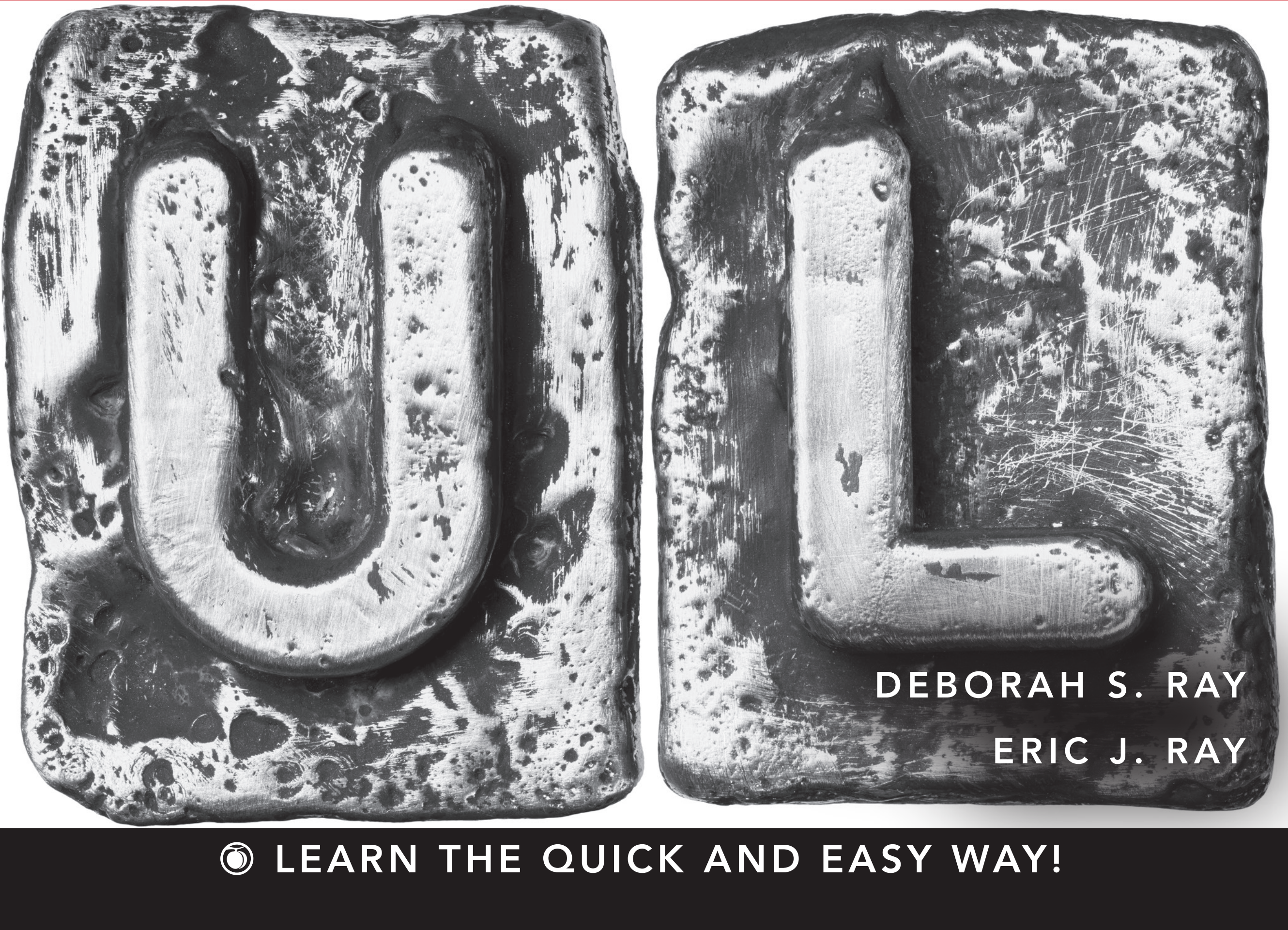
VISUAL QUICKSTART GUIDE

Unix and Linux

Fifth Edition



DEBORAH S. RAY

ERIC J. RAY

LEARN THE QUICK AND EASY WAY!

VISUAL QUICKSTART GUIDE

Unix and Linux

FIFTH EDITION

DEBORAH RAY • ERIC RAY

Peachpit Press

Visual QuickStart Guide
Unix and Linux, Fifth Edition
Deborah Ray and Eric Ray

Peachpit Press
www.peachpit.com

To report errors, please send a note to errata@peachpit.com.
Peachit Press is a dvision of Pearson Education.

Copyright © 2015 by Deborah Ray and Eric Ray

Editor: Clifford Colby
Development editor: Robyn G. Thomas
Senior production editor: Lisa Brazieal
Copyeditor: Scout Festa
Technical editor: Bruce Byfield
Compositor: Danielle Foster
Indexer: James Minkin
Cover design: RHDG / Riezebos Holzbaur Design Group, Peachpit Press
Interior design: Peachpit Press
Logo design: MINE™ www.minesf.com

Notice of Rights
All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information on getting permission for reprints and excerpts, contact permissions@peachpit.com.

Notice of Liability
The information in this book is distributed on an ”As Is” basis, without warranty. While every precaution has been taken in the preparation of the book, neither the author, Adobe Systems, Inc., nor the publisher shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Trademarks
Visual QuickStart Guide is a registered trademark of Peachpit Press, a division of Pearson Education.
Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Peachpit was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

ISBN 13: 978-0-321-99754-8
ISBN 10: 0-321-99754-9

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

Dedication

To Simon Hayes, who helped develop this book's first edition—the foundation for this book's long-term success.

Acknowledgments

This book came together with the invaluable assistance of a number of very talented and supportive people. Thanks to Clifford Colby for his continued confidence and support; it's always a treat to work with you! Our thanks to Robyn Thomas as well, who was a delight to work with and helped tremendously in pulling the various pieces together. Scout Festa was really helpful as copyeditor and was a great proofreader. Lisa Brazieal did a great job in production, even with our special needs. And our special thanks to Bruce Byfield, whose careful attention to detail and deep knowledge of the idiosyncrasies of Unix and Linux helped iron out technical rough spots. Thanks, all!

Table of Contents

Introduction ix

Chapter 1 Getting Started with Unix and Linux 1

Accessing a Unix System 3
Connecting to the Unix or Linux System 8
Logging In 11
Changing Your Password with **`passwd`** 13
Listing Directories and Files with **`ls`** 15
Changing Directories with **`cd`** 17
Finding Yourself with **`pwd`** 18
Piping Input and Output 19
Redirecting Output 21
Using Wildcards 23
Viewing File Contents with **`more`** 24
Displaying File Contents with **`cat`** 25
Exploring the System 27
Getting Help with **`man`** 28
Logging Out 30

Chapter 2 Using Directories and Files 31

Creating Directories with **`mkdir`** 32
Creating Files with **`touch`** 34
Copying Directories and Files with **`cp`** 36
Listing Directories and Files with **`ls`**
(More Goodies) 38
Moving Files with **`mv`** 41
Removing Files with **`rm`** 42
Removing Directories with **`rmdir`** 44
Finding Forgotten Files with **`find`** 46
Locating Lost Files with **`locate`** 48
Linking with **`ln`** (Hard Links) 49
Linking with **`ln -s`** (Soft Links) 51

Chapter 3 Working with Your Shell 53

Discovering Which Shell You're Using 54
Understanding Shells and Options 55
Changing Your Shell with **`chsh`** 57
Changing Your Shell Temporarily 59
Using Completion in the **`bash`** Shell 61
Viewing Session History in the **`bash`** Shell 62
Using Completion in the **`zsh`** Shell 64

Viewing Session History in the **zsh** Shell 65
Changing Your Identity with **su** 67
Fixing Terminal Settings with **stty** 69
Exiting the Shell 70

Chapter 4 Creating and Editing Files 71

Choosing an Editor: **nano**/**pico** or **vi**/**vim** 72
Starting **nano** and Dabbling with It. 75
Saving in **nano** 76
Cutting and Pasting Text Blocks in **nano** 77
Checking Spelling in **nano** 78
Getting Help in **nano** 79
Exiting **nano** 80
Starting **vi** (or **vim**) and Dabbling with It 81
Saving in **vi** 83
Adding and Deleting Text in **vi** 84
Importing Files into **vi** 85
Searching and Replacing in **vi**. 86
Exiting **vi** 88

Chapter 5 Controlling Ownership and Permissions 89

Understanding File Ownership and Permissions 90
Finding Out Who Owns What 91
Finding Out Which Group You're In 94
Changing the Group Association of Files and Directories with **chgrp** 96
Changing Ownership of Files and Directories with **chown** 98
Changing Permissions with **chmod**. 100
Translating Mnemonic Permissions to Numeric Permissions 103
Changing Permission Defaults with **umask** 105

Chapter 6 Manipulating Files 107

Counting Files and Their Contents with **wc** 108
Viewing File Beginnings with **head** 109
Viewing File Endings with **tail** 110
Finding Text with **grep** 111
Using Regular Expressions with **grep**. 112
Using Other Examples of Regular Expressions 114
Making Global Changes with **sed** 115
Changing Files with **awk** 116
Comparing Files with **cmp**. 118
Finding Differences in Files with **diff**. 119
Finding Differences in Files with **sdiff** 120
Sorting Files with **sort** 121

Eliminating Duplicates with **uniq** 122
Redirecting to Multiple Locations with **tee** 123
Changing with **tr** 124
Formatting with **fmt** 126
Splitting Files with **split** 128

Chapter 7 Getting Information About the System 131

Getting System Information with **uname** 132
Viewing File Systems with **df** 133
Determining Disk Usage with **du** 136
Finding Out File Types with **file** 137
Finding Out About Users with **finger** 138
Learning Who Else Is Logged In with **who** 141
Learning Who Else Is Logged In with **w** 142
Getting Information About Your Userid with **id** 144

Chapter 8 Configuring Your Unix or Linux Environment . . 145

Understanding Your Unix or Linux Environment 146
Discovering Your Current Environment 148
Adding or Changing Variables 150
Looking at Your **zsh** Configuration Files 152
Adding to Your **zsh** Path 156
Changing Your **zsh** Prompt 158
Looking at Your **bash** Configuration Files 161
Adding to Your **bash** Path 164
Changing Your **bash** Prompt 167
Setting Aliases with **alias** 169

Chapter 9 Running Scripts and Programs 171

Running a Command 172
Scheduling Onetime Jobs with **at** 173
Scheduling Regularly Occurring Jobs with **cron** 176
Suspending Jobs 178
Checking Job Status with **jobs** 179
Running Jobs in the Background with **bg** 180
Running Jobs in the Foreground with **fg** 181
Controlling Job Priority with **nice** 182
Timing Jobs with **time** 183
Finding Out What Processes Are Running with **ps**. . . .185
Deleting Processes with **kill** 187

Chapter 10 Writing Basic Scripts 189

Creating a Shell Script 191
Running a Shell Script. .193
Making a Script Executable 194
Getting a Head Start on Scripts with **history** 196

Embedding Commands 197
Looping Your Scripts 199
Creating If-Then Statements 201
Accepting Command-Line Arguments in
Your Scripts 204
Accepting Input While a Script Is Running 206
Debugging Scripts 208

Chapter 11 Sending and Reading Email 209

Choosing an Email Program and Getting Started 210
Reading Email with **alpine** 212
Sending Email with **alpine** 214
Customizing **alpine** 216
Reading Email with **mutt** 218
Sending Email with **mutt** 220
Reading Email with **mail** 222
Sending Email with **mail** 223
Creating a Signature File 225
Automatically Forwarding Incoming Messages 227
Announcing an Absence with **vacation** 228
Configuring **procmail**. 230
Managing Email with **procmail** 232

Chapter 12 Accessing the Internet 235

Getting Familiar with Internet Lingo 236
Logging In to Remote Systems with **ssh** 238
Logging In to Remote Systems with **telnet** 239
Communicating with Others Using **write** 241
Communicating with Others Using **talk** 242
Getting Files from the Internet with **ftp**. 243
Sharing Files on the Internet with **ftp**. 247
Surfing the Web with **links** 249
Surfing the Web with **lynx** 251
Downloading Websites with **wget** 253
Checking Connections with **ping** 254
Tracing Connections with **traceroute** 255
Matching Domain Names with IP Addresses 257

Chapter 13 Working with Encoded and
Compressed Files. 259

Encoding Files with **uuencode** 260
Decoding Files with **uudecode** 263
Archiving with **tar** 264
Unarchiving Files with **tar** 266
Compressing Files with **compress** 267

Uncompressing Files with **`uncompress`** 268
Zipping a File or Directory with **`gzip`** 269
Unzipping a **`gzip`** File with **`gunzip`**. 270
Zipping Files and Directories with **`zip`** 271
Unzipping Zipped Files with **`unzip`** 272
Combining Commands 273

Chapter 14 Using Handy Utilities 275
Calendaring with **`cal`** 276
Calculating with **`bc`** 278
Evaluating Expressions with **`expr`** 279
Converting with **`units`** 280
Looking It Up with **`look`** 281
Keeping a Record of Your Session with **`script`** 282
Getting Back to Your Place with **`screen`** 284
Acting On Found Files with **`xargs`** 286

Chapter 15 Being Root 287
Acting Like **`root`** with **`sudo`** 288
Becoming **`root`** with **`su`** 291
Starting, Stopping, and Restarting Daemons. 292
Changing the System Configuration 294
Monitoring the System 296
Keeping Up with **`watch`** 299
Checking Boot Messages with **`dmesg`**. 300
Setting the Date and Time 302

Chapter 16 Sensational Unix Tricks 303
Cleaning Up HTML Documents with **`tidy`** 304
Searching and Replacing Throughout Multiple
  Documents with **`sed`** 307
Generating Reports with **`awk`**. 310
Using Input to Customize Your Environment 311
Using ROT13 Encoding with **`sed`** 313
Embedding ROT13 Encoding in a Shell Script 315
Making Backups with **`rsync`** 318
Using Advanced Redirection with **`stderr`** 320

Appendix A Unix and Linux Reference 323

Appendix B What's What and What's Where 339

Appendix C Commands, Flags, and Arguments 343

Index 384

Introduction

Greetings, and welcome to Unix and Linux!

In this book, you'll find the information you need to get started with the operating system, advance your skills, and make Linux or Unix do the hard work for you. This book focuses on the most common Unix and Linux commands, but it also gives you ideas for working smartly and efficiently.

For the purposes of this book, Unix and Linux are pretty much interchangeable—the commands and usages are the same. You may find small differences among Unix versions or between specific Unix or Linux versions, but they'll be small indeed.

How Do You Use This Book?

We designed this book to be used as both a tutorial and a reference. If you're a Unix newbie, you should start at the beginning and work forward through the first several chapters. As you progress through the chapters, you'll build on concepts and commands you learned in previous chapters. Then, as you become more proficient, you can start choosing topics, depending on what you want to do. Be sure to reference the table of contents, the index, and the appendixes to find information at a glance.

The commands used throughout this book apply to any version of Unix (or Linux) you might be using, including Solaris, OpenIndiana, BSD, Linux (any or all), AIX or HP-UX, your OS X or Linux system at home, or any other *flavor* (that's the technical term) you can find. Heck, you can even run something very Unix-like from your Windows system with Cygwin. Or you can run the Unix or Linux of your choice under VirtualBox, VMWare, or other virtualization programs. You'll find more about flavors and getting access to Unix or Linux in Chapter 1.

By the way, if you're looking around on the Internet for information, you'll find that a number of people sidestep the awkward discussion of various flavors of Unix and Linux by using the term *Un*x*. That seems a little too geeky, even for us, so we opted out of that one. (*Un*x* sidesteps a perceived issue of trademark enforcement with the specific name Unix and manages to convey the "and Linux too, when it's like

Unix" concept as well. Useful, but geeky. In this book, we'll stick to *Unix* and *Linux*.)

Each chapter covers several topics, each of which is presented in its own section. Each section begins with a brief overview of the topic, often including examples or descriptions of how or when you'd use a command.

Next, you'll find a step-by-step list (or a couple of them) to show you how to complete a process. Note that the code you type appears as the numbered step, and a description follows it, like this:

1. **`The code you type will appear like`**
   `➝ this, in a blocky font.`

   An explanation will appear like this, in a regular font. Here, we often describe what you're typing, give alternatives, or provide cross-references to related information.

If a line of code in a numbered step is particularly long, the code might wrap to a second line. Just type the characters shown, without pressing Enter until the end of the command. Also, in code listings throughout the book, a single line of code onscreen might wrap to two lines in the book. If this happens, the continued line will start with an arrow, so it might look like this:

```
The beginning of the code starts here,
➝ but it continues on this line.
```

Sometimes you'll have to press a special key or key combination—like Ctrl+C, which means to hold down the Ctrl key and press C. We'll use this special keyboard font for these keys, but not for letters or numbers or symbols you might type.

Finally, most sections end with a couple of handy tips. Look here for ways to combine Unix and Linux commands, suggestions for using commands more efficiently, and ideas for finding out more information.

Who Are You?

We assume that you've picked up this book because you already have a need for or an interest in learning to use Unix or any Unix-like operating system, such as Linux, OpenIndiana, OS X, BSD, HP-UX, AIX, Solaris, or others.

Reasons and places to use Unix or Linux abound. Have an Android phone? Linux inside. How about a Kindle Fire tablet? Linux inside. Neighbors experimenting with a Raspberry Pi device? Yup, Linux. Your home Wi-Fi router probably contains Linux, as does your digital cable box. Unix and Linux are everywhere (even if they're hidden sometimes).

Many techie tasks are easier if you have some knowledge of Unix or Linux. For example, if you're installing Minecraft mods, knowing how your shell prompt works makes the process smoother. If you...or someone you know, that is...are rooting or flashing an Android device, knowing Linux can help there, too. And you get SuperGeek points for being able to watch movies and know that the villain is using Unix when writing that fatally flawed program to conquer the world.

We assume that

- You want to know how to use Unix or Linux to do things at work, school, or home.
- You may or may not already have experience with Unix or Linux.
- You don't necessarily have other geeky—er, techie—computer skills or experience.
- You want to learn to use Unix or Linux but probably do not want to delve into all the arcane details about the system.

In short, we assume you want to use Unix or Linux to achieve your computing goals. You want to know what you can do, get an idea of the potential that a command offers, and learn how to work smart. Very smart.

You can do all these things using this book. Basically, all you need is access to an account or system and a goal (or goals) that you want to achieve.

What Do You Need Computer-Wise?

You can learn or experiment with Unix using virtually any computer you might have available. If you're using a Mac with OS X, you're all set; it's all Unix under the hood. If you have an extra computer sitting around, even something as old as a Pentium III, you can install several different flavors of Unix or Linux, including Solaris, OpenIndiana, Ubuntu, Red Hat, or SuSE. Certainly you can install Unix or Linux on an extra hard drive (or empty space on your current hard drive) on your regular desktop computer, and generally without affecting your existing Windows configuration.

Alternatively, you can dabble in Unix less invasively by using an account on a system at work or through an Internet service provider. If you have a reasonably new computer and are concerned about not messing up what you have, though, the easiest options are to

- Use Cygwin to run Unix as part of your Windows environment
- Use VirtualBox or a similar program to run Unix in a "virtual machine" as an application in your Windows environment
- Use a bootable Unix (Linux will be the easiest) CD to experiment on without having to install anything at all on your computer (and save files to a flash drive as needed)

What Do You Need to Know to Get Started?

As you get started learning Unix or Linux, keep in mind the following Unix and Linux conventions for typing commands:

- The terminology and commands are typically arcane, cryptic, and funny-looking. For example, the command to list files or directories is just **`ls`**—short and cryptic. (The command **`ls`** is also short for **`lis`**t, as in "list the files." Many commands have these logical derivations.) We'll walk you through the commands one step at a time so that you know how to read them and apply them to your own uses. Just follow the steps in the order provided.

- Unix and Linux are case sensitive, so type commands following the capitalization used in the book.
- Whenever you type a command, you also have to press Enter. For example, if we say

1. **`funny-looking command goes here`**

   you'll type the code, then press Enter, which sends the command along to the Unix system.

   Often, we'll tell you to press a combination of keys on the keyboard, as in Ctrl+V. Here, all you do is press the Ctrl key plus the (lowercase) V key, both at the same time. Even though the keyboard uses capital letters (and, thus, the little key icons also do in this book), you would not take the extra step to capitalize the V (or whatever) in applying key combinations. If you really need to press Shift, we'll show you that.

   Some commands have flags or arguments (also called options) associated with them (you might think of flags as modifiers for the command) that give you additional control. For example, you might see the **`ls`** command used in variations like **`ls -la`** or **`ls -l -a`**. In either case, **`ls`** lists the files in a directory, the optional **`-l`** flag specifies that you want the long format, and the optional **`-a`** flag specifies all files, including hidden ones (don't worry, we'll go over this again!). Just keep in mind that flags and arguments are essentially options you can use with a given command.

- You can also put multiple commands on the same line. All you have to do is separate the commands with a semicolon (**`;`**), like this:

  ```
  ls ; pwd
  ```

  which would list the files in the current directory (**`ls`**) and find out what directory you're in (**`pwd`**)—all in one step!

So, with these things in mind, see you in Chapter 1!

Anything Else You Should Know?

Yup! Please feel free to send us a message at **`unixvqs@raycomm.com`**. We welcome your input, suggestions, and questions related to this book. Thanks, and we look forward to hearing from you!

Note to Mac Users

For simplicity, we consistently write Enter (not Return), Ctrl (not Control), and Alt (not Option), and we refer (not very often, though) to a Recycle Bin (not a Trash Can). No slight intended to those who do not use PCs or Windows—we just tried to keep the complexity of the instructions to a minimum.

4

Creating and Editing Files

Creating and editing files are likely the most common tasks you'll perform in Unix or Linux. If you're programming, developing webpages, sending email (uh-huh, really), writing a letter, configuring your environment (see Chapter 8), or just exploring the system, you'll spend a lot of time in an editor.

In this chapter, we'll introduce you to two of the most common editors: **`nano`** and **`vi`**. We'll launch this chapter with a general overview of each, and then discuss some how-tos of using each one. With the information presented here, you'll be able to choose an editor based on your needs and get started using it (or using both of them).

In This Chapter

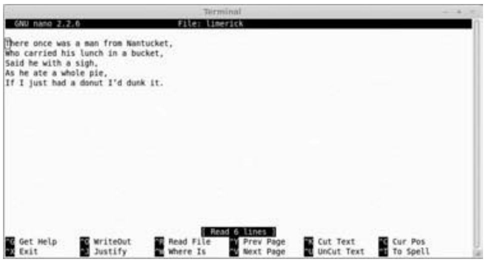
Choosing an Editor: **`nano/pico`** or **`vi`**/**`vim`**	72
Starting **`nano`** and Dabbling with It	75
Saving in **`nano`**	76
Cutting and Pasting Text Blocks in **`nano`**	77
Checking Spelling in **`nano`**	78
Getting Help in **`nano`**	79
Exiting **`nano`**	80
Starting **`vi`** (or **`vim`**) and Dabbling with It	81
Saving in **`vi`**	83
Adding and Deleting Text in **`vi`**	84
Importing Files into **`vi`**	85
Searching and Replacing in **`vi`**	86
Exiting **`vi`**	88

Choosing an Editor: `nano`/`pico` or `vi`/`vim`

Basically, all editors are designed to do the same things: enable you to create, modify, and save text files. These files could include configuration files, email messages, or shell scripts—essentially any text file you can create. Which editor you choose is up to you, depending on your specific needs and how much you're willing to learn.

In this book, we'll stick to the two biggies: **`nano`** and **`vi`**, which will likely give you all the capabilities you'll need. We chose these because **`nano`** is (arguably) the easiest Unix or Linux editor to use and **`vi`** is one of the most powerful and is available on almost every Unix or Linux system.

Which to choose? We recommend that you explore both. While you'll no doubt find **`nano`** easier to use, we highly recommend that you make a concerted effort to learn to get around in **`vi`**. You'll find that **`vi`**'s learning curve is steeper—much—but that being a skilled **`vi`** user will provide many benefits, not the least of which is that **`vi`** is really the only editor you can count on being on any Unix or Linux system you use.



```
Terminal
  GNU nano 2.2.6                File: limerick

There once was a man from Nantucket,
Who carried his lunch in a bucket,
Said he with a sigh,
As he ate a whole pie,
If I just had a donut I'd dunk it.









                               [ Read 6 lines ]
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

A **nano** offers onscreen command reminders to make it easier to use.

About nano or pico

nano is one of the more straightforward Unix editors and has become quite popular because it's extremely easy to use. In particular, as shown in **A**, it's menu-driven and intuitive. All the commands are visible, and you can open, modify, and close files with little effort. **nano** is a great choice if you're just getting started with Unix or Linux, or if you won't be needing an editor able to leap tall files in a single bound.

For a variety of reasons, mostly connected to open-source licensing issues, **nano** is a near-clone of **pico**, which in the past was included in a number of Linux/Unix distributions as well as on systems that you might be using today. The **nano** editor is command-for-command the same as **pico**, but it does offer some supplemental higher-end (yet still easy-to-use) features.

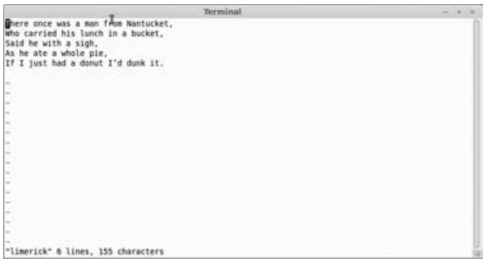
For the purposes of this book, we're going to treat **pico** and **nano** as equivalent—if you have **pico**, just mentally write that in wherever you see **nano**.

pico is distributed with the **pine** email program, so if you have **pine** available to you, you likely also have **pico**. (See Chapter 1 for a reminder on how to find out if specific programs are available to you.) If **pico** is not available to you, and if you cannot find **nano** either, ask your system administrator to install one or the other.

About **vi** or **vim**

Although **vi** is likely responsible for much of Unix's and Linux's reputation for being complicated and confusing, it offers enormous power and flexibility. Plus, **vi** is universally available (unlike **nano**), so for these two reasons, you should consider taking the time to learn it. You might find **vi** cryptic, counterintuitive, and nitpicky, and for this reason, you might want to choose a different editor if you won't require **vi**'s capabilities. As B shows, if you use **vi**, you won't have menus at your disposal—you'll have to get used to using commands like Esc**:q** or Esc**:%s/vi is arcane/vi is powerful/**.

Yes, continuing the theme from the previous section, there is an equivalent of **vi**, called **vim**, that's licensed differently and that's somewhat more powerful. For basic use—everything in this book and far more—the two are identical. In this case, though, you will always find **vi**, even if it's really **vim** (**vi** may actually be a *symlink*, or shortcut, to **vim**). If you find **vim**, though, it will assuredly be **vim**. All commands will be the same, so just dive in and enjoy.



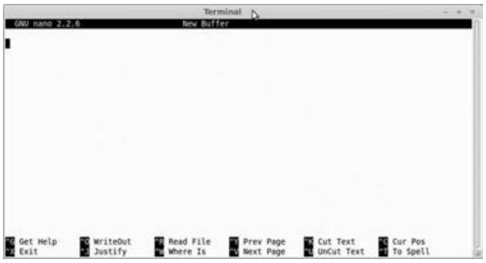
```
There once was a man from Nantucket,
Who carried his lunch in a bucket,
Said he with a sigh,
As he ate a whole pie,
If I just had a donut I'd dunk it.

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"limerick" 6 lines, 155 characters
```

B **vi** gives you a clean screen and makes you remember all of its cryptic commands.

Editors Abound

In addition to **nano** and **vi** (covered in this chapter), dozens of other editors exist:

- **emacs** offers a reasonable middle ground between the user-friendliness of **nano** and the power of **vi** (or **vim**). Do note that it's fully as powerful as **vi** for the experienced user, but it is still far easier to get started with. It's not available on all systems, though, so you'll just have to type in the command to see if you have access to it.
- **joe**, **jed**, and **e3**, which are fairly simple editors and comparable to **nano** in many ways. These are certainly worth a look if you're interested.
- **ed**, **ex**, and **red**, which are simple (in functionality, but not necessarily usage) line-by-line editors. They're probably not recommended for the novice user. We haven't had to use them in years.



A **`nano`** offers an intuitive interface for editing text.

Starting nano and Dabbling with It

You can start and dabble with **`nano`** using the following steps. Notice that the **`nano`** interface is intuitive and easy to navigate, as shown in **A**.

To start nano and dabble with it:

1. **`nano`**

   Type **`nano`** at the shell prompt. The program starts up, and you'll see something like **A**, with the text area up at the top of the window and the command hints at the bottom.

   If you know the name of the file you want to edit, type **`nano`** at the shell prompt followed by the path and name of the file you want to edit (**`hairyspiders`**, for example).

2. **`hairyspiders`**

   Go ahead. Type something—anything—just to try it out.

   - Use Delete and Backspace to help edit text.
   - Use the arrow keys to move up, down, right, or left.

TIP Start `nano` with the `-w` option (for example, `nano -w filename`) to disable word wrapping. You'll find this particularly useful when editing configuration files, as covered in Chapter 8.

TIP Throughout `nano`, you'll see `^C`, `^J`, and dozens of other `^`*`something`* characters hanging out in the menu at the bottom. The `^` stands for `Ctrl`, so `^C` is Ctrl+C, `^J` is Ctrl+J, and so on.

Saving in nano

You'll generally save your files frequently whenever you're editing them—and you should. Remember, Murphy is watching you!

To save in nano:

1. Ctrl+O

   Use Ctrl+O periodically to save ("write out") the text you're editing.

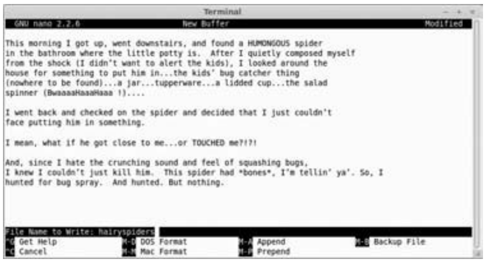
2. **`hairyspiders`**

   Specify the filename for your file A.

TIP After you save a file for the first time and want to save new changes, just press Ctrl+O and then press Enter to confirm the current filename and save it.

TIP When you exit `nano`, you'll get a last chance to save your changes. See "Exiting `nano`" in this chapter for the specifics.

TIP If you try to save a new file over an existing one—which would obliterate the original—`nano` carefully asks you if you want to overwrite the file. Answer Yes, and you'll no longer have the original; No, and you'll get to choose a new filename.

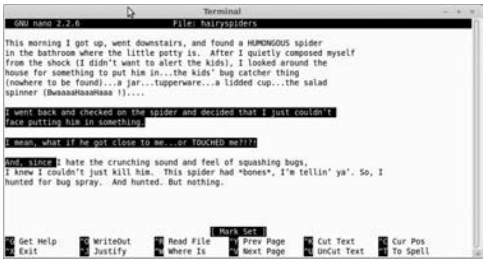


```
  GNU nano 2.2.6              New Buffer                          Modified

This morning I got up, went downstairs, and found a HUMONGOUS spider
in the bathroom where the little potty is.  After I quietly composed myself
from the shock (I didn't want to alert the kids), I looked around the
house for something to put him in...the kids' bug catcher thing
(nowhere to be found)...a jar...tupperware...a lidded cup...the salad
spinner (BwaaaaHaaaHaaa !)....

I went back and checked on the spider and decided that I just couldn't
face putting him in something.

I mean, what if he got close to me...or TOUCHED me?!?!

And, since I hate the crunching sound and feel of squashing bugs,
I knew I couldn't just kill him.  This spider had *bones*, I'm tellin' ya'. So, I
hunted for bug spray.  And hunted. But nothing.



File Name to Write: hairyspiders
^G Get Help          M-D DOS Format       M-A Append           M-B Backup File
^C Cancel            M-M Mac Format       M-P Prepend
```

A In **nano** lingo, "writing out" just means "saving."

Cutting and Pasting Text Blocks in nano

As you're typing along in **nano**, you'll probably need to cut and paste blocks of text, as shown in **A**.



```
Terminal
  GNU nano 2.2.6                File: hairyspiders

This morning I got up, went downstairs, and found a HUMONGOUS spider
in the bathroom where the little potty is.  After I quietly composed myself
from the shock (I didn't want to alert the kids), I looked around the
house for something to put him in...the kids' bug catcher thing
(nowhere to be found)...a jar...tupperware...a lidded cup...the salad
spinner (BwaaaaHaaaHaaa !)....

I went back and checked on the spider and decided that I just couldn't
face putting him in something.

I mean, what if he got close to me...or TOUCHED me?!?!

And, since I hate the crunching sound and feel of squashing bugs,
I knew I couldn't just kill him.  This spider had *bones*, I'm tellin' ya'. So, I
hunted for bug spray.  And hunted. But nothing.

                              [ Mark Set ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

A Marking, cutting, and pasting text in **nano** can be very handy.

To cut and paste text in nano:

1. **`nano hairyspiders`**

   At the shell prompt, type **nano** followed by the name of the file to edit.

2. Move the cursor to the first line of the text you want to cut.

3. Ctrl+^

   Press Ctrl+^ to mark the beginning of the text you want to cut. (Note that Ctrl+^ is really Ctrl+Shift+6—it might work without Shift, but it might not, depending on your terminal program. Try it out and see what happens.)

4. Use the arrow keys to move the cursor to the end of the text you want to cut.

   Note that the text gets highlighted as you select it **A**.

5. Ctrl+K

   This "kuts" the text.

6. Using the arrow keys, move the cursor to where you want to insert the cut text.

7. Ctrl+U

   Use this key combination to paste the cut text into the file at the new location.

TIP You can select and cut blocks of text without also pasting them back into a file. Just skip steps 6 and 7.

TIP You can paste text blocks as many times as you want. After you select and cut text, just press Ctrl+U at each place where you want to insert the cut text.

TIP If you don't select text, Ctrl+K just cuts a single line.

Checking Spelling in nano

Another handy thing you can do in **nano** is chek yoor speling, as shown in A and B.

To spell-check in nano:

1. **`nano hairyspiders`**

   At the shell prompt, type **nano** and the filename of the file to edit.

2. Ctrl+T

   Pressing these keys starts spell-checking the file. **nano** will stop at each misspelled word as shown in A.

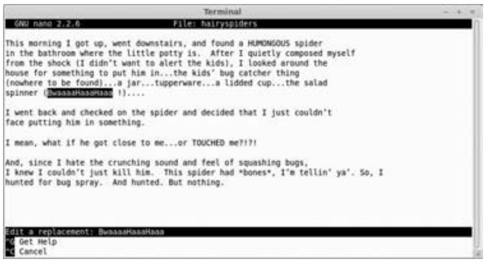
3. **`correctspelling`**

   Type in the correct spelling for any words flagged as misspelled, or press Enter to accept the current spelling and move along to the next word.

TIP You can press Ctrl+C to cancel spell-checking at any time.

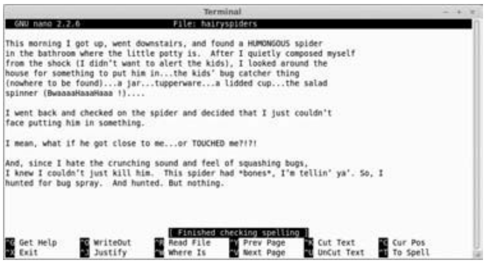
TIP Because the spell-checker in `nano` isn't full-featured, consider using an alternate spell-check program by specifying it on the command line, like `nano -s ispell hairyspiders`, so you can get a little more assistance. See Chapter 14 for more information.

TIP When the entire document has been spell-checked, `nano` will tell you that it's finished, and you can continue editing the file B.



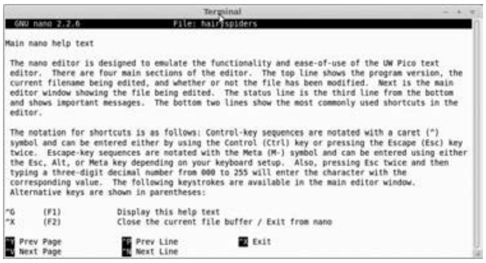
```
Terminal
  GNU nano 2.2.6                    File: hairyspiders

This morning I got up, went downstairs, and found a HUMONGOUS spider
in the bathroom where the little potty is.  After I quietly composed myself
from the shock (I didn't want to alert the kids), I looked around the
house for something to put him in...the kids' bug catcher thing
(nowhere to be found)...a jar...tupperware...a lidded cup...the salad
spinner (BwaaaaHaaaHaaa !)....

I went back and checked on the spider and decided that I just couldn't
face putting him in something.

I mean, what if he got close to me...or TOUCHED me?!?!

And, since I hate the crunching sound and feel of squashing bugs,
I knew I couldn't just kill him.  This spider had *bones*, I'm tellin' ya'. So, I
hunted for bug spray.  And hunted. But nothing.



Edit a replacement: BwaaaaHaaaHaaa
^G Get Help
^C Cancel
```

A **nano** prompts you to correct the spelling of misspelled words.



```
Terminal
  GNU nano 2.2.6                    File: hairyspiders

This morning I got up, went downstairs, and found a HUMONGOUS spider
in the bathroom where the little potty is.  After I quietly composed myself
from the shock (I didn't want to alert the kids), I looked around the
house for something to put him in...the kids' bug catcher thing
(nowhere to be found)...a jar...tupperware...a lidded cup...the salad
spinner (BwaaaaHaaaHaaa !)....

I went back and checked on the spider and decided that I just couldn't
face putting him in something.

I mean, what if he got close to me...or TOUCHED me?!?!

And, since I hate the crunching sound and feel of squashing bugs,
I knew I couldn't just kill him.  This spider had *bones*, I'm tellin' ya'. So, I
hunted for bug spray.  And hunted. But nothing.



                          [ Finished checking spelling ]
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

B **nano** informs you when the procedure is complete.



```
  GNU nano 2.2.6               File: hairyspiders

Main nano help text

 The nano editor is designed to emulate the functionality and ease-of-use of the UW Pico text
 editor.  There are four main sections of the editor.  The top line shows the program version, the
 current filename being edited, and whether or not the file has been modified.  Next is the main
 editor window showing the file being edited.  The status line is the third line from the bottom
 and shows important messages.  The bottom two lines show the most commonly used shortcuts in the
 editor.

 The notation for shortcuts is as follows: Control-key sequences are notated with a caret (^)
 symbol and can be entered either by using the Control (Ctrl) key or pressing the Escape (Esc) key
 twice.  Escape-key sequences are notated with the Meta (M-) symbol and can be entered using either
 the Esc, Alt, or Meta key depending on your keyboard setup.  Also, pressing Esc twice and then
 typing a three-digit decimal number from 000 to 255 will enter the character with the
 corresponding value.  The following keystrokes are available in the main editor window.
 Alternative keys are shown in parentheses:

^G      (F1)            Display this help text
^X      (F2)            Close the current file buffer / Exit from nano

^Y Prev Page        ^P Prev Line        ^X Exit
^V Next Page        ^N Next Line
```

A **nano** gives you all the information you need.

Getting Help in nano

A great way to find out more about **nano** is to access **nano** help. In addition to finding answers to your questions, you can discover its features and capabilities as shown in A.

To get help in nano:

1. Ctrl+G

   In **nano**, press Ctrl+G to access help.

2. Move through the help pages:

   - Ctrl+V moves you down through the help page.
   - Ctrl+Y moves you up through the help page.

3. Ctrl+X

   Use this combination to exit help.

To get help with nano startup options:

```
man nano
```

At the shell prompt, type **`man nano`** to learn more about startup options, including a variety of options that control how **nano** works.

TIP Keep your eye on the `nano` status line for current information, error messages, and occasional hints about using `nano`. The status line is the third line from the bottom of the screen, just above the menu, as shown in A.

TIP Keep in mind that `nano` really is a basic program. And if you're using `pico`, it's even more basic. If you're looking for a command or function that isn't readily available, it's probably not there. You might check out `vi` or `emacs` instead if you want more power than `nano` offers you.

Exiting nano

When you're finished editing in **nano**, you'll exit it using the following steps.

To exit nano:

1. Ctrl+X

   Within **nano**, press Ctrl+X. If you haven't made any changes to the text since you last saved the file, you'll find yourself immediately back at the shell prompt. If you have made changes, you'll be prompted to "Save modified buffer" A.

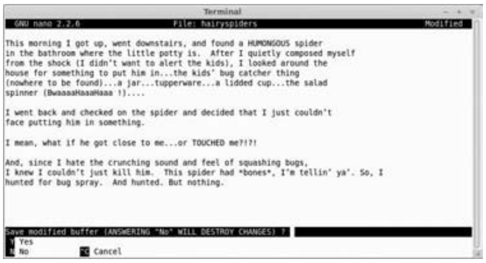
2. At the "Save modified buffer" prompt:

   - Press Y if you want to save your changes. Proceed to step 3.
   - Press N if you don't want to save your changes. You'll end up back at the shell prompt.

3. **`bighairyspiders`**

   Specify the filename for your file if it's the first time you've saved it. If you've saved it before, press Enter to confirm the current filename or change the name to save a copy and not change the original file.

TIP A *buffer* is what the computer uses to temporarily store information, and if it's modified, that means that it's temporarily storing something that you haven't saved to disk.



```
  GNU nano 2.2.6              File: hairyspiders                        Modified

This morning I got up, went downstairs, and found a HUMONGOUS spider
in the bathroom where the little potty is.  After I quietly composed myself
from the shock (I didn't want to alert the kids), I looked around the
house for something to put him in...the kids' bug catcher thing
(nowhere to be found)...a jar...tupperware...a lidded cup...the salad
spinner (BwaaaaHaaaHaaa !)....

I went back and checked on the spider and decided that I just couldn't
face putting him in something.

I mean, what if he got close to me...or TOUCHED me?!?!

And, since I hate the crunching sound and feel of squashing bugs,
I knew I couldn't just kill him.  This spider had *bones*, I'm tellin' ya'. So, I
hunted for bug spray.  And hunted. But nothing.



Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
 Y Yes
 N No           ^C Cancel
```

A **nano** gives you the opportunity to "Save modified buffer." Without the techno-babble, this means to save the text you just wrote or edited before you exit.

Ⓐ The **`vi`** editor inundates you with tons of onscreen help and advice, as shown here. Well, documentation is available, but the **`vi`** interface itself isn't really helpful at all!

Starting `vi` (or `vim`) and Dabbling with It

Before you go running off to use **`vi`**, understand that it has two modes (both of which look pretty much like Ⓐ):

- *Insert mode* (sometimes called input mode), in which the keys you press actually show up in the file that you're editing. You use this mode to add or change text.
- *Normal mode* (sometimes called command mode), in which every keystroke is interpreted as a command. You use this mode to do everything except enter text.

What's confusing for many people about **`vi`** is that it starts you in command mode, meaning that if you just start typing, you may see some blank spaces, characters, and bits of words that you type—essentially, a bunch of garbage that does not exactly represent what you're typing—and you'll hear a lot of beeping. So, as we'll show you in the following steps, you'll need to access the insert mode as soon as you start **`vi`**.

To start `vi`:

1. **`vi`**

   At the shell prompt, type **`vi`**. The program starts up, and you'll see something like A. The ~ symbols show blank lines below the end of the file.

2. **`i`**

   Type **`i`** to get into insert mode. This itself is a command issued in command mode, so it won't show up on the screen.

3. **`hairy spiders lurk`**

   In insert mode, type anything you want.

   Everything you type will show up on the screen until you return to normal mode by pressing Esc. When you are in normal mode, you can use the arrow keys to navigate up and down in the file line by line and use Ctrl+F and Ctrl+B to scroll one screen forward and backward, respectively.

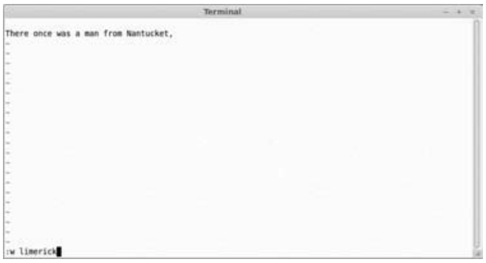
TIP To get help for `vi`, type `man vi`. See Chapter 1 for more about `man` pages.

TIP If you're not sure what mode you're in, press Esc to go into normal mode. If you're already in normal mode, you'll hear a beep. If you're in insert mode, you'll change to normal mode.

TIP Many Unix-like systems, including Linux and OS X, actually provide a program called `vim` in the place of `vi`. `vim` (for *VI iMproved*) is like `vi` but feature-rich and more flexible, and you can still start it with the command `vi`.

TIP You can open specific files or even multiple files when you access `vi`. At the shell prompt, type `vi filetoedit` (or whatever) to open a specific file. Or, for example, type `vi *.html` to open all the HTML documents in a directory, then use Esc`:n` (for "next") and then press Enter to move to each subsequent file.

TIP See "Adding and Deleting Text in `vi`" later in this chapter for more details about editing in `vi`.



```
There once was a man from Nantucket,
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:w limerick
```

A Save early, save often. That's the safe rule for **`vi`**.

Saving in vi

You'll want to save changes to your documents frequently, especially as you're learning to use **`vi`** **A**. Until you're accustomed to switching between insert and normal mode, you may accidentally type in commands when you think you're typing text, with unpredictable results. To save files, just follow these steps.

To save text in vi:

Esc**`:w limerick`**

Press Esc to get out of input mode and into command mode, then type **`:w`** (for "write," as in write to the disk) followed by a space and then the filename (**`limerick`**, in this example) you want to use for the file, then press Enter. If you've already saved the file once, just press Esc and type **`:w`**, then press Enter.

TIP If you've already saved your file at least once, you can save changes and exit `vi` in one fell swoop. In command mode, type `:wq` (for "write quit"). For more information about quitting `vi`, see the section "Exiting `vi`," later in this chapter.

TIP If you want to save a file over an existing file (obliterating the original as you do), use `:w! existingfilename` in command mode. The `!` forces `vi` to overwrite the original.

Adding and Deleting Text in vi

Adding and deleting text in **vi** is a bit more complicated than doing the same in **nano**. In **nano**, you basically just place your cursor where you want to make changes, whereas **vi** has a whole slew of commands that you use to specify where the changes should occur. (**Tables 4.1**, **4.2**, and **4.3** list only a very few of your options.) Plus, to issue the commands, you have to switch to normal mode.

To add or delete text in vi:

1. `vi`

   To begin, type **vi** at the shell prompt.

2. `i`

   Change into insert mode.

3. `There once was a man from Nantucket`

   Type some text that you'll want to add to.

4. Esc

   Press Esc to enter normal mode before you issue the commands.

5. Choose a command, based on what you want to do to the text.

   - Table 4.1 lists commands to add text.
   - Table 4.2 lists commands to delete text.
   - Table 4.3 lists miscellaneous editing commands.

6. `dd`

   Type the command. Here, we're deleting the current line of text.

TABLE 4.1 vi Commands to Add Text

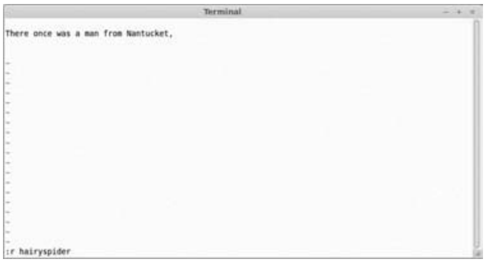
Command	Function
`a`	Adds text after the cursor
`A`	Adds text at the end of the current line
`i`	Inserts text before the cursor
`I`	Inserts text at the beginning of the current line
`o`	Inserts a blank line after the current line
`O`	Inserts a blank line before the current line

TABLE 4.2 vi Commands to Delete Text

Command	Function
`x`	Deletes one character (under the cursor)
`X`	Deletes one character (behind the cursor)
`dd`	Deletes the current line
`5dd`	Deletes five lines starting with the current line (any number would work here)
`dw`	Deletes the current word
`cw`	Changes the current word (deletes it and enters input mode)
`r`	Replaces the character under the cursor with the next character you type
`R`	Replaces the existing text with the text you type (like overtype mode in most word processors)

TABLE 4.3 Other Handy vi Editing Commands

Command	Function
`yy`	Copies the current line
`p`	Pastes any copied text after the cursor or line
`J`	Joins the current and following lines
`u`	Undoes the last change
`U`	Undoes all changes on the current line
`.`	Repeats the last command



```
There once was a man from Nantucket,


~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:r hairyspider
```

A Reading an additional file into the current one can make your editing tasks much easier.

Importing Files into `vi`

You can also merge multiple files in **`vi`** by reading additional files into the current one, as shown in **A**. Basically, all this means is that you insert one file into the file you're currently editing.

To import files in `vi`:

1. **`vi hairyspider`**

   At the shell prompt, type **`vi`** followed by the filename—in this case, the **`hairyspider`** file.

2. Esc**`:r filename`**

   At the point in the file where you want to import text, press Esc, then type **`:r`** and the filename you want to read into the file.

TIP `vi` also lets you read the output of commands into the file. For example, if you want to read the list of files in a specific directory into the file, use Esc`:r !ls` in normal mode.

Searching and Replacing in vi

One of **vi**'s better features (and advantages over **nano**) is that it allows you to search and replace throughout entire files. As shown in the next sections, you can just find a specific string of text (a *regular expression*, in Unix lingo; see A), or you can find the text and replace it with other text, as in B.

To find a string of text in vi:

1. **vi hairyspider**

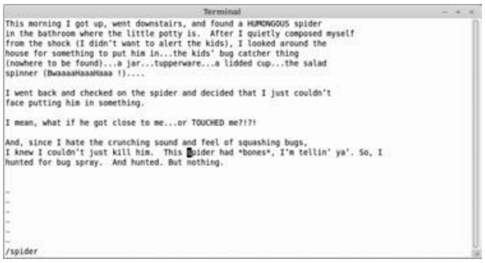
   For starters, access **vi** and a specific file.

2. Esc**/spider**

   Enter command mode, then type **/** followed by the text you're looking for. Here, we're looking for "spider," but you may be looking for "the fly" or "wiggled and jiggled and tickled inside her." Or whatever.

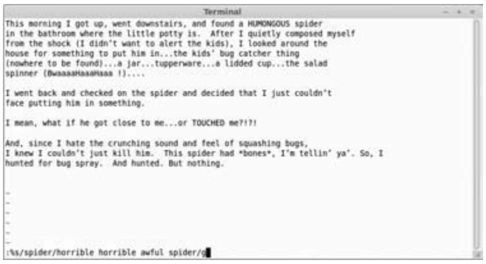
3. Enter

   Press Enter to find the first occurrence of the term. Type N to find the next one.



```
This morning I got up, went downstairs, and found a HUMONGOUS spider
in the bathroom where the little potty is.  After I quietly composed myself
from the shock (I didn't want to alert the kids), I looked around the
house for something to put him in...the kids' bug catcher thing
(nowhere to be found)...a jar...tupperware...a lidded cup...the salad
spinner (BwaaaaHaaaHaaa !)....

I went back and checked on the spider and decided that I just couldn't
face putting him in something.

I mean, what if he got close to me...or TOUCHED me?!?!

And, since I hate the crunching sound and feel of squashing bugs,
I knew I couldn't just kill him.  This spider had *bones*, I'm tellin' ya'. So, I
hunted for bug spray.  And hunted. But nothing.

~
~
~
~
~
~
/spider
```

A Searching for text in **vi** is quick and reliable.



```
This morning I got up, went downstairs, and found a HUMONGOUS spider
in the bathroom where the little potty is.  After I quietly composed myself
from the shock (I didn't want to alert the kids), I looked around the
house for something to put him in...the kids' bug catcher thing
(nowhere to be found)...a jar...tupperware...a lidded cup...the salad
spinner (BwaaaaHaaaHaaa !)....

I went back and checked on the spider and decided that I just couldn't
face putting him in something.

I mean, what if he got close to me...or TOUCHED me?!?!

And, since I hate the crunching sound and feel of squashing bugs,
I knew I couldn't just kill him.  This spider had *bones*, I'm tellin' ya'. So, I
hunted for bug spray.  And hunted. But nothing.

~
~
~
~
~
~
:%s/spider/horrible horrible awful spider/g
```

B Replacing text in **vi** requires a bit of arcane syntax, but you get used to it quickly.

To search and replace in `vi`:

1. **`vi hairyspider`**

   For starters, access **`vi`** and a specific file.

2. Esc**`:%s/spider/horrible horrible awful spider/`**

   Enter Esc**`:%s/`** plus the text to find, another **`/`**, followed by the replacement text, as in B. Here, we replace "swallowed a fly" with "swallowed a spider to catch the fly," but perhaps you might forgo the spider and simply go for some antacid.

TIP **A great use for the search-and-replace feature is if you end up with DOS text files in your Unix account (by transferring a text file from a Windows machine as a binary file, most likely). If you view DOS files through a Unix shell, all the lines in the file will end with `^M`. But if you try to type `^M` when you're doing a search and replace, the `^M` won't show up. What to do? Press Ctrl+V, then Ctrl+M. Just search and replace with `:%s/` Ctrl+V Ctrl+M`//g`. The Ctrl+V command "escapes" the following character, so you can press it without actually doing what the command would otherwise do. If you don't escape the Ctrl+M, `vi` thinks you just pressed Enter and tries to execute the unfinished command.**

TIP **See the section on `grep` in Chapter 6 for information about searching with regular expressions.**

TIP **Add a `g` at the end of the command to make it apply to all occurrences in the file. Otherwise, it applies only to the first occurrence on each line.**

Exiting vi

Whew! Time to exit **vi** A.

To exit vi:

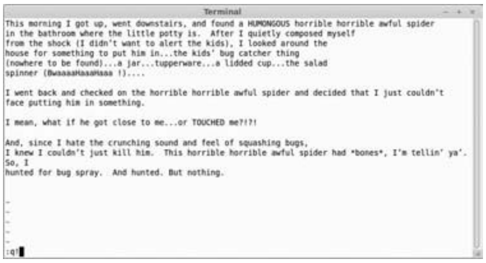
Esc**:q**

Enter command mode by typing Esc, then type **:q** to quit **vi**. If you haven't saved your latest changes, **vi** will not quit and will tell you to use **!** to override. To quit without saving your changes, use **:q!**, as shown in A.

TIP If you don't really want to quit but want to edit a different file instead, type `:e filename` to open a new file to edit.

TIP We recommend that you take a few minutes to try out some of the commands that you'll use throughout your `vi` experience. If you don't think you'll need this range of commands, consider using `nano` rather than `vi`.

TIP It takes some practice to get accustomed to `vi`, but the time spent is well worth it. With patience and practice, you'll quickly become proficient in using `vi`. Take your time, take deep breaths, and plow ahead.



```
This morning I got up, went downstairs, and found a HUMONGOUS horrible horrible awful spider
in the bathroom where the little potty is.  After I quietly composed myself
from the shock (I didn't want to alert the kids), I looked around the
house for something to put him in...the kids' bug catcher thing
(nowhere to be found)...a jar...tupperware...a lidded cup...the salad
spinner (BwaaaaHaaaHaaa !)....

I went back and checked on the horrible horrible awful spider and decided that I just couldn't
face putting him in something.

I mean, what if he got close to me...or TOUCHED me?!?!

And, since I hate the crunching sound and feel of squashing bugs,
I knew I couldn't just kill him.  This horrible horrible awful spider had *bones*, I'm tellin' ya'.
So, I
hunted for bug spray.  And hunted. But nothing.


~
~
~
~
~
:q!
```

A Use Esc**:q!** to quit **vi** without saving changes.

Index

Symbols

* (asterisk)
  function as regular expression, 112
  removing multiple files using, 43
  used as placeholder, 21, 23
\ (backslash), 112
^ (caret)
  as Ctrl in **nano**, 75
  function as regular expression, 112
$ (dollar sign), 112, 158
. (dot)
  find command using, 46
  function as regular expression, 112
  hiding files starting with, 40
  included in path statements, 157, 166
&& (double ampersand), 172
= (equals sign), 101
! (exclamation point), 248
/ (forward slash), 47
> (greater than symbol), 21
# (hash mark)
  comment indicator, 163, 191
  file transfer indicator, 245
  root prompt symbol, 291
| (pipe symbol), 19–20
? (question mark), 23
; (semicolon), xii
[ ] (square brackets), 112, 343
~ (tilde), 17

A

about this book, ix–x
absolute names, 37
accessing
  man pages, 28
  Unix, 3–7
alias command, 169–170, 344
aliases
  for email, 215
  recommended, 169
  setting, 169–170
alpine
  about, 209, 211
  composing email with, 214–215
  configuration options, 217
  customizing, 216–217
  deleting messages in, 213
  flags and arguments for, 344
  menu commands in, 213
  printing email with, 213
  reading email with, 212–213
  sending email with, 214–215
anonymous ftp command, 243–244
appending output to existing file, 22
applications
  installing on systems, 295
  See also programs; utilities
archiving files
  about, 264
  tar used for, 264–265
  zipping while, 274
arguments, xii, 47, 343–383
association, group, 96–97
asterisk (*)
  function as regular expression, 112
  removing multiple files using, 43
  used as placeholder, 21, 23
at command
  deleting scheduled jobs with, 175
  flags and arguments for, 344
  scheduling onetime jobs with, 173–174
  sequential job scheduling with, 174
atq command, 344
atrm command, 344
attachments, email, 224, 262
awk command
  changing files with, 116–117
  flags and arguments for, 344–345
  generating reports with, 310

B

background jobs, 180
backslash (\\), 112
backups
  configuration file, 170
  looping scripts for making, 199–200
  making with **cp** command, 43
  rsync for making, 318–319
bash shell
  adding variables in, 150–151
  alias setup for, 169–170
  changing variables in, 150–151

completion used in, 61
daisychains in, 163
features of, 55
flags and arguments for, 345
path changes, 164–166
prompt changes, 167–168
redirecting **stderr** in, 321–322
session history viewed in, 62–63
setting your prompt for, 168
showing current environment in, 148
viewing configuration files in, 161–163
writing scripts using, 190
batch command, 345
bc utility, 278
bg command, 178, 180, 345
binary files
cat command and, 26
downloading, 245
encoding, 260, 262
boot messages, 300–301
Bourne Again Shell. *See* **bash** shell
Bourne shell. *See* **sh** shell
breaking lines, 124
browsers
links, 249–250, 355
lynx, 251–252, 358–360
buffer, 80
bzip utility, 269

C

cal utility, 276–277, 345
calculator utility, 278
calendars, 276–277
caret (^)
as Ctrl in **nano**, 75
function as regular expression, 112
case sensitivity
of Unix/Linux commands, xii
of userid and password, 11
case translations, 124, 125
cat command
displaying file contents with, 25–26
flags and arguments for, 345
listing shells with, 56
cd command, 17, 346
changing
directories, 17
passwords, 13–14
paths, 156–157, 164–166
prompts, 158–159, 167–168
shells, 57–58
userid, 67–68
characters
Escape, used with **telnet**, 239
restricted for directory and filenames, 33
using as delimiters, 117
See also specific characters
chgrp command, 96–97, 346
chmod command, 100–102, 346
chown command, 98–99, 346
chsh command, 57–58, 346
clients, 236
cmp command, 37, 118, 346
code listings, x
combining commands, 172, 273–274
comma-delimited files, 121
command argument completion
bash shell and, 61
zsh shell and, 64
command mode in **vi**, 81
command-line arguments, 204–205
commands
about flags and, 343
argument completion for, 61, 64
basic Unix and Linux, 324
case sensitivity of, xii
combining, 172, 273–274
conventions for typing, xi–xii
creating and editing files, 326
directory and file management, 325
email-related, 334
embedding in scripts, 197–198
encoded and compressed file, 336
environment configuration, 331
file manipulation, 328–329
flags or arguments with, xii
Internet-related, 335
ownership and permission, 327
piping, 19–20
running, 172
script-related, 332–333
separating on same line, xii
shell management, 326
system information, 330
table of flags, arguments, and, 343–383
topical listing of, 323–337
utility-related, 337
vi editing, 84
See also specific commands
comments (#), 163, 191
comparing
directories, 37
files, 37, 118, 120
job times, 183
composing email
with **alpine**, 214–215
with **mail**, 223–224
with **mutt**, 220–221
vacation messages, 228–229
compress command, 267, 347
compressing files
about, 267
commands list for, 336
compress for, 267
gzip for, 269
zip for, 271

computer emulation, 6
computer requirements, xi
conditional statements, 201–203
configuration files
  backing up, 170
  bash shell, 161–163
  executing or sourcing, 163
  finding with **grep**, 155, 166
  order for running, 147, 161
  viewing, 152–155, 161–163
  zsh shell, 152–155
  See also editing configuration files
configuring
  aliases, 169–170
  alpine, 216–217
  bash shell, 161–168
  commands and flags for, 331
  environment variables, 146–147, 150–151, 331
  mutt, 219
  procmail, 230–231
  system files, 294–295
  zsh shell, 152–160
connections
  checking with **ping**, 254
  login information for, 10
  screen session, 284–285
  ssh (Secure Shell), 8–9
  tracing with **traceroute**, 255–256
  troubleshooting **ftp**, 246
  Unix or Linux, 8–10
converting measurements, 280
copying
  directories and files, 36–37
  man pages, 29
counting files and contents, 108
cp command
  copying directories and files with, 36–37
  flags and arguments for, 347
  making backups with, 43
CPU information, 143
crackers, 13
cron command, 176–177
crontab command, 176–177, 347
csh shell, 55
curl command, 347
customizing
  alpine, 216–217
  environment variables, 311–312
  mutt, 219
cutting/pasting, 77
Cygwin, xi, 6

D

daemons
  defined, 186, 292
  starting and stopping, 292–293
daisychains, 163
date
  formatting options for, 198
  manually setting, 302
  setting with **ntpdate**, 302
  updating time and, 35
date command, 198, 302
debugging scripts, 208
decoding files
  unzipping and, 273
  uudecode for, 263
default permissions, 105–106
deleting
  email messages, 213, 219
  files, 42–43
  processes, 187–188
  scheduled jobs, 175
  text in **vi**, 84
  See also removing
delimited files, 117
df command, 133–135, 348
dictionary lookup, 281
diff command, 37, 118, 119, 348
differences in files, 119–120
dig command, 258, 348
dircmp command, 37
directories
  archiving, 265
  backing up, 43, 318–319
  changing, 17
  commands for managing, 325
  common Unix and Linux, 27, 341–342
  comparing, 37
  compressing, 269
  copying, 36
  creating, 32–33
  determining disk usage for, 136
  displaying name for current, 18
  file system status for, 135
  finding files in, 46
  group association of, 96–97
  linking, 50, 51–52
  listing, 15–16, 18
  moving, 41
  naming, 33
  navigating, 30
  ownership of, 93, 98–99
  remote navigation of, 246
  removing, 44–45
  synchronizing, 318
  zipping/unzipping, 269–272
  See also files
disk usage
  determining with **du**, 136
  space required for uncompressed files, 268
  viewing number of blocks on device, 134
distrowatch.com website, 7
dmesg utility, 300–301

documents
  cleaning up HTML, 304–306
  searching and replacing in multiple, 307–309
dollar sign ($), 112, 158
domain names, 257–258
DOS text files, 87
dot (.)
  find command using, 46
  function as regular expression, 112
  hiding files starting with, 40
  included in path statements, 157, 166
downloading
  files with **ftp**, 243–246
  websites with **wget**, 253
dropbox, 248
du command, 136, 349
duplicate files, 122

E

echo $SHELL command, 54, 147
echo command, 192
editing configuration files
  alias setup, 169–170
  environment variables, 146–147, 150–151
  path changes, 156–157, 164–166
  prompt changes, 158–159, 167–168
  to request default editor, 311–312
  as **root** user, 294–295
editors
  alternative, 74
  choosing, 72–74
  emacs, 74, 79, 349
  nano, 72, 73, 75–80, 363
  pico, 73, 79, 363
  vi, 72, 74, 81–88, 196
egrep command, 113
else statement, 203
emacs editor, 71, 74, 79, 349
email
  attachments sent with, 224
  automatically forwarding, 227
  choosing a program for, 210–211
  commands for working with, 334
  composing, 214–215, 220–221, 223–224
  configuration options, 217
  deleting, 213, 219
  encoding files for, 262
  filtering, 230, 232, 233
  forwarding, 227, 233
  managing, 232–234
  printing, 213
  reading, 212–213, 218–219, 222
  sending, 214–215, 220–221, 223–224
  signature files for, 225–226
  spam filters for, 214, 233
  splitting files for, 128–129
  subject lines for, 214
  vacation messages for, 228–229, 233
  web-based, 210
  See also **alpine**; **mail**; **mutt**
embedding
  awk scripts in shell scripts, 310
  commands in scripts, 197–198
  ROT13 encoding in shell scripts, 315–317
encoding files
  about, 260
  commands list for, 336
  decoding and, 263
  emailing and, 262
  ROT13 encoding for, 313–317
  sed command used for, 313–314
  uuencode for, 260–262
environment variables
  about, 146–147
  adding or changing, 150–151
  commands for configuring, 331
  input used to customize, 311–312
  not to change, 149
  safe to change, 151
  setting **TERM**, 312
  showing current, 148
  su for setting, 291
equals sign (=), 101
error, standard, 320–322
Escape character, 239
/etc directory, 294, 295
exclamation point (!), 248
executable scripts, 194–195
execute permission, 90, 106
executing configuration files, 163
exit command
  exiting shells with, 70
  returning to previous shell with, 68
exiting
  nano editor, 80
  returning to previous shell by, 68
  shells at end of session, 70
  temporary shells, 60
  vi editor, 88
expr utility, 279, 349
expressions
  calculating, 278
  evaluating, 279
  regular, 112–114

F

failed login attempts, 12
fg command, 178, 181, 349
fgrep command, 113
fields, sorting, 121
figlets, 226
file command, 137, 349
file systems, 133–135
File Transfer Protocol (FTP), 237

files
  appending output to, 22
  archiving/unarchiving, 264–266
  backing up, 43, 199–200, 318–319
  beginnings viewed in, 109
  breaking lines in, 124
  changing with **awk**, 116–117
  commands for working with, 325, 328–329
  comparing, 37, 118, 120
  compressing/uncompressing, 267–272, 336
  copying directories and, 36–37
  counting contents of, 108
  creating, 34–35, 326
  date/time updates, 35
  delimited, 117
  discovering differences in, 119–120
  displaying contents of, 24, 25–26
  downloading with **ftp**, 243–246
  editing, 71, 326
  eliminating duplicate, 122
  emailing text, 224
  encoding/decoding, 260–263, 336
  endings viewed in, 110
  finding, 46–47
  formatting, 126–127
  global changes to, 115
  group association of, 96–97
  hiding, 40
  identifying types of, 137
  importing into **vi**, 85
  information about, 39
  linking, 49–52
  listing, 15–16, 38–40
  locating lost, 48
  moving, 41
  naming, 33
  ownership of, 90, 91–93, 98–99
  permissions for, 90, 91–93, 100–106
  redirecting output to, 21–22, 123
  regular expressions in, 112–114
  removing with **rm**, 42–43
  saving, 76, 80, 83
  searching/replacing in, 86–87
  sharing with **ftp**, 247–248
  signature, 225–226
  sorting, 121
  splitting, 128–129
  synchronizing, 318
  text strings found in, 111
  translating case in, 124, 125
  zipping/unzipping, 269–272
  See also directories
filtering email
  configuring **procmail** for, 231
  tossing spam messages, 233
  writing **procmail** recipes, 232
find command, 46–47, 350–351
finding
  available shells, 56
  configuration files with **grep**, 155, 166
  directory names, 18
  files with **find**, 46–47
  group membership, 94–95
  info on logged-in users, 138–143
  lines with specific characteristics, 114
  ownership information, 91–93
  permission information, 92, 93
  temporary shells, 59
  text strings, 111
  viewing files after, 286
  See also searching and replacing
finger command, 54, 138–140, 351
firewalls
  ftp connections and, 246
  talk chats and, 242
  traceroute problems with, 256
flags, xii
  command-based listing of, 343–383
  topical listing of, 323–337
flavors of Unix/Linux, ix, 7
fmt command, 126–127, 351
foreground jobs, 181
formatting
  date display, 198
  files, 126–127
fortune program, 48
.forward file, 227, 229, 230
forward slash (/), 47
forwarding email messages
  automatically, 227
  with **procmail**, 233
found files, acting on, 286
free utility, 297
ftp command
  downloading files with, 243–246
  flags and arguments for, 352
  sharing files with, 247–248
FTP sites, 237, 243
  downloading files from, 243–246
  sharing files via, 247–248
  troubleshooting connections to, 246

G

get command, 245
getting started, 1–30
git command, 352
global changes to files, 115
greater than symbol (>), 21
grep command
  configuration files found with, 155, 166
  flags and arguments for, 352
  regular expressions used with, 112–114
  specific lines found with, 114
  text strings found with, 111
  w command used with, 143

groups
  association changes for, 96–97
  determining by userid, 144
  file ownership by, 90
  finding membership in, 94–95
groups command, 95
GUI tool, 295
gunzip command, 268, 270, 272, 273
gz command, 248
gzip command, 265, 269, 274, 352–353

H

hackers, 13
hard drive information, 133–135
hard links, 49–50
hash mark (#)
  comment indicator, 163, 191
  file transfer indicator, 245
  root prompt symbol, 291
head command, 109, 263, 353
help
  expr utility, 279
  man command for, 27, 28–29
  mutt, 219
  nano, 79
  ssh, 238
  telnet, 240
  vi, 82
hidden files, 38, 40
history command
  creating scripts with, 196
  viewing session history with, 63, 65
home directory
  files stored in, 12
  shortcut to, 17
  system root directory vs., 17
host names, 236
HTML (Hypertext Markup Language)
  document cleanup for, 304–306
  searching and replacing tags in, 307–309
HTTP (Hypertext Transfer Protocol), 237
human-readable output, 135

I

id command, 95, 144, 353
if-then statements, 201–203
  about, 201
  conditions for, 202
  steps for writing, 202–203
importing files into **vi**, 85
incoming directory, 248
info command, 353
information about files, 39
input
  accepting while running scripts, 206–207
  command-line arguments as, 204–205
  customizing your environment using, 311–312
  piping, 19–20
  standard, 320
insert mode in **vi**, 81, 82
installing
  software, 295
  Unix or Linux, 5–6
Internet
  checking connections on, 254
  commands for accessing, 335
  communicating with others via, 241–242
  domain names and IP addresses on, 257–258
  downloading files from, 243–246
  downloading websites from, 253
  FTP sites on, 243–248
  links browser for, 249–250
  lynx browser for, 251–252
  ports and protocols, 237
  remote system login, 238–240
  sharing files on, 247–248
  surfing websites on, 249–252
  terminology related to, 236–237
  tracing connections on, 255–256
IP (Internet Protocol) addresses
  defined, 236
  matching domain names with, 257–258
ISPs (Internet service providers)
  forwarding email when changing, 227
  interface for changing shells, 58
  shell accounts offered by, 5

J

jobs
  checking status of, 179
  controlling priority of, 182
  defined, 171
  deleting scheduled, 175
  info on processes running, 185–186
  killing, 179, 187–188
  onetime, 173–174
  regularly occurring, 176–177
  running background/foreground, 180–181
  scheduling, 173–174, 176–177
  sequential, 174
  suspending, 178
  timing, 183–184
  See also scripts
jobs command, 178, 179, 354
Julian calendar, 276

K

keystrokes
  for **links** browser, 249
  for **lynx** browser, 251
kill command, 178, 179, 187–188, 354
killing jobs, 179, 187–188
ksh shell, 55

L

last utility, 297
less command, 24, 354–355
lines
  breaking with **tr** command, 124
  counting in files, 108
  viewing specified number of, 109
linking files
  hard links for, 49–50
  soft links for, 51–52
links browser
  flags and arguments for, 355
  keystrokes for, 249
  navigating with, 250
  surfing the web with, 249–250
Linux
  accessing, 3–5
  choosing flavor of, 7
  command reference for, 323–327
  common directories in, 341–342
  connecting to, 8–10
  installing, 5–6
  list of key files in, 340
  logging in to, 11–12
  logging out of, 30
  passwords for, 13–14
  reasons for using, 4
  shells available in, 55–56
  software installation on, 295
  table of commands, flags, and arguments, 343–383
  typing commands in, xi–xii
  See also Unix
listing
  available shells, 56
  directories and files, 15–16, 38–40
  files by type, 137
  screen sessions, 285
ln command
  flags and arguments for, 356
  hard links and, 49–50
  soft links and, 51–52
locate command, 48, 356
locating lost files, 48
logged-in users
  finger for finding out about, 138–140
  w for getting info on, 142–143
  who for snooping on, 141
logging in
  cal utility used on, 276
  as different user, 68
  remotely, 238–240
  root user safeguards for, 291
  steps in process of, 11
  writing details about, 10
logging out, 30, 70
logout command, 30, 70
logs, monitoring, 296
look utility, 281, 356
looping scripts, 199–200
lp command, 357
ls command
  flags and arguments for, 357–358
  listing directories and files with, 15–16, 18, 38–40
lynx browser
  flags and arguments for, 358–360
  keystrokes for, 251
  navigating with, 252
  surfing the web with, 251–252

M

Macintosh computers
  accessing Unix on, 7
  author's note to users of, xii
  viewing hard drives on, 133
MacSSH program, 8
mail
  about, 211
  composing email with, 223–224
  flags and arguments for, 361
  reading email with, 222
  sending email with, 223–224
  text files sent with, 224
mail clients, 210
mail loops, 229
man command
  flags and arguments for, 361
  getting help with, 27, 28–29
managing email
  configuring **procmail** for, 230–231
  writing **procmail** recipes, 232
measurement conversions, 280
memory check, 297
Message of the Day (**motd**) file, 294–295
messaging
  talk command, 242
  write command, 241
mget command, 246
mkdir command, 32–33, 361
mnemonic permissions, 103
monitoring
  logs, 296
  memory, 297
  scripts for, 297
  system load, 297
  users, 297, 298
  watch utility for, 299
more command, 24, 361–362
motd file, 294–295
mounted file systems, 133
mountpoint, 134
moving
  files or directories, 41
  up/down in directories, 30

mput command, 248
multiple files
  downloading with **ftp**, 246
  making global changes to, 115
  viewing beginnings/endings in, 109–110
mutt
  about, 211
  composing email with, 220–221
  customizing, 219
  deleting messages in, 219
  flags and arguments for, 362
  reading email with, 218–219
  sending email with, 220–221
mv command, 41, 362

N

naming
  directories and files, 33
  using absolute or relative names, 37
nano editor, 72, 73, 75–80
  about, 73
  cutting/pasting in, 77
  exiting, 80
  flags and arguments, 363
  getting help in, 79
  saving files in, 76, 80
  spelling checks in, 78
  starting, 75
  status line in, 79
navigating
  in directories, 30
  with **links** browser, 250
  with **lynx** browser, 252
nice command, 182, 363
normal mode in **vi**, 81, 82
nslookup command, 257, 258
ntpdate command, 302
numbers
  sorting numerically, 121
  specifying for **cron** jobs, 176
numeric permissions, 103–104

O

OpenIndiana, 7
output
  appending to existing files, 22
  error messages as, 320–322
  human-readable, 135
  piping, 19–20
  redirecting, 21–22, 123, 316, 320–322
  standard (**stdout**), 320–322
overwriting files, 37
ownership
  changing with **chown**, 98–99
  commands for controlling, 327
  finding for files/directories, 91–93
  group association changes and, 96–97
  group membership and, 94–95
  levels of, 90
  See also permissions

P

packets, 255
passwd command, 13–14, 363
passwords
  case sensitivity of, 11
  changing, 13–14
  checking new, 14
  root user, 291
  tips on choosing, 14
path statement, 156, 164
paths
  changing **bash**, 164–166
  changing **zsh**, 156–157
  executable scripts and, 195
  finding name of, 18
performance
  checking running processes, 185–186
  controlling job priority, 182
  monitoring system load, 297
  time of day related to, 184
  tracing bottlenecks in, 255–256
permissions
  adding, 102
  categories of, 90
  changing defaults for, 105–106
  characters indicating, 92, 103
  checking current, 100
  commands for controlling, 327
  finding file/directory, 91–93
  numeric, 103–104
  removing, 102
  setting, 101
  See also ownership
pfexec utility, 290
pgrep command, 365
pico editor, 73, 79, 363
  See also **nano** editor
PID number, 185, 187, 188
pine, 73, 209, 364
  See also **alpine**
ping command, 254, 364
pipe symbol (|), 19–20
pkill command, 188, 365
plan files, 140
ports, 237
pr command, 366–367
-print flag, 47
printing
  email with **alpine**, 213
  scripts onscreen, 208
priority of jobs, 182

processes
  defined, 185
  deleting, 187–188
  info on running, 185–186
  ownership and, 90
  system info about, 142–143
procmail
  about, 230
  configuring, 230–231
  flags and arguments for, 366
  forwarding email with, 227, 233
  invoking **vacation** with, 233
  sample recipes for, 234
  tossing spam messages with, 233
  writing recipes for, 232
programs
  commands for running, 332
  exploring locally installed, 27
  installing on systems, 295
  See also running scripts and programs
project files, 140
prompt command, 246
prompts
  about default, 158
  changing **bash**, 167–168
  changing **zsh**, 158–159
  shell, 12, 53
  sudo, 290
protocols, 237
ps command, 185–186, 366
put command, 247
PuTTY program, 8, 9
pwd command, 18, 367

Q

question mark (?) wildcard, 23
quit command, 246, 285
quota command, 367

R

RAM check, 297
read permission, 90
reading email
  with **alpine**, 212–213
  with **mail**, 222
  with **mutt**, 218–219
real time, 184
recipes for **procmail**, 232, 234
reconnecting to screen sessions, 285
recording scripts, 282–283
redirecting output
  to files, 21–22
  to multiple locations, 123
  of scripts, 316
  with **stderr**, 320–322
reget command, 246
regular expressions
  finding lines using, 114
  summary table describing, 112
  using with **grep**, 112–114
relative names, 37
remote systems
  checking connections to, 254
  logging in to, 238–240
  navigating directories on, 246
  screen sessions on, 284–285
  ssh connections to, 238
  telnet connections to, 239–240
  tracing connections to, 255–256
removing
  directories, 44–45
  files, 42–43
  hard links, 50
  line numbering in **vi**, 196
  permissions, 102
  scheduled jobs, 175
  See also deleting
renice command, 182, 367
replacing. *See* searching and replacing
reports, generating, 310
reset command, 69, 367–368
restarting daemons, 293
rm command
  file removal with, 42–43
  flags and arguments for, 367
  hard link removal with, 50
rmdir command, 44–45, 368
root directory, 16, 17, 47
root users
  about, 287
  becoming **root** with **su**, 291
  caution on logging in as, 43, 44
  changing system configuration, 294–295
  checking boot messages, 300–301
  monitoring the system, 296–298, 299
  responsibilities of, 287, 288, 291
  safeguards for logging in, 291
  setting date and time, 302
  starting and stopping daemons, 292–293
  sudo utility and, 288–290
  watch utility and, 299
  See also system administrators
ROT13 encoding
  embedding in shell scripts, 315–317
  using with **sed**, 313–314
rsync utility, 318–319, 368–370
runique command, 246
running scripts and programs
  accepting input while, 206–207
  background jobs, 180
  checking job status, 179
  command-line arguments for, 204–205
  commands and flags for, 332
  controlling job priority, 182

deleting processes, 187–188
foreground jobs, 181
info on processes running, 185–186
making executable scripts, 194–195
onetime jobs, 173–174
regularly occurring jobs, 176–177
removing scheduled jobs, 175
running commands, 172
scheduling jobs, 173–174, 176–177
suspending jobs, 178
timing jobs, 183–184

S

saving files
in **nano**, 76, 80
in **vi**, 83
scheduling
onetime jobs, 173–174
regularly occurring jobs, 176–177
screen sessions, 284–285
screen utility, 284–285, 370–371
script utility, 282–283
scripts
accepting input while running, 206–207
awk scripts embedded in, 310
background, 180
checking status of, 179
command-line arguments in, 204–205
commands for working with, 332–333
controlling priority of, 182
creating shell, 191–192
debugging, 208
embedding commands in, 197–198
example of using, 189
foreground, 181
history for creating, 196
if-then statements in, 201–203
looping, 199–200
making executable, 194–195
numbers for **cron** jobs, 176
onetime, 173–175
printing onscreen, 208
recording with **script** utility, 282–283
redirecting output of, 316
regularly occurring, 176–177
removing scheduled, 175
ROT13 encoding embedded in, 315–317
running, 180–181, 193, 332
scheduling, 173–174, 176–177
sed search and replace with, 307–309
suspending, 178
system monitoring, 297
tidy used with **sed**, 306
timing, 183–184
verifying first line of, 195
writing, 189–190, 333
See also running scripts and programs
sdiff command, 37, 120
searching and replacing
loops used for, 200
in multiple documents with **sed**, 307–309
text in **vi**, 86–87
See also finding
security
su used for root access, 68
telnet vs. **ssh** connection, 8
sed command
flags and arguments for, 370
loops used with, 200
making global changes with, 115
multiple document search-and-replace with, 307–309
ROT13 encoding used with, 313–314
tidy used with, 306
semicolon (;), xii
sending email
with **alpine**, 214–215
with **mail**, 223–224
with **mutt**, 220–221
from shell prompt, 215
vacation messages, 228–229
sequential onetime jobs, 174
servers, 236
sessions
exiting at end of, 70
recording with **script**, 282–283
viewing history of, 62–63, 65–66
working with screen, 284–285
set command, 370
setenv command, 370
sh shell, 55
sharing files, 49–50, 89
shell prompt, 12, 53
See also prompts
shell scripts, 175, 189–190
See also scripts
shell variables, 146
shells
accessing shell accounts, 5
adding or changing variables in, 150–151
alias setup for, 169–170
changing with **chsh**, 57–58
command argument completion in, 61, 64
commands for working with, 326
configuration files viewed in, 152–155, 161–163
creating shell scripts, 191–192
determining one in use, 147
discovering default, 54
exiting, 70
finding available, 56
fixing terminal settings, 69
killing current process in, 188
mail announcements in, 211
redirecting **stderr** in, 321–322
running shell scripts, 193, 194–195

shells (*continued*)
  sending email from prompt in, 215, 221
  session history for, 62–63, 65–66
  temporary, 59–60
  types of available, 55–56
  userid changes for, 67–68
  See also **bash** shell; **zsh** shell
signature files, 225–226
sniffing, 8
soft links, 51–52
software
  installing on systems, 295
  See also programs
Solaris, 7
sort command, 121, 122, 372
sorting
  and eliminating duplicates, 122
  files using **sort** command, 121
sourcing configuration files, 163
spam filters, 214, 233
special characters. *See* characters
spelling checks
  dictionary lookup for, 281
  looping scripts for, 200
  nano editor, 78
  piped commands for, 20
split command, 128–129, 372
splitting files, 128–129
square brackets ([ ]), 112, 343
SSH Preferences dialog box, 12
ssh program
  about connecting with, 8
  flags and arguments for, 371–372
  logging in remotely with, 8–9, 238
  setting preferences for, 12
standard error (**stderr**)
  defined, 320
  redirecting in shells, 321–322
standard input (**stdin**), 320
standard output (**stdout**), 320
starting
  daemons, 292–293
  nano, 75
  vi, 82
status line in **nano**, 79
sticky bits, 93
stopped jobs, 178
stopping
  daemons, 293
  vacation emails, 229
strings, text, 111
stty command, 69
su command, 67–68, 291, 373
su - yourid command, 14
subject line for email, 214, 224
subshells, 59–60
sudo utility, 99, 288–290, 373
suspending jobs, 178
synchronizing files/directories, 318–319
system administrators
  changing system configuration, 294–295
  checking boot messages, 300–301
  diagnosing system problems, 135
  monitoring the system, 296–298
  password security, 8
  rm command used by, 43
  setting date and time, 302
  starting and stopping daemons, 292–293
  sudo utility and, 288
  wall command for, 238
  watch utility for, 299
  whoami command used by, 141
  See also **root** users
system information
  commands for getting, 330
  daemons running, 186
  determining disk usage, 136
  finding out file types, 137
  getting with **uname**, 132
  obtaining on logged-in users, 138–143
  setting **zsh** prompt to show, 159
  userid information, 144
  viewing file systems, 133–135
system load, 297
system root directory, 17
systemwide configuration files, 147

T

tac command, 26
tail command, 110, 373
talk command, 242, 374
tar command
  archiving files with, 264–265
  flags and arguments for, 374–375
  gzip used with, 274
  transferring multiple files with, 248
  unarchiving files with, 266
tcsh shell, 55
tee command, 123, 375
telnet
  Escape character in, 239
  flags and arguments for, 375
  logging in remotely with, 239–240
  ssh program compared to, 8
temporary shells, 59–60
TERM environment variable, 312
terminal settings, 69
text
  adding and deleting in **vi**, 84
  cutting and pasting in **nano**, 77
  finding in files, 86, 111
  searching/replacing in **vi**, 86–87
tidy utility, 304–306, 375–376
tilde (~) character, 17

time
  manually setting, 302
  real vs. user and system, 184
  setting with **ntpdate**, 302
  updating date and, 35
time command, 183–184, 321, 322, 376
timing jobs, 183–184
tmux utility, 285
tn3270 program, 240
top utility, 297
touch command, 34–35, 376
tr command, 124–125, 376
traceroute command, 254, 255–256, 376–377
translating case, 124, 125
troubleshooting
  ftp connections, 246
  terminal settings, 69
  traceroute problems, 256

U

umask command, 105–106, 377
unalias command, 377
uname command, 132, 377
unarchiving files
  tar command for, 266
  uncompressing while, 274
uncompress command, 268, 274
uncompressing files
  about, 268
  decoding and, 273
  gunzip for, 270
  unarchiving while, 274
  uncompress for, 268
  unzip for, 272
uniq command, 122, 377
units utility, 280, 377
Unix
  accessing, 3–5
  choosing flavor of, 7
  command reference for, 323–327
  common directories in, 27, 341–342
  connecting to remotely, 8–10
  installing, 5–6
  list of key files in, 340
  logging in to, 11–12
  logging out of, 30
  passwords for, 13–14
  reasons for using, 4
  root users in, 43, 287, 288, 291
  shells available in, 55–56
  software installation on, 295
  table of commands, flags, and arguments, 343–383
  typing commands in, xi–xii
untarring files, 274
unzip command, 272, 378
unzipping files, 270, 272, 273
user and system time, 184
userid
  case sensitivity of, 11
  changing with **su**, 67–68
  getting info about, 144
users
  communicating with other, 241–242
  file ownership by, 90
  info on logged-in, 138–143
  monitoring, 297, 298
  See also **root** users
utilities
  bc, 278
  cal, 276–277, 345
  commands list for, 337
  defined, 275
  dmesg, 300–301
  expr, 279, 349
  free, 297
  last, 297
  look, 281, 356
  rsync, 318–319, 368–370
  screen, 284–285, 370–371
  script, 282–283
  sudo, 99, 288–290, 373
  tidy, 304–306, 375–376
  tmux, 285
  top, 297
  units, 280, 377
  watch, 299, 379
  xargs, 286
uudecode command, 263, 273, 378
uuencode command, 260–262, 378

V

vacation program, 228–229, 233, 378–379
vi editor, 72, 74, 81–88
  about, 74
  adding/deleting in, 84
  composing email in, 220
  exiting, 88
  getting help in, 82
  importing files into, 85
  modes used in, 81, 82
  removing line numbers in, 196
  saving files in, 83
  searching/replacing in, 86–87
  starting, 82
viewing
  bash configuration files, 161–163
  beginning of files, 109
  ending of files, 110
  file contents with **more**, 24
  file systems, 133–135
  hidden files, 38
  session history, 62–63, 65–66
  specified number of lines in files, 109
  zsh configuration files, 152–155

vim editor, 74, 82
  See also **vi** editor
VirtualBox, xi, 6

W

w command, 142–143, 379
wall command, 238
watch utility, 299, 379
wc command, 108, 379
web browsers
  links, 249–250, 355
  lynx, 251–252, 358–360
websites
  downloading with **wget**, 253
  surfing, 249–252
  See also Internet
wget command, 253, 379–381
whereis command, 381–382
who command, 141, 382
whoami command, 141
whois query server, 258
wildcards
  find command and, 46, 47
  guidelines for using, 23
  removing multiple files using, 43
Windows
  installing Unix and, 6
  using **zip** for files accessed in, 269, 271
  viewing hard drives in, 133
WinZip program, 271
words
  counting in files, 108
  looking up, 281
world permissions, 93
write command, 241, 382
write permission, 90
writing
  if-then statements, 202–203
  procmail recipes, 232
  scripts, 189–190, 333

X

xargs utility, 286

Y

ydecode command, 382
yencode command, 382
yppasswd command, 14

Z

zip command, 271, 382–383
zipping files
  gzip for, 269
  zip for, 271
zsh shell
  adding variables in, 150–151
  alias setup for, 169–170
  changing variables in, 150–151
  completion used in, 64
  features of, 55
  flags and arguments for, 382
  path changes, 156–157
  prompt changes, 158–160
  redirecting **stderr** in, 321–322
  session history viewed in, 65–66
  setting your prompt for, 159
  showing current environment in, 148
  viewing configuration files in, 152–155
  writing scripts using, 190