tools, 626-627
 error checking, 630-631
 GUIs, creating, 634-635
 libraries, searching, 631-635
 MonoDevelop, 627-628
 website, 636
 parameters, printing out, 629
 references, 636
 variables, creating, 629-630
 website, 636
MonoDevelop, 627-628
 monolithic kernels, 705
 motherboards, troubleshooting, 747
 website, 636
mount command, mounting Samba shares, 391-392
mouse configurations, 41
mov files, 180
moving files via mkdir shell command, 680
Mozilla Thunderbird, 118
MP3 files, 156, 177
MPEG files, 156, 177, 180
MPlayer, 182

MPM (multiprocessing modules), Apache Server configuration, 419
mpm_common MPM (multiprocessing module), Apache Server configuration, 419
MS-DOS environments, 80
MTA (mail transport agents)
 choosing, 474
 Exim, 473
 mbox email directory format, 474
 pine directory format, 474
 Postfix
 configuring, 476-477
 forwarding email with aliases, 478
 masquerading, 477
 message delivery intervals, setting, 477-478
 relaying mail, 478
 smart hosts, 477
 Postfix, 472
 Qmail, maildir email directory format, 473
 Sendmail, 473
MUA (mail user agent), 474-475
multisession CDs, creating, 175
multicast addressing, 334
multicasting mode (NIC), 343
multidimensional arrays, 582
multimedia
 formats, 156, 177
 hardware, 180
 preformatted DVDs, 177
 related commands, 182
 sound cards, 177
 storage capacity, 176
 TV/video, 178
 websites, 182-183
multimedia applications
 capturing screen images, 168
 CDs/DVDs, burning, 170-172

- creating from command line, 174–177
- creating with graphical clients, 172
- DVD/video players, 182
- file formats, 180–181
- formatting, 156–157, 177
- GIMP, 163–164
- modifying volume, 155, 177
- personal video recorders, 181–182
- Rhythmbox, 162
- scanners, 166–167
- sound cards, 154, 177
- TV hardware, 178–180
- video, 178
- viewing, 181

multimedia applications, 153

multiple files, copying between servers, 374–375

multiple kernel versions

- compiling, 710
- patching, 711

multiple login feature, disabling, 66

multiple terminals, screen shell command, 686–687

mutability

- lists, 567
- Python dictionaries, 569

mv shell command, 680

myenv shell script

- aliases, 254
- executing, 254–255
- systemwide access, storing for, 255–256

MySQL

- access/permission issues, 518
- CONCAT() function, 506
- data directories, 510
- database clients
 - command-line clients, 518
 - graphical clients, 523

- local GUI database access, 520

- MySQL command-line client, 521–522

- SSH database access, 518–521

- databases, creating, 511–512

- grant tables, initializing, 510

- installing, 510–511

- optimizing

- measuring key buffer usage, 658–659

- query caches, 660–661

- query optimization, 661

- read buffers, 661–661

- PostgreSQL comparisons

- ACID compliance, 509

- data locking, 508

- procedural languages, 510

- speed, 507

- SQL subqueries, 509

- triggers, 510

- privileges

- column-level privileges, 512

- database-level privileges, 512

- global-level privileges, 512

- granting/revoking, 512–513

- table-level privileges, 512

- root user passwords, 511

- user accounts, adding, 512

mysql command (MySQL), 524

MySQL command-line client, 521–522

mysqladmin command (MySQL), 524– 524

- MySQL database creation, 512

MySQLGUI, website, 523

mysql_install_db command, MySQL configuration, 510–511

N

name-based virtual hosts, 432–434

nano command, 96, 286

NAS (Network Attached Storage), 309

NAT (network address translation), 332

nautilus command, 470

Nautilus, 172

navigating directories, 84

ncftp command, 470

NcFTPD servers, 440–441

needs assessments (backups), 303

Nessus, system vulnerability assessments, 641–642

nested lists, 567

NetBoot, 235

netmasks, 334

network classes, 330

netpbm tools, 167

netstat network configuration tool, 346–347

network configuration files

/etc/host.conf, 349

/etc/hosts, 347

/etc/nsswitch.conf, 348

/etc/resolv.conf, 348

/etc/services, 347

network configuration tools

ifconfig, 342–345

netstat, 346–347

network-admin, 350

activating/deactivating dial-up
Internet connections, 366–368

route, 345–346

Network Manager, configuring wireless networks, 48–49

network printers, 397

CUPS GUI, 399–400

LAN, enabling on, 397–398

SMB printing, 398

network storage, 309

network-admin network configuration tool, 342, 351, 369

dial-up Internet connections,
activating/deactivating, 366, 367

networking

bridges, 339

broadcast addressing, 334

cable

fiber optic cable, 338

UTP, 337

connections, troubleshooting, 339

DHCP

activating at boot time, 352

activating at installation, 352

advantages/disadvantages of, 352

DHCP server, 354

dhcpd.conf files, 355–357

identifying information pairs, 351

leases, 351

server configuration, 354–356

disaster recovery plans, 647–648

distribution systems, 359

FDDI, 336

hubs, 338

initializing hardware

editing /etc/modprobe.conf files, 340

manually loading kernel module, 341

initializing hardware, 339

multicast addressing, 334

netmasks, 334

NIC, 335

100BASE-T, 335

10BASE-T, 335

1000BASE-T, 336

1000BASE-X, 336

fiber optic, 336

token rings, 335

routers, 339

security

backups, 643

bridges, 645

- firewalls, 646
- passwords, 643
- physical security, 643
- Tripwire security tool, 644–645
- updates, 648
- subnetting, 333
- switches, 338
- TCP/IP, 328
 - classes, 329–330
 - IP masquerading, 331
 - IPv4 addressing, 329–331
 - IPv6 addressing, 331
 - ports, 332
- unicast addressing, 334
- uplink ports, 338
- web resources, 369
- wireless, 336
 - advantages of, 359
 - choosing available protocols, 359–360
- configuring, 48–49
- encryption, 49
 - support for, 357–358
- New Printer toolbar button, 190
- next statments (Perl), 548**
- NFS (Network File System)**
 - client configuration, 384–385
 - installing, 382
 - server configuration, 383–384
 - starting/stopping, 382
- NIC (network interface cards)**
 - 100BASE-T, 335
 - 10BASE-T, 335
 - 1000BASE-T, 336
 - 1000BASE-X, 336
 - fiber optic, 336
 - multicasting mode, 343
 - promiscuous mode, 343, 645
 - token rings, 335
 - troubleshooting, 341

- nice command, 292
- Nmap, 642
- NT servers, DHCP, 356
- number comparisons (comparison of expression), 268–273
- number type conversions in Python, 564
- numeric comparison operators (Perl), list of, 542–543
- numeric operators in Python, 563

O

- object variables, class object variables (Python), 574
- OCR (optical character recognition), 165
- octets, 329
- one-liner Perl code examples, 556
- oocalc command, 151
- oog files, 156, 177
- oointpress command, 151
- OOP (Object-Oriented Programming)**
 - Python
 - class definitions, 573
 - class methods, 573
 - classes, creating instances of, 573
 - classes, inheritance, 575–577
 - classes, object variables, 574
 - constructor/destructor methods, 575
- oowriter command, 151
- Open Sound System (OSS) sound card drivers, 154, 177**
- Open-Xchange, 486**
- OpenLDAP, 525**
 - file permissions, 530
 - management tools, 532–533
- OpenOffice.org**
 - Calc, 140–144
 - configuring, 136–137
 - development of, 135–136
 - office tools, 144–149

- overview of, 134
- spelling/hyphenation dictionaries, 137
- Writer, 137–140

OpenSSH, 440

- servers, 372
- tools, wireless network security, 643

operator overloading, 567

operators (PHP), 589. *See also* operators (Perl)

- execution operators, 591
- list of, 587–588
- ternary operators, 591

operators (Perl)

- comparison operators
 - numeric comparison operators list, 542–543
 - string comparison operators list, 543
- compound operators, 543
- math operators, 544
- miscellaneous operators, 544

optical character recognition (OCR), 165

optimizing

- Apache web server, 656–658
- file systems
 - badblocks command, 655
 - disabling atime, 655
 - e2fsck command, 655
 - tune2fs command, 654
- GNOME desktop environment, 656
- hard disks
 - benchmarks, 652
 - BIOS adjustments, 652
 - file system adjustments, badblocks command, 655
 - file system adjustments, disabling atime, 655
 - file system adjustments, e2fsck command, 655
 - file system adjustments, tune2fs command, 654

- hdparm command, 653

- KDE desktop environment, 656

- Linux kernel, 655–656

- MySQL

- measuring key buffer usage, 658–659

- query caches, 660–661

- query optimization, 661

- read buffers, 661

- table caches, 661

- SQL statements, 661

Options configuration directive (Apache Web server), 420–421

options field (ftpconversions file), 462

OR statements (SQL), 506–507

Oracle Collaboration Suite, 486

output redirection, 248–252

overclocking, 748

P

packages

- dvd+rw-tools, 176
- FTP servers, 440

packet writing, DVD burning, 177

paging through file output, less shell command, 674–676

PAM (Pluggable Authentication Modules), 223

partitioning, Ubuntu installation options, 12–13, 17

passive command, 447

passwords

- bootloaders, 643
- brute-forcing, 375
- FTP users, 442
- GRUB, 238
- LDAP root users, assigning to, 527
- MySQL root users, 511
- PostgreSQL database users, creating, 516
- Ubuntu installation, 18–19

patch command, 622

patch command, 711

patched kernels, 708– 711

pattern-matching, 248–250

pci=biosirg command, hard disk optimization, 652

PCRE (Perl-Compatible Regular Expressions), functions

preg match all(), 606

preg match(), 605

preg replace(), 606

pcregrep utility, 557

pcx graphics file format, 166

pdksh shells, 247

comparison of expression

file comparisons, 270–271

logical comparisons, 271–272

number comparisons, 268– 270

string comparisons, 267–268

comparison of expression, 267

select statements, 280

test command

file comparisons, 270–271

logical comparisons, 271–272

number comparisons, 268–270

string comparisons, 267–268

test command, 267

PEAR::DB (PHP), 608– 610

performance tuning

Apache Web server, 656–658

file systems

badblocks command, 655

disabling atime, 655

e2fsck command, 655

tune2fs command, 654

GNOME desktop environment, 656

hard disks

benchmarks, 652

BIOS adjustments, 652

file system adjustments, badblocks command, 655

file system adjustments, disabling atime, 655

file system adjustments, e2fsck command, 655

file system adjustments, tune2fs command, 654

hdparm command, 653

KDE desktop environment, 656

Linux kernel, 655– 656

MySQL

measuring key buffer usage, 658–659

query caches, 660–661

query optimization, 661

read buffers, 661

table caches, 661

SQL statements, 661

Perl

#! (she-bang), 539

appeal of, 537

code examples

command-line processing, 556

one-liners, 556

posting to Usenet, 555–556

purging logs, 554–555

sending email, 553–554

command-line errors, 539

conditional statements

if/else, 546

overview of, 545

unless, 546

development of, 537

functions

die, 555

use, 552

looping constructs

- do...until loops, 549
- do...while loops, 549
- for loops, 547
- foreach loops, 547
- last statements, 548
- next statements, 548
- redo statements, 548–549
- until loops, 548
- while loops, 548

modules

- installing CPAN modules, 553–554
- Mail::Sendmail, 553–554
- standard modules list, accessing, 552

operators

- comparison operators, 542–543
- compound operators, 543
- math operators, 544
- miscellaneous operators, 544

perldoc command, 540

perlfunc document, accessing, 540

regular expressions, 549–550

sendmail command, 552–553

shell access, 550–551

simple programming example, 538–540

string constants, 545

system logging, 538

variables

- arrays, 540–541
- hashes, 540–541
- scalars, 540
- special variables, 541

versions of, 538

perlcc (Perl compiler), 557

perldoc (Perl documentation reader utility) , 540, 557

perlfunc document, accessing, 540

permission control directives (ftppaccess configuration file)

- allow users to change file permissions, 457
- allowing users to compress files, 458
- allowing users to rename files, 458
- assigning users file-delete permissions, 458
- assigning users file-overwrite permissions, 458
- assigning/denying tar command usage permissions, 458–459
- user-created upload file permissions, 459
- Wu-FTPd server configuration, 457–459

permissions

- file access permissions, changing via chmod shell command, 669
- OpenLDAP files, 530

pgaccess command (PostgreSQL), 524

pg_ctl command (PostgreSQL), 524

Photoshop (Adobe), comparing to GIMP, 164

PHP

array functions

- array keys(), 600
- array unique(), 600
- array values(), 600
- arsort(), 600
- asort(), 600
- extract(), 601
- krsort(), 600
- ksort(), 600
- shuffle(), 600

arrays, 582–583

backreferencing, 606–607

comments, 585

conditional statements, 589–592

constants, 584

development of, 579

entering/exiting, 580

escape sequences, 585–586

file functions

`fclose()`, 603–604

`file_get_contents()`, 602

file handles, 603

`file_put_contents()`, 602

`filesize()`, 603

`fopen()`, 602–603

`fread()`, 603

`fwrite()`, 603

functions

`isset()`, 604

`unset()`, 604

`var_dump()`, 605–606

HTML forms, handling, 607–608

include keyword, 596

loops

actions, 594

break statements, 595

conditions, 594

declarations, 594

do...while loops, 595

for loops, 594

foreach loops, 594

infinite loops, 593

while loops, 593

manual page URL, 607

operators

execution operators, 591

list of, 587–588

ternary operators, 591

operators, 589

PCRE functions

`preg_match_all()`, 606

`preg_match()`, 605

`preg_replace()`, 606

PEAR::DB (PHP), 608–610

references, 584

resources, 582

string functions

`str_replace()`, 597

`strlen()`, 596

`strpos()`, 599–600

`substr()`, 598

`trim()`, 597

strings

defining, 587

variable substitution, 587

switch/case blocks, 592–593

variables, 580

arrays, 582–583

resources, 582

types of, 581

variable substitution, 587

phpgroupware, 487

phpLDAPadmin administration tool, 533

PHPProjekt, 487

physical security, 643

PIDs (process IDs), 235, 290

Squid proxy server, 492

pine directory format (email), 474

pipes, 248, 252

Planner (OpenOffice.org), 135

planner command, 151

Pluggable Authentication Modules (PAM), 223

png graphics file format, 166

podcasts, 162

poff command, disconnecting dial-up connections, 367

pon command, configuring dial-up connections manually, 367

ports

TCP/IP, 332

uplink, 338

positional arguments, 258

positional parameters, 259–261

Postfix MTA (mail transport agent), 472

configuring, 476–477

forwarding email with aliases, 478

masquerading, 477

message delivery intervals, setting,
477–478

relaying mail, 478

smart hosts, 477

PostgreSQL

access/permission issues, 518

data directories, initializing, 514–515

data locking website, 508

database clients

command-line clients, 518

local GUI database access, 520

PostgreSQL command-line client, 523

SSH database access, 518–520

web database access, 520–521

database users

creating, 516

deleting, 517

databases, creating, 515

installing, 513

MySQL comparisons

ACID compliance, 509

data locking, 508

procedural languages, 510

speed, 507

SQL subqueries, 509

triggers, 510

postmaster program, starting, 515

privileges, granting/revoking, 517–518

psql command-line client, exiting, 517

RPM distribution, 513

website, 513

|| (double pipe) string concatenation
function, 506

PostgreSQL command-line client, 523

postmaster program (PostgreSQL), starting, 515

posts to Usenet, Perl coding example, 555–556

power management (Ubuntu), configuring, 43

PowerPC-based Linux distribution websites, 764

ppd (PostScript Printer Description) files, 185

**PPP (Point-to-Point Protocol), dial-up Internet
connections, 366**

**pppconfig command, configuring dial-up
Internet connections manually, 365**

**PPPoE (Point-to-Point Protocol over Ethernet),
configuring manual connections, 363**

precompiled kernels, 708

Preferred Applications, 37–38

prefork MPM (multiprocessing module)

Apache Server configuration, 419

preg match all() PCRE function, 606

preg match() PCRE function, 605

preg replace() PCRE function, 606

prelogin banners, 453

print/fax/scan devices, troubleshooting, 403

printenv command, 92

**Printer model dialog box (system-config-printer
tool), 190**

Printer Name dialog box, 190

printers. See also printing

commands, 404

driver/printer support cross-reference
websites, 191

troubleshooting

legacy printers, 403–404

print/fax/scan devices, 403

USB printers, 403–404

printing

command location via top shell command,
684

configuring, 187–189

- disk usage, `du` shell command, 669–670
- file contents, `cat` shell command, 666–667
- GUI-based printers, 187
- last lines of files, via `tail` shell command, 682
- local printers, 189, 192
- Mono parameters, 629
- network printers
 - CUPS GUI, 399–400
 - LAN, enabling on, 397–398
 - SMB printing, 398
- overview of, 185–187
- references, 195
- resource usage, via `top` shell command, 682–684

privileges

MySQL

- column-level privileges, 512
- database-level privileges, 512
- global-level privileges, 512
- granting/revoking in, 512–513
- table-level privileges, 512
- PostgreSQL, granting/revoking in, 517–518

procedural language, database comparisons, 510

Process Listing view, 297

processes, listing via `ps` shell command, 680–681

Procmail MDA (mail delivery agents), 483–484

productivity applications, 133–134

productivity suites

- related commands, 151
- websites, 152

programming languages, C/C++

- development of, 613–614
- graphical development tools, 620–621
- project management tools, 614–619

project management tools, C/C++ programming language, 614

- autoconfig, 617
- debugging tools, 618–619
- make command, 615–617
- Subversion system, 617–618

Project Planner (GNOME Office), 149

promiscuous mode (NIC), 343, 645

prompts, interactive interpreter (Python), 562

protocols, FTP, 439

proxy servers

- defining, 489
- Squid
 - ACL, 491–495
 - client configuration, 490
 - configuration examples, 496–497
 - installing, 490
 - kill-SIGHUP command, 495
 - specifying client IP addresses, 495–496
- uses of, 489

`ps` command, 290

`ps` shell command, 680–681

psql client program, PostgreSQL database creation, 516

psql command (PostgreSQL), 524

psql command-line client, exiting, 517

purging logs, Perl coding example, 554–555

`pwd` command, 83

PXE, 235

Python

- `*` operator, 566
- `+` operator, 566–567
- conditional statements, conditional checks, 570
- dictionaries, 569
- for loops, 570
- functions, defining, 572–573
- infinite loops, 571

- installing, 561
- interactive interpreter, 562
- lists
 - built-in methods, 568
 - copying, 568
 - mutability, 567
 - nested lists, 567
 - operator overloading, 567
- loop blocks, 571
- loops
 - break keyword, 572
 - continue keyword, 572
 - multiline, 571
- number-handling
 - floating-point values, 563
 - integers, 563
 - large numbers, 564
 - number type conversions, 564
 - numeric operators, 563
- OOP
 - class definitions, 573
 - class methods, 573
 - classes, creating instances of, 573
 - classes, inheritance, 575–577
 - classes, object variables, 574
 - constructor/destructor methods, 575
- scripts, executing, 562
- standard library modules, 577
- strings
 - assigning value to, 564
 - built-in methods, 566
 - concatenating, 566
 - immutable sequences, 564
 - indexing, 565
 - repeating, 566
- unofficial scripts/add-ons websites, 577
- van Rossum, Guido, 561

- while loops, 571
- Qmail MTA (mail transport agent), 473**
- qt files, 180**
- query caches, MySQL optimization, 660–661**
- question mark (?), shell pattern-matching searches, 250**
- queues (print), creating, 190–193**

R

- RAID arrays, 307**
- RAM disk images (kernel), 720**
- RARP (Reverse Address Resolution Protocol), 334**
- raw files, 156, 177**
- rc.sysinit script, 236**
- rcp command, SSH servers, 373**
- RDBMS (Relational Database Management Systems)**
 - data storage, 501
 - SQL
 - commands, whitespace, 504
 - CREATE statements, 504
 - creating tables, 503
 - creating tables, 504
 - INSERT statements, 504–505
 - inserting data into tables, 504–505
 - reserved keywords, 504
 - retrieving data from databases, 505–507
 - SELECT statements, 505
 - table relations, 501–503
- reading man pages, 108–109**
 - man shell command, 679
- recipes (Procmail), 483**
- recompiling kernel, 708**
- recovery mode, booting into, 322**
- recovery plans, 647–648**
- redirection, 248–252**

redo statments (Perl), 548–549

references (PHP), 584

regular expressions, 430, 549–550

relational databases, 501

relaying email, Postfix MTA, 478

remote access

key-based logins, 375–376

SSH servers

configuring, 372–373

disabling SSH1, 373

ftp command, 373

rcp command, 373

scp command, 374

sftp command, 374–375

ssh-keygen command, 375–376

Telnet servers, configuring, 371

VNC, 378

XDMCP, 377

remote clients, X Window System support, 55

remote files, copying to other remote servers, 374

remote servers

copying files to, 374

security, 375–376

Removable Drives and Media, 39

removable storage media, 308

removing modules, 707

renice command, 292, 298

repeat statements, 280

repeating strings in Python, 566

repositories (software), configuring, 30–33

rescue disc, 320

reserved SQL keywords, 504

resetting date/time, 44

date command, 45

hwclock command, 45

time-admin client, 46

resource usage, printing via top shell command, 682–684

resources (PHP), 582

restoring

files from backups, tar command-line

backup software, 311–312

MBR, 320

Restricted Drivers Manager, 199

retrieving data from SQL databases, 505–507

return on investment (ROI), 729

REVOKE statement (PostgreSQL), 517–518

REVOKE statement (SQL), 513

revoking privileges

MySQL, 512–513

PostgreSQL, 517–518

RhythmBox, 157–161

ripping music tracks, 171

rm shell command, 681

rmmod command, modular kernel management, 706

ROI (return on investment), 729

root account, 104

creating, 105–106

deleting, 106

rebooting, 107

remote server security, 376

system shutdown, 106–107

root privileges, granting, 227

root users

in LDAP

assigning passwords to, 527

defining, 526

passwords, 511

route network configuration tool, 345–346

routers, 339

rpm command, Apache Server source code, building, 411–412

rsh shell, 247

Run() method, 633

run-parts command, 245

runlevels, 234–236

changing, 240

default runlevels, booting to, 237

defining, 236

nondefault runlevels, booting to, 237–238

troubleshooting, 240–241

runtime errors (kernel), troubleshooting, 722

S

s2p filters, 557

Samba

connection status reports, 390

installing, 386

manual configuration via
/etc/samba/smb.conf files

[global] section, 388

[homes] section, 388

[printers] section, 389

manual configuration via
/etc/samba/smb.conf files, 387–388

mounting shares, 391–392

SMB protocol, 385

smbd daemon, starting

smbclient command, 391

smbstatus command, 390

SWAT, 385

configuring via, 392–394

testing via testparm command, 390

Sample /var/log/xferlog File with Inbound and Outbound Logging listing (20.5), 469

scalars (Perl), 540

scanbus command, 171

Scanner Access Now Easy (SANE), 165

scanners, 403

schedulers, 186

scheduling tasks

at command, 242–244

batch command, 243–244

cron daemon, 245–247

scheduling tasks, 242

Schwartzian Transforms, 556

scp command, 322, 374

screen grabs, 168

screen shell command, 686–687

scripts directory (/usr/src/linux-2.6 directory), 704

SCSI disk drivers, 705

Seamonkeychat, 124

searching

files

find shell command, 670–672

index searches via locate shell
command, 677

input strings, grep shell command,
673–674

Mono libraries, Beagle, 631–634

pattern-matching, 248–250

Web

Google website, 761

tips for, 760

security

Apache Web server

Internet security, 422

security report websites, 409

autohacking, 640

backups, 643

bootloaders, passwords, 643

bridges, 645

data integrity, database comparisons, 509

data locking, database comparisons, 508

disaster recovery plans, 647–648

external computer attacks, defining, 640

firewalls, 646

internal computer attacks, defining,
639–640

key-based logins, 375–376

LAN, 332

Linux

virus protection, 646

sysctl command, 639

networks, physical security, 643

NICs, Promiscuous mode, 645

passwords

bootloaders, 643

brute-forcing, 375

MySQL root users, 511

PostgreSQL database user creation, 516

remote servers, root accounts, 376

remote servers, 375–376

tcpdump command, Promiscuous mode,
645

Tripwire security tool, 644–645

updates, 648

virus scanners, 646

vulnerability assessments

Nessus, 641–642

Nmap, 642

wireless networks, 642

access points, 643

encryption, 49

OpenSSH tools, 643

war driving, 643

Security Focus website, 648

SELECT statements (SQL), 505

select statements, 280

selecting

backup strategies, 307

kernel type, 705

partitioning schemes, 756

SendAsync() method, 633

sending email via Perl, 553–554

sendmail command (Perl), 552–553

Sendmail MTA (mail transport agent), 473

sequences, slices of, 565

serial-port modems, detecting/configuring, 42

**ServerAdmin configuration directive
(Apache Server), 418**

**ServerName configuration directive
(Apache Server), 418**

**ServerRoot configuration directive
(Apache Server), 417**

servers

anonymous FTP servers, configuring, 448

FTP servers

administration commands, 464–469

Bsdfp-ssl servers, 441

choosing, 439–440

connected user information, 464–465

file-conversion actions, 460–462

NcFTPd servers, 440–441

packages, 440

LDAP, configuring, 526–527

remote, security, 375–376

SSH servers

configuring, 372–373

disabling SSH1, 373

ftp command, 373

rcp command, 373

scp command, 374

sftp command, 374–375

ssh-keygen command, 375–376

versus Telnet servers, 372

Telnet servers

configuring, 371

versus SSH servers, 372

vsftpd servers, 440

anonymous access control, 446

configuring, 445–448

Web, 407

Sun ONE, 436

Zeus, website, 436

Zope, website, 436

wu-ftpd servers

configuring, 448–460

xinetd daemon configuration, 444

session writing, DVD burning, 176

Settings dialog box, 192

sftp command, 374–375, 470

sh shell, 247

Shadow Password File ftp User Entry listing (20.1), 442

shadow passwords, FTP users, 442–443

shar command, 286

SharpDevelop, 636

shebang lines (#!), 256, 258, 539

shell commands. See also shell command line; shell scripts; shells

basic commands list, 665–666

cat command, 666–667

cd command, 667–668

chmod command, 669

commands, combining, 684–685

cp command, 669

du command, 669–670

find command, 670–672

grep command, 673–674

less command, 674–676

ln command, 676–677

locate command, 677

ls command, 678–679

man command, 679

mkdir command, 680

mv command, 680

ps command, 680–681

rm command, 681

screen command, 686–687

tail command, 682

top command, 682, 684

which command, 684

shell command line

background processing, 252–253

input/output redirection, 248–252

job-control commands, 249

pattern-matching, 248–250

pipes, 248–252

positional parameters, accessing/retrieving
command line variables, 259–261

shell scripts

#! (shebang lines), 256–258

built-in variables, viewing, 258, 263

commands as, 253

environment variables, 258

executing, 254–255

functions in, 285–286

positional parameters, accessing/retrieving
command line variables, 259–261

reasons for using, 253

special characters, list of, 263–264

startx command, 253

tasks, automating, 261–262

testing, 262

trojan scripts, 255

user variables, 258

variables

accessing values, 259

assigning values to, 258

storing strings in, 259

writing

aliases, 254

comments, 253

text wrapping, 253

shells

bash

comparison of expression, 267

test command, file comparisons,
270–271

test command, logical comparisons,
271–272

- test command, number comparisons, 268, 270
- test command, string comparisons, 267–268
- test command, 267
- bash, 247–248
- break statements, 285
- case statements, 283–284
- changing, 256
- endless loops, 277–278
- escape characters, / (backslashes) as, 266
- exit statements, 285
- Fedora Core shells list, 247–248
- for statements, 276–277
- if statements, 281–282
- job-control commands, 249
- ksh, 247
- man pages, 248
- pdcksh, 247
 - comparison of expression, 267
 - select statements, 280
 - test command, 267
 - test command, file comparisons, 270–271
 - test command, logical comparisons, 271–272
 - test command, number comparisons, 268–270
 - test command, string comparisons, 267–268
- positional arguments, 258
- repeat statements, 280
- rsh, 247
- select statements, 280
- sh, 247
- shell command line
 - background processing, 252–253
 - input/output redirection, 248–252
 - job-control commands, 249
 - pattern-matching, 248–250
 - pipes, 248–252
- shift statements, 281
- strings
 - embedded spaces, resolving variables in, 264–265
 - unexpanded variables, maintaining, 265–266
- tcsh
 - repeat statements, 280
 - test command, file comparisons, 274–275
 - test command, logical comparisons, 275
 - test command, number comparisons, 273
 - test command, string comparisons, 272
- tcsh, 247
- until statements, 279–280
- while statements, 277–279
- zsh, 247
 - ` (backticks), replacing strings with output, 266
- shift statements, 281**
- shortcuts (keyboards), 40**
- shred command, 676**
- shuffle() array function, 600**
- shutdown command, 460**
- shutdown files, list of magic cookies, 460**
- shutting down Ubuntu, 24**
- simple backup strategies, 306**
- single quotes ('), maintaining shell strings with unexpanded variables, 265–266**
- Skolnick, Cliff, 407**
- slappasswd tool, 527**
 - Slashdot.org website, 657
- slices (sequences), 565**
- copying lists in Python, 568**
- small enterprise backup strategies, 305**
- small office backup strategies, 305**

small office/home office (SOHO), 729

smart gateways, 339

smart hosts, Postfix MTA, 477

SMB (Session Message Blocking) protocol, 385

printing, 398

smbclient command, Samba connections, 391

smbclient command, 470

smbd daemon, starting, 390–391

smbstatus command, Samba connection status reports, 390

SMPs (symmetric multiprocessors), 731

SMTP, MDA, 474

sockets, 346

soft links, 676

software

backups

afio, 316

Amanda backup application, 315

ark, 313

cdbbackup, 316

File Roller, 312

flexbackup, 316

kdat, 313

tar, command-line options, 310

tar, find command, 311

tar, full backups, 311

tar, incremental backups, 311

tar, restoring files from backups,
311–312

tar, 310

FTP software, installing, 441–442

managing

Add/Remove Applications, 689

APT, 694–698

Synaptic, 691–692

tarballs, 698

Update Manager, 693

repositories, configuring, 30–33

software modems, Linux support, 367

SOHO (small office/home office), 729

source code

checking, 618

kernel source code in /usr/src/linux-2.6
directory, 704

source tree (kernel), makefiles, 702

Spamassassin MDA (mail delivery agents), 484

speed, database comparisons, 507

splint command (C/C++ programming language), 618, 622

split() method, Python lists, 569

SQL (Structured Query Language)

!= (is not equal) symbol, 506

AND statements, 506–507

commands, whitespace, 504

CREATE statements, 504

databases, retrieving data from, 505–507

INSERT statements, 504–505

OR statements, 506–507

reserved keywords, 504

statements, optimizing, 661

subqueries, database comparisons, 509

tables

creating, 503–504

inserting data, 504–505

WHERE statements, 506

Squid proxy server

ACL, 491–495

client configuration, 490

client IP addresses, specifying, 495–496

configuration examples, 496–497

installing, 490

kill-SIGHUP command, 495

Squirrelmail MDA (mail delivery agents), 484

SSH servers

configuring, 372–373

ftp command, 373

- rcp command, 373
- scp command, 374
- sftp command, 374–375
- ssh-keygen command, 375–376
- SSH1, disabling, 373
- versus Telnet servers, 372

ssh-keygen command, SSH servers, 375– 376

stable kernels versus kernel development releases, 709

Stallman, Richard M., 97, 728

starting

starting/stopping

- Apache Server
 - /etc/init.d/apache2 script, 414–416
 - manually starting, 413–414
 - postmaster program (PostgreSQL), 515
- system services, manually starting/
 - stopping, 241–242

startx command, 253

statements

- conditional statements (PHP), 589–591

static IP addresses, 361

stereotypes, 213

storing

- data in RDBMS, 501
- deb files, APT, 696
- shell scripts for systemwide access,
 - 255–256
- strings in shell script variables, 259

str replace() string function, 597

string comparisons (comparison of expression), 267–272, 543

string constants (Perl), 545

string functions (PHP)

- str replace(), 597
- strlen(), 596
- strpos(), 599–600
- substr(), 598
- trim(), 597

strings

- assigning value to, 564
- built-in methods, 566
- concatenating, 566
- immutable sequences, 564
- indexing, 565
- PHP
 - defining in, 587
 - variable substitution, 587
- repeating, 566
- searches, grep shell command, 673–674
- storing in shell script variables, 259

strip postfixes, 461

strip prefixes, 461

strlen() string function, 596–600

StumbleUpon, 115

sublevel number section (kernel), 709

Subnet masks, 334

subnetting, 333

subqueries (SQL), database comparisons, 509

substr() string function, 598

Subversion system (C/C++ programming language), 617–618

sudo command, Ubuntu configurations, 27

sudo command, 105

summarizing data (Calc), 143

Sun ONE Web server, 436

support (commercial), websites, 762

svg graphics file format, 166

- svn command, 622

SWAT (Samba Web Administration Tool), 385

- Samba configurations, 392–394

switch/case blocks (PHP)

- break statements, 593

switch/case blocks (PHP), 592

switches

- gcc (GNU C compiler), 619
- uplink ports, 338

- symbolic debugging**, 619
- symbolic links (kernel), changing**, 710
- symlinks**, 676–676
- symmetric multiprocessors (SMPs)**, 731
- SYN flooding**, 89
 - SYN Stealth scans (Nmap), 642
- Synaptic**, 699
 - installed packages list, viewing, 761
 - software management, 691–692
- sync command, file system synchronization**, 654
- sysctl command**, 722
 - Linux kernel optimization, 655–656
 - Linux kernel security, 639
- sysrq.txt files**, 703
- system boots, from generic floppy disks**, 320–321
- system fonts, modifying**, 36
- system jobs**
 - /etc/crontab files, 245
 - editing, 245–246
- system logging directives (ftpdaccess configuration file)**, 538
 - logging all user-issued commands, 456–457
 - logging security violations, 457
 - redirecting logging records, 456
- system rescue**, 320
- system services**
 - bootup, operation at
 - booting to default runlevel, 237
 - booting to nondefault runlevel, 237–238
 - booting to runlevel, 236
 - controlling services via administrative tools, 239
 - init scripts, 238–239
 - initiating boot loading process, 234–235
 - loading Linux kernel, 235–236
 - runlevels, changing, 240

- system-config-display application**, 57
- system-config-display client, X Window system configuration**, 64
- system-config-printer tool**, 189–190, 194
- system-config-printer utility**, 188
- system-config-printer-tui program**, 187
- system-config-printer-tui tool**, 194
- system-monitoring tools**
 - console-based monitoring, 289–291
 - disk quotas, 296
 - disk space, 295
 - graphical processes, 296–299
 - KDE, 298–299
 - kill command, 291–292
 - priority scheduling, 292–294
 - viewing memory, 294
 - watch command, 294
 - websites, 299

T

- table caches, MySQL optimization**, 661
- table-level privileges (MySQL)**, 512
- tables**
 - RDBMS, relations in, 501–503
 - SQL tables
 - creating, 503–504
 - inserting data, 504–505
- tail shell command**, 682
- tape archives**, 86
- tape drive backups**, 309
- tar command-line backup software**
 - compressing directories, 317
 - copying directory structures, 317
 - copying files, 316–317
 - find command, 311
 - full backups, 311
 - incremental backups, 311

- options, 310

- restoring files from backups, 311–312

tar command, 309–311, 322

tar streams, 317

tarballs, 86, 698

targets, creating makefiles, 615–616

tasks, automating

- batch job numbers, viewing, 244

- scheduling tasks

- at command, 242–244

- batch command, 243–244

- cron daemon, 245–247

- scheduling tasks, 242

- shell scripts, 261–262

- #! (shebang lines), 256–258

- built-in variables, viewing, 263

- commands as, 253

- environment variables, 258

- executing, 254–255

- positional parameters, accessing/retrieving command line variables, 259–261

- reasons for using, 253

- special characters, list of, 263–264

- storing for systemwide access, 255–256

- testing, 262

- trojan scripts, 255

- user variables, 258

- variables, accessing values, 259

- variables, assigning values to, 258

- variables, storing strings in, 259

- writing, aliases, 254

- writing, comments, 253

- writing, text wrapping, 253

- shells

- ~ (backticks), replacing strings with output, 266

- / (backslashes) as escape characters, 266

- changing, 256

- Fedora Core shells list, 247–248

- job-control commands, 249

- maintaining shell strings with unexpanded variables, 265–266

- man pages, 248

- resolving variables in strings with embedded spaces, 264–265

- shell command line, background processing, 252–253

- shell command line, input/output redirection, 248–252

- shell command line, job-control commands, 249

- shell command line, pattern-matching, 248–250

- shell command line, pipes, 248–252

- shell command line, positional parameters, 259–261

- system services operation at startup

- booting to default runlevel, 237

- booting to nondefault runlevel, 237–238

- booting to runlevel, 236

- changing runlevels, 240

- controlling services via administrative tools, 239

- init scripts, 238

- init scripts, 239

- initiating boot loading process, 234–235

- loading Linux kernel, 235–236

- manually starting/stopping, 241–242

- troubleshooting runlevels, 240–241

- system services operation at startup, 234

Taylor, David, 154, 177

TCP/IP (Transport Control Protocol/Internet Protocol), networking, 328

- classes, 329–330

- IP masquerading, 331

- IPv4 addressing, 329–331

- IPv6 addressing, 331

- ports, 332

tcpdump command, Promiscuous mode, 645

tcsh shell, 247

- comparison of expression
 - file comparisons, 274–275
 - logical comparisons, 275
 - number comparisons, 273
 - string comparisons, 272
- repeat statements, 280

test command

- file comparisons, 274–275
- logical comparisons, 275
- number comparisons, 273
- string comparisons, 272

telinit command, changing runlevels, 240

Telnet servers

- configuring, 371
- versus SSH servers, 372

terminal client, 56

terminal multiplexers, 686

terminals, screen shell command, 686–687

ternary operators (PHP), 591

test command

- file comparisons, 270–275
- logical comparisons, 271–275
- number comparisons, 268–273
- string comparisons, 267–272
- test command, 267

testing

- programs (kernel) in /usr/src/linux-2.6 directory, 704
- shell scripts, 262

testparm command, testing Samba, 390

text editors

- emacs, 98–99
- vi, 95–97, 557

text wrapping, shell scripts, 253

Thunderbird email client, 532

tif graphics file format, 166

time command, 292

time-admin client, changing time/date settings, , 46

time/date resets

- date command, 45
- hwclock command, 45
- time-admin client, 46

time/date resets, 44

timewarp, 219

TiVo, 182

token ring NIC (network interface cards), 335

tools

- Mono, 626–627
- netpbm, 167
- system-config-printer, 189–190

top command, 293

top shell command, 682–684

Torvalds, Linus, 702, 727

touch command, 99

tracking function time, 619

Transmeta, Inc., 727

triggers, database comparisons, 510

trim() string function, 597

Tripwire security tool, 644–645

Trojan horses, 644

trojan scripts, 255

troubleshooting

- Internet connections, 368
- kernel compilation, 721
- kernel runtime errors, 722
- network connections, 339
- NIC, 341
- printers
 - legacy printers, 403–404
 - print/fax/scan devices, 403
 - USB printers, 403–404

runlevels, 240–241

Ubuntu configurations, 26

tune2fs command, file system optimization,
654

Tuxmobil-Mobile Unix website, 765

twm (Tab Window Manager), 76

typecasting, Python number type conversion,
564

types field (ftpconversions file), 462

U

Ubuntu

64-bit, 733

configuring

CD/DVD drive configurations, 46–47

date/time resets, 44–46

first updates, 28–30

modems, 42–43

power management, 43

software repositories, 30–33

sudo command, 27

troubleshooting, 26

wireless networks, 48–49

customizing

desktop backgrounds, 34

desktop color, 35

input devices, 40–41

mouse configurations, 41

preferred applications, 37–38

Removable Drives and Media, 39

documentation, 733–735

for business, 731–732

for home use, 732–733

history of, 730

installing

CD-based installations, 13

distribution options, 12

DVD-based installations, 13

first updates, 20–22

GRUB, 13

hardware requirements, 12

partitioning options, 12–13, 17

passwords, 18–19

step-by-step guide, 14–20

networking in, 329

overview of, 730

PPC, 733

shutting down, 24

versions, 731

Ubuntu Project

mailing lists, 767

website, 763

Ubuntu rescue disc, 320

UDP (Universal Datagram Protocol), 328

unexpanded variables (shells), 265–266

unicast addressing, 334

University of Helsinki, Finland, 727

Unix backup levels, 305–306

unless conditional statements (Perl), 546

unset() function, 604

until loops (Perl), 548

until statements, 279–280

Update Manager

package updates, 699

software management, 693

Ubuntu first updates, 20–22

updates, 20–23

network security, 648

software management, 693

upgrading

Apache Server file locations, 410

to other Ubuntu versions, 33

uplink ports, 338

USB (Universal Serial Bus)

- printers, troubleshooting, 403–404
- scanners, 166
- troubleshooting, 747

use function (Perl), 552**Usenet**

- newsgroups websites, 766
- posts to (Perl coding example), 555–556

user accounts, 81

- Fetchmail, configuring in, 481–483
- MySQL, adding to, 512

User configuration directive (Apache Server), 417–418**user information**

- FTP servers, displaying, 464–465
- wu-ftp servers, configuring, 452–455

user information directives (ftppass configuration file)

- displaying administrator email address, 455
- displaying files, 453–455
- displaying prelogin banners, 452
- last modification date notifications, 455–456

user jobs

- /var/spool/cron directories, 246
- running, 245

user variables (shell scripts), 258**useradd command, 105****UserDir configuration directive (Apache Server), 419****users**

- FTP users, 442–444
- granting, 224–229

UTP (Unshielded Twisted Pair) cable, 337**V****values function, Perl hashes, 541****van Rossum, Guido, 561****var dump() function, 605–606****variable substitution (PHP), 587****variables**

- class object variables in Python, 574
- interactive interpreter (Python), handling in, 562
- Mono, creating, 629–630
- Perl variables
 - arrays, 540–541
 - hashes, 540–540
 - special variables, 541
- PHP variables, 580
 - arrays, 582–583
 - resources, 582
 - types of, 581
 - variable substitution, 587
- shell scripts
 - built-in, 258–263
 - built-in, viewing, 263
 - environment, 258
 - storing strings in, 259
 - unexpanded variables, 265–266
 - user, 258
 - values, accessing, 259
 - values, assigning, 258

Vaults of Parnassus website, 577**vi command, 286****vi text editor, 557****video cards, X Window System updates, 61****VideoLAN HOWTO, 182****viewing**

- batch job numbers, 244
- built-in variables, 263

vim, 96**virtual file systems, 87****virtual hosts, Apache Web server**

- address-based hosts, 432

intranet websites, 433

name-based hosts, 432–434

virus scanners, 484, 646

visual effects, modifying, 37

visudo command, 227

vmstat tool, 294

VNC (Virtual Network Computing), 378

vncviewer tool, 296

vsftpd command, 470

vsftpd servers

anonymous access, controlling, 446

configuring, 445–448

vsftpd, 470, 440

vulnerability assessments (security)

Nessus, 641–642

Nmap, 642

W

w command, 249

Wall, Larry, 622

wallpaper, changing, 34

war driving, 643

wav files, 156, 177

weaknesses, assessing (security)

Nessus, 641–642

Nmap, 642

Web searches

Google website, 761

tips for, 760

Web servers

Apache

building source code, via configure script, 411

building source code, via ln command, 411–412

building source code, via rpm command, 411–412

development of, 407–408

documentation websites, 408

downloading, 409

file system access control, 422–425

file system authentication, 423–425

installing, file locations, 412

installing, from APT, 409–410

installing, 409

Internet security, 422

logging, common log format, 434

logging, CustomLog directive, 435

logging, 434

mod_access module, 427

mod_alias module, 427

mod_asis module, 427

mod_auth module, 428

mod_auth_anon module, 428

mod_auth_dbm module, 428

mod_auth_digest module, 428

mod_autoindex module, 429

mod_cgi module, 429

mod_dir module, 429

mod_env module, 429

mod_expires module, 429

mod_headers module, 429

mod_include module, 429

mod_info module, 430

mod_log_config module, 430

mod_mime module, 430

mod_mime_magic module, 430

mod_negotiation module, 430

mod_proxy module, 430

mod_rewrite module, 430

mod_setenvif module, 430

mod_sll module, 431

mod_speling module, 431

mod_status module, 431

mod_unique_id module, 431

- mod_userdir module, 431
- mod_usertrack module, 431
- mod_vhost_alias module, 431
- mod_vhost_alias module, 432
 - optimizing, 656–658
- quick start guide, 412
- runtime configuration, configuration directives, 416–421
- runtime configuration, httpd.conf configuration file, 416
- runtime configuration, MPM, 419
- security report websites, 409
- source code website, 410
- starting/stopping, manually starting, 413–414
- starting/stopping, /etc/init.d/apache2 script, 414–416
- upgrading, file locations, 410
- usage statistics, 407
- version information, 409
- via allow/deny directives, 422–423
- virtual hosting, address-based hosts, 432–433
- virtual hosting, name-based hosts, 432–434
- Sun ONE, 436
- Zeus, website, 436
- Zope, website, 436
- webcam command, 470**
- welcome.msg files, 454**
- WEP encryption, 49**
- whatis command, 85**
- WHERE clauses, SELECT statements, 505**
- WHERE statements (SQL), 506**
- whereis command, 85, 94**
- which shell command, 684**
- while loops, 593**
 - Perl, 548
 - Python, 571
- while statements, 277–279**
- whitespace, SQL commands, 504**
- wildcards, 85, 250**
- window managers, modifying, 73–76**
- Windows, productivity applications, 150–151**
- Wine, 150–151**
- WinModems, laptop configurations, 42–43**
- wireless networking, 336**
 - advantages of, 359
 - choosing available protocols, 359–360
 - configuring, 48–49
 - encryption, 49
 - security, 642
 - access points, 643
 - OpenSSH tools, 643
 - support for
 - iwconfig command, 357–358
 - iwlist command, 357
 - iwpriv command, 357
 - iwspy command, 357
 - war driving, 643
- wish command, 257**
- WITH PASSWORD segment (CREATE USER statement), 516**
- worker MPM (multiprocessing module), Apache Server configuration, 419**
- WPA Personal encryption, 49**
- WriteLine() method, 629**
- Writer (OpenOffice.org), 134**
 - options, 139
 - styles and formatting, 139–140
- writing shell scripts**
 - aliases, 254
 - comments, 253
 - text wrapping, 253

wu-ftpd servers

- access control, configuring, 449–452
- ftppass file command, 448–460
- permission control, configuring, 457–459
- system logging, configuring, 456–457
- user information, configuring, 452–455
- xinetd daemons, configuring for, 444

X-Y-Z**X Window System**

- applying, 56–57
- components list, 57–58
- components of, 57
- configuring, 62
- display managers, 56
- displayconfig-gtk client, 63–64
- distributed processing, 55
- distribution components, 56
- elements of, 55–59, 62
- hard drive requirements, 56
- Module section, 59
- Monitor section, 60
- overview of, 55
- references, 78
- ServerLayout section, 58
- starting, 64
- via display managers, 65
 - websites, 765
- xorg.conf file, 64

X-Chat, 124–125**x-x, shell pattern-matching searches, 250****X.Org Foundation, 728****X11, 51****xdm display manager, applying, 73****XDMCP (X Display Manager Control Protocol), 68, 71, 377****Xfce desktop, 78, 656****Xine, 182****xinetd daemons, configuring for wu-ftpd servers, 444****xorg.conf files**

- Device section, 60–61
- Files section, 58
- InputDevice section, 59
- Module section, 59–60
- Screen section, 61–62
- X Window System configuration, 64

Xsane scanners, 165–166**Zeus Web server**

- website, 436

zImage directive

- kernel compilation, 714

Zope Web server

- website, 436

zsh shell, 247