

SECOND EDITION

bash

Bourne

zsh

Korn

**SAMS**

***Teach Yourself***

# Shell Programming

*in* **24**  
***Hours***

Sriranga Veeraraghavan

Sriranga Veeraraghavan



**SAMS**  
**Teach Yourself**

# Shell Programming

in **24** Hours

SECOND EDITION

**SAMS**

800 East 96th St., Indianapolis, Indiana, 46240 USA

# Sams Teach Yourself Shell Programming in 24 Hours, Second Edition

**Copyright © 2002 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. Neither is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-672-32358-3

Library of Congress Catalog Card Number: 2001096631

Printed in the United States of America

First Printing: April 2002

06 05            13 12 11 10 9 8 7

## Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams 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.

## 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@pearsoned.com**

### ACQUISITIONS EDITOR

Katie Purdum

### DEVELOPMENT EDITOR

Steve Rowe

### TECHNICAL EDITOR

Michael Watson

### MANAGING EDITOR

Charlotte Clapp

### PROJECT EDITOR

Natalie Harris

### COPY EDITORS

Kezia Endsley  
Rhonda Tinch-Mize

### INDEXER

Kelly Castell

### PROOFREADERS

Linda Seifert  
Karen Whitehouse

### INTERIOR DESIGN

Gary Adair

### COVER DESIGN

Aren Howell

### PAGE LAYOUT

Stacey Richwine-DeRome

# Contents at a Glance

Introduction	1
<b>PART I Introduction to UNIX and Shell Tools</b>	<b>7</b>
Hour 1 Shell Basics	9
2 Script Basics	21
3 Working with Files	37
4 Working with Directories	53
5 Input and Output	71
6 Manipulating File Attributes	89
7 Processes	105
<b>PART II Shell Programming</b>	<b>119</b>
Hour 8 Variables	121
9 Substitution	135
10 Quoting	147
11 Flow Control	159
12 Loops	181
13 Parameters	197
14 Functions	213
15 Text Filters	231
16 Filtering Text with Regular Expressions	249
17 Filtering Text with awk	267
18 Other Tools	293
<b>PART III Advanced Topics</b>	<b>311</b>
Hour 19 Signals	313
20 Debugging	325
21 Problem Solving with Functions	341
22 Problem Solving with Shell Scripts	359
23 Scripting for Portability	389
24 Shell Programming FAQs	403

<b>PART IV</b>	<b>Appendixes</b>	<b>417</b>
Appendix A	Command Quick Reference	419
B	Glossary	433
C	Answers to Questions	441
D	Shell Function Library	461
	Index	465

# Contents

Introduction	1
<b>PART I Introduction to UNIX and Shell Tools</b>	<b>7</b>
<b>Hour 1 Shell Basics</b>	<b>9</b>
What Is a Command? .....	10
Simple Commands.....	11
Complex Commands .....	11
Compound Commands .....	12
What Is the Shell?.....	13
The Shell Prompt.....	14
Different Types of Shells.....	14
Summary .....	18
Questions.....	19
Terms.....	19
<b>Hour 2 Script Basics</b>	<b>21</b>
The UNIX System .....	22
Logging In .....	23
Shell Modes and Initialization .....	24
Initialization Procedures .....	24
Initialization File Contents .....	26
Interactive and Non-Interactive Shells .....	28
Getting Help .....	31
man .....	31
Online Resources.....	34
Summary .....	35
Questions.....	35
Terms.....	35
<b>Hour 3 Working with Files</b>	<b>37</b>
Listing Files .....	38
Hidden Files.....	39
Option Grouping .....	40
File Contents .....	41
cat .....	41
wc .....	43
Manipulating Files .....	46
Copying Files (cp).....	46
Renaming Files (mv) .....	48
Removing Files (rm) .....	49

Summary .....	50
Questions.....	51
Terms.....	51
<b>Hour 4 Working with Directories</b>	<b>53</b>
The Directory Tree .....	54
Filenames.....	54
Pathnames .....	55
Switching Directories .....	57
Home Directories.....	57
Changing Directories.....	58
Listing Files and Directories.....	60
Listing Directories.....	60
Listing Files.....	61
Manipulating Directories .....	62
Creating Directories.....	62
Copying Files and Directories.....	63
Moving Files and Directories .....	64
Removing Directories .....	66
Summary .....	68
Questions.....	68
Terms.....	69
<b>Hour 5 Input and Output</b>	<b>71</b>
Output .....	71
Output to the Terminal .....	72
Output Redirection .....	77
Input .....	79
Input Redirection.....	79
Reading User Input .....	81
Pipelines.....	81
File Descriptors.....	82
Associating Files with a File Descriptor.....	82
General Input/Output Redirection.....	83
Summary .....	87
Questions.....	87
Terms.....	87
<b>Hour 6 Manipulating File Attributes</b>	<b>89</b>
File Types .....	89
Determining a File's Type .....	90
Regular Files .....	90
Links .....	91
Device Files.....	94
Named Pipes .....	95

Owners, Groups, and Permissions .....	95
Viewing Permissions .....	96
Changing File and Directory Permissions.....	98
Changing Owners and Groups .....	101
Summary .....	103
Questions.....	103
Terms.....	104
<b>Hour 7 Processes</b> .....	<b>105</b>
Starting a Process .....	105
Foreground Processes .....	106
Background Processes .....	106
Listing and Terminating Processes .....	111
jobs .....	112
ps Command .....	112
Killing a Process (kill Command).....	114
Parent and Child Processes.....	114
Subshells .....	115
Process Permissions.....	116
Overlaying the Current Process (exec Command) .....	116
Summary .....	117
Questions.....	117
Terms.....	117
<b>PART II Shell Programming</b> .....	<b>119</b>
<b>Hour 8 Variables</b> .....	<b>121</b>
Working with Variables.....	121
Scalar Variables .....	122
Array Variables .....	124
Read-Only Variables .....	128
Unsetting Variables .....	129
Environment and Shell Variables .....	129
Exporting Environment Variables .....	130
Shell Variables .....	131
Summary .....	132
Questions.....	132
Terms.....	133
<b>Hour 9 Substitution</b> .....	<b>135</b>
Filename Substitution (Globbing) .....	136
The * Meta-Character .....	136
The ? Meta-Character .....	138
Matching Sets of Characters .....	139



Variable Substitution.....	141
Default Value Substitution.....	141
Default Value Assignment .....	142
Null Value Error.....	142
Substitute When Set .....	143
Command and Arithmetic Substitution .....	143
Command Substitution .....	143
Arithmetic Substitution .....	144
Summary .....	146
Questions.....	146
Terms.....	146

## Hour 10 Quoting 147

Quoting with Backslashes.....	148
Meta-Characters and Escape Sequences .....	149
Using Single Quotes .....	149
Using Double Quotes .....	150
Quoting Rules and Situations .....	151
Quoting Ignores Word Boundaries .....	152
Combining Quoting in Commands .....	152
Embedding Spaces in a Single Argument .....	152
Quoting Newlines to Continue on the Next Line .....	153
Quoting to Access Filenames Containing Special Characters .....	154
Quoting Regular Expression Wildcards .....	155
Quoting the Backslash to Enable echo Escape Sequences .....	155
Quoting Wildcards for cpio and find .....	156
Summary .....	157
Questions.....	158
Terms.....	158

## Hour 11 Flow Control 159

The if Statement .....	160
An if Statement Example .....	160
Using test .....	163
The case Statement.....	175
A case Statement Example .....	175
Using Patterns .....	177
Summary .....	178
Questions.....	178
Terms.....	179

## Hour 12 Loops 181

The while Loop .....	181
Nesting while Loops .....	183
Validating User Input with while .....	184

---

Input Redirection and while .....	185
The until Loop .....	187
The for and select Loops .....	188
The for Loop .....	188
The select Loop .....	190
Loop Control .....	192
Infinite Loops and the break Command .....	192
The continue Command .....	194
Summary .....	195
Questions .....	195
Terms .....	196
<b>Hour 13 Parameters</b> .....	<b>197</b>
Special Variables .....	198
Using \$0 .....	198
Options and Arguments .....	200
Dealing with Arguments .....	201
Using basename .....	201
Common Argument Handling Problems .....	203
Option Parsing in Shell Scripts .....	205
Using getopt .....	206
Summary .....	210
Questions .....	210
Terms .....	211
<b>Hour 14 Functions</b> .....	<b>213</b>
Using Functions .....	213
Executing Functions .....	214
Aliases Versus Functions .....	217
Unsetting Functions .....	218
Understanding Scope, Recursion, Return Codes, and Data Sharing .....	218
Scope .....	218
Recursion .....	221
Return Codes .....	223
Data Sharing .....	223
Moving Around the File System .....	223
Summary .....	228
Questions .....	228
Terms .....	229
<b>Hour 15 Text Filters</b> .....	<b>231</b>
The head and tail Commands .....	231
The head Command .....	232
The tail Command .....	233

Using grep .....	234
Looking for Words.....	235
Reading From STDIN .....	236
Line Numbers .....	237
Listing Filenames Only .....	238
Counting Words .....	238
The tr Command .....	239
The sort Command.....	241
The uniq Command.....	241
Sorting Numbers .....	242
Using Character Classes with tr.....	244
Summary .....	245
Questions.....	246
Terms.....	247
<b>Hour 16 Filtering Text with Regular Expressions</b>	<b>249</b>
The Basics of awk and sed .....	250
Invocation Syntax .....	250
Basic Operation .....	250
Regular Expressions .....	251
Using sed .....	257
Printing Lines .....	258
Deleting Lines .....	259
Performing Substitutions .....	260
Using Multiple sed Commands.....	262
Using sed in a Pipeline .....	263
Summary .....	264
Questions.....	264
Terms.....	265
<b>Hour 17 Filtering Text with awk</b>	<b>267</b>
What Is awk? .....	267
Basic Syntax .....	268
Field Editing .....	269
Taking Pattern-Specific Actions .....	270
Comparison Operators.....	271
Using STDIN as Input.....	274
Using awk Features .....	275
Variables .....	276
Flow Control .....	283
Summary .....	288
Questions.....	289
Terms.....	291

<b>Hour 18 Other Tools</b>	<b>293</b>
The Built-In Commands .....	293
The eval Command.....	294
The : Command .....	294
The type Command.....	296
The sleep Command .....	297
The find Command .....	298
find: Starting Directory .....	299
find: -name Option .....	300
find: -type Option .....	300
find: -mtime, -atime, -ctime .....	301
find: -size Option .....	302
find: Combining Options.....	302
find: Negating Options .....	303
find: -print Action.....	303
find: -exec Action.....	303
xargs.....	304
The expr Command .....	306
expr and Regular Expressions.....	307
The bc Command .....	307
Summary .....	308
Questions.....	309
Terms.....	309
 <b>PART III Advanced Topics</b>	 <b>311</b>
<b>Hour 19 Signals</b>	<b>313</b>
How Are Signals Represented? .....	314
Getting a List of Signals .....	314
Default Actions .....	315
Delivering Signals .....	315
Dealing with Signals.....	316
The trap Command.....	317
Cleaning Up Temporary Files .....	317
Ignoring Signals.....	319
Setting Up a Timer .....	320
Summary .....	324
Questions.....	324
Terms.....	324
 <b>Hour 20 Debugging</b>	 <b>325</b>
Enabling Debugging .....	326
Using the set command .....	327

Using Syntax Checking .....	328
Why Syntax Checking Is Important .....	329
Using Verbose Mode .....	331
Shell Tracing.....	332
Finding Syntax Bugs Using Shell Tracing .....	333
Finding Logical Bugs Using Shell Tracing.....	335
Using Debugging Hooks .....	337
Summary .....	339
Questions.....	339
Terms.....	340
<b>Hour 21 Problem Solving with Functions</b>	<b>341</b>
Library Basics .....	341
What Is a Library?.....	342
Using a Library .....	342
Creating a Library.....	343
Naming the Library .....	343
Naming the Functions .....	344
Displaying Error and Warning Messages .....	344
Asking Questions.....	345
Checking Disk Space.....	351
Obtaining a Process ID by its Process Name .....	354
Getting a User's Numeric User ID .....	355
Summary .....	356
Questions.....	356
Terms.....	357
<b>Hour 22 Problem Solving with Shell Scripts</b>	<b>359</b>
Startup Scripts.....	360
System Startup.....	360
Developing an Init Script .....	364
Maintaining an Address Book .....	373
Showing People .....	375
Adding a Person .....	377
Deleting a Person.....	380
Summary .....	385
Questions.....	385
Terms.....	387
<b>Hour 23 Scripting for Portability</b>	<b>389</b>
Determining UNIX Versions.....	390
BSD .....	390
System V .....	390
Linux .....	391
Using uname to Determine the UNIX Version.....	392
Determining the UNIX Version Using a Function .....	394

Techniques for Increasing Portability .....	396
Conditional Execution .....	396
Abstraction .....	397
Summary .....	400
Question .....	401
Terms .....	401
<b>Hour 24 Shell Programming FAQs</b>	<b>403</b>
Shell and Command Questions .....	404
Variable and Argument Questions .....	409
File and Directory Questions .....	412
Summary .....	416
<b>PART IV Appendixes</b>	<b>417</b>
<b>APPENDIX A Command Quick Reference</b>	<b>419</b>
Reserved Words and Built-in	
Shell Commands .....	420
Conditional Expressions .....	423
File Tests .....	423
String Tests .....	424
Integer Comparisons .....	424
Compound Expressions .....	424
Arithmetic Expressions (ksh, bash, and zsh Only) .....	424
Integer Expression Operators .....	425
Parameters and Variables .....	426
User-Defined Variables .....	426
Special Variables .....	427
Shell Variables .....	428
Input/Output .....	428
Input and Output Redirection .....	429
Here Document .....	429
Pattern Matching and Regular Expressions .....	430
Filename Expansion and Pattern Matching .....	430
Limited Regular Expression Wildcards .....	430
Extended Regular Expression Wildcards .....	430
<b>APPENDIX B Glossary</b>	<b>433</b>
<b>APPENDIX C Answers to Questions</b>	<b>441</b>
<b>APPENDIX D Shell Function Library</b>	<b>461</b>
Index	465

# About the Author

**SRIRANGA VEERARAGHAVAN** is a material scientist by training and a software engineer by trade. He has several years of software development experience in C, Java, Perl, and Bourne Shell and has contributed to several books, including *Solaris 8: Complete Reference*, *UNIX Unleashed* and *Special Edition Using UNIX*. Sriranga graduated from the University of California at Berkeley in 1997 and is presently pursuing further studies. He is currently employed in the Server Appliance group at Sun Microsystems, Inc. Before joining Sun, Sriranga was employed at Cisco Systems, Inc. Among other interests, Sriranga enjoys mountain biking, classical music, and playing Marathon with his brother Srivathsa. Sriranga can be reached via e-mail at [ranga@soda.berkeley.edu](mailto:ranga@soda.berkeley.edu).

# Dedication

*For my grandmother, who taught me to love the English language.*

*For my mother, who taught me to love programming languages.*

# Acknowledgments

Writing a book on shell programming is a daunting task, due to the myriad UNIX versions and shell versions that are available. Thanks to the hard work of my development editor Steve Rowe, my technical editor Michael Watson, and my copy editor Kezia Endsley, I was able to make sure the book covered the material completely and correctly. Their suggestions and comments have helped enormously.

In addition to the technical side of the book, the task of coordinating and managing the publishing process is a difficult one. The assistance of my acquisitions editor, Kathryn Purdum, in handling all of the editorial issues and patiently working with me to keep this book on schedule was invaluable.

Working on a book takes a lot of time and makes it difficult to concentrate on work and family activities. Thanks to the support of my manager, Larry Coryell, my parents, my brother Srivathsa, and my uncle and aunt Srinvasa and Suma, I was able to balance work, family, and authoring.

Thanks to everyone else on the excellent team at Sams who worked on this book. Without their support, this book would not exist.



# Tell Us What You Think!

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

You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

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

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

Email:      [opensource@sampublishing.com](mailto:opensource@sampublishing.com)

Mail:        Mark Taber  
              Sams Publishing  
              800 East 96th Street  
              Indianapolis, IN 46240 USA

# Introduction

In recent years, the UNIX operating system has seen a huge boost in its popularity, especially with the emergence of Linux. For programmers and users of UNIX, this comes as no surprise: UNIX was designed to provide an environment that's powerful yet easy to use.

One of the main strengths of UNIX is that it comes with a large collection of standard programs. These programs perform a wide variety of tasks from listing your files to reading e-mail. Unlike other operating systems, one of the key features of UNIX is that these programs can be combined to perform complicated tasks and solve your problems.

One of the most powerful standard programs available in UNIX is the shell. The *shell* is a program that provides a consistent and easy-to-use environment for executing programs in UNIX. If you have ever used a UNIX system, you have interacted with the shell.

The main responsibility of the shell is to read the commands you type and then ask the UNIX kernel to perform these commands. In addition to this, the shell provides several sophisticated programming constructs that enable you to make decisions, repeatedly execute commands, create functions, and store values in variables.

This book concentrates on the standard UNIX shell called the Bourne shell. When Dennis Ritchie and Ken Thompson were developing much of UNIX in the early 1970s, they used a very simple shell. The first real shell, written by Stephen Bourne, appeared in the mid 1970s. The original Bourne shell has changed slightly over the years; some features were added and others were removed, but its syntax and its resulting power have remained the same.

The most attractive feature of the shell is that it enables you to create scripts. *Scripts* are files that contain a list of commands you want to run. Because every script is contained in a file and every file has a name, scripts enable you to combine existing programs to create completely new programs that solve your problems. This book teaches you how to create, execute, modify, and debug shell scripts quickly and easily. After you get used to writing scripts, you will find yourself solving more and more problems with them.

## How This Book Is Organized

This book assumes that you have some familiarity with UNIX and know how to log in, create, and edit files, as well as how to work with files and directories to a limited extent. If you haven't used UNIX in a while or you aren't familiar with one of these topics, don't worry; the first part of this book reviews this material thoroughly.

This book is divided into three parts:

- Part I is an introduction to UNIX, the shell, and some common tools.
- Part II covers programming using the shell.
- Part III covers advanced topics in shell programming.

Part I consists of Chapters 1 through 7. The following material is covered in the individual chapters:

- Chapter 1, “Shell Basics,” discusses several important concepts related to the shell and describes the different versions of the shell.
- Chapter 2, “Script Basics,” describes the process of creating and running a shell script. It also covers the login process and the different modes in which the shell executes.
- Chapters 3, “Working with Files,” and 4, “Working with Directories,” provide an overview of the commands used when working with files and directories. These chapters show you how to list the contents of a directory, view the contents of a file, and manipulate files and directories.
- Chapter 5, “Input and Output” covers the `echo`, `printf`, and `read` commands along with the `<` and `>` input redirection operators. This chapter also covers using file descriptors.
- Chapter 6, “Manipulating File Attributes,” introduces the concept of file attributes. It covers the different types of files along with how to modify a file’s permissions.
- Chapter 7, “Processes,” shows you how to start and stop a process. It also explains the term *process ID* and how you can view them.

By this point, you should have a good foundation in the UNIX basics. This will enable you to start writing shell scripts that solve real problems using the concepts covered in Part II. Part II is the heart of this book, consisting of Chapters 8 through 18. It teaches you about all the tools available when programming in the shell. The following material is covered in these chapters:

- Chapter 8, “Variables,” explains the use of variables in shell programming, shows you how to create and delete variables, and explains the concept of environment variables.
- Chapters 9, “Substitution,” and 10, “Quoting,” cover the topics of substitution and quoting. Chapter 9 shows you the four main types of substitution: filename, variable, command, and arithmetic substitution. Chapter 10 shows you the behavior of the different types of quoting and its affect on substitution.

- Chapters 11, “Flow Control,” and 12, “Loops,” provide complete coverage of flow control and looping. The flow control constructs `if` and `case` are covered along with the loop constructs `for` and `while`.
- Chapter 13, “Parameters,” shows you how to write scripts that use command-line arguments. The special variables and the `getopts` command are covered in detail.
- Chapter 14, “Functions,” discusses shell functions. Functions provide a mapping between a name and a set of commands. Learning to use functions in a shell script is a powerful technique that helps you solve complicated problems.
- Chapters 15, “Text Filters,” 16, “Filtering Text with Regular Expressions,” and 17, “Filtering Text with `awk`,” cover text filtering. These chapters show you how to use a variety of UNIX commands including `grep`, `tr`, `sed`, and `awk`.
- Chapter 18, “Other Tools,” provides an introduction to some tools that are used in shell programming. Some of the commands that are discussed include `type`, `find`, `bc`, and `expr`.

At this point, you will know enough about the shell and the external tools available in UNIX that you can solve most problems. The last part of the book, Part III, is designed to help you solve the most difficult problems encountered in shell programming. Part III spans Chapters 19 through 24 and covers the following material:

- Chapter 19, “Signals,” explains the concept of signals and shows you how to deliver a signal and how to deal with a signal using the `trap` command.
- Chapter 20, “Debugging,” discusses the shell’s built-in debugging tools. It shows you how to use syntax checking and shell tracing to track down bugs and fix them.
- Chapters 21, “Problem Solving with Functions,” and 22, “Problem Solving with Shell Scripts,” cover problem solving. Chapter 21 covers problems that can be solved using functions. Chapter 22 introduces some real-world problems and shows you how to solve them using a shell script.
- Chapter 23, “Scripting for Portability,” covers the topic of portability. In this chapter, you will rewrite several scripts from previous chapters to be portable to different versions of UNIX.
- Chapter 24, “Shell Programming FAQs,” is a question-and-answer chapter. Several common programming questions are presented along with detailed answers and examples.

Each chapter in this book includes complete syntax descriptions for the various commands along with several examples to illustrate the use of commands. The examples are designed to show you how to apply the commands to solve real problems. At the end of

each chapter are a few questions that you can use to check your progress. Some of the questions are short answers, whereas others require you to write scripts.

After Chapter 24, four appendixes are available for your reference:

- Appendix A, “Command Quick Reference,” provides a complete command reference.
- Appendix B, “Glossary,” contains the terms used in this book.
- Appendix C, “Answers to Questions,” contains the answers to all the questions in the book.
- Appendix D, “Shell Function Library,” contains a listing of the shell function library discussed in Chapter 21, “Problem Solving with Functions.”

## About the Examples

As you work through the chapters, try typing in the examples to get a better feeling for how the computer responds and how each command works. After you get an example working, try experimenting with the example by changing commands. Don’t be afraid to experiment. Experiments (both successes and failures) teach you important things about UNIX and the shell.

Many of the examples and the answers to the questions are available for downloading from the following URL:

<http://www.csua.berkeley.edu/~ranga/downloads/tysp2.tar.Z>

After you have downloaded this file, change to the directory where the file was saved and execute the following commands:

```
$ uncompress tysp2.tar.Z
$ tar -xvf tysp2.tar
```

This creates a directory named `tysp2` that contains the examples from this book.

There is no warranty of any kind on the examples in this book. Much effort has been placed into making the examples as portable as possible. To this end the examples have been tested on the following versions of UNIX:

- Sun Solaris versions 2.5.1 to 8
- Hewlett-Packard HP-UX versions 10.10 to 11.0
- OpenBSD versions 2.6 to 2.9
- Apple MacOS X 10.0 to 10.1.2
- Red Hat Linux versions 4.2, 5.1, 5.2, 6.0, and 6.2
- FreeBSD versions 2.2.6 and 4.0 to 4.3

It is possible that some of the examples might not work on other versions of UNIX. If you encounter a problem or have a suggestion about improvements to the examples or the content of the book, please feel free to contact me at the following e-mail address:

ranga@soda.berkeley.edu

I appreciate any suggestions and feedback you have regarding this book.

## Conventions Used in This Book

Features in this book include the following:



Notes give you comments and asides about the topic at hand, as well as full explanations of certain concepts.



Tips provide great shortcuts and hints on how to program in shell more effectively.



Cautions warn you against making your life miserable and avoiding the pitfalls in programming.

### NEW TERM

New terms appear in *italic*. Each of the new terms covered in a chapter is listed at the end of that chapter in the “Terms” section.

At the end of each chapter, you’ll find the handy Summary and Quiz sections (with answers found in Appendix C).

In addition, you’ll find various typographic conventions throughout this book:

- Commands, variables, directories, and files appear in text in a special monospaced font.
- Commands and such that you type appear in **boldface type**.
- Placeholders in syntax descriptions appear in a *monospaced italic* typeface. This indicates that you will replace the placeholder with the actual filename, parameter, or other element that it represents.

*This page intentionally left blank*

# HOUR 3



## Working with Files

In UNIX there are two basic types of files: ordinary and special. An *ordinary* file contains data, text, or program instructions. Almost all of the files on a UNIX system are ordinary files. This chapter covers operations on ordinary files.

*Special* files are mainly used to provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Some special files are similar to aliases or shortcuts and enable you to access a single file using different names. Special files are covered in Chapter 6, “Manipulating File Attributes.”

Both ordinary and special files are stored in directories. *Directories* are similar to folders in the Mac OS or Windows, and they are covered in detail in Chapter 4, “Working with Directories.”

In this chapter, we will examine ordinary files, concentrating on the following topics:

- Listing files
- File contents
- Manipulating files



## Listing Files

We'll start by using the `ls` (short for **list**) command to list the contents of the current directory:

```
$ ls
```

The output will be similar to the following:

Desktop	Icon	Music	Sites
Documents	Library	Pictures	Temporary Items
Downloads	Movies	Public	

We can tell that several items are in the current directory, but this output does not tell us whether these items are files or directories. To find out which of the items are files and which are directories, we can specify the `-F` option to `ls`. An option is an argument that starts with the hyphen or dash character, '- '.

The following example illustrates the use of the `-F` option of `ls`:

```
$ ls -F
```

Now the output for the directory is slightly different:

Desktop/	Icon	Music/	Sites/
Documents/	Library/	Pictures/	Temporary Items/
Downloads/	Movies/	Public/	

As you can see, some of the items now have a `/` at the end, indicating each of these items is a directory. The other items, such as `icon`, have no character appended to them. This indicates that they are ordinary files.

When the `-F` option is specified to `ls`, it appends a character indicating the file type of each of the items it lists. The exact character depends on your version of `ls`. For ordinary files, no character is appended. For special files, a character such as `!`, `@`, or `#` is appended to the filename. For more information on the `-F` options, check the UNIX manual page for the `ls` command. You can do this as follows:

```
$ man ls
```

### Options Are Case Sensitive

The options that can be specified to a command, such as `ls`, are case sensitive. When specifying an option, you need to make sure that you have specified the correct case for the option. For example, the output from the `-F` option to `ls` is different from the output produced when the `-f` option is specified.

So far, you have seen `ls` list more than one file on a line. Although this is fine for humans reading the output, it is hard to manipulate in a shell script. Shell scripts are geared toward dealing with lines of text, not the individual words on a line. Although external tools, such as the `awk` language covered in Chapter 17, “Filtering Text with `awk`,” can be used to deal with multiple words on a line, it is much easier to manipulate the output when each file is listed on a separate line. You can modify the output of `ls` to this format by using the `-1` option. For example,

```
$ ls -1
```

produces the following listing:

```
Desktop
Documents
Downloads
Icon
Library
Movies
Music
Pictures
Public
Sites
Temporary Items
```

3

## Hidden Files

In the examples you have seen thus far, the output has listed only the visible files and directories. You can also use `ls` to list invisible or hidden files and directories. An *invisible* or *hidden* file is one whose first character is a dot or period (`.`). Many programs, including the shell, use such files to store configuration information. Some common examples of invisible files include

- `.profile`, the Bourne shell (`sh`) initialization script
- `.kshrc`, the Korn Shell (`ksh`) initialization script
- `.cshrc`, the C Shell (`csh`) initialization script
- `.rhosts`, the remote shell configuration file

All files that do not start with the `.` character are considered *visible*.

To list invisible files, specify the `-a` option to `ls`:

```
$ ls -a
```

The directory listing now resembles this:

```
.          .FBCLockFolder  Icon          Public
..         .ssh          Library       Sites
.CFUserTextEncoding Desktop        Movies        Temporary Items
```

```
.DS_Store      Documents      Music
.FBCIndex      Downloads      Pictures
```

As you can see, this directory contains several invisible files.

Notice that in this output, the file type information is missing. To get the file type information, specify the `-F` and the `-a` options as follows:

```
$ ls -a -F
```

The output changes to the following:

```
./              .ssh/           Movies/
../            Desktop/        Music/
.CFUserTextEncoding Documents/      Pictures/
.DS_Store      Downloads/     Public/
.FBCIndex      Icon?          Sites/
.FBCLockFolder/ Library/        Temporary Items/
```

With the file type information, you see that there are two invisible directories (`.` and `..`). These directories are special entries present in all directories. The first one, `.`, represents the current directory, whereas the second one, `..`, represents the parent directory. These concepts are discussed in greater detail in Chapter 4.

## Option Grouping

In the previous example, you specified the options to `ls` separately. You could have grouped the options together, as follows:

```
$ ls -aF
$ ls -Fa
```

Both of these commands are equivalent to the following command:

```
$ ls -a -F
```

The order of the options does not matter to `ls`. As an example of option grouping, consider the following equivalent commands:

```
ls -1 -a -F
ls -1aF
ls -a1F
ls -Fa1
```

All permutations of the options `-1`, `-a`, and `-F` produce the same output:

```
./
../
.CFUserTextEncoding
.DS_Store
```

```
.FBCIndex
.FBCLockFolder/
.ssh/
Desktop/
Documents/
Downloads/
Icon?
Library/
Movies/
Music/
Pictures/
Public/
Sites/
Temporary Items/
```

## File Contents

3

In the last section we looked at listing files and directories with the `ls` command. In this section we will look at the `cat` and `wc` commands. The `cat` command lets you view the contents of a file. The `wc` command gives you information about the number of words and lines in a file.

### cat

To view the contents of a file, we can use the `cat` (short for **concatenate**) command as follows:

```
cat [opts] file1 ... fileN
```

Here `opts` are one or more of the options understood by `cat`, and `file1...fileN` are the names of the files whose contents should be printed. The options, `opts`, are optional and can be omitted. Two commonly used options are discussed later in this section.

The following example illustrates the use of `cat`:

```
$ cat fruits
```

This command prints the contents of a file called `fruits`:

Fruit	Price/lbs	Quantity
Banana	\$0.89	100
Peach	\$0.79	65
Kiwi	\$1.50	22
Pineapple	\$1.29	35
Apple	\$0.99	78

If more than one file is specified, the output includes the contents of both files concatenated together. For example, the following command outputs the contents of the files `fruits` and `users`:

```
$ cat fruits users
Fruit      Price/lbs      Quantity
Banana     $0.89          100
Peach      $0.79          65
Kiwi       $1.50          22
Pineapple  $1.29          35
Apple      $0.99          78

ranga
vathsa
amma
```

## Numbering Lines

The `-n` option of `cat` will number each line of output. It can be used as follows:

```
$ cat -n fruits
```

This produces the output

```
 1 Fruit      Price/lbs      Quantity
 2 Banana     $0.89          100
 3 Peach      $0.79          65
 4 Kiwi       $1.50          22
 5 Pineapple  $1.29          35
 6 Apple      $0.99          78
 7
```

From this output, you can see that the last line in this file is blank. We can ask `cat` to skip numbering blank lines using the `-b` option as follows:

```
$ cat -b fruits
```

Now the output resembles the following:

```
 1 Fruit      Price/lbs      Quantity
 2 Banana     $0.89          100
 3 Peach      $0.79          65
 4 Kiwi       $1.50          22
 5 Pineapple  $1.29          35
 6 Apple      $0.99          78

```

The blank line is still presented in the output, but it is not numbered. If the blank line occurs in the middle of a file, it is printed but not numbered:

```
$ cat -b hosts
 1 127.0.0.1      localhost      loopback

 2 128.32.43.52   soda.berkeley.edu  soda
```

If multiple files are specified, the contents of the files are concatenated in the output, but line numbering is restarted at 1 for each file. As an illustration, the following command,

```
$ cat -b fruits users
```

produces the output

```

1 Fruit           Price/lbs      Quantity
2 Banana          $0.89         100
3 Peach           $0.79         65
4 Kiwi            $1.50         22
5 Pineapple       $1.29         35
6 Apple           $0.99         78

1 ranga
2 vathsa
3 amma
```

## WC

Now let's look at getting some information about the contents of a file. Using the `wc` command (short for **w**ord **c**ount), we can get a count of the total number of lines, words, and characters contained in a file. The basic syntax of this command is:

```
wc [opts] files
```

Here `opts` are one or more of the options given in Table 3.1, and `files` are the files you want examined. The options, `opts`, are optional and can be omitted.

**TABLE 3.1** `wc` Options

<i>Option</i>	<i>Description</i>
-l	Count of the number of lines.
-w	Count of the number of words.
-m	Count of the number of characters. This option is available on Mac OS X, OpenBSD, Solaris, and HP-UX. This option is not available on FreeBSD and Linux systems.
-c	Count of the number of characters. This option is the Linux and FreeBSD equivalents of the <code>-m</code> option.

When no options are specified, the default behavior of `wc` is to print out a summary of the number of lines, words, and characters contained in a file. For example, the command

```
$ wc fruits
```

produces the following output:

```
      8      18    219 fruits
```

The first number, in this case 8, is the number of lines in the file. The second number, in this case 18, is the number of words in the file. The third number, in this case 219, is the number of characters in the file. At the end of the line, the filename is listed. When multiple files are specified, the filename helps to identify the information associated with a particular file.

If more than one file is specified, `wc` gives the counts for each file along with a total. For example, the command

```
$ wc fruits users
```

produces output similar to the following:

```
      8      18    219 fruits
      3       3     18 users
     11     21    237 total
```

The output on your system might be slightly different.

## Counting Lines

To count the number of lines, the `-l` (as in lines) option can be used. For example, the command

```
$ wc -l fruits
```

produces the output

```
      8 fruits
```

The first number, in this case 8, is the number of lines in the file. The name of the file is listed at the end of the line.

When multiple files are specified, the number of lines in each file is listed along with the total number of lines in all of the specified files. As an example, the command

```
$ wc -l fruits users
```

produces the output

```
      8 fruits
      3 users
     11 total
```

## Counting Words

To count the number of words in a file, the `-w` (as in words) option can be used. For example, the command

```
$ wc -w fruits
```

produces the output

```
18 hosts
```

The first number, in this case 18, is the number of words in the file. The name of the file is listed at the end of the line.

When multiple files are specified, the number of words in each file is listed along with the total number of words in all of the specified files. As an example, the command

```
$ wc -w fruits users
```

produces the output

```
18 fruits
 3 users
21 total
```

## Counting Characters

To count the number of characters, we need to use either the `-m` or the `-c` option. The `-m` option is available on Mac OS X, OpenBSD, Solaris, and HP-UX. On FreeBSD and Linux systems, the `-c` option should be used instead.

For example, on Solaris the command

```
$ wc -m fruits
```

produces the output

```
219 fruits
```

The same output is produced on Linux and FreeBSD systems using the command

```
$ wc -c fruits
```

The first number, in this case 219, is the number of characters in the file. The name of the file is listed at the end of the line.

When multiple files are specified, the number of characters in each file is listed along with the total number of characters in all the specified files. As an example, the command

```
$ wc -m fruits users
```



produces the output

```
219 hosts
 18 users
237 total
```

## Combining Options

The options to `wc` can be grouped together and specified in any order. For example, to obtain a count of the number of lines and words in the file `fruits`, we can use any of the following commands:

```
$ wc -w -l fruits
$ wc -l -w fruits
$ wc -wl fruits
$ wc -lw fruits
```

The output from each of these commands is identical:

```
8      18 fruits
```

The output lists the number of words in the files, followed by the number of lines in the file. The filename is specified at the end of the line. When multiple files are specified, the information for each file is listed along with the appropriate total values.

# Manipulating Files

In the preceding sections, you looked at listing files and viewing their content. In this section, you will look at copying, renaming, and removing files using the `cp`, `mv`, and `rm` commands.

## Copying Files (`cp`)

The `cp` command (short for copy) is used to make a copy of a file. The basic syntax of the command is

```
cp src dest
```

Here `src` is the name of the file to be copied (the source) and `dest` is the name of the copy (the destination). For example, the following command creates a copy of the file `fruits` in a file named `fruits.sav`:

```
$ cp fruits fruits.sav
```

If `dest` is the name of a directory, a copy with the same name as `src` is created in `dest`. For example, the command

```
$ cp fruits Documents/
```

creates a copy of the file `fruits` in the directory `Documents`.

It is also possible to specify multiple source files to `cp`, provided that the destination, `dest`, is a directory. The syntax for copying multiple files is

```
$ cp src1 ... srcN dest
```

Here `src1 ... srcN` are the source files and `dest` is the destination directory. As an example, the following command

```
$ cp fruits users Documents/
```

creates a copy the files `fruits` and `users` in the directory `Documents`.

## Interactive Mode

The default behavior of `cp` is to automatically overwrite the destination file if it exists. This behavior can lead to problems. The `-i` option (short for interactive) can be used to prevent such problems. In interactive mode, `cp` prompts for confirmation before overwriting any files.

Assuming that the file `fruits.sav` exists, the following command

```
$ cp -i fruits fruits.sav
```

results in a prompt similar to the following:

```
overwrite fruits.sav? (y/n)
```

If `y` (yes) is chosen, the file `fruits.sav` is overwritten; otherwise the file is untouched. The actual prompt varies among the different versions of UNIX.

## Common Errors

When an error is encountered, `cp` generates a message. Some common error conditions follow:

- The source, `src`, is a directory.
- The source, `src`, does not exist.
- The destination, `dest`, is not a directory when multiple sources, `src1 ... srcN`, are specified.
- A non-existent destination, `dest`, is specified along with multiple sources, `src1 ... srcN`.
- One of the sources in `src1 ... srcN` is not a file.

The first error type is illustrated by the following command:

```
$ cp Downloads/ fruits
```

Because `src` (`Downloads` in this case) is a directory, an error message similar to the following is generated:

```
cp: Downloads: is a directory
```

In this example, *dest* was the file *fruits*; the same error would have been generated if *dest* was a directory.

The second error type is illustrated by the following command:

```
$ cp fritus fruits.sav
cp: cannot access fritus: No such file or directory
```

Here the filename *fruits* has been misspelled *fritus*, resulting in an error. In this example *dest* was the file *fruits.sav*; the same error would have been generated if *dest* was a directory.

The third error type is illustrated by the following command:

```
$ cp fruits users fruits.sav
usage: cp [-R [-H | -L | -P]] [-f | -i] [-p] src target
        cp [-R [-H | -L | -P]] [-f | -i] [-p] src1 ... srcN directory
```

Because *dest*, in this case *fruits.sav*, is not a directory, a usage statement that highlights the proper syntax for a *cp* command is presented. The output might be different on your system because some versions of *cp* do not display the usage information.

If the file *fruits.sav* does not exist, the error message is

```
cp: fruits.sav: No such file or directory
```

This illustrates the fourth error type.

The fifth error type is illustrated by the following command:

```
$ cp fruits Downloads/ users Documents/
cp: Downloads is a directory (not copied).
```

Although *cp* reports an error for the directory *Downloads*, the other files are correctly copied to the directory *Documents*.

## Renaming Files (*mv*)

The *mv* command (short for move) can be used to change the name of a file. The basic syntax is

```
mv src dest
```

Here *src* is the original name of the file and *dest* is the new name of the file. For example, the command

```
$ mv fruits fruits.sav
```

changes the name of the file `fruits` to `fruits.sav`. There is no output from `mv` if the name change is successful.

If `src` does not exist, an error will be generated. For example,

```
$ mv cp fritus fruits.sav
mv: fritus: cannot access: No such file or directory
```

Similar to `cp`, `mv` does not report an error if `dest` already exists. The old file is automatically overwritten. This problem can be avoided by specifying the `-i` option (short for interactive). In interactive mode, `mv` prompts for confirmation before overwriting any files.

Assuming that the file `fruits.sav` already exists, the command

```
$ mv -i fruits fruits.sav
```

results in a confirmation prompt similar to the following:

```
overwrite fruits.sav?
```

If `y` (yes) is chosen, the file `fruits.sav` is overwritten; otherwise the file is untouched. The actual prompt varies among the different versions of UNIX.

## Removing Files (`rm`)

The `rm` command (short for remove) can be used to remove or delete files. Its syntax is

```
rm file1 ... fileN
```

Here `file1 ... fileN` is a list of one or more files to remove. For example, the command

```
$ rm fruits users
```

removes the files `fruits` and `users`.

Because there is no way to recover files that have been removed using `rm`, you should make sure that you specify only those files you really want removed. One way to ensure this is by specifying the `-i` option (short for interactive). In interactive mode, `rm` prompts before removing every file. For example, the command

```
$ rm -i fruits users
```

produces confirmation prompts similar to the following:

```
fruits: ? (n/y) y
users: ? (n/y) n
```

In this case, you answered `y` (yes) to removing `fruits` and `n` (no) to removing `users`. Thus, the file `fruits` was removed, but the file `users` was untouched.

## Common Errors

The two most common errors when using `rm` are

- One of the specified files does not exist.
- One of the specified files is a directory.

The first error type is illustrated by the following command:

```
$ rm users fritus hosts
rm: fritus non-existent
```

Because the file `fruits` is misspelled as `fritus`, it cannot be removed. The other two files are removed correctly.

The second error type is illustrated by the following command:

```
$ rm fruits users Documents/
rm: Documents directory
```

The `rm` command is unable to remove directories and presents an error message stating this fact. It removes the two other files correctly.

## Summary

In this chapter, the following topics were discussed:

- Listing files using `ls`
- Viewing the content of a file using `cat`
- Counting the words, lines, and characters in a file using `wc`
- Copying files using `cp`
- Renaming files using `mv`
- Removing files using `rm`

Knowing how to perform each of these basic tasks is essential to becoming a good shell programmer. In the chapters ahead, you will use these basics to create scripts for solving real-world problems.

## Questions

1. What are invisible files? How can they be listed with `ls`?
2. Is there any difference in the output of the following commands?
  - a. `$ ls -a1`
  - b. `$ ls -1 -a`
  - c. `$ ls -1a`
3. Which options should be specified to `wc` to count just the number of lines and characters in a file?
4. Given that `hw1`, `hw2`, `ch1`, and `ch2` are files and `book` and `homework` are directories, which of the following commands generates an error message?
  - a. `$ cp hw1 ch2 homework`
  - b. `$ cp hw1 homework hw2 book`
  - c. `$ rm hw1 homework ch1`
  - d. `$ rm hw2 ch2`

## Terms

**Directories** Directories are used to hold ordinary and special files. Directories are similar to folders in Mac OS or Windows.

**Invisible Files** An invisible file is one whose first character is a dot or period (.). Many programs (including the shell) use such files to store configuration information. Invisible files are also referred to as hidden files.

**Option** An option is an argument that starts with the hyphen or dash character, '- '.

**Ordinary File** An ordinary file is a file that contains data, text, or program instructions. Almost all the files on a UNIX system are ordinary files.

**Special Files** Special files are mainly used to provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Some special files are similar to aliases or shortcuts and enable you to access a single file using different names.

*This page intentionally left blank*



# INDEX

## Symbols

- & (ampersand), back-ground processes, 106**
- && and compound operator, 273**
- atime option, find command, 301**
- ` (backquote), command substitution, 143**
- \ (backslash)**
  - echo command escape sequences, 155-156
  - newline character, 154
  - quoting, 148-149
  - tr command, 239
- #!/bin/sh, 404**
- { } (braces), while statement, 286**
- c option, uniq command, 242**
- \$ character, 10**
- ; character, 12**
- : character, 24**
  - shell command, 294-296
  - if statement, 295
  - while statement, 295-296
- / character, 53**
- # character, comments, 30**
- character, getopts command, 206**
- + character, shell tracing, 333**
- : (colon), 420**
- ;; command, case statement, 175**
- . command, including functions and variable definitions in other files, 409**
- ctime option, find command, 301**
- \$ (dollar sign)**
  - field operator, 269
  - newline character, 153
  - quoting with double quotes, 151
- " (double quote), quoting, 150**
- exec action, find command, 303-304**
- f option, tail command, 234**
- i option, grep command, 236**
- k option, sort command, 243**
- l option, grep command, 238**
- [>] (less than sign), quoting, 150**
- ^M (carriage return) removing from files, 415-416**



**-m option, uname command, 393**  
**\$ (meta-character), 252**  
**\* (meta-character), 252**  
**. (meta-character), 252**  
**\ (meta-character), 252**  
**^ (meta-character), 252**  
**-mtime option, find command, 301**  
**-n option, 328**  
     find command, 300  
     grep command, 237  
     sort command, 242-243  
**\$n variable, 198**  
**^ (negation operator), 254**  
**! operator, 171**  
     until loop, 187  
**>> operator, here documents, 80**  
**!= operator, test command, 169**  
**|| operator, 171, 408**  
**&& operator, 171, 408**  
**|| (or) compound operator, 273**  
**% (percent sign), job number prefixes, 109**  
**. (period), 39, 420**  
**-print action, find command, 303**  
**-r option**  
     sort command, 242-243  
     uname command, 393  
**-s option, tr command, 240**  
**[<] (redirection sign), eval command, 294**  
**;; (semicolon), 148**  
     awk command, 269

**! sign, find command, 303**  
**' (single quote), filtering, 244**  
**-size option, find command, 302**  
**-type option, find command, 300**  
**\$USAGE variable, 202**  
**-v option, 331**  
     grep command, 236-237  
**\$! variable, 198**  
 **\$# variable, 198, 203**  
 **\$\$ variable, 198**  
 **\$\* variable, 198**  
     compared to \$@, 204  
**\$0 variable, 198-199, 404**  
     usage statements, 199-200  
 **\$? variable, 198**  
 **\$@ variable, 198**  
     compared to \$\*, 204  
**variable values, 124**  
**\* wildcard**  
     basename command, 202  
     globbing, 136  
**\* wildcard, globbing, 139**  
**? wildcard, globbing, 138**  
     common errors, 138-139  
**-x option, 332**

## A

**a- option, 39**  
**absolute pathnames, 56**  
     find command, 299  
**abstraction, portability, 397-400**  
**accounts, 14**

**actions (find command)**  
     -exec, 303-304  
     -print, 303  
**adaptability, init script, 372-373**  
**addperson script, 378-379**  
**address books, 373-374**  
     adding people, 377-380  
     deleting people, 380-385  
     interactive mode, 377  
     listing people in, 375-377  
     noninteractive mode, 377  
**ALARM signals, handler function, 321**  
**alias command, 217**  
**aliases, 217, 420**  
     C shells, 16  
     displaying pathnames for, 296  
     functions, comparing, 217-218  
     unaliases, 218  
**ampersand (&), background processes, 106**  
**anchoring, regular expressions, 254-256**  
**and-and operator (&&), 273**  
**appending output to files, 78**  
**arguments, 200**  
     basename command, 201  
         emulating, 202  
     cd command, 59  
     considering one at a time, 409  
     example, 201  
     forwarding to another command, 410

- functions, executing, 215-216
- mkdir command, 63
- passing to commands with xargs command, 304
- shell tracing, 335
- troubleshooting, 203-205
- arithmetic**
  - bc command, 307
  - expr command, 306
- arithmetic expressions, 425**
- arithmetic substitution, 144**
  - common errors, 145-146
  - operators, 144-145
  - precedence, 145
- array variables, 121-127, 427**
- arrays**
  - accessing values, 127-128
  - indices, 126
  - notation, 126
  - support arrays, 427
- assigning variables, awk, 276**
- assignment operators, numeric expressions, 278-279**
- associating files with file descriptors, 82-83**
- AT&T System V UNIX.**  
*See* **System V UNIX**
- awk**
  - invocation syntax, 250
  - operations, 250-251
  - versus sed, 250

- awk command, 268-269**
  - comparison operators, 271-272
  - compound expressions, 273
  - next command, 273-274
- field editing, 269-270
- flow control, 283
  - do statement, 286
  - for statement, 286-288
  - if statement, 284-285
  - while statement, 285
- formatting address book with, 375
- FS, 282
- numeric variables, 277
- pattern-specific actions, 270-271
- STDIN as input, 274-275
- variables, 276
  - numeric expressions, 276-283

## B

- background processes, 106-107**
  - fg command, 110
  - input, requiring, 107-108
  - moving foreground processes to, 108-110
  - preventing termination, 110
  - waiting for, 111
- backquote (`), command substitution, 143**
- backslash (\), 148-149**
  - echo command escape sequences, 155-156
  - newline character, 154
- backslash character (\), tr command, 239**
- basename command, 201-202, 412**
- bash (Bourne Again shell), 17, 25**
  - exporting variables, 130
  - initialization, 25
  - online resources, 34
- Bash shell**
  - integer expressions, 425
  - support arrays, 427
  - wildcards, 430
- bc command, 307-308**
- beeps, sounding a series with sleep command, 297**
- BEGIN pattern, numeric expressions, 279-280**
- Berkeley Software Distribution (BSD), 390**
- bg, 420**
- bg command, 109**
- bit bucket, 405**
- block special files, 94**
- Bourne Again shell (bash), 17**
  - arrays, 125
  - initialization, 25
  - online resources, 34
  - wildcards, 430

**Bourne-type shells, 14-15****braces { }, while statement, 286****break command, 192-193, 420**

nested loops, 194

**BSD (Berkeley Software Distribution), 390****BSD UNIX**

abstraction, getpid function, 399-400

versus System V, 391

**BSD Web site, 390****built-in shell commands, 293****built-in variables, 281-283**

## C

**C shell**

(:) character, 296

starting from Korn Shell, 116

**-c option (wc command), 43****c-based shells tcsh, 16****C-type shells, 14-16****carriage returns, removing from files, 415-416****case statement, 175-176, 420**common errors, 176-177  
patterns, 177**case-sensitivity, options, 38****cat command, 41**

-n option, 42

**cd command, 420**

arguments, 59

changing directories, 58-59

errors, 59

navigating directory trees, 57

**CDPATH variable, 428****changing directories, 58-59****character special files, 94****characters**

counting in file contents, 45

matching, regular expressions, 252-253

sets of, regular expressions, 253-254

**child directories, 54****child processes, 114-115**

permissions, 116

subshells, 115-116

**chmod command, 98**

common errors, 101

octal method, 100-101

symbolic expression, 98-100

**chown command, 101-102**

groups, 102-103

restrictions, 102

**closing file descriptors, 86****command interpreter, 13****command line, options, 200****command substitution, 143-144****commands, 22**

(:) character, 294-296

if statement, 295

while statement, 295-296

(:) colon symbol, 420

(.) period, 420

accessing by shell,

#!/bin/sh, 404

alias, 217

aliases, 420

arguments

forwarding to another

command, 410

passing with xargs

command, 304

awk, 268-269

comparison operators, 271-274

field editing, 269-270

flow control, 283-288

pattern-specific

actions, 270-271

STDIN as input,

274-275

variables, 276-283

basename, 201, 412

emulating, 202

bc, 307-308

bg, 109, 420

break, 192-193, 420

nested loops, 194

case statement, 420

cd, 420

chmod, 98

common errors, 101

octal method,

100-101

symbolic expression,

98-100

chown, 101-102

groups, 102-103

restrictions, 102

- complex, 11
- compound, 12
- compound expressions, 424
- continue statement, 420
- copying files, 46
  - errors, 47
  - interactive mode (cp command), 47
- default behavior, 11
- determining if shell can find, 407-408
- dirname, 412
- do statement, 420
- done statement, 420
- echo, 420
  - conditional execution, 397
  - modifying with single quote, 149
  - output, 72
- esac statement, 420
- eval, 294, 420
- exec, 116-117, 421
- executing in separate shells, 408
- exit, 223
- exit n, 421
- export, 130, 421
- expr, 306-307
- false, 421
- fg, 110, 421
- fi statement, 421
- file, 90
- file descriptors, 82
- file tests, 423
- find, 298-299, 413
  - atime option, 301
  - ctime option, 301
  - exec action, 303-304

- mtime option, 301
- n option, 300
- print action, 303
- size option, 302
- type option, 300
- combining options, 302
- negating options, 303
- starting directory, 299-300
- for statement, 421
- function statement, 421
- getopts, 421
- globbing, 136
- grep, 234
  - line numbers, 237
  - listing filenames, 238
  - searching for words, 235-236
- head, 232-233
- hostname, 394
- if statement, 160-161, 421
  - common errors, 161-163
- integer statement, 421
- integers tests, 424
- jobs, 112, 421
- kill, 114, 421
  - l option, 314
  - signals, 315
- let, 421
- ls
  - character special files, 94
  - d- option, 90
  - file types, 90
  - l- option, 90

- man, 31, 33
- mv, renaming files, 414
- nohup, 110
- option case-sensitivity, 38
- options, 200
  - grouping, 40
- output. *See* output
- overview, 10
- passwd, SUID bit, 97
- pausing with sleep command, 297
- print, with awk, 269
- printf, output, 75-77
- prompt, 10
- ps, 112-113, 366-368
- pwd, 421
- quoting
  - combining, 152
  - echo escape sequences, 155-156
  - embedding spaces, 152-153
  - filenames with special characters, 154-155
  - newline character, 153-154
  - wildcards, 155
  - word boundaries, 152
- read, 81, 421
- readonly, 128, 422
- redirecting to /dev/null, 405-406
- removing directories, 66
- removing files, 49
  - errors, 50
- renaming files, 48
- return, 223, 422
- rsh, 396

- sed, multiple, 262-264
- select, 422
- separators, 12
- set, 327-328, 422
- shift, 208, 422
- simple, 9, 11
- sleep, 297
- sort, 241
  - sorting numbers, 242-243
- STDERR, 406-407
- string tests, 424
- stty, 108
  - addperson script, 380
- tail, 233-234
  - follow option, 234
- test, 163, 422
  - compound expressions, 171-174
  - empty strings, 166-167
  - file tests, 164-165
  - numerical comparisons, 170-171
  - string comparisons, 166-169
  - string equality, 167-168
  - string inequality, 169
- tr, 239
  - character classes, 244-245
  - removing carriage returns, 416
  - removing spaces, 240-241
- trap, 317, 422
  - cleaning up temporary files, 318-319

- type, 296-297, 422
- typeset, 220, 422
- ulimit, 422
- umask, 422
- unalias, 218, 422
- uname, 392-393
  - determining versions with a function, 394-395
  - hardware type, 393-394
- uniq, 241-242
- unset, 129, 218, 422
- until, 422
- using operators conditionally to execute, 408
- viewing file contents, 41-43
  - combining options, 46
  - counting characters, 45
  - counting lines, 44
  - counting words, 45
- wait, 111, 422
- whence, 422
- while, 422
- while loops, 182
- xargs, 304-305

## **comments, 30**

## **common errors, chmod command, 101**

## **comparing aliases and functions, 217-218**

- comparisons operators (awk command), 271-272**
  - compound expressions, 273
  - next command, 273-274

## **complex commands, 11**

## **compound commands, 12**

## **compound expressions**

- comparison operators, 273
- test command, 171-174
- test commands, 424

## **conditional execution**

### **operators, 171**

## **conditional executions,**

### **portability, 396-397**

## **conditional expressions, 423**

## **conditional statements. *See* flow control**

## **continue command, 194**

## **continue statement, 420**

## **copying**

- directories, 63
- directories (multiple), 64
- files, cp command, 46-47

## **counter variables (for statement), 287**

## **counting**

- characters in viewed file information, 45
- lines in viewed file information, 44
- words in viewed file information, 45

## **cp command, 46**

- r option, 63-64
- errors, 47
- interactive mode, 47

## **cpio command, quoting wildcards, 156-157**

## **csh, stack, 224**

## D

**date command, 10**

**debug mode, variable substitution, 143**

**debugging**

- debugging mode, 327
  - invocation activated, 326-327
- enabling, 326
- execution tracing mode, 332
- set command, 327-328
- shell tracing, 332-333
  - debugging hooks, 337-339
  - logical bugs, 335-337
  - syntax bugs, 333-335
- syntax, 328-331
  - verbose mode, 331-332

**debugging hooks, shell tracing, 337-339**

**default actions (signals), 315**

**defining variables, 122**

**deleting**

- directories, 66
- files (rm command), 49-50
- lines, sed, 259-260
- persons from address book, 381

**delimiters, deleting from input file, 239**

**delivering signals, 315**

**delperson script, 381-383**

**dev directory, device files, 94**

**device drivers, block special files, 94**

**device files, 94**

**directories**

- (/), 53
- BSD and System V equivalents, 390-391
- changing, 58-59
- cleaning up files, 414
- copying, 63
- copying multiple, 64
- creating, 62
  - common errors, 63
  - parents, 62
- determining full path-name, 412
- disk space, 352
- find command, -type option, 300
- greping every file in, 413
- home, 24
- info on (ls ld- command), 90
- listing, 60
- listing files in, 38
- moving, 64
- moving (multiple), 65
- permissions, 96-97
  - changing, 98-101
- removing, 66
- run-levels, 362-363
- trees, 53-54
  - filenames, 54
  - navigating, 57
  - pathnames, 55-57

**directory stack**

- adding directories to, 225-226
- listing, 224-225
- manipulating (popd function), 226

**dirname command, 412**

**dirs function, 224-225**

**disk space**

- file ownership, 102
- find command, 304
- function libraries, 351-354
- removing temporary files, 414

**divide and conquer, 222**

**division operation (expr command), 306**

**do statement, 182, 420**

- awk command, flow control, 286

**documents, here documents, 80**

**dollar sign (\$)**

- field operator, 269
- newline character, 153
- quoting with double quotes, 151
- variables, accessing values, 124

**done statement, 420**

**double quotes, 150**

## E

**e- option (ps command), 114**

**echo command, 420**

- conditional execution, 397
- modifying with double quotes, 150
- modifying with single quote, 149

- output, 72
  - formatting, 73-75
  - punctuation marks, 73
  - passing arguments to, 305
- echo\_prompt function, 397**
- editors, stream (sed), 249, 257**
- elif statement, with else statements, 160**
- else if statements, 284**
- else statement, with elif statement, 160**
- embedding in output**
  - formatting, 73-75
  - printf command, 76-77
  - punctuation marks, 73
- END pattern, numeric expressions, 279-280**
- environment variables, 129**
  - exporting, 130
- error messages**
  - background processes, 107
  - output, 72
  - redirecting, 84-85
- error messages (function libraries), 344-345**
- errors. *See also* troubleshooting**
  - cd command, 59
  - cp command, 47
  - functions, 216-217
  - if statement, 161-163
  - ln command, symlinks, 94
  - ls command, 61
  - mkdir command, 63
  - mv command, 65
  - rm command, 50, 67
  - rmdir command, 66
  - variable substitution, 142
- esac statement, 420**
- escape characters, formatting output with, 74-75**
  - echo command, 73
- escape sequence, 149**
  - echo command, 155-156
- etc/shadow file, 97**
- eval command, 294, 420**
- exclamation (!) (find command), 303**
- exec command, 116-117, 421**
- exec system call, 404**
- execution tracing mode, 332**
- exit command, 223, 421**
- export, 421**
- export command, 130**
- exporting**
  - environment variables, 130
  - variables in ksh, bash, and zsh, 130
- expr command, 306-307**
- expressions**
  - arithmetic, 425
  - compound, 171
  - conditional, 423
  - regular expressions, 249-252
    - anchoring, 254-256
    - examples, 252-257
    - matching characters, 252-253
    - meta-characters, 251-252, 256-257
    - sets of characters, 253-254
  - symbolic, 98

## F

- F option, 38**
- false command, 421**
- fg command, 110, 421**
- fi statement, 421**
- field editing (awk command), 269-270**
- fields, 269**
- file command, 90**
- file descriptors, 82**
  - associating files with, 82-83
  - closing, 86
  - redirecting, 85-86
  - STDERR, 82
  - STDIN, 82
  - STDOUT, 82
- file handles. *See* file descriptors, 82**
- file types, determining, 90**
- filename substitution. *See* globbing**
- FILENAME variable, 281**
- filenames, 54**
  - rules for expansion, 430
  - setting to lowercase, 415
  - special characters, 155
- files**
  - appending output to, 78
  - associating with descriptors, 82-83
  - block special, 94
  - changing owners, 101-102
    - restrictions, 102
  - character special, 94
  - copying (cp command), 46-47

- determining full path-name, 412-413
- device, 94
- file command, 90
- filtering
  - grep command, 234-238
  - head command, 232-233
  - tail command, 233-234
- finding with find command, 299
- greping every file in a directory, 413
- hidden, 39
- links, 91-92
- listing, 61
  - visible, 39
- listing in directories, 38
- listing lines, 235
- locating, 413
- manipulating with for loop, 189-190
- most recently accessed, listing, 232
- nohup.out, 111
- ownership, 95
- passwords stored, 97
- permissions
  - changing, 98-101
  - viewing, 96
- printing input lines with awk, 268
- read permissions, 96
- regular, 90
- removing (rm command), 49-50
- removing carriage returns, 415-416

- removing temporary files
  - with matching names, 414
- renaming, 414-415
  - mv command, 48
- SGID permission, 97-98
- shell initialization, 25
- shell scripts, 29
- special, 37
- STDERR, 82
- STDIN, 82
- STDOUT, 82
- SUID permission, 97-98
- symbolic links, 92-93
- symlinks, common
  - errors, 94
- temporary, cleaning up, 318-319
- test command, 164-165
  - compound expressions, 171-174
  - empty strings, 166-167
  - numerical comparisons, 170-171
  - string comparisons, 166-169
  - string equality, 167-168
  - string inequality, 169
- test commands, 423
- viewing contents, 41
  - combining options, 46
  - counting characters, 45
  - counting lines, 44
  - counting words, 45
  - getting information about, 43
  - numbering lines, 42

## filtering text, 249

- awk command, 268-269
  - comparison operators, 271-274
- field editing, 269-270
- flow control, 284-288
- pattern-specific
  - actions, 270-271
- STDIN as input, 274-275
- variables, 276-283

## filtering text files

- grep command, 234
  - line numbers, 237
  - listing filenames, 238
  - searching for words, 235-236
- head command, 232-233
- tail command, 233-234
  - follow option, 234

## find command, 298-299, 413

- atime option, 301
- ctime option, 301
- exec action, 303-304
- mtime option, 301
- n option, 300
- print action, 303
- size option, 302
- type option, 300
- combining options, 302
- negating options, 303
- quoting wildcards, 156-157
- starting directory, 299-300



**finding files, 413****flow control, 159**

- awk command, 283-285
  - flow control, 285-288
- case statement, 175-176
  - common errors, 176-177
  - patterns, 177
- if statement, 160-161
  - common errors, 161-163
- test, 163
  - compound expressions, 171-174
  - empty strings, 166-167
  - file tests, 164-165
  - numerical comparisons, 170-171
  - string comparisons, 166-169
  - string equality, 167-168
  - string inequality, 169

**flow of the script, 159****for loops, 188**

- manipulating sets of files, 189-190

**for statement, 421**

- awk command, flow control, 286-288

**foreground processes, 106**

- fg command, 110
- moving to background, 108-110

**forked child processes, 115****format specifications**

(**printf command**), 76-77

**formatting output**

- echo command, 73-75
- printf command, 76-77

**FreeBSD, 390****FS property (awk command), 282****function chaining, 216**

- recursion, 221-223

**function libraries, 344**

- checking disk space, 351-354
- error messages, 344-345
- retrieving process ID name, 354-355
- retrieving user numeric user ID, 355-356
- user input, 345-351

**function statement, 421****functions, 213-214**

- aliases, comparing, 217-218
- data sharing, 223
- debugging, set command, 328
- debugging hooks, 337
- determining UNIX version, 395
- dirs, 224-225
- echo\_prompt, 397
- getopts, 380
- getOSName, 395
- getPID, 399-400
- getSpaceFree, abstraction, 397-398
- getUID, 356
- including variables definitions in other files, 409
- init script, 368-372
- invoking, 214-215, 217
  - arguments, 215-216
  - errors, 216-217
  - function chaining, 216

- main code, 342
- naming, 344
- popd, 226
  - wrapper, 227-228
- popd\_helper, 226-227
- pushd, 225-226
- SetTimer, 322
- undefined, 218

**G****gawk command, 268****general input/output redirection, 83-84****getopts command, 198, 205-210, 421****getopts function, 380****getOSName function, 395****getPID function, abstraction, 399-400****getSpaceFree function, abstraction, 397-398****getUID function, 356****global scope, 218-220****global variables, 218-220****globally regular expression**

**print. See grep**

**globbing, 136**

- \* wildcard, 136
- ? wildcard, 138
  - common errors, 138-139
- matching sets of characters, 139-141
- matching suffixes and prefixes, 137-138
- \* wildcard, 139

**GNU (gawk command),**  
**268**

**grep command, 234**

-l option, 238

-n option, 237

-v option, 236-237

address book, extracting  
names, 375

greping a string in every  
file, 413

line numbers, 237

listing filenames, 238

regular expressions,  
quoting, 155

searching for words, 235

case independent,

235-236

STDIN, 236

**grouping options, 40**

**groups, changing owners,**  
**102-103**

## H

**hard links, 91-92**

**hardware, determining,**  
**393-394**

**head command, 232-233**

**help features, 31**

UNIX system manuals,  
33

**help. See online help**

**here documents, 80, 429**

**hidden files, 39**

**hierarchies, directories, 53**

**home directories, 24, 57**

**HOME variable, 132, 428**

**hostname command, 394**

**HP-UX**

/bin, /sbin directories,  
391

abstraction, getSpaceFree  
function, 397-398

remote system command,  
396

wc command, counting  
file characters, 45

## I

**i- option (cp command), 47**

**I/O (Input/Output), 428**

**I/O redirection, 429**

**IEEE, awk standard, 268**

**if statement, 160-161, 295,**  
**421**

awk command, flow con-  
trol, 284-285

common errors, 161-163

script portability, 396

syntax checking, 329

**IFS variable, 131, 428**

**ignoring signals, 319-320**

**index numbers, 125**

arrays variables, access-  
ing, 127

**infinite loops**

(:) character, 295

break command, 192-193

nested loops, 194

continue command, 194

**init scripts, 361-366**

adaptability, 372-373

functions, 368-372

platform variations, 363

**initialization, System V**  
**UNIX, 363**

**initialization scripts,**  
**accessing current shell**  
**name, 404**

**initializing shells, 24**

Bourne Again (bash), 25

file contents, 26

setting MANPATH  
variable, 27

setting PATH vari-  
able, 27

Korn (ksh), 25

Z (zsh), 26

**inner loops, 183**

**input, 79**

background processes,  
107-108

pipelines, 81-82

printing lines with awk,  
268

reading, 81

redirecting, 79

general redirection,  
83-84

here documents, 80

while loops, 185-187

xargs command, 304

**Input/Output. See I/O**

**integer arithmetic, 306**

**integer statement, 421**

**integers, test commands,**  
**424**

**interactive mode, address**  
**book, 377**

**interactive shells, 28**

- determining, 405
- starting, 28

**interpreter, 404****interrupt signals, 313****invisible files, 39****invocation activated**

- debugging modes, 326-327

**invocation syntax**

- awk, 250
- sed, 250

**invoking functions,**

- 214-215, 217
- arguments, 215-216
- errors, 216-217
- function chaining, 216

**J****job ID, 107****jobs (kill command), 114****jobs command, 112, 421****K****kernel, 22**

- accessing features with system calls, 404

**kill command, 114, 421**

- l option, 314
- signals, 315

**Korn, ksh shells, 16-17, 25****Korn shell**

- integer expressions, 425
- starting C Shell from, 116
- support arrays, 427
- wildcards, 430

**ksh (Korn shell), 16, 25**

- exporting variables, 130
- initialization, 25

**L****-l option (wc command), 43****let command, 421****libraries, 342-344**

- checking disk space, 351-354
- naming, 343-344
- retrieving process ID name, 354-355
- retrieving user numeric user ID, 355-356
- user input, 345-351

**line numbers (grep command), 237****lines (sed)**

- deleting, 259-260
- printing, 258-259

**links, 91**

- files, hard links, 91-92

**Linux**

- compared to BSD and System V, 391
- gawk command, 268
- wc command, counting file characters, 45

**listing**

- directories, 60
- files, 61
- visible files, 39

**listing signals, 314****listings**

- addperson script, 378-379
- delperson script, 381-383
- function libraries, 461-464
- showperson script, 375-376
- sshd init script, 371-372

**local scope, 218-220****local variables, 129, 218, 220****logging in, 23****logic, checking with shell tracing, 335-337****logins, logging, 297****looping**

- controlling
  - break command, 192-194
  - continue command, 194
- for, 188
  - manipulating sets of files, 189-190
- infinite loops, 192-193
  - continue command, 194
  - nested loops, 194
- select, 190-192
  - changing prompt, 192
- until, 187
- while, 181-182
  - nesting, 183-184
  - until loop, 187-188
  - validating user input, 184-185

**loops (while), input redirection, 185-187**  
**lowercase, setting file-names to, 415**  
**ls command**  
 character special files, 94  
 d- option, 90  
 errors, 61  
 file types, determining, 90  
 l- option, 90  
 listing directories, 60  
 listing files, 61  
 listing visible files, 39  
 options  
     case-sensitivity, 38  
     grouping, 40

## M

**m- option (wc command), 43**  
**mail command, quoting with embedding spaces, 153**  
**mail spools, listing oldest, 233**  
**main loops, 183**  
**man command, 31, 33**  
**man pages, 31-32**  
**manipulating directories, 62**  
     copying, 63  
         multiple, 64  
     creating, 62  
     moving, 64  
     moving multiple, 65  
     removing, 66

**MANPATH variable, 27**  
**manuals (UNIX system), 33**  
**matching**  
     characters, regular  
         expressions, 252-253  
         meta-characters, 256-257  
**memory**  
     commands, 22  
     kernel, 22  
     utilities, 22  
**messages**  
     displaying on STDERR, 406  
     printing to STDOUT, 85  
**meta-characters, 135. *See also* wildcards**  
     double quotes, 150  
     quoting with backslash, 148-149  
     regular expressions  
         escaping, 256  
         matching, 256-257  
         single quotes, 149-150  
**meta-characters (regular expressions), 251-252**  
**mkdir command, 62**  
     -p option, 62  
     common errors, 63  
**modulus function, 306**  
**moving directories, 64**  
**multiple sed commands, 262-264**  
**mv command, 48**  
     errors, 65  
     moving directories, 64  
     renaming files, 414

## N

**-n option (cat command), 42**  
**name value pairs, 122**  
**named pipes, 95**  
**naming**  
     files (mv command), 48  
     libraries, 343-344  
     variables, 122-123  
**negation operator (^), 254**  
**nesting, 183**  
     loops, breaking infinite  
         loops, 194  
     while loops, 183-184  
**NetBSD, 390**  
**newline character, 153**  
**newlines, converting to spaces, 239**  
**newsgroups, shell programming resources, 34**  
**next command, comparison operators, 273-274**  
**nohup command, 110**  
**nohup.out file, 111**  
**noninteractive shells, starting, 28**  
**noninteractive mode, address book, 377**  
**noninteractive shells, determining, 405**  
**notation, strings sets, 251**  
**numbers, sorting, 242**  
     different columns, 243  
**numeric expressions, 276**  
     awk command  
         assignment operators, 278-279  
         built-in variables, 281-283

shell variables, 283

special patterns,

BEGIN, END,

279-280

**numeric tests, 335**

## O

**octal method (chmod command), 100-101**

**online help**

man command, 31, 33

MANPATH variable, 27

**OpenBSD, 390**

**operations**

awk, 250-251

sed, 250-251

**operators**

(!), 171

(!=), test command, 169

(&&), 171, 408

(>>), here documents, 80

(||), 171, 408

arithmetic substitution,

144-145

comparison, 272

Korn/Bash integer

expressions, 425

negation (^), 254

**OPTARG variable, 428**

**OPTIND variable, 428**

**option parsing, 205-206**

getopts command,

206-210

**options, 200**

combining

find command, 302

when viewing file

contents, 46

compared to arguments,

200

debugging options, 326

grouping, 40

negating, find command,

303

ps command, 114

uname command, 392

wc command, 43

**or-or operator (||), 273**

**outer loops, 183**

**output, 71**

redirecting, 77

appending to files, 78

general redirection,

83-84

pipelines, 81-82

to files and screens,

78

redirecting to /dev/null,

405-406

STDERR, 72

redirecting, 84-85

STDOUT, 72

printing messages to,

85

redirecting, 84-85

to terminal, 72

echo command, 72-75

printf command,

75-77

**owners, changing owners**

files, 101-102

groups, 102-103

**ownership, files, 95**

## P-Q

**p- option (mkdir command), 62**

errors, 63

**parent directories, 54**

**parent processes, 114-115**

permissions, 116

subshells, 115-116

**passwd command, SUID bit, 97**

**passwd file, login, 23**

**password files, process**

**permissions, 116**

**passwords**

file stored in, 97

logging in, 23

**PATH variable, 132, 428**

setting, 27

**pathnames, 54**

absolute, 56

determining directory

full pathnames, 412

determining file full

pathnames, 412-413

displaying for a com-

mand, 296

displaying for files, 298

find command, 299

relative, 56-57

types, 55

**pattern matching, 430**

awk command, 270

if statement, 284

**patterns (\*), 307. See also**

**regular expressions**

**percent sign (%), job**

**number prefixes, 109**

**permissions**

- changing with chmod
- command, 98
- common errors, 101
- octal method, 100-101
- symbolic expression, 98-100

- directory, 96-97
- file ownership, 95
- files, viewing, 96
- octal expression values, 100
- processes, 116
- read, 96
- SGID file permission, 97-98
- SUID file permission, 97-98
- world read, 99
- world write, 100
- write, 97

**pid (process ID), 106****pipelines, 81-82**

- sed in, 263-264

**pipes, named, 95****pipng, most recently**

- accessed files, 233

**plus (+) character, shell**

- tracing, 333

**popd function, 226**

- wrapper, 227-228

**popd\_helper function, 226-227****portability**

- abstraction, 397-400
- conditional execution, 396-397
- determining versions with a function, 394-395

- hardware type, 393-394
- improving, 396
- uname command, 392-393
- UNIX versions, 390

**POSIX, awk, 268****pound sign (#), comments, 30****precedence, arithmetic**

- substitution, 145

**prefixes, matching in globbing, 136-137****print command, with awk, 269-270****printf command, 270**

- output, 75
- formatting, 76-77

**printing**

- lines, sed, 258-259
- messages, to STDOUT, 85

**processes**

- background, 106-107
- fg command, 110
- moving foreground processes to, 108-110
- preventing termination, 110
- waiting for, 111
- child, 114-115
- permissions, 116
- subshells, 115-116
- exec command, 116-117
- foreground, 106
- function libraries
- ID names, retrieving, 354-355
- user numeric user ID, retrieving, 355-356

- job numbers, assigning, 110

- jobs command, 112

- kill command, 114

- limit, 106

- parent, 114-115

- permissions, 116

- subshells, 115-116

- ps command, 112-113

- starting, 105

- suspending, 108

**profile file, shell initialization, 27****profiles, shell specific startup with \$0 variable, 404****programmer activated modes, 327****programs**

- executing with SGID bit, 97

- shells, 13, 23

- Bourne Again, 17

- Bourne-type, 15

- C-type, 16

- Korn, 16-17

- prompt, 14

- types of, 14

- Z, 18

- signals, 316

- utilities, 22

**prompts, 10**

- background processes, 107

- changing with select loop, 192

- echo command, 397

- shell, 14

**ps command**, 112-113, 366-368  
**PS1 variable**, 428  
**PS2 variable**, 428  
**public directory**, disk space, 352  
**punctuation marks**, embedding in output, 73  
**pushd function**, 225-226  
**pwd command**, 421  
**PWD variable**, 131, 428

## quoting

combining quoting, 152  
 echo escape sequences, 155-156  
 embedding spaces, 152-153  
 filenames with special characters, 154-155  
 newline character, 153-154  
 wildcards, 155  
     cpio and find commands, 156-157  
     with backslash, 148-149  
     with double quotes, 150  
     with less than sign, 150  
     with single quotes, 149-150  
     word boundaries, 152

**quoting values**, 123

## R

**-r option (cp command)**, 63-64  
     rmrmdir command, 67

**RANDOM variable**, 131, 428  
**read command**, 81, 421  
**read permissions**, 96  
**read-only variables**, 128  
**reading input**, 81  
**readonly command**, 128, 422  
**recursion**, 221-223  
**redirecting**  
     file descriptors, 85-86  
     input, 79  
         general redirection, 83-84  
         here documents, 80  
         while loops, 185-187  
     output, 77  
         appending to files, 78  
         general redirection, 83-84  
         pipelines, 81-82  
         STDOUT, 84-85  
     to files and screens, 78

**redirection signs (eval command)**, 294

**regex**. *See* regular expressions

**regular expression wildcards**, 431

**regular expressions**, 249-252

(.\*), 307  
 anchoring, 254-256  
 examples, 252-257  
 matching characters, 252-253  
 meta-characters, 251-252  
     escaping, 256  
     matching, 256-257  
 quoting, 155  
 sets of characters, 253-254

**regular files**, 90  
**relative pathnames**, 56-57  
     find command, 300  
**remainders**, 306  
**remote commands**, conditional execution, 396  
**removing**  
     directories, 66  
     files (rm command), 49-50  
**renaming files**, 414-415  
     mv command, 48  
**REPLY variable**, 131, 428  
**RESPONSE variable**, 295, 349-351  
**return codes**, 223  
**return command**, 223, 422  
**rm command**, 49  
     errors, 50, 67  
**rmdir command**  
     -r option, 67  
     error, 66  
     removing directories, 66  
     syntax, 66  
**root accounts**, 14  
**root directories**, 53  
**rsh command**, 396  
**run-level S**, 362  
**run-levels**, 361  
     directories, 362-363

## S

**scalar variables**, 121  
**scale (bc command)**, 308  
**scope**, 218-219  
     global scope, 218-220  
     local scope, 218-220

**scripts**

- \$0 shell variable, 199
- comments, 30
- globbing, 136
- init, 361-363
  - adaptability, 372-373
  - functions, 368-372
  - platform variations, 363
- init, 364-366
- operation failures, 204
- option parsing, 205-206
  - getopts command, 206-210
- variable substitution, 142
- while loop, 181-182
  - nesting, 183-184
  - until loop, 187-188
  - validating user input, 184-185

**searching files with wild-cards, 140****SECONDS variable, 131, 428****sed**

- in pipelines, 263-264
- invocation syntax, 250
- operations, 250-251
- versus awk, 250

**sed (stream editor), 249, 257**

- actions, 257
- deleting lines, 259-260
- printing lines, 258-259
- substitutions, 260-262
- syntax, 257
- troubleshooting, 261

**sed command**

- multiple, 262-264
- using shell variables in, 410-411

**select command, 422****select loops, 190-192**

- changing prompt, 192

**semicolon (;), 148**

- awk command, 269
- if then statement, 161

**separators (command), 12****set command, 327-328, 422**

- x option, 332

**Set Group ID. See SGID****Set User ID. See SUID****SetTimer function, 322****SGID file permission, 97-98****shadow file, 97****shell scripts, 29**

- comments, 30
- debugging, 326-331
  - set command, 327-328
  - verbose mode, 331-332
- making executable, 29
- portability
  - abstraction, 397-400
  - conditional execution, 396-397
  - determining versions with a function, 394-395
  - hardware type, 393-394
  - improving, 396
- signals, 314
- temporary files, cleaning up, 317
- UNIX versions, 392

**shell tracing, 332-333**

- debugging, single functions, 328
- debugging hooks, 337-339
- disabling, 328
- logical bugs, 335-337
- set command, 327
- syntax bugs, 333-335

**shell variables, 129, 131, 198, 428****shells, 13, 23**

- accessing name, 404
- arrays, 125
- awk command variables, 283
- Bourn Again, 17
- Bourne-type, 15
- built-in variables, 427
- C-type, 16
- default, 24
- executing commands in separate shells, 408
- find commands, 407-408
- initialization, 24
  - Bourne Again shell (bash), 25
  - Korn shell (ksh), 25
  - Z shell (zsh), 26
- initializing
  - file contents, 26
  - setting MANPATH variable, 27
  - setting PATH variable, 27
- interactive mode, 28
- Korn, 16-17
- login, 23
- making scripts executable, 29



- non-interactive mode,
  - starting, 28
- prompt, 14
- subshells, 115
- types of, 14
- uninitialized, 24
- using operators conditionally to execute, 408
- using variables in sed command, 410-411
- variables, listed, 428
- Z (zsh), 18
- shift command, 208, 422**
- SHLVL variable, 131, 428**
- showperson script, 375-376**
- SIGALARM signals, 320**
  - example timer script, 323
  - setting timer, 322
  - unsetting timer, 322
- SIGHUP signals, 315**
- SIGINT signals, 316**
- SIGKILL signals, 316**
- signals, 313-314**
  - ALARM, handler function, 321
  - cleaning up temporary files, 318-319
  - dealing with, 316
  - default actions, 315
  - delivering, 315
  - ignoring, 319
    - during critical operations, 320
  - kill command, 315
  - list of, 314
  - listing, 314
  - multiple handlers, 318
  - setting actions, 317
- SIGALARM, 320**
  - example timer script, 323
  - setting timer, 322
  - unsetting timer, 322
- SIGHUP, 315**
- SIGINT, 316**
- SIGKILL, 316**
- SIGQUIT, 316**
- SIGTERM, 315**
- SIGQUIT signals, 316**
- SIGTERM signals, 315**
- simple commands, 9, 11**
- single quotes ('), 149-150**
  - filtering, 244
- sleep command, 297**
- Solaris**
  - uname command, 393
  - wc command, counting characters, 45
- sort command, 241**
  - k option, 243
  - n option, 243
  - r option, 243
  - sorting numbers, 242
    - different columns, 243
- spaces**
  - converting tabs/newlines to, 239
  - removing with tr command, 240-241
- special characters**
  - backslash (\), 148
  - filenames, accessing by quoting, 154-155
- special files, 37**
- special variables, 198**
  - \$0, 198-199
    - usage statements, 199-200
- stacks, 224**
  - csh, 224
  - directory
    - adding directories to, 225-226
    - listing, 224-225
    - manipulating (popd function), 226
- standard error. See STDERR**
- standard input. See STDIN**
- standard output. See STDOUT**
- startup**
  - system, 360
  - system scripts, 360
- startup scripts, 360**
- statements**
  - case, 175-176
    - common errors, 176-177
    - patterns, 177
  - if, 160-161, 295
    - common errors, 161-163
  - while, 295-296
- STDERR (standard error), 72, 82**
  - command execution, 406-407
  - displaying messages on, 406
  - redirecting, 84-85

**STDIN (standard input), 82**  
     grep command, 236  
     input for awk command, 274-275  
     xargs command), 304

**STDOUT (standard output), 72, 82**  
     printing messages to, 85  
     redirecting, 84-85

**stream editors (sed), 249, 257**  
     actions, 257  
     deleting lines, 259-260  
     printing lines, 258-259  
     substitutions, 260-262  
     syntax, 257  
     troubleshooting, 261

**string comparisons (test command), 166**

**strings**  
     sets of, notation, 251  
     test commands, 424

**stty command, 108**  
     addperson script, 380

**subdirectories, 54**

**subshells, 115-116**  
     while loop, 186-187

**substitution variables, 426**

**substitutions (sed), 260-262**

**suffixes, matching in globbing, 137**

**SUID, octal expression values, 101**

**SUID file permission, 97-98**

**SunOS (uname command), 393**

**support arrays, 427**

**suspending processes, 108**

**symbolic expressions (chmod command), 98-100**

**symbolic links. *See* symlink files**

**symlinks, 92-93**  
     common errors, 94

**syntax**  
     checking with shell tracing, 333-335  
     debugging, 328-331  
         verbose mode, 331-332  
     invocation, 250  
     rmdir command, 66

**system startup, 360**

**system startup scripts, 360**

**System V (SysV), 390-391**

**System V UNIX, 361**  
     initialization, 363

**SysV (System V), 390-391**

## T

**tabs, converting to spaces, 239**

**tail command, 233-234**  
     -f option, 234  
     follow option, 234

**tar files**  
     arguments, 201  
     listing contents with \$0 variable, 199

**tcsh shell, 16**

**temporary files, cleaning up, 317, 414**  
     trap command, 318-319

**terminal, output to, 72**  
     echo command, 72-75  
     printf command, 75-77

**test command, 163, 422**  
     compound expressions, 171-174  
     empty strings, 166-167  
     file test options, 164  
     file tests, 164-165  
     numerical comparisons, 170-171  
     string comparisons, 166-169  
     string equality, 167-168  
     string inequality, 169

**text, filtering, 249**  
     awk command, 268-288

**text files, filtering**  
     grep command, 234-238  
     head command, 232-233  
     tail command, 233-234

**then statement, troubleshooting, 161**

**timers**  
     ALARM signals, handler function, 321  
     SIGALARM signals, 320  
         example timer script, 323  
         setting timer, 322  
         unsetting timer, 322

**tr command, 239**  
     -s option, 240  
     character classes, 244-245

- removing carriage returns, 416
- removing spaces, 240-241
- versions of, 240
- tracing, 332-333**
  - debugging hooks, 337-339
  - disabling, 328
  - logical bugs, 335-337
  - set command, 327
  - syntax bugs, 333-335
- transliterating words, tr command, 239**
- trap command, 317, 422**
  - cleaning up temporary files, 318-319
- trees (directory), 54**
  - filenames, 54
  - navigating
    - changing directories, 58-59
    - home directories, 57
  - pathnames, 55
    - absolute, 56
    - relative, 56-57
- troubleshooting**
  - address book, 377
  - arguments, 203-205
  - background processes, 107
  - sed, 261
- type command, 296-297, 422**
- typeset command, 220, 422**

## U

- UID variable, 131, 428**
- ulimit command, 422**
- umask command, 422**
- unalias command, 218, 422**
- unaliases, 218**
- uname command, 392-393**
  - m option, 393
  - r option, 393
  - determining versions
    - with a function, 394-395
  - hardware type, 393-394
  - SunOS, 393
- undefined functions, 218**
- uniq command, 241-242**
- UNIX**
  - commands, 10
    - complex, 11
    - compound, 12
    - default behavior, 11
    - separators, 12
    - simple, 11
  - directories, 53
    - cd command, 57
    - changing, 58-59
    - copying, 63
    - copying multiple, 64
    - creating, 62
    - creating parents, 62
    - filenames, 54
    - listing, 60
    - manipulating, 62
    - moving, 64
    - moving multiple, 65
    - pathnames, 55-57
    - removing, 66
    - trees, 54
- kernel, 22
- man pages, 31
  - sections, 32
- online resources, 34
- shells, 13
  - Bourne Again, 17
  - Bourne-type shells, 15
  - C-type shells, 16
  - default, 24
  - Korn shells, 16-17
  - prompt, 14
  - types of, 14
  - Z (zsh), 18
- system manuals, 33
- unset command, 129, 218, 422**
- unsettling variables, 129**
- until command, 422**
- until loop, 187-188**
- usage statements, \$0 variable, 199-200**
- user IDs, retrieving, 355**
- user input**
  - function libraries, 345-351
  - validating with while loop, 184
- user-defined variables, 426**
- usernames, 23**
- users. *See also* input**
  - logging in, 23
  - logging logins with sleep command, 297
  - process ID, 113
  - profiles, shell specific startup with \$0 variable, 404
  - shells, interactive mode, 28

**utilities, 22**  
**uuencode, 206**  
**uuencode command,**  
     **option parsing, 208**

## V

**validating user input,**  
     **while loops, 184-185**  
**validity (variables), 122**  
**values**  
     accessing (array variables), 127  
     quoting, 123  
     variables, 123  
**variable substitution, 135, 141**  
     default values  
         assigning, 142  
         substituting, 141  
     option parsing, 208  
     variable errors, 142  
**variables**  
     \$!, 198  
     \$, 198  
     \$\$, 198  
     \$\*, 198  
     \$, 198-199, 404  
         usage statements,  
         199-200  
     \$?, 198  
     \$@, 198  
     \$n, 198  
     \$USAGE, 202  
     arguments, troubleshooting,  
         203-205  
     array, 121, 125-127, 427

arrays, 124  
     accessing values,  
         127-128  
     awk command, 276  
         numeric expressions,  
         276-283  
     built-in shell, 427  
     checking for values, 411  
     considering arguments  
         one at a time, 409  
     defining, 122  
     environment, 129  
         exporting, 130  
     exporting, 130  
     FILENAME, 281  
     global, 218-220  
     including functions and  
         definitions in other  
         files, 409  
     local, 129, 218  
     naming, 122-123  
     read-only, 128  
     RESPONSE, 295,  
         349-351  
     scalar, 121  
     sed command, using  
         shell variable values in,  
         410-411  
     shell, 129, 131, 428  
     special, 198  
     substitution, 426  
     unsetting, 129  
     user-defined, 426  
     validating user input, 185  
     validity, 122  
     values, 123  
         accessing, 123  
     YESNO, 345-349

**verbose mode, 331-332**

### versions

awk command, 268  
     determining, 390  
     determining versions  
         with a function,  
         394-395  
     tr command, 240  
     uname command,  
         392-393  
         hardware type,  
         393-394

### viewing

file contents, 41  
     combining options, 46  
     counting characters,  
         45  
     counting lines, 44  
     counting words, 45  
     getting information  
         about, 43  
         numbering lines, 42  
     file permissions, 96

**visible files, listing, 39**

## W-Y

**w- option (wc command), 43**

**wait command, 111, 422**

**wc command, 43**

### Web sites

BSD, 390  
     online help resources, 31  
     UNIX resources, 34

**whence command, 422**

**while command, 422**

**while loop, 181-182**

nesting, 183-184

until loop, 187-188

validating user input,  
184-185

**while loops, input redirection, 185-187**

**while statement, 295-296**

awk command, flow control,  
285

**who command, 10**

default behavior, 11

**wildcards, 430. *See also***

**meta-characters**

expr command, 307

find command, 300

globbing, 136

\* wildcard, 136, 139

? wildcard, 138-139

matching sets of characters,  
139-141

quoting, 155

with cpio and find,  
156-157

regular expression, 431

**words**

count occurrences,  
241-242

counting, 238

counting in file contents,  
45

transliterating, 239

**world read permission, 99**

**world write permission,  
100**

**wrapper scripts, forwarding arguments onto other commands, 410**

**write permission, 97**

**xargs command, 304-305**

**YESNO variable, 345-349**

## Z

**Z shell (zsh), 18**

initialization, 26

online resources, 34

**zero completion code, 294**

**zsh (Z shell), 18, 26**

exporting variables, 130

initialization, 26

online resources, 34