

INDEX

Symbols

- & (ampersand), 24
- && (and) operator, 145
- * (asterisk)
 - parameter variable, 178
 - regex quantifier, 51
 - wildcard character, 16, 34, 49, 124, 264, 304, 359, 402
- @ (at sign), 49, 51–52
 - parameter variable, 178, 179
- \ (backslash), 37, 55, 88, 112, 264
 - looping through lines in a file, 167
 - starting continuation using, 229
- ` (backtick), 306, 359
- { } (braces), 104, 105
 - in variable syntax, 127
 - in xargs examples and man pages, 105
- ^ (caret or circumflex), 55, 90, 93, 366, 412
- :
- , (comma), 74, 356
- (dash or hyphen), 53, 217, 348, 416
- \$ (dollar sign), 122, 133, 222
 - end-of-line anchor, 55, 298
 - parameter expansion, 298, 299–300
 - in regular expressions, 89
- . (dot) character, 4, 49
- ./ (dot-slash), 25
- " (double quotes), 125, 126
- >> (double right angle brackets), 22, 26
- ;; (double semicolon), 148
- ! (exclamation mark), 10
- / (forward slash), 78, 87, 264, 304
- # (hash mark), 29, 87, 111, 348
 - in command prompt, 222
 - parameter variable, 178
- ## (hash marks, double), 359
- || (or) operator, 145, 150, 152, 167, 190
- % (percent sign), 6, 367
- . (period), 304
- | (pipe) operator, 415
- + (plus sign), 51, 73, 88, 135, 192, 229
- ? (question mark), 57, 126
 - searching for files or folders using, 35
 - as wildcard, 310
- > (redirect symbol), 24, 25, 27
- > (right angle bracket), 113, 229
- ; (semicolon), 39, 145–146, 152, 308, 356
- #! (shebang), 116–117
 - configuring options in, 203
 - to locate python3 program, 324
 - omitting, 202
- ' (single quote), 124–125, 126
- [] (square brackets), 75
 - in command prompt, 225
 - in conditional expressions, 150
- ~ (tilde), 88, 125–126, 272, 298, 299, 396

A

- absolute paths, 397
- ack tool, 70
- actions in find expressions, 38
- addresses in sed functions, 87
- ag tool, 70
- ai command, 378–379
- AI-generated scripts, 379–380
- alias command, 414
- aliases, 209, 414
 - with different users, 216
 - saving time with, 209–210
- all parameters array (\$a), 196
- alternate value in parameter expansion, 301
- Amazon Web Services, 328
 - creating virtual machine on, 331–335
 - credit card use for paid services from, 330
 - setting up account, 329–330

- ampersand (&), 24
- anchors in regexes, 55–56
- AND expressions, 36
- and (&&) operator, 145
- angle bracket (>), 113, 229
- ANSI C quoting, 124–125
- ANSI escape sequences, 197, 199, 225, 227
- ANSI formatting sequences, 226
- anti-patterns for shell scripts, 201–203
 - configuring options in shebangs, 203
 - omitting shebangs, 202
 - using complex logic, 203
- appending
 - to files, 22, 26
 - text with sed, 89–90
- append redirection operator (>>), 22, 26
- Applied Cryptography* (Schneier), 328
- apropos command, 418
- Apt package manager, 419
- argparse Python module, 321
- arguments, use of term, 396
- arithmetic expansion, 126, 136, 298, 306
- arithmetic operations, 135–137
- arrays
 - associative, 130
 - expanding, 303
 - finding length of, 303
 - looping through with for loop, 157
 - overview, 128–129
 - sparse, 129
 - using operators in parameter variables, 179
- ASCII EOT (end of transmission) character, 201
- asterisk (*)
 - parameter variable, 178
 - regex quantifier, 51
 - wildcard character, 16, 34, 49, 124, 264, 304, 359, 402
- atomic groups, 59
- awk standard output applications as
 - alternative to sed, 97
- AWS. *See* Amazon Web Services

B

- backslash (\), 37, 55, 88, 112, 264
 - looping through lines in a file, 167
 - starting continuation using, 229
- backtick (`), 306, 359
- backtracking, 58, 59
- basename command, 192
- BASH_ENV variable, 216, 218
- bash parameter, 217
- BASH_REMATCH array, 151
- basics of shell, 393–420
 - clipboard, 412–415
 - getting help, 415–420
 - accessing online cheat sheets, 420
 - summarizing output with tldr, 418–419
 - using manual, 415–418
 - managing files and folders, 400–412
 - copying files, 404–405
 - creating files, 408–409
 - creating folders, 407–408
 - deleting files, 403
 - deleting folders, 409–410
 - downloading files, 400–402
 - renaming and moving files, 405–407
 - showing text content, 410–411
 - unzipping files, 402–403
 - viewing directory tree, 404
 - working with wildcards, 409
 - zipping files, 411
 - navigating filesystem, 393–400
 - changing directory, 395–396
 - going back to previous directory, 398–399
 - identifying working directory, 394
 - listing contents of working directory, 394–395
 - navigating with dot and double-dot folders, 397–398
 - pushing and popping working directory, 399–400
 - returning to home directory, 396
 - using absolute and relative paths, 397
 - bin* directory for custom commands, 210–211
- bindkey command, 10

- bind -p command, 10
- brace expansion, 100, 126, 159, 164–165, 298–299
- braces ({}), 104, 105
 - in variable syntax, 127
 - in xargs examples and man pages, 105
- branches in Git, 250, 257–262
 - creating, 257–260
 - diverged, 261–262
 - merges
 - fast-forward, 260
 - recursive, 261–262
 - resolving conflicts between, 263–267
- break statements, 169–170
- brew command, 391
- BSD Unix, 382
- bugs, GitHub repository for
 - reporting, xxx
- builtins, 416–417

C

- C (programming language), 315
- capture groups, 56–57, 94, 150
- caret or circumflex (^), 55, 90, 93, 366, 412
- case-insensitive searches
 - with find, 36
 - with grep, 66
- case sensitivity in Vim, 346
- case statement, 148–149, 196, 199
- cat (concatenate) command, 17, 410, 411, 414
- catastrophic backtracking, 58
- cd command, 211, 398, 401
- CDPATH settings, 244
- Chacon, Scott, 292
- chaining commands, 151–152
- character classes, 75
- characters
 - counting number of, 19
 - lowercase, translating to uppercase, 23
 - sequences of, creating, 298–299
- character sets and metacharacters, 52–57
 - adding special characters, 53–54
 - negating characters, 54–55
 - specifying ranges, 52–53
- ChatGPT, 378–379, 380

- cheat sheets
 - accessing online, 420
 - Vim example, 347, 359–360
- chmod command, 42, 115, 119, 337, 341
- chords in Vim, 346
- chroot command, 240
- chsh command, 219, 391
- clearing screen, 9
- Click Python package, 325
- clipboard, 412–415
 - content as standard input, 20
 - copying and pasting with pbcopy and pbpaste, 414–415
 - creating custom commands, 413–414
 - essentials, 412–413
- collaboration techniques, 286–288
 - forking, 286–287
 - making pull requests, 287–288
 - using multiplexer, 365
 - using tmux, 372–373
- colon (:), 106, 347
- color
 - adding to command prompt, 225–228
 - adding to text displayed in shell, 197–199
- comma (,), 74, 356
- command history
 - searching through, 6–7
 - viewing, shortcut for, 9–10
- command line, 3–11
 - editing in place, 7–9
 - navigation techniques, 4–6
 - searching through command history, 6–7
 - shortcuts, 9–10
- command mode in Vim, 346
- “command not found” error
 - message, 401
- command prompt, 221–235
 - adding color and text formatting to, 225–228
 - adding data to, 228–229
 - shell prompt variables, 229–231
 - showing Git information in, 290–291
 - structure of, 222–225
 - writing shell script to customize, 231–235

- commands. *See also* command prompt;
 - common command functions; xargs
 - command
- building, 99–106
 - handling whitespace, special characters, and tracing, 101–102
 - organizing parameters, 103–105
 - xargs, 100–103
- built-in shell commands, 416–417
- chaining, 151–152
- connecting. *See* pipelines
- custom, 413–414
- determining whether available, 200
- differentiating filepaths from, 25
- executing on search results, 39–40
- format of in this book, xxix–xxx
- Git
 - quick reference, 272–273
 - recap, 293–294
- in history file, rerunning, 10
- output
 - in functions, 182–183
 - storing in variables, 127–128
- running, 105–106
 - SSH, 338
 - in subshells, 201
 - too many on startup, 214
- searching for using `grep`, 63
- searching through command history, 6–7
- sending to nested sessions, `tmux`
 - configuration, 371
- in shell scripts, 112–115
- shell startup, 210–214
- using `Vim`, 352, 358–359
- command substitution, 126, 298, 305–306
- comments
 - adding to shell scripts, 111
 - stripping with `sed`, 89
- committing changes in `Git`, 254–257
 - commit messages, conventions for, 257
 - commit signing, 292
- common command, 119, 146, 147, 195
 - enhancing with variables, 137–138
 - extending to handle different shells, 152–154
 - simplifying with functions, 187–188
 - updating to loop through results, 171–172
- compact loops, creating, 170–171
- complex logic, problems requiring, 314
- concatenating
 - contents of files, 411
 - strings, 136
- conditional expressions, 149–151
- conditional logic, 141–154
 - case statement, 148–149
 - chaining commands, 151–152
 - combining statements on single line, 145–146
 - conditional expressions, 149–151
 - `elif` clause, 146–147
 - `else` clause, 146
 - extending common command to handle different shells, 152–154
 - `if` statements, 142
 - `test` command, 143–145
- configuration files. *See* dot files
- configuring shell
 - changing shell, 219–220
 - interactive shells, 207–214
 - common startup file
 - customizations, 209–214
 - default shell startup file, 208–209
 - in Linux, 392
 - login shells, 216–217
 - in macOS, 390–392
 - in Microsoft Windows, 383–392
 - installing Linux tools, 383–384
 - running virtual machine, 384–389
 - Windows Subsystem for Linux, 389–390
 - non-interactive shells, 214–216
- conflicts in `Git`, resolving, 264–267
- connecting commands. *See* pipelines
- content, adding to file, 22
- continuation character, 112
- continuations, 229
- `continue` statements, 169–170
- copying
 - files, 404–405
 - with `pbcopy`, 414–415
- counting words lines, characters, and bytes, 19

- count variable, 176
- cp command, 86, 87, 88, 89, 157, 398,
 - 404, 406, 415
- cryptography, 340
- C-style loops, iterating with, 163–164
- CTRL-C, handling, 325
- curl command, 320, 321, 420
- cursor, maneuvering, 4–6
- Cursor tool website, 380
- cut command, 75
- cutting text, 75–78
- Cywin, 383–384

D

- dash (-), 53, 217, 348, 416
- date command, 306
- debian_chroot variable, 240
- debugging with trace option,
 - 191–192
- declare command, 130, 192
- default shell
 - dot file, 238–240
 - setting in tmux, 370
 - startup file, 208–209
- delete action, 39
- deleting
 - files, 403
 - folders, 409–410
 - interactively, 403
- dictionary lookup tool
 - improving, 325
 - installing, 324
 - writing in Python, 316–323
 - defining basic structure, 317–319
 - downloading definition, 319–321
 - formatting output, 321–323
- directories
 - changing, 395–396
 - directory stack, 399
 - directory tree, viewing, 404
 - home directory, returning to, 396
 - previous directory, going back to,
 - 398–399
 - working directory
 - identifying, 394
 - listing contents of, 394–395
 - pushing and popping, 399–400
- disconnections in SSH, 339

- diverged branches, handling in Git,
 - 261–262
- do keyword, 169
- dollar sign (\$), 122, 133, 222
 - end-of-line anchor, 55, 298
 - parameter expansion, 298, 299–300
 - in regular expressions, 89
- dot (.) character, 4, 49
- dot files, 237–248
 - custom, creating, 241–244
 - cleaning up variables and
 - configuring keyboard shortcuts, 243
 - creating folder for, 241
 - creating *shell.sh*, 241
 - setting preferred editor, 242–243
 - setting shell history options, 244
 - working with folders, 243–244
 - default shell dot file, 238–240
 - defined, 238
 - installation script, 247–248
 - location of, 238, 245
 - sharing on GitHub, 285–286
 - sourcing from folder, 245–247
 - testing, 244–245
- dot folder, 33
 - navigating with, 397–398
- dot notation to source scripts, 117
- dot-slash (./), 25
- double-dot folders, navigating with,
 - 397–398
- double quotes ("), 125, 126
- double right angle brackets (>>), 22, 26
- double semicolon (; ;), 148
- downloading files, 400–402
- duplicates, removing, 17, 79–80
- dynamic scoping, 175

E

- echo command, 22, 90, 103, 134, 137,
 - 142, 166, 192, 214, 216, 219
- editing in place, 7–9, 96–97
- editors, setting in dot file, 242–243
- EDITOR variable, 9, 212, 242
- elif clause, 146–147
- else clause, 146
- Emacs, 344
- empty folders, finding and removing, 43

- end of transmission (EOT), 14
- engines, regex, 50–52
- env (set or print environment and execute command) command, 116, 117, 122
- environment variables
 - configuring, 212
 - scope of, 122–123
 - shell variables as, 123–124
- envsubst (substitute environment variables) command, 96
- errors, 183, 325
 - in functions, 185–187
 - logging, 25
 - in parameter expansion, 301
 - redirecting to null, 25–26
 - standard, 23–26
 - suppressing display of, 26
- escape character, 125
- escape sequences
 - coloring text displayed in shell using, 197–199
 - customizing command prompt, 223–225
- escaping
 - regex characters, 55
 - special characters, 37
- events, trapping, 193–195
- exclamation mark (!), 10
- ex command in Vim, 347
- exec action, 39, 42
- executable scripts, making non-executable, 42
- exit command, 168, 184, 336
- exit on failure, ensuring, 190
- expansion operations, 297–311. *See also*
 - parameter expansion
 - arithmetic expansion, 306
 - brace expansion, 298–299
 - command substitution, 305–306
 - pathname expansion, 309–310
 - tilde expansion, 299
 - word splitting, 306–309
- export keyword, 123, 124
- expressions
 - in case statements, 148
 - grouping parts of, 36–37
 - in sed, 85
- extended regular expressions, 50

- extracting
 - information with sed, 91
 - text, first and last part of file, 71–73

F

- failure, ensuring exit on, 190
- fast-forward merges in Git, 260
- fetching changes in GitHub, 281–284
- file command, 402
- file descriptor, 16
- filepaths, differentiating from
 - commands, 25
- files. *See also* finding files and folders; streams
 - adding content to, 22
 - appending to, 22, 26
 - with changed permissions, finding, 42
 - configuration files, sourcing
 - additional, 212–213
 - copying, 404–405
 - creating, 348–350, 408–409
 - deleting, 403
 - downloading, 400–402
 - extracting text from, 71–73
 - looping through with for loop, 160–161
 - moving, 405–407
 - redirecting to, 22, 25, 380
 - renaming, 405–407
 - in repositories, 267–270
 - running operations on set of, 160–161
 - sample, xxx
 - searching through, 67–68
 - showing text content, 410–411
 - startup files
 - customizing, 209–214
 - default, 208–209
 - loading, 216–219
 - template, creating with sed, 96
 - transferring with scp, 339–341
 - unzipping, 402–403
 - using as input, 19–20
 - writing to, 25
 - zipping, 411
- filesystem, navigating, 393–400
 - changing directory, 395–396
 - with dot and double-dot folders, 397–398

- going back to previous directory, 398–399
 - identifying working directory, 394
 - listing contents of working directory, 394–395
 - pushing and popping working directory, 399–400
 - returning to home directory, 396
 - using absolute and relative paths, 397
 - filtering
 - search results using `grep`, 68–69
 - standard input, 21
 - `find` command, 39, 306
 - extra options, 41–43
 - searching by file or folder name, 34–35
 - searching by path, 35
 - searching for only files or folders, 33
 - searching with, 31–32
 - finding files and folders, 31–43
 - acting on search results, 38–40
 - excluding search results with `NOT` operator, 38
 - grouping parts of expression, 36–37
 - handling symbolic links, 40–41
 - large files, 41
 - recently modified files, 42
 - running case-insensitive searches, 36
 - specifying multiple search options, 35–36
 - flags, use of term, 396
 - folders. *See also* finding files and folders
 - copying text from to another location, 104–105
 - dot files
 - configuring options for working with folders, 243–244
 - creating folders for, 241
 - empty, finding and removing, 43
 - looping through with `for` loop, 160–161
 - managing in shell, 400–412
 - creating, 407–408
 - deleting, 409–410
 - viewing directory tree, 404
 - working with wildcards, 409
 - navigating with dot and double-dot folders, 397–398
 - running operations on set of, 160–161
 - searching for, 33–35
 - forking in Git, 286–287, 293
 - `for` loop, 156–164
 - iterating with C-style loops, 163–164
 - looping over sequences, 164–165
 - looping through arrays, 157
 - looping through files and folders, 160–161
 - looping through `find` command results, 162–163
 - splitting loop input into words, 157–160
 - wildcards in, 157, 160–161
 - `for` statement, 170
 - forward slash (/), 78, 87, 264, 304
 - function keyword, 174
 - functions, 173–188. *See also* commands
 - adding to startup file of interactive shells, 210
 - checking for existing, 192–193
 - creating, 174–176
 - error handling, 185–187
 - making more flexible, 177
 - to open pull request, 288–289
 - passing parameters to, 177–180
 - return values, 180–184
 - simplifying common command with, 187–188
 - unneeded, cleaning up, 193
- ## G
- `getopts` command, 195, 196
 - Git, 249–273
 - adding and resetting changes to index, 251–254
 - branches, 257–262
 - creating, 257–260
 - diverged, 261–262
 - performing fast-forward merges, 260
 - performing recursive merges, 261–262
 - commands, 272–273, 294, 297
 - `git add`, 252, 256, 273, 294
 - `git branch`, 257, 273
 - `git checkout`, 251, 257, 258, 259, 269, 270, 294
 - `git clean`, 292
 - `git clone`, 294
 - `git commit`, 255, 256, 261, 268, 273, 294
 - `git diff`, 292

Git (*continued*)

commands (*continued*)

- git fetch, 280, 281, 283–284, 294
- git init, 67, 251, 273, 294
- git log, 262, 273
- git merge, 260, 294
- git mv, 270
- git pull, 280, 284, 294
- git push, 280–281, 289, 294
- git remote, 280–281
- git reset, 253, 269, 273, 294
- git rev-list, 291
- git rm, 268, 273
- git status, 251, 255, 257–258, 268, 273, 292
- committing changes, 254–257
- creating repositories, 250–251
- definition of, 250
- log, 262–264
- managing files in repositories, 267–270
- remote repositories, 275–294
 - collaborating on, 286–288
 - key concepts and commands, 293–294
 - sharing dot files, 285–286
 - showing Git information in
 - command prompt, 290–291
 - writing shell function to open pull request, 288–289
- resolving conflicts, 264–267
- restoring working tree, 270–272
- workflows in, 267

GitHub, 276–285

- changes
 - fetching, 281–284
 - pulling, 284–285
 - pushing, 280–281
- creating repositories, 276–280
- forking with, 286–287
- making pull requests, 287–288
- Vim cheatsheet in, 360

globally scoped variables, 175

globs, 149, 161

GNOME Terminal, 413

GNU Screen multiplexer, 365

Go programming language, 314

Graham-Cumming, John, 58

graphical user interface (GUI), xxiv, 393–395

greedy regexes, 57, 58

grep command, 61–70

- advanced features, 65–69
 - combining with other commands, 69
 - filtering and piping of commands, 68–69
 - getting additional context for search results, 66–67
 - making search case-insensitive, 66
 - searching through multiple files, 67–68
- alternatives to, 69–70
- definition, 62–63
- origin of name, 62
- pipelines and, 69
- recursive searching using, 68
- searching through text, 63–64
- using with regular expressions, 64–65

GUI (graphical user interface), xxiv, 393–395

H

hash mark (#), 29, 87, 111, 348

- in command prompt, 222
- parameter variable, 178

head command, 21, 73

help, getting, 415–420

- accessing online cheat sheets, 420
- summarizing output with `tlldr`, 418–419
- using manual, 415–418

--help option, 323

highlighting of syntax, 197–199

HISTFILE variable, 10, 62, 64, 154

history

- rerunning commands in, 10
- searching using `grep`, 63
- setting options in dot files, 244
- using with shell scripts, 112–113

history command, 64

Hogan, Brian, 375

Homebrew, 390–391, 419

home directory, returning to, 396

HOME variable, 122, 299

host alias, 337

HostName setting, 337

hosts, SSH configuring, 337–338

hunks, interactive staging for, 292
hyphen. *See* dash

I

IdentityFile setting, 337
IDEs (integrated development environments), 344
if command, 146
if...else statement, 193
IFS (internal field separator) variable, 162, 163, 307, 308
if statement, 142, 146, 148–149, 154, 184, 187
index
 in Git, 251–254, 293
 retrieving element after defining array, 128
indirection, 132, 192
infinite loop, 168
input, standard, 18–21
 clipboard content as, 20
 files as, 19–20
 filtering as, 21
 output from shell code as, 19
input, user
 hiding, 134
 limiting, 134–135
 prompting for, 133–134
 reading, 132–133
 into custom variable, 133
input-process-output (IPO) pattern, 14–16
insert mode in Vim, 346
installed programs, checking for, 200
integrated development environments (IDEs), 344
interactive shells, 207–214, 218
 default shell startup file, 208–209
 distinction between login shells and, 217
 startup file customizations, 209–214
 adding functions, 210
 configuring environment variables, 212
 creating local *bin* directory for custom commands, 210–211
 pitfalls, 213–214
 shell options, 211–212

 shell startup commands, 213
 sourcing additional configuration files, 212–213
 using aliases, 209–210
interactive staging (Git), 292
interrupt signal, 412
IPO (input-process-output) pattern, 14–16
iterate expression (i), 164

J

jail, 240
Java, 315
JavaScript, 314

K

keyboard
 input from, 16
 shortcuts
 configuring in dot files, 243
 in Vim, 346
keys, 130
 creating pair, 328–329
 dealing with permission errors, 336–337
key-value format, 92
Konsole, 413

L

large files, finding, 41
large language models (LLMs), 380
lazy regexes, 57
less program, 21–22, 80, 122, 416
let keyword, 136
level of indirection, 192
lexical scoping variables, 175
line mode editors, 242
lines
 combining statements on single, 145–146
 counting number of, 19
 deleting, 5–6
 input, customizing processing of in *xargs*, 102–103
 looping through in files, 166–167
 moving to beginning or end of, 4
 removing parts of with *sed*, 88
 replacing text on with *sed*, 86–87
 of text, removing duplicate, 79–80

- Linux
 - clipboard essentials, 413
 - creating custom commands, 414
 - shell
 - accessing, 382
 - configuring, 392
- LLMs (large language models), 380
- ln (create link) command, 119
- local keyword, 175, 176
- locally scoped variables, 175
- log, Git, 262–264
- logging errors, 25
- logic, conditional. *See* conditional logic
- login shells, 216–217
- lookaround, 58
- lookbehind, 58
- lookup command, 325
- loops, 155–172
 - adding to common command, 171–172
 - compact, creating, 170–171
 - continue and break statements, 169–170
 - for, 156–164
 - iterating with C-style loops, 163–164
 - looping over sequences, 164–165
 - looping through arrays, 157
 - looping through files and folders, 160–161
 - looping through find command results, 162–163
 - splitting loop input into words, 157–160
 - while, 165–169
- lowercase
 - converting parameter value to, 305
 - searching files regardless of case
 - using grep, 66
 - transforming variable value to, 132
 - translating to uppercase, 23
- lowercase function, 181–182
- lower character class, 75
- ls command, 157, 228, 394, 397, 402

M

- macOS
 - accessing shell, 382
 - appearance of shell in, xxiv
 - clipboard essentials, 413
 - configuring shell, 390–392

- main branch, 257–264, 266, 281, 285, 291
- man command, 64, 415
- manual pages (man pages), 325, 415–418
 - builtins, 416–417
 - man page titles and summaries, 418
 - pager, 416
 - sections, 417–418
- Markdown styling tips website, 360
- menu, showing in bash, 200–201
- merges (Git)
 - fast-forward, 260
 - recursive, 261–262
- metacharacters. *See* character sets and metacharacters
- Microsoft Windows
 - accessing shell, 381–382
 - appearance of shell in, xxiv
 - clipboard essentials, 413
 - configuring shell, 383–392
 - installing Linux tools, 383–384
 - running virtual machine, 384–389
 - setting up Windows Subsystem for Linux, 389–390
 - creating custom commands, 413
- mkdir command, 7, 23–26, 407–408
- modal editing, 345
- Modern Vim* (Neil), 361
- modified files, finding, 42
- motions in Vim, 350, 356–357
- mouse support, enabling in tmux, 371
- multiplexer, 363–375. *See also* tmux
 - benefits of, 365
 - session management, 368–369
 - window management, 367
- mv command, 157, 406

N

- nano editor, 242
- navigation techniques using cursor, 4–6
 - deleting lines, 5–6
 - deleting words, 5
 - moving back or forward one word, 4–6
 - moving to beginning or end of line, 4
 - undoing changes, 6
- Neil, Drew, 361

- nested sessions, sending commands to
 - in tmux, 371
- nesting of if statements, 154
- network connectivity, losing, 339
- newlines
 - character for, 74
 - when looping through lines in a
 - file, 167
 - word splitting and, 162
- Node.js, 97, 314
- non-interactive shells, 214–216
- normal mode in Vim, 346
- NOT operator, to exclude search
 - results, 38
- npm (Node Package Manager)
 - program, 378
- null, redirecting errors to, 25–26
- nullglob command, 161
- numbers, sequences of, 298–299

O

- offset, specifying in parameter
 - expansion, 302
- Oh-My-Zsh project, 233–234
- ok action, 40
- OpenSSH, 328
- operating system, checking,
 - 199–100
- operations
 - running on set of files or folders,
 - 160–161
 - in Vim, 354–355
- operators in test command, 143–144
- options
 - common shell options, 212
 - use of term, 396
- option strings, 195
- or (||) operator, 145, 150, 152,
 - 167, 190
- OR expressions, 36
- output
 - avoiding printing during shell
 - startup, 213
 - of commands
 - in functions, 182–183
 - storing in variables, 127–128
 - printing during shell startup, 213
 - from shell code as input, 19

- standard
 - appending to files, 22
 - displaying onscreen, 21–22
 - redirecting standard error to,
 - 24–25
 - redirecting to files, 22
 - writing results of function to,
 - 181–182
- storing, 22
- summarizing with tldr, 418–419

P

- package manager, 390
- pager
 - paging through text, 80–81
 - scrolling through man pages, 416
- PAGER variable, 122, 212
- Panes (Tmux), 367
- parameter expansion, 299–300
 - arrays, expanding, 303
 - converting to lowercase or
 - uppercase, 305
 - default values, 300–301
 - displaying error if value is null or
 - unset, 301
 - finding length of parameter or
 - array, 303
 - patterns, removing/replacing, 304
 - specifying offset and length, 302
 - using alternate value, 301
 - using parameter indirection, 305
 - variable names, expanding,
 - 302–303
- parameters
 - available, checking for, 416
 - for commands, organizing with
 - xargs, 103–105
 - expanding, 130–132
 - of find command, 38
 - passing to functions, 177–180
 - passing to scripts, 184
 - processing complex script, 195–196
 - term usage, 396
 - variables of, common, 177–178
- parentheses
 - in command prompt, 225
 - grouping parts of expression using,
 - 36–37

- passwords, using SSH keys instead of, 279
- pasting with `pbpaste`, 414–415
- patch staging (Git), 292
- pathname expansion, 126, 160, 298
- paths
 - absolute and relative, 397
 - pathname expansion, 309–310
 - searching by, 35
 - in search results, printing, 39
- `PATH` variable, 106, 116, 119, 210, 216, 217, 218, 308
- patterns
 - in parameter expansion, 304–305
 - supplying, 34
- `pbcopy` command, 413, 414–415
- `pbpaste` command, 413, 414–415
- PEM (Privacy Enhanced Mail), 328
- percent sign (%), 6, 367
- period (.), 304
- Perl, 315
- permissions, changed, 42
- `pipefail` option, 190
- pipelines, 13–31
 - commands in, 113–115
 - composing for `grep` commands, 68–69
 - example of, 17–18
 - exiting script when commands fail, 190
 - filtering input, 21
 - `grep` command and, 69
 - input-process-output (IPO) pattern, 14–16
 - redirection with both `stdout` and `stderr`, 26–27
 - in standard error applications, 23–26
 - in standard input applications, 18–21
 - in standard output applications, 21–22
 - T-pipe, 27–28
 - and Unix philosophy, 28–29
 - using with copying and pasting, 414–415
- placeholders in `xargs` examples and `man` pages, 105
- playground folder, 400
- plus sign (+), 51, 73, 88, 135, 192, 229
- `popd` command, 399–400
- portability across systems, 314

- Practical Vim* (Neil), 361
- prepending text with `sed`, 90–91
- `-print` (print to `stdout`) action, 102
- `-print` action, 39
- `-print0` action, 102
- `printf` command, 197, 214
- printing output during shell startup, avoiding, 213
- Privacy Enhanced Mail (PEM), 328
- profile loading, differences in
 - depending on operating systems, 218
- Pro Git* (Chacon, Straub), 292
- programming languages
 - alternatives to `sed`, 97
 - choosing, 314–315
- programs
 - connecting. *See* pipelines
 - installed, checking for, 200
- prompting for input, 133–134
- prompt string, customizing, 223
- prompt variables, 229–231
 - `PROMPT_COMMAND`, 229, 231
 - `PROMPT_DIRTRIM`, 229, 230–231
 - `PS1`, 222–225, 227–229, 232, 233–234, 240
 - `PS2`, 229–230
 - `PS3`, 229, 230
 - `PS4`, 229, 230
- `pstree` command, 118
- pulling changes from GitHub, 284–285
 - making pull requests, 287–288
 - writing shell function to open pull requests, 288–289
- pushing
 - changes to GitHub, 280–281
 - working directory, 398–400
- `pwd` (print working directory) command, 394
- Python, 97, 314, 315
 - dictionary lookup tool in
 - improving, 325
 - installing, 324
 - writing, 316–323

Q

- `q!` (quit without saving) command, 347
- quantifiers in regex, 51–52

- question mark (?), 57, 126
 - searching for files or folders
 - using, 35
 - as wildcard, 310
- quotes in variable syntax, 124–126

R

- randomart, 329
- RANDOM variable, 166
- rc (run commands), 208
- read command, 131, 133, 134, 135, 167
- rebasing (Git), 292
- recursive merges (Git), 261–262
- redirecting
 - with both stdout and stderr, 26–27
 - in standard error applications, 24–26
 - standard output to file, 22
 - symbol (>), 24, 25, 27
- redirection operator, 411
- regexes (regular expressions), 47–59
 - advanced concepts, 57–59
 - to avoid backtracking, 59
 - basic, 65
 - breaking up into smaller parts, 56
 - building, 48–57, 93
 - anchors, 55–56
 - capture groups, 56–57
 - character sets and metacharacters, 52–57
 - lazy and greedy expressions, 57
 - quantifiers, 51–52
 - regex engines, 50–52
 - complexity of, 48
 - conditional expressions and, 150–151
 - continuing until finds no further matches, 57
 - edge cases and, 51
 - for email validation, 48–49
 - engines, 50–52
 - escaping characters in, 55
 - extended, 65
 - identifying greedy matches, 58
 - matching patterns of text at certain points on a line, 55–56
 - online, 93
 - overview, 48–52
 - Regular Expressions 101 website, 49–50, 51, 58, 59

- stopping search as soon as finds match, 57
- testing, 49–50
- using grep with, 64–65
- using in different languages, 59
- using with sed, 85–94
- relative paths, using, 397
- remotes, Git, 276
- REPLY variable, 133
- repositories, Git, 293
 - creating, 250–251, 276–280
 - managing files, 267–270
 - deleting, 267–268
 - restoring and renaming, 268–270
- requests module, 321
- resources, online, xxx
- restructuring text with sed, 92–95
- return values of functions, 180–184
 - avoiding pitfalls with command output, 182–183
 - returning status codes, 183–184
 - writing to standard output, 181–182
- rev (reverse) command, 78
- reversing text, 78
- rev tool, 315
- right angle bracket (>), 113, 229
- ripgrep tool, 70
- rm command, 40, 100, 101, 103, 190, 209, 403
- rmdir command, 85–88, 409–410
- root of filesystem, 397
- Ruby, 314
- Rust, 315

S

- sample files, xxx
- sandbox, Linux, 384
- Schneier, Bruce, 328
- scp program, transferring files with, 339–341
- screen, clearing, 9
- scripting alternatives, 313–325
 - characteristics of shell-friendly tools, 315–316
 - choosing programming language, 314–315

- scripting alternatives (*continued*)
 - dictionary lookup tool, writing in
 - Python, 316–323
 - defining basic structure, 317–319
 - downloading definition, 319–321
 - formatting output, 321–323
 - whether to use, 314
- scripts, xxiv, 109–120
 - AI-generated, executing, 379–380
 - behavior of, 214–216
 - benefits of, 110–115
 - creating, 110–118
 - adding and formatting
 - commands, 112–113
 - adding code comments, 111
 - making shell scripts executable, 115
 - pipelining commands, 113–115
 - sourcing shell scripts, 117–118
 - specifying program to run script, 116–117
 - customizing command prompt, 231–235
 - executable scripts, 42
 - for installing dot files, 247–248
 - installing locally, 118–119
 - patterns for, 189–204
 - adding syntax highlighting, 197–199
 - anti-patterns, 201–203
 - checking for existing variables or functions, 192–193
 - checking for installed programs, 200
 - checking operating system, 199–100
 - debugging with trace option, 191–192
 - ensuring exit on failure, 190
 - processing complex parameters, 195–196
 - running commands in
 - subshells, 201
 - showing menu, 200–201
 - trapping signals and events, 193–195
 - unsetting values, 193
 - tidying up, 302–303
 - using history file with, 112–113
- search expression, 33
- searching
 - through command history, 6–7
 - with `find` command
 - case-insensitive, 36
 - by file or folder name, 34–35
 - introduction to, 31–32
 - for only files or folders, 33
 - by path, 35
 - with `grep`
 - case-insensitive, 66
 - through multiple files, 67–68
 - through text, 63–64
 - by path, 35
 - specifying multiple options, 35–36
- search motions in Vim, 355–356
- search results
 - acting on, 38–40
 - excluding with NOT operator, 38
 - getting additional context with `grep`, 66–67
- Secure Hash Algorithm (SHA), 262–263, 270
- secure shell. *See* SSH
- `sed` command, 19, 83–98, 114, 134, 315, 316
 - advanced applications
 - creating template files, 96
 - editing in place, 96–97
 - restructuring text, 92–95
 - alternatives to, 97
 - text manipulation with, 83–98
 - transformations with
 - appending text, 89–90
 - applying multiple expressions, 85–88
 - extracting information, 91
 - prepending text, 90–91
 - replacing text, 84–85
 - stripping comments, 89
 - using addresses in `sed` functions, 87
 - using regexes with, 85–94
- `select` command, 200, 229
- semicolon (;), 39, 145–146, 152, 308, 356
- sequences
 - looping over, 164–165
 - of numbers or characters, creating, 298–299
 - in Vim, 346

- ul style="list-style-type: none;">
- session management with tmux, 365, 368–369
- set (set option) command, 186
- SHA (Secure Hash Algorithm), 262–263, 270
- sh command, 114, 123
- shebang (#!), 116–117
 - configuring options in, 203
 - to locate python3 program, 324
 - omitting, 202
- shell. *See also* basics of shell; configuring shell
 - accessing, 381–382
 - configuration cheat sheet, 218
 - future of, 380
 - idioms, handling errors using, 316
 - overview, xxiv–xxv
 - reasons for using, xxiv
- shell-friendly tools, characteristics of, 315–316
- shell options (\$-) parameter, 242
- SHELL variable, 122, 154, 219, 336
- shifting parameters, 179–180
- shopt command, 161, 209
- shortcuts. *See also* symbolic links (symlinks)
 - clear screen, 9
 - for maneuvering cursor, 4–6
 - show all, 10
 - transpose text, 10
 - view command history, 9–10
- signals, trapping, 193–195
- single quote ('), 124–125, 126
- sleep command, 225
- sort command, 17, 28, 79, 113, 114
- sorting text, 79–80
- sourcing shell scripts, 117–118
- spaces. *See* whitespace
- sparse arrays, 129
- special characters
 - escaping, 37
 - handling in xargs, 101–102
- splitting
 - loop input into words, 157–160
 - streams, 27–28
 - text into words, 126, 159–160, 298, 306–309
 - in txux, 367
- square brackets ([]), 75
 - in command prompt, 225
 - in conditional expressions, 150
- squashing (Git), 292
- SSH, 327–341
 - configuring hosts, 337–338
 - connecting to virtual machine, 335–336
 - creating a key pair, 328–329
 - creating virtual machine on AWS, 331–335
 - definition of, 327–328
 - handling disconnections, 339
 - key permission errors, dealing with, 336–337
 - keys, 279
 - running commands, 338
 - setting up AWS account, 329–330
 - transferring files with scp, 339–341
- ssh-keygen OpenSSH authentication key utility, 328
- ssh program, 336, 338, 339, 372
- standard error, 23–26
 - appending to file, 26
 - redirecting, 24–26
 - writing to file, 25
- standard input, 18–21
 - clipboard content, 20
 - files, 19–20
 - filtering input, 21
 - output from shell code as, 19
 - reading from, 315
- standard output, 21–22
 - appending to file, 22
 - displaying onscreen, 21–22
 - redirecting standard error to, 24–25
 - redirecting to file, 22
 - writing to, 315
- startup commands, 213–214
- startup files
 - for default shell, 208–209
 - interactive shells, 209–214
 - adding functions, 210
 - configuring environment variables, 212
 - creating local bin directory for custom commands, 210–211
 - pitfalls to avoid, 213–214
 - setting shell options, 211–212

- startup files (*continued*)
 - interactive shells (*continued*)
 - setting shell startup commands, 213
 - sourcing additional configuration files, 212–213
 - using aliases, 209–210
 - loading for login shell, 217–219
 - loading with BASH_ENV, non-interactive shells, 216
- statements, combining on single line, 145–146
- status codes, returning in functions, 183–184
- stderr stream
 - input-process-output (IPO) pattern, 15–16
 - and stdout, redirection with both, 26–27
- stdin stream, 15–16
- stdout stream
 - input-process-output (IPO) pattern, 15–16
 - and stderr, redirection with both, 26–27
- storing output, 22
- Storti, Brian, 27
- Straub, Ben, 292
- stream redirection operator, 20
- streams
 - input-process-output (IPO) pattern, 15
 - splitting in two, 27–28
- stripping comments with sed, 89
- subject line of commit messages, 257
- subshells, 123, 201
- substring, returning, 131
- sudo (run command as superuser)
 - command, 119
- summaries of man pages, 418
- symbolic links (symlinks), 40–41, 119
- syntax
 - highlighting, adding, 197–199
 - of variables, 124–127
 - using braces, 127
 - using quotes, 124–126

T

- tabs, word splitting and, 162
- tags, 292
- tail command, 72–73, 113, 154
- template files, creating with sed, 96
- Terminal AI, 377–380
 - chatting with ChatGPT, 378–379
 - copying or saving results, 379
 - executing AI-generated scripts, 379–380
 - installing and running, 378
 - redirecting responses to files, 380
 - shell of future, 380
- terminal editors, 343–361. *See also* Vim
 - adding command count, 352
 - editing commands, 358–359
 - inserting text at specific positions, 352–354
 - navigating through text, 350–352
 - operating on range of text, 354–355
 - reasons to use, 344
 - searching for text patterns, 355–358
- test command, 143–145, 161
 - advantages of conditional expressions over, 150
 - checking multiple conditions simultaneously, 145
 - operators for expressions and files, 143–144
- text
 - appending with sed, 89–90
 - displayed in shell, coloring, 197–199
 - formatting, adding to command prompt, 225–228
 - inserting at specific positions using Vim, 352–354
 - navigating through using Vim, 350–352
 - operating on range of using Vim, 354–355
 - patterns, searching for using Vim, 355–358
 - prepending with sed, 90–91
 - replacing with sed, 84–85
 - on specific lines, 86–87
 - restructuring with sed, 92–95
 - searching through using grep, 63–64
 - showing content of files, 410–411
 - splitting into words, 298
 - transposing shortcut, 10
- text editor. *See* terminal editors

- text manipulation, 71–81. *See also*
 - sed command
 - cutting text, 75–78
 - extracting first and last part of file, 71–73
 - paging through text, 8–81
 - replacing text, 74–75
 - reversing text, 78
 - with sed, 83–98
 - sorting text and removing duplicate lines, 79–80
- tilde (~), 88, 125–126, 272, 298, 299, 396
- tlldr tool, 62, 415, 418–419
- tmux
 - collaboration with, 372–373
 - commands in, 373–374
 - configuring, 369–372
 - installing, 365–366
 - keyboard shortcuts, 366
 - quick guide, 373–374
 - session management with, 368–369
 - session persistence, 365
 - window management with, 367
- tmux 3: Productive Mouse-Free Development* (Hogan), 375
- Torvalds, Linus, 250
- touch command, 408
- T-pipe, 27–28
- tput command, 199
- trace option, debugging with, 191–192
- tracing, handling in xargs, 101–102
- transposing text, shortcuts for, 10
- trap command, 193
- trapping signals and events, 193–195
- tr command, 23–25, 74, 181
- tree, working, 270–272
- tree command, 258–259, 407
- Typer package, 325

U

- Ubuntu's Desktop Edition, 384
- uname command, 199
- undoing changes, 6
- uniq (omit duplicate lines) command, 17, 27–28, 79, 113
- unset command, 193
- unsettling values, 193
- until loop, 168–169

- unzip command, 402, 411
- unzipping files, 402–403
- uppercase
 - converting parameter value to, 305
 - searching files regardless of case, 66
 - transforming variable value to, 132
 - translating lowercase characters to, 23
- upper character class, 75
- urllib library, 321
- user input. *See* input, user
- User setting, 337
- USER variable, 122

V

- values variable, 179
- variables, 121–139
 - checking for existing, 192–193
 - cleaning up in dot files, 243
 - common operations, 127–137
 - arrays, 128–129
 - associative arrays, 130
 - expanding shell parameters, 130–132
 - performing arithmetic operations, 135–137
 - storing command's output in variables, 127–128
 - user input, reading and storing in variables, 132–135
 - common parameter values, 177–178
 - configuring environment variables, 212
 - enhancing common command with variables, 137–138
 - environment variables, 123–124
 - expanding names of, 302–303
 - in functions, 174–175
 - indirection, 132
 - returning default value of, 131
 - returning length of, 130–131
 - scope of
 - environment vs. shell variables, 122–123
 - in functions, 175–176
 - syntax, 124–127
 - transforming value of to uppercase/lowercase, 132
 - unnneeded, cleaning up, 193

- verbatim insert command, 412
- version control system. *See* Git
- Vim, 345–350, 360–361
 - adding command count, 352
 - buffers, 346, 348
 - case sensitivity in, 345
 - configuring tmux to interface with, 372
 - editing commands, 358–359
 - “enter insert mode”
 - commands, 353
 - keyboard shortcuts in, 345
 - motions in, 350, 356–357
 - operators in, 354–355
 - text
 - inserting at specific positions, 352–354
 - navigating through, 350–352
 - operating on range of, 354–355
 - searching for patterns in, 355–358
 - updating and styling cheat sheet, 359–360
- Vimcasts website, 361
- vimtutor program, 361
- VirtualBox, 384–385
- virtual machine
 - creating on AWS, 331–335
 - shutting down, 335
 - using SSH to connect to, 335–336
- VISUAL variable, 242

W

- website-and-repo tab of tmux
 - multiplexer, 364
- wget command, 190, 401–402
- whatis command, 418
- while loop, 165–169, 195
 - looping forever, 168
 - looping through lines in files, 166–167
 - until loop, 168–169
- while statement, 170
- whitespace
 - handling in xargs, 101–102
 - when looping through lines in a file, 167
- word splitting and, 162

- whoami command, 389
- wildcards, 409
 - character (*), 16, 34, 49, 124, 264, 304, 359, 402
 - in for loops, 157, 160–161
 - in pathnames, expanding, 298
 - searching for files or folders using, 34
- windows
 - creating and moving between, 367
 - managing using multiplexer, 365
 - naming and numbering, 370–371
 - splitting, 371
 - zooming panes, 367
- Windows, Microsoft. *See* Microsoft Windows
- Windows Subsystem for Linux (WSL), 383, 389–390, 413
- words
 - counting number of, 19
 - deleting, 5
 - moving back or forward one word, 4–6
 - splitting text into, 126, 159–160, 298, 306–309
 - in Vim, 352
- working directory
 - identifying, 394
 - listing contents of, 394–395
 - pushing and popping, 399–400
 - specifying in tmux
 - configuration, 370
- working tree in Git, 251, 270–272, 293
- wq (write and quit) command, 359
- WSL (Windows Subsystem for Linux), 383, 389–390, 413

X

- xargs command
 - braces used as placeholders, 105
 - handling whitespace, special characters, and tracing, 101–102
 - input lines, 102–103
 - organizing parameters for commands, 103–105
 - overview, 99–101
 - running commands, 105–106
- xclip command, 20

Z

ZIP file

- containing files found, 42

- unzipping, 402

- zipping, 411

Z shell, 233–234, 390

zsh_regex variable, 150