

# Index

## Symbols and Numerics

### & (ampersand)

for redirecting standard output and standard error, 260–261

for running in the background, 37–38  
as sed metacharacter, 218–219

### < > (angle brackets)

in awk comparison operators, 250  
for end-of-file marker (<<), 177  
for redirecting and appending output (>), 261–262  
for redirecting input (<), 259  
for redirecting output (>), 106–108, 258–259  
for redirecting standard error (>), 260  
for truncating files when redirecting, 262–263

### \* (asterisk). See \* (star)

### \ (backslash) for continuing lines, 81–82

### ` (backtick)

for command substitution, 287–289  
nesting, 292  
reading files into variables, 292  
setting variables from commands, 289–290  
using parentheses instead of, 289

### ! (bang). See ! (exclamation mark)

### [ ] (brackets)

for regular expressions, 4  
as test command shorthand ([]), 123–125

### ^ (caret) for beginning of line in sed, 212, 213

### :

as Mac directory delimiter, 175  
as sed string separator, 205–206  
vi commands using, 58

### { } (curly braces)

for referencing variables, 86  
for sed commands, 224

### - (dash) preceding command-line options, 23, 26

### \$ (dollar sign)

for end of line in sed, 213  
as environment variable constructor, 136  
PID-related variables (\$\$ and \$!), 278  
for repeating parts of commands (!\$), 30–31  
for return codes variable (\$?), 295–296, 309  
in shell prompt, 2, 16, 20–21  
as variable name prefix, 22, 72, 85–86, 88

### . (dot)

for character matching in sed, 213  
as prefix for hidden files, 91, 153  
source command compared to, 152

### ! (exclamation mark)

for address negation with sed, 201–202  
in awk comparison operators (!=), 250  
with command history, 32–33  
in csh special commands, 5  
in magic line (!#), 161–163  
for negation with test command, 121, 122–123  
for repeating commands, 27–30, 32–33  
for repeating parts of commands (!\$), 30–31  
for variable holding PID of last command (\$!), 278

### / (forward slash). See / (slash)

### > (greater-than sign). See < > (angle brackets)

### # (hash mark)

for comment lines in scripts, 78–80  
in magic line (!#), 161–163  
for sed comments, 209–210, 224  
in shell prompt, 20

### < (less-than sign). See < > (angle brackets)

### ( ) (parentheses)

for command substitution, 289  
double parentheses syntax with for loop, 96–97

### % (percent sign)

in printf format control characters, 243–244  
in shell prompt, 20

### . (period). See . (dot)

### | (pipe character). See piping commands

### # (pound sign). See # (hash mark)

### ? (question mark)

for return codes variable (\$?), 295–296, 309  
as wildcard, 37

### “ (quotation marks) for variable values including spaces, 72

### ; (semicolon) for separating commands, 94

### # (sharp sign). See # (hash mark)

### / (slash)

for command options in MS-DOS, 23  
for searching in vi, 58  
as sed string separator, 203, 205, 206  
surrounding sed regular expression addresses, 212

## \* (star)

---

### \* (star)

as case statement catch-all value, 127–129  
for character matching in sed, 213  
hidden files and, 91  
as wildcard, 35–36

~ (tilde) for user's home directory, 20

\_ (underscore) beginning variable names, 71

zero, as success return code, 100, 102–103

## A

**AbiWord word processor, 410–411**

**active select and paste model, 16–17**

**addresses in sed**

advanced addressing, 211–217  
combining line addresses with regular expressions, 217  
negating, 201–202  
ranges, 200–201  
regular expression address ranges, 216  
regular expression addresses, 212–214  
stepping, 202  
substitution, 206–207

**administrative scripting. See also MRTG (Multi Router Traffic Grapher)**

for automating daily work, 392  
checking disk space, 379–382  
checking USB port versions, 391–392  
for cleaning up data, 387–392  
for complicated commands, 376–379  
exercises and answers, 392–393, 458–459  
guidelines, 376  
listing active processes, 382–384  
monitoring HTTP data, 377–378  
playing favorite songs, 387  
for removing minor annoyances, 385–387  
running a command in a directory, 386–387  
starting USB networking, 376–377  
for troubleshooting systems, 379–385  
uses for, 375, 392  
verifying processes are running, 384–385  
viewing Linux USB port details, 387–391

**Advanced Bash-Scripting Guide, 436**

**alarm clock script, 411–413**

**Almquist, Kenneth (shell creator), 7**

**Alt key, as emacs Meta key, 48**

**ampersand (&)**

for redirecting standard output and standard error, 260–261  
for running in the background, 37–38  
as sed metacharacter, 218–219

**AND operation for binary tests, 121–123**

**angle brackets (< >)**

in awk comparison operators, 250  
for end-of-file marker (<<), 177  
for redirecting and appending output (>), 261–262

for redirecting input (<), 259  
for redirecting output (>), 106–108, 258–259  
for redirecting standard error (>), 260  
for truncating files when redirecting, 262–263

**Apache Axis TCP monitor program, 378**

**append command (sed), 210–211**

**Apple. See AppleScript (Mac OS X); Mac OS X The AppleScript Language Guide (Apple publication), 417**

**AppleScript (Mac OS X)**

alarm clock script using, 411–413  
AppleScript Studio environment, 419  
dictionaries, 415–417  
further information, 417  
going from shell to, 418–420  
going to shell from, 420–425  
Open Scripting Architecture extension (OSAX), 415  
OSA and, 414  
targeting applications, 415  
verbosity of, 414

**applications. See also programs; specific applications**

monitoring remotely with MRTG, 366–372  
“native” Mac OS X, 413–414  
targeting (AppleScript), 415

**archives, combining files into, 168–169**

**arguments**

command-line, 23, 156–160  
debugging options, 327–333  
with functions, 308–309  
for ooffice command, 396–397

**arithmetic functions (awk), 251–252**

**ash shell, 7**

**asterisk. See star (\*)**

**Audrey Internet appliances, running shells on, 19**

**awk command, 485–486**

**awk language**

arithmetic functions, 251–252  
awk command, 485–486  
basic functionality, 234  
built-in variables, 245–248  
checking version of, 231–232  
comparison operators, 250–251  
control statements, 248–254  
creating self-contained programs, 236–237  
exercises and answers, 256, 450  
field separators, 240–241, 246  
for loops, 253–254  
FS variable, 245–246  
functions, 254–255  
further information, 255  
gawk (GNU awk), 230–231  
if statements, 249–250  
installing gawk, 232–233  
invoking, 234–237  
output redirection, 252–253  
overview, 229–230

print command, 237–241  
 printf command, 241–244  
 running scripts with `-f` flag, 236  
 sed compared to, 234  
 simple command-line program, 235  
 sprintf command, 244  
 user-defined variables, 245  
 user-defined versus built-in variables, 244  
 versions, 230  
 while loops, 253

**Axis TCP monitor program (Apache), 378**

## B

### background

running commands in, 37–38, 285–286  
 running MRTG in, 352–353

**backslash (\) for continuing lines, 81–82**

### backtick (`)

for command substitution, 287–289  
 nesting, 292  
 reading files into variables, 292  
 setting variables from commands, 289–290  
 using parentheses instead of, 289

**backup scripts, 90, 93**

**bang. See exclamation mark (!)**

**Bare Bones Software's BBEEdit text editor, 62–63**

**basename command, 471**

**bash (Bourne Again shell)**

Advanced Bash-Scripting Guide, 436  
 downloading, 8  
 file-name completion feature, 34–35  
 looping in, 96–97  
 overview, 6  
 --posix command-line option, 8  
 prompt, 21  
 specifying multiple sed commands using, 208  
 startup by, 154–155

**.bash\_login file, 154, 155**

**.bash\_profile file, 154, 155**

**BBEEdit text editor (Bare Bones Software), 62–63**

**bc command**

checking disk space, 382  
 overview, 491–492  
 running from scripts, 294–295  
 running interactively, 293–294

**binary tests, 121–123**

**bit bucket, redirecting output to, 106–108, 263**

**bookmarks, konsole support for, 16**

**bootstrap installation of sed, 192**

**Bourne Again shell. See bash**

**Bourne shell (sh)**

compatibility with Korn shell, 5  
 incompatibility with C shell, 5  
 limitations for editing commands, 27  
 overview, 4

reading command-line arguments, 156–159  
 startup by, 153

**Bourne, Steven (shell creator), 4, 436**

**brackets ([ ])**

for regular expressions, 4  
 as test command shorthand ([]), 123–125

**breaksw statement, 129**

**BuddySpace IM client, script for starting, 66–67, 386–387**

**buffers in emacs, 54**

**built-in variables. See also variables**

FS (awk), 245–246  
 NR (awk), 247–248  
 table describing, 248  
 user-defined variables versus, 244

## C

**C shell (csh)**

command-history feature, 31–32  
 file-name completion feature, 34  
 gathering keyboard input, 130–131  
 incompatibility with Bourne shell, 5  
 looping in, 98–99  
 overview, 4–5  
 printenv command, 146–147  
 reading command-line arguments, 160  
 rebuilding list of executables, 161  
 set command, 144–145  
 setenv command, 145–146  
 setting environment variables, 152  
 startup by, 154  
 switch statement, 129–131  
 variables and, 73

**calculator. See bc command**

**calling**

exec command from scripts, 286–287  
 functions in shells, 303  
 functions, incorrect invocation errors, 307  
 functions within other functions, 305–306  
 programs, errors in, 322–323

**caret (^) for beginning of line in sed, 212, 213**

**case statements**

C shell switch statement, 129–131  
 esac ending, 126  
 example for choosing favorites, 126–127  
 handling unexpected input, 127–129  
 syntax, 125–126  
 uses for, 127

**cat command**

outputting here files using, 178–179  
 overview, 471–472  
 viewing Linux USB port details, 387–389  
 viewing /proc file contents, 281–282

**CDE (Common Desktop Environment), 7, 18, 64**

**change command (sed), 211**

**character class keywords (sed), 215–216**

## checking environment variables, 147–150

### child shells. *See* subshells

#### chmod command

- adding execute permissions, 161, 163
- changing permissions, 170–171
- overview, 472–474

#### chown command, 474

#### chpass command, 9, 10

#### chsh command, 9, 10

#### cmd.exe shell (Windows), 18. *See also* MS-DOS

#### Coding Monkeys' SubEthaEdit text editor, 62

#### colon (:)

- as Mac directory delimiter, 175
- as sed string separator, 205–206
- vi commands using, 58

#### color editor (Eclipse), 61

#### Color settings (Mac OS X Terminal), 429–430

#### combining command-line options, 26

#### command history, 31–33

#### command pipelines. *See* piping commands

#### command substitution

- backtick for, 287–289
- for math, 290–292
- nesting backticks, 292
- parentheses for, 289
- reading files into variables, 292
- for setting variables from commands, 289–290
- tracing with, 331–332

#### command.com shell (Windows), 18. *See also* MS-DOS

#### command-line arguments

- debugging options, 327–333
- defined, 23
- exercises and answers, 165, 445
- listing, 157–159
- for `office` command, 396–397
- reading with Bourne shell, 156–159
- reading with C shell, 160
- reading with Korn shell, 156
- reading with T C shell, 160
- using, 159

#### command-line editing

- cycling through previous commands, 31
- need for, 27
- repeating parts of previous commands, 30–31
- repeating previous commands, 27–30
- shell differences and, 27
- shells and support for, 4, 5, 6
- text editor for, 33–34
- viewing the command history, 31–33

#### command-line options

- combining, 26
- defined, 23
- inconsistency in, 26–27
- on Mac OS X, 25–26
- `man` command for information about, 24–25
- on Unix and Linux, 23–25

#### commands. *See also* specific commands and programs

- arguments, defined, 23
- basic format, 21
- building up with variables, 74–76
- command-line options, 23–27
- complex, storing in scripts, 66–67
- defined, 275
- determining type of, 467–468
- editing, 27–35
- entering, 20–27
- executing from the prompt, 2
- for files and directories, 471–485
- interactive, in shell scripts, 41–42
- math using command substitution, 290–292
- mixing with text output, 69–70
- mixing with variables, 73–74
- for operating system navigation, 461–470
- path for, 2, 22
- reading command-line arguments, 156–160
- for resolving expressions, 491–492
- return codes, 102–106, 309
- running from variables, 74–75
- running in directories, 386–387
- running in subshells, 286
- running in the background, 37–38, 285–286
- running in the foreground, 285
- running with `exec`, 286–287
- sed commands, 224–226
- separating with semicolon, 94
- setting variables from, 289–290
- storing in variables, 75–76
- for text manipulation, 485–487
- for transforming data, 487–491
- viewing history of, 31–33
- wildcards, 35–37

#### comments

- for avoiding errors, 334
- defined, 78
- for documenting scripts, 376
- entering in scripts, 78–80
- need for, 78
- sed comment command, 209–210
- tips for, 80

#### Common Desktop Environment (CDE), 7, 18, 64

#### comparison operators (awk), 250–251

#### compatibility issues, 5, 6, 7, 9

#### compilers, setting variable for, 75

#### compiling sed, 193

#### compress command, 168

#### compressing files, 168, 169

#### concatenating. *See* cat command

#### configuring

- MRTG, 345–350, 369–372
- sed text-processing tool, 193

#### continuing lines in scripts, 81–82

#### Control key for emacs, 48, 49

**control statements (awk)**

- arithmetic functions, 251–252
- comparison operators, 250–251
- defined, 248
- for loops, 253–254
- if statements, 249–250
- output redirection, 252–253
- while loops, 253

**controlling how scripts run**

- bash shell loops, 96–97
- C shell loops, 98–99
- case statement for complex decisions, 125–131
- checking conditions with if, 100–113
- exercises and answers, 134, 442–444
- looping for fixed number of iterations, 93–96
- looping over files, 90–93
- looping overview, 89–90
- looping until condition is true, 132–133
- looping while condition is true, 131–132
- nested loops, 99–100
- nesting if statements, 113–114
- referencing variables, 85–89
- test command for, 114–125

**controlling processes. See processes****copying**

- active select and paste model for, 16–17
- emacs commands for, 52–53
- files with cp, 474–475
- vi commands for, 58

**cp command, 474–475****CPU, graphing usage with MRTG, 358–360****Cream for Vim graphical front-end, 59****cron program**

- alarm clock script using, 411–413
- configuring for MRTG, 352–353

**crontab file, 412, 413****cross-platform text editors, 59–61****csh. See C shell****.cshrc file, 154, 155****curly braces ( { } )**

- for referencing variables, 86
- for sed commands, 224

**customizing your account**

- bash startup, 154–155
- Bourne shell startup, 153
- C shell startup, 154
- Korn shell startup, 153–154
- system files versus personal startup files, 152–153
- T C shell startup, 154

**cut command, 487–488****Cute text editor, 64****Cygwin environment**

- default shell, 7, 8, 18
- installing on Windows, 18
- listing the environment, 140–141
- shells included with, 18

**D****dash (-) preceding command-line options, 23, 26****data transformation commands, 487–491****dd command, 26****de-assigning variables, 76****debugging mode**

- combining -n and -v options, 329
- disabling the shell with -n option, 328
- displaying commands with -v option, 328–329
- tracing execution with -x option, 329–333

**debugging scripts. See also error messages**

- asking for help, 327
- avoiding errors, 333–335
- basic steps, 317–318
- breaking the script into pieces, 326–327
- checking syntax with -n option, 327, 329
- debuggers for, 327
- deciphering error messages, 318–323
- divide and conquer technique, 326
- errors in calling programs, 322–323
- exercises and answers, 336–337, 455–457
- finding missing syntax, 319–321
- finding syntax errors, 321–323
- looking for hidden assumptions, 325
- looking for obvious mistakes, 324
- looking for weird things, 324–325
- missing spaces, 321–322
- need for, 317
- techniques for, 323–327
- tracing the execution, 327, 329–333
- tracking errors to right location, 320–321
- verbose mode for, 328–329
- working backward, 323–324

**declaring functions, 303–305****default shell, 6, 8, 9–12****deleting**

- directories, 483
- emacs commands for, 53
- files, 482–483
- sed command for, 195–196

**desktop-related scripting**

- for AbiWord word processor, 410–411
- exercises and answers, 436–437, 459
- on Mac OS X, 411–432
- for multimedia, 432–435
- for NEdit text editor, 411
- for OpenOffice.org suite, 396–410
- tips for scripting applications, 435–436
- uses for, 395

**/dev directory, 106****/dev/null device, redirecting output to, 106–108, 263****df command**

- checking disk space, 380–382
- diskusage function for, 300–304

### **df command (continued)**

- graphing disk usage, 361
- overview, 475–476
- repeating, 28–29

### **dictionaries (AppleScript), 415–417**

“die a flaming death” alert, 147, 165, 445

### **directories**

- checking disk space used by, 476–477
- commands for, 471–485
- creating, 481–482
- deleting, 483
- Mac OS X versus Unix standard, 172–173

### **disk images, mobile file systems and, 174**

### **disk usage**

- checking disk space, 379–382, 475–477
- graphing with MRTG, 361–363
- by MRTG, 342

### **diskusage function, 300–304**

### **Display settings (Mac OS X Terminal), 429**

.dmg files (Mac OS X), 174

### **do shell script command (AppleScript), 420–425**

### **dollar sign (\$)**

- for end of line in sed, 213
- as environment variable constructor, 136
- PID-related variables (\$\$ and \$!), 278
- for repeating parts of commands (!\$), 30–31
- for return codes variable (\$?), 295–296, 309
- in shell prompt, 2, 16, 20–21
- as variable name prefix, 22, 72, 85–86, 88

### **DOS. See MS-DOS**

### **dot (.)**

- for character matching in sed, 213
- as prefix for hidden files, 91, 153
- source command compared to, 152

### **down arrow for cycling through commands, 31**

### **downloading, resources for. See Internet resources**

### **downloading web pages with wget command, 366–368**

### **du command, 476–477**

## **E**

### **echo command**

- determining which shell is running, 21–22
- listing command-line arguments, 157–159
- mixing commands with text output, 69–70
- mixing commands with variables, 73–74
- without newlines (-n option), 68–69
- overview, 486–487
- for simple text output, 67–68
- using interactively with read, 76–78
- variables with, 21–22, 71–74

### **Eclipse IDE, 61**

### **editing. See also text editors**

- command-line editing, 27–35
- emacs commands for, 50

- sed commands, 195–196, 207–209
- shell commands using text editors, 33–34

### **elif construct, 111–113**

### **emacs command, 49**

### **emacs text editor**

- buffer-related commands, 54
- buffers, 54
- conflicts with other program usages, 52–53
- Control and Meta keys for, 48, 49
- copy and paste commands, 52–53
- as cross-platform editor, 59
- downloading, 48
- editing commands, 50
- editing shell commands using, 33–34
- frames, 54
- graphical version, 49–50
- help commands, 50–51
- help online, 49
- interactive shell within, 54
- kill buffer, 53
- loading files on startup, 50
- long names for commands, 50
- minibuffer area, 51
- navigation commands, 51–52
- running long commands, 50
- searching with, 52
- shell commands for, 33
- starting, 49
- text deletion commands, 53
- usefulness of, 48

### **Emulation settings (Mac OS X Terminal), 427, 428**

### **enabling command history, 31**

### **end-of-file marker for here files, 177**

### **env command, 142, 146**

### **environment variables**

- C shell and, 144–147
- checking, 147–150
- commonly available variables, 136–137
- defined, 135, 141
- documentation for program use of, 139
- guidelines for using, 142
- listing on Linux, 137–138
- listing on Mac OS X, 138–139
- listing on Windows XP, 140–141
- listing only environment variables, 142–144
- printing value of, 464–465
- reading the environment, 136–150
- reading values into current variables, 152
- setting, 150–152
- shell variables compared to, 22, 135–136
- uppercase names for, 138

### **error messages. See also debugging scripts**

- deciphering, 318–323
- function declaration errors, 302
- function formatting errors, 302

- incorrect function invocation, 307
  - informative, creating, 334–335
  - script execution stopped by, 42
  - from sed, 199, 201
  - standard output for, 107–108
  - esac **ending case statements, 126**
  - Esc key, as emacs Meta key, 48**
  - `/etc/passwd` **file**
    - creating pipelines for, 266–270
    - invoking sed with data from, 194–195
  - exclamation mark (!)**
    - for address negation with sed, 201–202
    - in awk comparison operators (!=), 250
    - with command history, 32–33
    - in csh special commands, 5
    - in magic line (!#), 161–163
    - for negation with test command, 121, 122–123
    - for repeating commands, 27–30, 32–33
    - for repeating parts of commands (!\$), 30–31
    - for variable holding PID of last command (\$!), 278
  - exec **command, 286–287**
  - executable files**
    - defined, 160
    - magic line (!#) for, 161–163
    - making scripts executable, 163–164
    - marking files as, 160–161
    - rebuilding list of, 161
  - executing. See loading; running**
  - execution trace mode. See tracing script execution**
  - exit **command**
    - if statements testing return codes, 102–106
    - overview, 461
    - shell exit codes, 309
  - export **command, 150–152**
  - expr **command**
    - checking disk space, 382
    - evaluating math expressions, 290–291
    - overview, 492
    - using variables with, 291
  - extracting files from archives, 168**
- ## F
- false **command, 103–104**
  - fi **ending if statements, 100, 102**
  - field separators (awk), 240–241, 246**
  - file **command, 462**
  - file descriptors, 258**
  - file systems**
    - Mac OS X versus Unix standard, 175–177
    - mobile, Mac OS X and, 173–174
    - `/proc`, 279–284
  - files**
    - changing ownership of, 474
    - checking disk space used by, 476–477
    - checking permissions, 161
    - classifying type with file, 462
    - combining into archives, 168–169
    - commands for, 471–485
    - compressing and uncompressing, 168–169
    - copying, 474–475
    - creating from OpenOffice.org Basic scripts, 407–410
    - deleting, 482–483
    - extracting from archives, 168
    - file-name completion feature, 34–35
    - finding files, 477–478
    - finding strings in, 490
    - for functions, 306–307
    - here files, 177–186
    - listing for current directory, 2
    - loading on awk startup, 234–235
    - loading on emacs startup, 50
    - loading on vi startup, 55–56
    - locking down permissions, 171–172
    - login files, 153–155
    - logout files, 155
    - looping over, 90–93
    - Mac OS X files, 172–177
    - magic line (!#) for, 161–163
    - marking executable, 160–161
    - modes, 169–171
    - moving, 482
    - MRTG configuration file, 370–372
    - personal startup files, 153–155
    - reading into variables, 292
    - redirecting output to, 106–108
    - shells versus file managers, 13
    - sorting line by line, 266, 489–490
    - standard input, output, and error files, 107–108
    - system files, 152–153
    - test command for, 120–121, 171–172
    - updating with touch, 484–485
    - vi file-related commands, 57–58
    - viewing first few lines, 479
    - viewing last few lines, 483–484
  - find **command, 477–478**
  - finding**
    - with emacs text editor, 52
    - files, 477–478
    - installed shells, 108–111
    - missing syntax, 319–321
    - strings in files, 490
    - syntax errors, 321–323
    - with vi text editor, 58
  - FireWire, mobile file systems and, 173–174**
  - for **loop**
    - in awk language, 253–254
    - in bash shell, 96–97
    - basic syntax, 90
    - checking for shells, 110–111
    - in diskcheck script, 381
    - double parentheses syntax, 96–97

## for loop (continued)

---

### **for loop (continued)**

- looping for fixed number of iterations, 93–96
- looping over files, 90–93
- nested loops, 99–100
- in simple backup script, 93
- sleep command for pausing iterations, 95–96
- tracing, 332–333

### **foreach loop (C shell), 98–99**

### **foreground, running commands in, 285**

### **format control characters (awk printf command), 242–244**

### **forward slash. See slash (/)**

### **frames in emacs, 54**

### **FS variable (awk), 245–246**

### **ftp command, driving with here files, 183–184**

### **functions (awk), 254–255**

### **functions (shell)**

- arguments with, 308–309
- calling within other functions, 305–306
- declaration errors, 302
- declaring before use, 303–305
- diskusage example, 300–304
- exercises and answers, 316, 453–455
- formatting errors, 302
- function files for, 306–307
- incorrect invocation errors, 307
- as named code blocks, 299, 300
- naming blocks of code, 300–302
- recursion, 314–315
- return codes with, 309–311
- syntax for defining, 299–300
- undeclaring, 307–308
- uses for, 299
- using (calling), 303
- variable scope and, 311–314

## G

### **gawk (GNU awk) language. See also awk language**

- checking version of, 231–232
- further information, 255
- installing, 232–233
- obtaining, 233
- overview, 230–231

### **gedit text editor (GNOME), 63–64**

### **Gisin, Eric (shell creator), 6**

### **Glimmer text editor, 64**

### **globs or globbing. See wildcards**

### **GNOME desktop**

- gedit text editor, 63–64
- running the shell window, 15–16

### **gnome-terminal window, 15–16**

### **GNU Linux**

- gawk (GNU awk) language, 230–231, 255
- sed commands, 225–226
- sed information, 222
- sed version, 191

### **graphical text editors**

- Cream for Vim, 59
- cross-platform editors, 59–61
- emacs graphical version, 49–50
- for Linux, 63–64
- for Mac OS X, 61–63
- for Microsoft Windows, 65
- for Unix, 64–65

### **graphical user interfaces. See also specific interfaces and operating systems**

- file managers versus shells, 13
- power increased by shells, 1, 3
- shell support by, 1, 7
- shell use reduced by, 13

### **graphing. See MRTG (Multi Router Traffic Grapher)**

### **greater-than sign (>). See angle brackets (< >)**

### **grep command, 478–479**

### **gunzip program, 169**

### **gzip program, 168**

## H

### **hash mark (#)**

- for comment lines in scripts, 78–80
- in magic line (!#), 161–163
- for sed comments, 209–210, 224
- in shell prompt, 20

### **head command, 479**

### **help. See also Internet resources; man command**

- asking for, when debugging, 327
- emacs commands for, 50–51
- for OpenOffice.org Basic, 404

### **here files**

- basic syntax, 177
- changing input with variables, 181–182
- defined, 177
- displaying messages with, 178–179
- driving interactive programs with, 183–186
- end-of-file marker, 177
- redirecting input versus, 179, 180
- turning off variable substitution, 186
- variables in, 181–186

### **HFS+ file system (Mac OS X), 175–177**

### **hidden files, wildcards and, 91**

### **history of commands used, 31–33**

### **hold space (sed), 220–222**

### **HTTP data, monitoring, 377–378**

## I

### **IAC (Interapplication Communication), 414**

### **if statements**

- in awk language, 249–250
- basic syntax, 100
- checking for shells, 108–111
- in diskcheck script, 381–382
- elif construct with, 111–113

else block with, 101–102, 249–250  
 fi ending, 100, 102  
 nesting, 111, 112, 113–114  
 redirecting output with, 106–108  
 test command with, 115–123  
 testing make command, 104–106  
 for true and false return codes, 102–106  
 uses for, 101

**ifconfig command**  
 directory for, 377  
 for starting USB networking, 71, 376

**IM (instant messaging), 66–67, 386–387**

**incremental search, 52**

**indenting**  
 in Python scripting language, 45–46  
 for readability, 333–334, 376

**initialization files for shells, 153–155**

**input redirection. See redirecting input**

**insert command (sed), 210–211**

**inserting**  
 awk print command for, 238–239  
 vi commands for, 56

**installing**  
 Cygwin on Windows, 18  
 gawk (GNU awk), 232–233  
 Korn shell on Windows, 19  
 MRTG, 341  
 sed text-processing tool, 191–193

**instant messaging (IM), 66–67, 386–387**

**interactive commands or programs. See also specific commands**  
 driving with here files, 183–186  
 expect package for, 184  
 gathering keyboard input, 76–78, 130–131  
 running bc interactively, 293–294  
 using in scripts, 41–42

**Interapplication Communication (IAC), 414**

**Internet resources**  
 AbiWord information, 410–411  
 Advanced Bash-Scripting Guide, 436  
 Apache Axis information, 378  
 AppleScript information, 417  
 awk information, 255  
 bash shell, 8  
 Bourne shell information, 436  
 Cream for Vim, 59  
 emacs text editor, 48  
 gawk (GNU awk), 233  
 Java Runtime Environment, 59  
 Java-based text editors, 59, 61  
 Linux text editors, 64  
 Motif libraries, 64  
 MRTG home page, 341  
 NEdit text editor, 64  
 OpenOffice.org Basic help, 404  
 round-robin database (RRD), 342

sed information, 222  
 sed text-processing tool, 191–192  
 SubEthaEdit text editor, 62  
 Textpad text editor, 65  
 tksh shell information, 7  
 Windows Services for UNIX information, 19  
 xemacs text editor, 54  
 Yopy PDA information, 377  
 Z shell information, 7

**interpreters, 162**

**invoking. See running**

**iterations, looping for fixed number of, 93–96**

**iTunes, alarm clock script using, 411–413**

## J

**J text editor, 60–61**  
**Java Runtime Environment, 59**  
**jEdit text editor, 59–60, 411**  
**Jext text editor, 61**  
**Joy, Bill (shell creator), 4**

## K

**kate text editor (KDE), 64**

**KDE desktop**  
 kate text editor, 64  
 konsole application, 16  
 running the shell window, 16

**keyboard**  
 Control and Meta keys on, 48  
 gathering input with read, 76–78  
 gathering input with set, 130–131  
 as standard input, 107, 258  
 Terminal settings (Mac OS X), 431–432

**Keychain Scripting application (Mac OS X), 420**

**keywords for character classes (sed), 215–216**

**kill command, 284, 462–463**

**konsole application (KDE), 16**

**Korn, David (shell creator), 5**

**Korn shell (ksh)**  
 command-history feature, 31, 32–33  
 file-name completion feature, 34  
 installing on Windows, 19  
 overview, 5–6  
 POSIX standard and, 5, 8  
 public domain version (pdksh), 6  
 reading command-line arguments, 156  
 startup by, 153–154  
 tksh as extension of, 7

## L

**LANG environment variable, 136**

**launching. See loading; running**

**legacy editors. See emacs text editor; vi text editor**

**less-than sign (<). See angle brackets (< >)**

## libraries (OpenOffice.org Basic)

creating, 398–400  
defined, 397

## Linux. *See also* GNU Linux; Unix

bash as default shell for, 6  
bash popularity and, 6  
changing the default shell, 9  
checking USB port versions, 391–392  
command-line options on, 23–25  
graphical text editors, 63–64  
Korn shell not supported by, 6  
listing the environment, 137–138  
running shells on, 15–17  
tosh popularity and, 6  
viewing USB port details, 387–391

## LISP, 47, 48

### listing. *See also* `ls` command; viewing

active processes, 276–278, 382–384  
command-line arguments, 157–159  
environment variables on Linux, 137–138  
environment variables on Mac OS X, 138–139  
environment variables on Windows XP, 140–141  
files for current directory, 2  
`myls` script for, 90–91  
`myls2` script for, 91–93  
only environment variables, 142  
PIDs with `ps` command, 276–278  
`/proc` files, 280–281

### loading. *See also* running

AppleScript dictionaries, 417  
files on `awk` startup, 234–235  
files on `emacs` startup, 50  
files on `vi` startup, 55–56

## local variables, 141

## locking down file permissions, 171–172

## logic constructs. *See* controlling how scripts run

## login files, shells and, 153–155

## logout files, shells and, 155

## looping. *See also* `for` loop

in bash shell, 96–97  
basic `for` loop syntax, 90  
in C shell, 98–99  
defined, 89  
for fixed number of iterations, 93–96  
nested loops, 99–100  
over files, 90–93  
overview, 89–90  
`sleep` command for pausing iterations, 95–96  
until condition is true, 132–133  
while condition is true, 131–132

## `ls` command

bracket syntax with, 3–4  
for checking permissions, 160–161  
file-name completion with, 34–35  
listing files in current directory, 2  
listing `/proc` files, 280–281

overview, 480–481

redirecting output to file, 106–108, 258–259  
redirecting output to null file, 263  
redirecting standard error and output, 260–261  
running from a variable, 74–75  
scripts mimicking, 90–93  
storing in a variable, 75–76  
testing with `if` statement, 105–108  
truncating files when redirecting output, 262–263  
variables as arguments for, 74

## `lsusb` command (Linux)

checking USB port versions, 391–392  
viewing USB port details, 389–391

# M

## Mac OS X. *See also* AppleScript

alarm clock script, 411–413  
bash as default shell for, 6, 8  
changing the default shell, 10–12  
command-line options on, 25–26  
compatibility issues, 6  
HFS+ file system, 175–177  
Interapplication Communication (IAC), 414  
layout not standard Unix, 172–173  
listing the environment, 138–139  
mobile file systems and, 173–174  
naming issues, 175  
NeXT legacy in, 172–173  
Open Scripting Architecture (OSA), 413–414  
resource forks, 176–177  
running shells on, 17  
`sed` version, 191  
Target Disk Mode (TDM), 174  
Terminal application, 10, 17, 425–432  
text editors, 61–63  
`vim` text editor, 58–59, 65

## macros (OpenOffice.org Basic)

creating a macro, 400–405  
for creating files from scripts, 407–410  
defined, 397  
passing parameters from the command line, 405–406  
recording, 405  
running from the command line, 405

## magic line (! #), 161–163

## `make` command, 104–106, 193

## Makefile file, 105

## `man` command

for command-line option information, 24–25  
`MANPATH` environment variable for, 139  
overview, 463–464  
for shells, 13

## math using command substitution, 290–292

## mathematical expressions, resolving, 491–492

## memory usage, graphing with MRTG, 354–358

**messages. See also error messages**

- displaying with here files, 178–179
- instant messaging (IM), 66–67, 386–387
- sending to all logged-in users, 179–180

**Meta key for emacs, 48, 49****methods (OpenOffice.org Basic), 397****Microsoft Windows. See Windows****minibuffer area (emacs), 51****mkdir command, 481–482****modules (OpenOffice.org Basic)**

- creating, 398–400
- defined, 397

**monitoring. See also MRTG (Multi Router Traffic Grapher)**

- applications remotely with MRTG, 366–372
- computer and resources with MRTG, 353–363
- HTTP data, 377–378
- networks with MRTG, 363–365
- other data with MRTG, 341
- routers with MRTG, 340
- system uptime with MRTG, 343–344
- web servers with MRTG, 368–372

**more command, 259****Motif libraries, 64****mount command, mobile file systems and, 174****mouse**

- active select and paste model and, 16–17
- replacing Mac single-button mouse, 17

**moving files, 482****MRTG (Multi Router Traffic Grapher)**

- basic syntax, 350
- configuration file for, 345
- configuring `cron` for, 352–353
- configuring global values, 345–346
- configuring graphs, 348–349
- configuring target HTML outputs, 347–348
- configuring targets, 346–347
- configuring to monitor web servers, 369–372
- customizing output from, 347–350
- disk space required by, 342
- exercises and answers, 373, 457–458
- graphing CPU usage, 358–360
- graphing disk usage, 361–363
- graphing memory usage, 354–358
- installing, 341
- maximizing performance, 353
- monitoring applications remotely, 366–372
- monitoring computer and resources, 353–363
- monitoring networks, 363–365
- monitoring other data, 341
- monitoring routers, 340
- monitoring system uptime, 343–344
- obtaining, 341
- running, 350–351
- SNMP protocol used by, 340–341
- steps for configuring, 345

targets, 339–340, 346–348

testing scripts, 344

uses for, 339, 373

verifying your configuration, 349–350

viewing output, 351–352

working with, 339–340

writing scripts for, 342–344

`mrtg_sys.cfg` file, 370–372

**MS-DOS**

batch files as scripting language, 47

primitive shell provided by, 18

slash for command options, 23

wildcards and, 35

MsgBox dialog box types (OpenOffice.org Basic), 404

**Multi Router Traffic Grapher. See MRTG**

**multimedia scripting**

Rhythmbox music player, 433–435

totem movie player, 435

xmms music player, 66, 387, 432–433

**multitasking, shell support for, 37–38****music players**

Rhythmbox, 433–435

xmms, 66, 387, 432–433

**mv command, 482**

**mysls script, 90–91**

**mysls2 script, 91–93**

**N****names and naming**

emacs long command names, 50

for environment variables, uppercase, 138

extracting base file name from path, 471

file-name completion feature, 34–35

functions as named code blocks, 299, 300

insulting, avoiding, 80

Mac naming issues, 175

naming blocks of code, 300–302

personal startup file names, 153

storing paths in variables, 334

test command for file names, 120–121, 171–172

variable names, 71, 334

**navigation**

emacs commands for, 51–52

of operating system, commands for, 461–470

vi commands for, 56–57

**nc command, 65, 411**

**NEdit text editor, 64–65, 411**

**negating addresses with sed, 201–202**

**negation test, 121, 122–123**

**nesting**

backticks, 292

if statements, 111, 112, 113–114

loops, 99–100

tracing nested statements, 330–331

**NetInfo Manager (Mac OS X), 10–12**

**netstat command, 364–365**

## networks

- monitoring with MRTG, 363–365
- starting USB networking, 70–71, 376–377

## newlines

- echo command without, 68–69
- printing with awk, 239

**NeXTSTEP, Mac OS X legacy from, 172–173**

**noexec mode, 327, 329**

**nohup command, 464**

**Notepad text editor, 65**

**NR variable (awk), 247–248**

**null device, redirecting output to, 106–108, 263**

**numbers, test command for, 114–117**

## O

**octal file modes, 169–170**

**od command, 282–283**

**office command. See also OpenOffice.org suite**

- command-line arguments, 396–397
- running macros with, 405

**Open Scripting Architecture (OSA), 413–414**

**Open Scripting Architecture extension (OSAX), 415**

**opening. See loading; running**

**OpenMotif libraries, 64**

**OpenOffice.org Basic**

- creating files from scripts, 407–410
- creating macros, 400–405
- creating modules, 398–400
- help online, 404
- numeric codes for MsgBox dialog box types, 404
- overview, 396, 397
- passing parameters from the command line, 405–406
- terminology, 397

**OpenOffice.org suite. See also OpenOffice.org Basic**

- emacs conflicts with Writer, 53
- name explained, 396
- office command-line arguments, 396–397
- passing parameters from command line to, 405–406
- programming API with, 396
- programs in, 396
- recording macros, 405
- running macros from the command line, 405
- Sun Microsystems StarOffice version, 400

**operating systems. See also specific operating systems**

- commands for navigating, 461–470
- default shells, 8
- determining which is running, 70
- if statements for commands specific to, 101
- printing system information, 468–469
- single-shell systems, 3

**operators, comparison (awk), 250–251**

**options. See command-line options**

**OR operation for binary tests, 121–123**

**OSA (Open Scripting Architecture), 413–414**

**osacompile command (Mac OS X), 418–419**

**osalang command (Mac OS X), 418**

**osascript command (Mac OS X), 420**

**OSAX (Open Scripting Architecture extension), 415**

**output redirection. See redirecting output**

## P

**parentheses [( )]**

- for command substitution, 289
- double parentheses syntax with for loop, 96–97

**pasting, active select and paste model for, 16–17**

## paths

- for commands, 2, 22
- in Cygwin environment, 18
- extracting base file name from, 471
- magic line (!#), 161–163
- mobile file systems and, 173
- not depending on, 22
- shell and interpreter locations, 162
- storing in variables, 334

**pattern space (sed), 195**

## PDAs

- running shells on, 19
- script for starting USB networking, 70–71, 376–377

**pdsh (public domain Korn shell), 6**

**percent sign (%)**

- in printf format control characters, 243–244
- in shell prompt, 20

**period. See dot (.)**

**Perl (Practical Extraction and Reporting Language), 43–45, 63**

## permissions

- adding execute permissions, 161, 163
- changing with chmod, 170–171, 472–474
- checking with ls, 160–161
- checking with test, 121, 171
- file modes, 169–171
- locking down file permissions, 171–172
- mobile file systems and, 173

## personal startup files

- bash shell, 154–155
- Bourne shell, 153
- C shell, 154
- Korn shell, 153–154
- naming, 153
- system files versus, 152–153
- T C shell, 154

## PIDs (process IDs)

- defined, 276
- killing processes, 284
- listing with ps command, 276–278
- reading, 278–279

**ping command, 29–30, 363–364**

**pipng commands**

- basic syntax, 264
- checking spelling, 265–266
- creating pipelines, 266–271
- exercises and answers, 273, 451
- power of, 257
- processing `/etc/passwd` user names, 267–270
- with scripts, 270–271
- `tee` command for multiple locations, 271–272
- Unix commands, 264–266

**playing**

- movies, 435
- songs, 66, 387, 432–435

**positional variables**

- Bourne shell, 156–159
- defined, 156
- Korn shell, 156

**POSIX (Portable Operating System Interface for Computer Environments) standard**

- bash command-line option for, 8
- Korn shell and, 5, 8
- sed version, 191

**pound sign. See hash mark (#)****Practical Extraction and Reporting Language (Perl), 43–45, 63****print command (awk)**

- field separators with, 240–241
- inserting text, 238–239
- overview, 237–238
- printing newlines, 239

**printenv command**

- under the C shell, 146–147
- listing only environment variables, 142–144
- overview, 464–465

**printf command (awk)**

- format control characters, 242–243
- overview, 241–242
- using format modifiers, 243–244

**/proc file system**

- defined, 279
- information in, 279–280
- listing files in, 280–281
- viewing file contents, 281–282
- viewing process data, 282–284

**process IDs. See PIDs****processes**

- active, listing with `ps` command, 276–278, 382–384
- backticks for command substitution, 287–289
- capturing output of, 287–296
- defined, 275
- exercises and answers, 297, 452–453
- killing, 284, 462–463
- reading PIDs, 278–279
- reading the `/proc` file system, 279–284

- running from scripts, 285–287, 296–297
- Terminal settings (Mac OS X), 427
- verifying processes are running, 384–385

**processing text. See awk language; sed text-processing tool****.profile file, 153, 154–155****programs. See also applications; interactive commands or programs; processes; specific programs**

- defined, 275
- errors in calling, 322–323
- in OpenOffice.org suite, 396
- running, defined, 276
- running within scripts, 296–297
- self-contained, creating with `awk`, 236–237

**prompt. See shell prompt****ps command**

- listing active processes, 276–278, 382–384
- overview, 465–466
- verifying processes are running, 384–385

**public domain Korn shell (pdksh), 6****Python scripting language, 45–46, 63****Q****question mark (?)**

- for return codes variable (`$?`), 295–296, 309
- as wildcard, 37

**quotation marks (“”) for variable values including spaces, 72****R****rc shell, 7****read command, 76–78, 258****reading**

- command-line arguments, 156–160
- environment variables, 136–147
- file data with `sed -e` flag, 196–197, 208
- files into variables, 292
- PIDs, 278–279
- `/proc` file system, 279–284

**reading command-line arguments**

- Bourne shell, 156–159
- C shell, 160
- Korn shell, 156
- T C shell, 160

**reading environment variables**

- with C shell, 144–147
- checking, 147–150
- on Linux, 137–138
- listing only environment variables, 142–144
- on Mac OS X, 139–140
- need for, 136
- on Windows XP, 140–141

**recording macros (OpenOffice.org), 405****recursion, 314–315**

### **redirecting input. See also piping commands**

here files versus, 179, 180  
standard input, 258, 259

### **redirecting output. See also piping commands**

appending to file, 261–262  
with `awk`, 252–253  
destructive nature of, 261  
exercises and answers, 273, 451  
with `if` statements, 106–108  
to null file, 106–108, 263  
from `sed`, 197  
standard error, 108, 259–260  
standard output, 106, 108, 258–259  
standard output and standard error, 260–261  
`tee` command for multiple locations, 271–272  
truncating files, 262–263

### **referencing variables**

curly braces for, 86  
dollar sign as prefix for, 22, 72, 85–86, 88  
script illustrating, 86–89

### **regular expressions**

for address ranges in `sed`, 216  
for addresses in `sed`, 212–214  
for back references in `sed`, 219–220  
for combining line addresses in `sed`, 217  
for `FS` variable (`awk`), 246  
`grep` searches using, 478–479  
for referencing matched expressions in `sed`, 218–219  
syntax and, 4  
wildcards for, 35–40

`rehash` command, 161

### **repeating**

parts of previous commands, 30–31  
previous commands, 27–30

### **resource forks (Mac OS X), 176–177**

### **return codes**

with functions, 309–311  
`if` statements testing, 102–106  
shell exit codes, 309  
variable for (`$?`), 295–296, 309

### **reverse address ranges with `sed`, 200–201**

### **Rhythmbox music player, 433–435**

`rm` command, 482–483

`rmdir` command, 483

### **round-robin database (RRD), 342**

### **routers, monitoring, 340**

`rrdtool` program, 342

### **running. See also loading**

`awk`, invoking, 234–237  
`awk` scripts with `-f` flag, 236  
checking syntax with `-n` option, 327, 329  
commands from the prompt, 2  
commands from variables, 75–76  
commands in directories, 386–387  
commands in subshells, 286  
commands in the background, 37–38, 285–286

commands in the foreground, 285  
commands with `exec`, 286–287  
in debugging mode, 327–333  
defined, 276  
`emacs` text editor, 49  
functions, calling, 303, 305–306  
initialization files for shells, 153–155  
making scripts executable, 160–164  
`MRTG`, 350–351  
`OpenOffice.org` macros from the command line, 405  
processes from scripts, 285–287  
programs or scripts within scripts, 296–297  
`sed`, advanced invocation, 207–211  
`sed`, invoking with `/etc/passwd` data, 194–195  
`sed`, invoking with flags, 196–199  
shell scripts with `sh` command, 40  
shell within a shell, 12  
shells on Linux, 15–17  
shells on Mac OS X, 17  
shells on PDAs and other systems, 19  
shells on Unix, 18  
shells on Windows, 18–19  
startup process for shells, 153–155  
tracing script execution, 327, 329–333  
in verbose mode, 328–329  
`vi` text editor, 55  
`xmms` music player from scripts, 66, 387, 432–433

## S

### **SciTE text editor, 64**

### **scope of variables, 311–314**

### **screen, as standard output, 107, 258**

### **scripting languages**

as alternatives to shells, 39, 42–43  
choosing, 43  
MS-DOS batch files, 47  
Perl, 43–45, 63  
programming languages versus, 89  
Python, 45–46, 63  
Tcl, 46–47

### **scripting with files**

combining files into archives, 168–169  
exercises and answers, 187, 446–448  
file modes, 169–171  
here files, 177–186  
Mac OS X files, 172–177  
overview, 167  
testing with `test` command, 120–121, 171–172

### **Scrollbar settings (Mac OS X Terminal), 427–428**

### **searching. See finding**

`sed` command, 488–489

### **sed text-processing tool**

address negation, 201–202  
address ranges, 200–201  
address stepping, 202

- address substitution, 206–207
- advanced addressing, 211–217
- advanced invocation, 207–211
- advanced substitution, 217–220
- alternative string separators, 205–206
- ampersand metacharacter, 218–219
- append command, 210–211
- awk compared to, 234
- back references, 219–220
- bootstrap installation, 192
- change command, 211
- character class keywords, 215–216
- checking version of, 191
- combining line addresses with regular expressions, 217
- commands, 224–226
- comment command, 209–210
- common one-line scripts, 222–223
- configuring, 193
- deleting all lines using, 195–196
- disabling automatic printing, 197, 224
- editing commands, 195–196, 207–209
- error messages, 199, 201
- exercises and answers, 227, 448–450
- f flag, 209
- further information, 222
- GNU sed commands, 225–226
- hold space, 220–222
- insert command, 210–211
- installing, 191–193
- invoking with `/etc/passwd` data, 194–195
- invoking with flags, 196–199
- n, --quiet, or --silent flags, 197–198, 224
- obtaining, 191–192
- option usages, 194–195
- overview, 189–190
- pattern space, 195
- printing with `p` command, 197–198
- reading file data using `-e` flag, 196–197, 208
- redirecting output from, 197
- referencing matched regular expressions, 218–219
- regular expression address ranges, 216
- regular expression addresses, 212–214
- replacing with empty space, 206
- running on command line, 189
- sed command, 488–489
- sed scripts, 209
- selecting lines to operate on, 199–202
- specifying multiple editing commands, 207–209
- as stream editor, 193–194
- substitution command, 203–207
- substitution flags, 204–205
- versions, 190–191
- selecting, active select and paste model for, 16–17**
- semicolon (;) for separating commands, 94**
- set **command**
  - C shell and, 144–145
  - for exporting environment variables, 151
  - for listing the environment, 137–141
- setenv **command**
  - for listing the environment, 137, 145–146
  - for setting environment variables, 152
- setting environment variables, 150–152**
- sh. See Bourne shell**
- sh **command**
  - for running shell scripts, 40
  - shells masquerading as, 6, 8
- sharp sign. See hash mark (#)**
- shell prompt**
  - convention in this book, 21
  - defined, 2
  - in gnome-terminal window, 16
  - overview, 20–21
- shell scripts. See also specific kinds**
  - continuing lines in, 81–82
  - defined, 40
  - entering comments in, 78–80
  - exercises and answers, 83, 439–442
  - first example, 40–41
  - for gathering input, 76–78
  - interactive commands in, 41–42, 76–78
  - making executable, 160–164
  - need for, 39
  - for outputting text, 67–71
  - for piping commands, 270–271
  - running programs or scripts from, 296–297
  - for storing complex commands, 66–67
  - using variables, 71–76
  - wildcards in, 40
- SHELL variable, 21, 22**
- shell variables. See environment variables; variables**
- shells. See also specific shells**
  - automatic logout time for, 21
  - changing the default shell, 9–12
  - choosing, 9
  - default shells, 6, 8
  - defined, 2
  - determining which are installed, 108–111
  - determining which is running, 21–22
  - editing commands, 27–35
  - emacs interactive shell, 54
  - entering commands, 20–27
  - file managers versus, 13
  - file-name completion feature, 34–35
  - further information, 13
  - going from AppleScript to, 418–420
  - going to AppleScript from, 420–425
  - graphical environments and, 7, 13–20
  - hierarchy of, 142
  - locations for, 162

### shells (continued)

- login versus others, 153–155
- origin of term, 3
- POSIX standard shell, 8
- running a shell within a shell, 12
- running commands in the background, 37–38
- single-shell systems, 3
- startup process for, 153–155
- Terminal settings (Mac OS X), 426
- types of, 4–7
- uses for, 3–4
- wildcards, 35–37
- window as standard output, 107, 258

**Simple Network Management Protocol (SNMP), 340–341, 372**

**simplifying scripts, 335**

**slash (/)**

- for command options in MS-DOS, 23
- for searching in vi, 58
- as sed string separator, 203, 205, 206
- surrounding sed regular expression addresses, 212

**sleep command, 95–96, 466–467**

**SlickEdit text editor, 61**

**.smi files (Mac OS X), 174**

**SNMP (Simple Network Management Protocol), 340–341, 372**

**soffice command, 400**

**sort command, 266, 489–490**

**source command**

- for function files, 306–307
- reading values into current environment, 152

**spaces**

- as awk field separators, 240, 246
- Mac naming issues and, 175
- missing, debugging, 321–322

**spell command, 265–266**

**splat. See star (\*)**

**sprintf command (awk), 244**

**Stallman, Richard (emacs creator), 48**

**standard error (stderr)**

- defined, 107, 258
- redirecting standard output and, 260–261
- redirecting to file, 108, 259–260

**standard input (stdin). See also keyboard**

- defined, 107, 258
- redirecting, 258, 259

**standard output (stdout)**

- defined, 107, 258
- redirecting and appending to file, 261–262
- redirecting standard error and, 260–261
- redirecting to file, 106, 108, 258–259
- redirecting to null file, 106–108, 263
- truncating files when redirecting, 262–263

**star (\*)**

- as case statement catch-all value, 127–129
- for character matching in sed, 213

- hidden files and, 91
- as wildcard, 35–36

**StarOffice (Sun Microsystems), 400**

**starting. See loading; running**

**stderr. See standard error**

**stdin. See standard input**

**stdout. See standard output**

**strings command, 490**

**strings, test command for, 117–120**

**Stuffit program (Mac OS X), 169**

**SubEthaEdit text editor (Coding Monkeys), 62**

**subshells**

- defined, 141
- hierarchy of, 142
- running commands in, 286
- scripts run in, 152

**substitution. See command substitution**

**substitution command (sed)**

- address substitution, 206–207
- advanced substitution, 217–220
- alternative string separators, 205–206
- back references, 219–220
- flags, 204–205
- overview, 203–204
- referencing matched regular expressions, 218–219
- replacing with empty space, 206

**Sun Microsystems StarOffice, 400**

**swapfiles, Mac OS X and, 172**

**switch statement (C shell), 129–131**

**syntax errors. See also debugging scripts**

- avoiding, 333–334
- checking with `-n` option, 328, 329
- finding, 321–322
- missing syntax, 319–321

**system files, leaving alone, 152–153**

**system uptime, monitoring, 343–344**

## T

### T C shell (tcsh)

- file-name completion feature, 34
- overview, 6–7
- prompt, 20, 21
- reading command-line arguments, 160
- rebuilding list of executables, 161
- `set` command, 144–145
- `setenv` command, 145–146
- startup by, 154
- variables and, 73

**tail command, 483–484**

**tar command, 168**

**Target Disk Mode (TDM), 174**

**targeting applications (AppleScript), 415**

**targets (MRTG)**

- configuring, 346–347
- HTML output configuration, 347–348
- overview, 339–340

**Tcl (Tool Command Language), 46–47**tcpmon **program**, 378tcsh. *See* **T C shell**.tcshrc **file**, 154, 155**TDM (Target Disk Mode), 174**tee **command**, 271–272**Terminal application (Mac OS X)**

changing the default shell, 10

Preferences, 425–426

running shells using, 17

Window Settings, 426–432

**Terminal Inspector (Mac OS X)**

Color settings, 429–430

Display settings, 429

Emulation settings, 427, 428

Keyboard settings, 431–432

Processes settings, 427

Scrollback settings, 427–428

Shell setting, 426

Window settings, 431

**test command**

binary tests, 121–123

bracket as shorthand for, 123–125

for files, 120–121, 171–172

negation test, 121, 122–123

for numbers, 114–117

overview, 114

for text strings, 117–120

**testing scripts, 335, 344****text editors. *See also specific editors***

cross-platform, 59–61

editing shell commands using, 33–34

graphical, 59–65

legacy, 47–59

for Linux, 63–64

for Mac OS X, 61–63

need for, 47

for Unix, 64–65

for Windows, 65

**text manipulation commands, 485–487****text processing. *See* awk language; sed text-processing tool****TextEdit text editor**, 61**Textpad text editor**, 65**tilde (~) for user's home directory**, 20**Tkl scripting language**, 7**tksh shell**, 7**Tool Command Language (Tcl), 46–47**totem **movie player**, 435touch **command**, 484–485tr **command**, 283–284, 490–491**tracing script execution**

with command substitution, 331–332

for loops, 332–333

listusers example, 330

manually looking over commands, 327

nested statements, 330–331

-x option for, 329–333

true **command**, 103–104type **command**, 467–468**U****uname command**

combining options with, 26

on Mac OS X, 25–26

overview, 468–469

on Unix and Linux, 23–24

using with echo command, 70

uncompress **command**, 168**uncompressing files**, 168, 169**undeclaring functions**, 307–308**underscore (\_) beginning variable names**, 71uniq **command**, 266**Unix. *See also* Linux; Mac OS X**

Bourne shell support on, 4

C shell and distributions of, 4, 5

changing the default shell, 9

command-line options on, 23–25

default shells for systems, 8

design philosophy of, 257

graphical text editors, 64–65

Korn shell and commercial versions of, 5

piping commands, 264–266

running shells on, 18

unset **command**, 76until **loop**, 132–133**up arrow for cycling through commands**, 31uptime **command**, 262, 358–359**USB**

checking port versions, 391–392

lsusb command (Linux), 389–392

port oddities, 389

script for starting networking, 70–71, 376–377

viewing port details in Linux, 387–391

**user-defined variables, 244, 245. *See also* variables****V****variables. *See also* environment variables**

building up commands with, 74–76

C shell or TC shell and, 73

combining in scripts, 73–74, 88–89

for command-line arguments, 156–160

de-assigning, 76

defined, 71

double quotes with, 72–73

echo command with, 21–22, 71–74

gathering keyboard input to, 76–78

in here files, 181–186

local, defined, 141

naming, 71, 334

PID-related (\$\$ and \$!), 278

### variables (continued)

- positional, 156–159
- reading files into, 292
- referencing values for, 72, 85–89
- for return codes (\$?), 295–296, 309
- running commands from, 75–76
- scope of, 311–314
- scripting languages versus programming languages and, 89
- setting from commands, 289–290
- shell variables, 22
- with `sprintf` command (awk), 244
- storing commands in, 75–76
- user-defined versus built-in, 244

### verbose mode, 328–329

### vi text editor

- colon starting commands, 58
- copy and paste commands, 58
- as cross-platform editor, 59
- driving with here files, 184–185
- editing shell commands using, 33–34
- file-related commands, 57–58
- insert commands, 56
- jumping between files, 56
- loading files on startup, 55–56
- navigation commands, 56–57
- search commands, 58
- shell commands for, 33
- starting, 55
- usefulness of, 48
- using in scripts, 41–42
- vim (vi improved), 58–59, 65

### viewing. *See also* listing

- command history, 31–33
- environment variables values, 464–465
- first few lines of files, 479
- last few lines of files, 483–484
- MRTG output, 351–352
- /proc file contents, 281–282
- process data in /proc, 282–284
- USB port details in Linux, 387–391

### vim text editor, 58–59, 65

### vmstat program, 354–358

### vscanx command, AppleScript for, 420–425

## W

### wall command, 179–180

### wc command, 92, 259

### web servers

- downloading pages with `wget` command, 366–368
- monitoring with MRTG, 368–372
- webalizer command for log files, 372

### webalizer command, 372

### wget command, 366–368

### while loop, 131–132, 253

### who command, 469–470

### whoami command, 69–70, 470

### wildcards

- defined, 35
- as globs, 35
- MS-DOS and, 35
- question mark (?), 37
- in shell scripts, 41
- star (\*), 35–36

### Window settings (Mac OS X Terminal), 431

### Windows (Microsoft). *See also* Cygwin environment

- installing Cygwin, 18
- listing the environment, 140–141
- running shells on, 18–19
- as single-shell system, 3
- text editors, 65

### Windows Services for UNIX (Microsoft), 19

### Word (Microsoft), emacs conflicts with, 53

### Writer (OpenOffice.org), emacs conflicts with, 53

### writing scripts

- commenting, 78–80, 334
- for complex commands, 66–67
- continuing lines, 81–82
- creating informative error messages, 334–335
- for gathering input, 76–78
- guidelines for administrative scripts, 376
- guidelines for tidy scripts, 333–334
- for MRTG, 342–344
- for outputting text, 67–71
- practices for avoiding errors, 333–335
- simplifying, 335
- testing, 335
- using variables, 71–76

## X

### X Window System active select and paste model, 16–17

### X11 application (Mac OS X), 17

### xemacs text editor, 54

### xmms music player, 66, 387, 432–433

## Y

### Yopy PDAs

- further information, 377
- running shells on, 19
- script for starting USB networking, 70–71, 376–377

## Z

### Z shell (zsh), 7

### Zaurus PDAs, running shells on, 19

### zero, as success return code, 100, 102–103

### zip program, 169