How to do the most with the least

Minimal Perl

for UNIX and Linux People

Tim Maher

MANNING

contents

foreword xvii
preface xix
acknowledgments xxii
about this book xxiii
about the cover illustration xxxiv
list of tables xxxv

Part 1 Minimal Perl: for UNIX and Linux Users 1

1 Introducing Minimal Perl 3

1.1 A visit to Perlistan 3
Sometimes you need a professional guide 5

1.2 Perl can be simple 7

1.3 About Minimal Perl 7
What Minimal Perl isn't 8 ✦ What Minimal Perl is 8

1.4 Laziness is a virtue 9

1.5 A minimal dose of syntax 10
Terminating statements with semicolons 10

1.6 Writing one-line programs 11
Balancing simplicity and readability 12
Implementing simple filters 12

1.7 Summary 14

2 Perl essentials 16

2.1 Perl's invocation options 17
One-line programming: `-e` 18 ✦ Enabling warnings: `-w` 18
Processing input: `-n` 19 ✦ Processing input with automatic printing: `-p` 19 ✦ Processing line-endings: `-l` 20 ✦ Printing without newlines: `printf` 21 ✦ Changing the input record separator: `-0`*`digits`* 22

2.2 Using variables 23
Using special variables 23 ✦ Using the data variable: $_ 24
Using the record-number variable: $. 24 ✦ Employing user-defined variables 25

2.3 Loading modules -M 27

2.4 Writing simple scripts 29
Quoting techniques 30 ✦ True and False values 32
Handling switches: -s 32 ✦ Using warn and die 35
Using logical and, logical or 37 ✦ Programming with BEGIN and END blocks 39 ✦ Loading modules with use 41

2.5 Additional special variables 42
Employing I/O variables 42 ✦ Exploiting formatting variables 43

2.6 Standard option clusters 44
Using aliases for common types of Perl commands 46

2.7 Constructing programs 47
Constructing an output-only one-liner 49 ✦ Constructing an input/output script 50

2.8 Summary 51
Directions for further study 51

3 Perl as a (better) grep command 53

3.1 A brief history of grep 53

3.2 Shortcomings of grep 54
Uncertain support for metacharacters 54 ✦ Lack of string escapes for control characters 56 ✦ Comparing capabilities of greppers and Perl 57

3.3 Working with the matching operator 60
The one-line Perl grepper 61

3.4 Understanding Perl's regex notation 63

3.5 Perl as a better fgrep 64

3.6 Displaying the match only, using $& 64

3.7 Displaying unmatched records (like grep -v) 65
Validating data 66 ✦ Minimizing typing with shortcut metacharacters 67

3.8 Displaying filenames only (like grep -l) 67

3.9 Using matching modifiers 68
Ignoring case (like grep -i) 70

3.10 Perl as a better `egrep` 70
Working with cascading filters 72
3.11 Matching in context 75
Paragraph mode 75 ✦ File mode 77
3.12 Spanning lines with regexes 77
Matching across lines 79 ✦ Using `lwp-request` 80
Filtering `lwp-request` output 80
3.13 Additional examples 81
Log-file analysis 81 ✦ A scripted grepper 84
Fuzzy matching 85 ✦ Web scraping 86
3.14 Summary 86
Directions for further study 88

4 Perl as a (better) sed command 89

4.1 A brief history of `sed` 89
4.2 Shortcomings of `sed` 91
4.3 Performing substitutions 93
Performing line-specific substitutions: `sed` 96 ✦ Performing line-specific substitutions: Perl 96 ✦ Performing record-specific substitutions: Perl 97 ✦ Using backreferences and numbered variables in substitutions 99
4.4 Printing lines by number 100
Printing lines by number: `sed` 100 ✦ Printing lines by number: Perl 100 ✦ Printing records by number: Perl 101
4.5 Modifying templates 101
4.6 Converting special characters 103
4.7 Editing files 105
Editing with commands 105 ✦ Editing with scripts 107
Safeguarding in-place editing 111
4.8 Converting to lowercase or uppercase 113
Quieting spam 113
4.9 Substitutions with computed replacements 114
Converting miles to kilometers 114 ✦ Substitutions using function results 116
4.10 The `sed` to Perl translator 118
4.11 Summary 118
Directions for further study 120

5 Perl as a (better) awk command 121

5.1 A brief history of AWK 122

5.2 Comparing basic features of `awk` and Perl 123
Pattern-matching capabilities 124 ✦ Special variables 126 Perl's variable interpolation 128 ✦ Other advantages of Perl over AWK 129 ✦ Summary of differences in basic features 129

5.3 Processing fields 130
Accessing fields 130 ✦ Printing fields 132 ✦ Differences in syntax for `print` 134 ✦ Using custom field separators in Perl 136

5.4 Programming with Patterns and Actions 138
Combining pattern matching with field processing 142 Extracting data from tables 143 ✦ Accessing cell data using array indexing 145

5.5 Matching ranges of records 151
Operators for single- and multi-record ranges 152 ✦ Matching a range of dates 153 ✦ Matching multiple ranges 155

5.6 Using relational and arithmetic operators 157
Relational operators 157 ✦ Arithmetic operators 158

5.7 Using built-in functions 159
One-liners that use functions 161 ✦ The legend of `nexpr` 162 How the `nexpr*` programs work 164

5.8 Additional examples 165
Computing compound interest: `compound_interest` 165 Conditionally pluralizing nouns: `compound_interest2` 166 Analyzing log files: `scan4oops` 168

5.9 Using the AWK-to-Perl translator: `a2p` 175
Tips on using `a2p` 175

5.10 Summary 175
Directions for further study 177

6 Perl as a (better) find command 178

6.1 Introducing hybrid `find/perl` programs 180

6.2 File testing capabilities of `find` vs. Perl 180
Augmenting `find` with Perl 183

6.3 Finding files 184
Finding files by name matching 184 ✦ Finding files by pathname matching 187

6.4 Processing filename arguments 188
Defending against `grep`'s messes 189 ✦ Recursive grepping 191
Perl as a generalized argument pre-processor 192
6.5 Using `find | xargs` vs. Perl alternatives 192
Using Perl for reliable timestamp sorting 193
Dealing with multi-word filenames 196
6.6 `find` as an argument pre-processor for Perl 197
6.7 A Unix-like, OS-portable `find` command 198
Making the most of `find2perl` 198 ✦ Helping non-Unix friends with `find2perl` 199
6.8 Summary 200
Directions for further study 201

Part 2 Minimal Perl: for UNIX and Linux Shell Programmers 203

7 Built-in functions 205

7.1 Understanding and managing evaluation context 206
Determinants and effects of evaluation context 207
Making use of evaluation context 208
7.2 Programming with functions that generate or process scalars 210
Using `split` 211 ✦ Using `localtime` 214 ✦ Using `stat` 215 ✦ Using `chomp` 219 ✦ Using `rand` 221
7.3 Programming with functions that process lists 223
Comparing Unix pipelines and Perl functions 223
Using `sort` 224 ✦ Using `grep` 227 ✦ Using `join` 229
Using `map` 232
7.4 Globbing for filenames 234
Tips on globbing 237
7.5 Managing files with functions 239
Handling multi-valued return codes 240
7.6 Parenthesizing function arguments 242
Controlling argument-gobbling functions 242
7.7 Summary 243
Directions for further study 245

8 Scripting techniques 247

8.1 Exploiting script-oriented functions 248
Defining `defined` 249 ✦ Exiting with `exit` 253
Shifting with `shift` 254

8.2 Pre-processing arguments 256
Accommodating non-filename arguments with implicit loops 256
Filtering arguments 257 ✦ Generating arguments 259

8.3 Executing code conditionally with `if/else` 259
Employing `if/else` vs. `and/or` 260 ✦ Mixing branching techniques: The `cd_report` script 261 ✦ Tips on using `if/else` 264

8.4 Wrangling strings with concatenation and repetition operators 265
Enhancing the `most_recent_file` script 267 ✦ Using concatenation and repetition operators together 267 ✦ Tips on using the concatenation operator 268

8.5 Interpolating command output into source code 269
Using the `tput` command 271 ✦ Grepping recursively: The `rgrep` script 273 ✦ Tips on using command interpolation 274

8.6 Executing OS commands using `system` 275
Generating reports 277 ✦ Tips on using `system` 280

8.7 Evaluating code using `eval` 283
Using a Perl shell: The `psh` script 284 ✦ Appreciating a multi-faceted Perl grepper: The `preg` script 286

8.8 Summary 292
Directions for further study 294

9 List variables 295

9.1 Using array variables 296
Initializing arrays with piecemeal assignments and `push` 299
Understanding advanced array indexing 300 ✦ Extracting fields in a friendlier fashion 301 ✦ Telling fortunes: The `fcookie` script 304 ✦ Tips on using arrays 308

9.2 Using hash variables 308
Initializing hashes 311 ✦ Understanding advanced hash indexing 312 ✦ Understanding the built-in `%ENV` hash 313 ✦ Printing hashes 314 ✦ Using `%ENV` in place of switches 315 ✦ Obtaining uniqueness with hashes 316 ✦ Employing a hash as a simple database: The `user_lookup` script 319 ✦ Counting word frequencies in web pages: The `count_words` script 323

9.3 Comparing list generators in the Shell and Perl 325
Filename generation/globbing 326 ✦ Command substitution/interpolation 327 ✦ Variable substitution/interpolation 327
9.4 Summary 328
Directions for further study 329

10 Looping facilities 330

10.1 Looping facilities in the Shell and Perl 331
10.2 Looping with `while/until` 333
Totaling numeric arguments 333 ✦ Reducing the size of an image 335 ✦ Printing key/value pairs from a hash using `each` 336 ✦ Understanding the implicit loop 337
10.3 Looping with `do while/until` 338
Prompting for input 339
10.4 Looping with `foreach` 340
Unlinking files: the `rm_files` script 341 ✦ Reading a line at a time 341 ✦ Printing a hash 342 ✦ Demystifying acronyms: The `expand_acronyms` script 343 ✦ Reducing image sizes: The `compress_image2` script 344
10.5 Looping with `for` 345
Exploiting `for`'s support for indexing: the `raffle` script 347
10.6 Using loop-control directives 349
Nesting loops within loops 350 ✦ Enabling loop-control directives in bottom-tested loops 351 ✦ Prompting for input 352 ✦ Enhancing loops with `continue` blocks: the `confirmation` script 353
10.7 The CPAN's `select` loop for Perl 355
Avoiding the re-invention of the "choose-from-a-menu" wheel 356
Monitoring user activity: the `show_user` script 357
Browsing man pages: the `perlman` script 358
10.8 Summary 360
Directions for further study 361

11 Subroutines and variable scoping 362

11.1 Compartmentalizing code with subroutines 363
Defining and using subroutines 365 ✦ Understanding `use strict` 368
11.2 Common problems with variables 370
Clobbering variables: The `phone_home` script 371 ✦ Masking variables: The `4letter_word` script 372 ✦ Tips on avoiding problems with variables 373

11.3 Controlling variable scoping 373
Declaring variables with `my` 374 ✦ Declaring variables with `our` 374 ✦ Declaring variables with `local` 375 ✦ Introducing the Variable Scoping Guidelines 375

11.4 Variable Scoping Guidelines for complex programs 376
Enable `use strict` 377 ✦ Declare user-defined variables and define their scopes 377 ✦ Pass data to subroutines using arguments 383 ✦ Localize temporary changes to built-in variables with `local` 383 ✦ Employ user-defined loop variables 383
Applying the Guidelines: the `phone_home2` script 384

11.5 Reusing a subroutine 386

11.6 Summary 387
Directions for further study 387

12 Modules and the CPAN 388

12.1 Creating modules 389
Using the Simple Module Template 390 ✦ Creating a module: `Center.pm` 393 ✦ Testing a new module 395

12.2 Managing modules 398
Identifying the modules that you want 398 ✦ Determining whether you have a certain module 400 ✦ Installing modules from the CPAN 401

12.3 Using modules 403
`Business::UPS`—the `ups_shipping_price` script 403
`LWP::Simple`—the `check_links` script 405
`Shell::POSIX::Select`—the `menu_ls` script 408
`File::Find`—the `check_symlinks` script 411
`CGI`—the `survey.cgi` script 414 ✦ Tips on using Object-Oriented modules 422

12.4 Summary 424
Directions for further study 425

epilogue 426
appendix A: Perl special variables cheatsheet 427
appendix B: Guidelines for parenthesizing code 430
glossary 432
index 443