Perl Cookbook(影印版)

2nd Edition

Perl Cookbook™

O'REILLY®
東南大學出版社

Tom Christiansen & Nathan Torkington 著

第二版

Perl Cookbook（影印版）

Perl Cookbook

om Christiansen &
Nathan Torkington

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

Foreword

They say that it's easy to get trapped by a metaphor. But some metaphors are so magnificent that you don't mind getting trapped in them. Perhaps the cooking metaphor is one such, at least in this case. The only problem I have with it is a personal one—I feel a bit like Betty Crocker's mother. The work in question is so monumental that anything I could say here would be either redundant or irrelevant.

However, that never stopped me before.

Cooking is perhaps the humblest of the arts; but to me humility is a strength, not a weakness. Great artists have always had to serve their artistic medium—great cooks just do so literally. And the more humble the medium, the more humble the artist must be in order to lift the medium beyond the mundane. Food and language are both humble media, consisting as they do of an overwhelming profusion of seemingly unrelated and unruly ingredients. And yet, in the hands of someone with a bit of creativity and discipline, things like potatoes, pasta, and Perl are the basis of works of art that "hit the spot" in a most satisfying way, not merely getting the job done, but doing so in a way that makes your journey through life a little more pleasant.

Cooking is also one of the oldest of the arts. Some modern artists would have you believe that so-called ephemeral art is a recent invention, but cooking has always been an ephemeral art. We can try to preserve our art, make it last a little longer, but even the food we bury with our pharoahs gets dug up eventually. So too, much of our Perl programming is ephemeral. This aspect of Perl cuisine has been much maligned. You can call it quick-and-dirty if you like, but there are billions of dollars out there riding on the supposition that fast food is not necessarily dirty food. (We hope.)

Easy things should be easy, and hard things should be possible. For every fast-food recipe, there are countless slow-food recipes. One of the advantages of living in California is that I have ready access to almost every national cuisine ever invented. But even within a given culture, There's More Than One Way To Do It. It's said in Russia that there are more recipes for borscht than there are cooks, and I believe it. My

mom's recipe doesn't even have any beets in it! But that's okay, and it's more than okay. Borscht is a cultural differentiator, and different cultures are interesting, and educational, and useful, and exciting.

So you won't always find Tom and Nat doing things in this book the way I would do them. Sometimes they don't even do things the same way as each other. That's okay—again, this is a strength, not a weakness. I have to confess that I learned quite a few things I didn't know before I read this book. What's more, I'm quite confident that I still don't know it all. And I hope I don't any time soon. I often talk about Perl culture as if it were a single, static entity, but there are in fact many healthy Perl subcultures, not to mention sub-subcultures and supercultures and circumcultures in every conceivable combination, all inheriting attributes and methods from each other. It can get confusing. Hey, I'm confused most of the time.

So the essence of a cookbook like this is not to cook for you (it can't), or even to teach you how to cook (though it helps), but rather to pass on various bits of culture that have been found useful, and perhaps to filter out other bits of "culture" that grew in the refrigerator when no one was looking. You in turn will pass on some of these ideas to other people, filtering them through your own experiences and tastes, your creativity and discipline. You'll come up with your own recipes to pass to your children. Just don't be surprised when they in turn cook up some recipes of their own, and ask you what you think. Try not to make a face.

I commend to you these recipes, over which I've made very few faces.

——Larry Wall
June, 1998

Preface

The investment group eyed the entrepreneur with caution, their expressions flickering from scepticism to intrigue and back again.

"Your bold plan holds promise," their spokesman conceded. "But it is costly and entirely speculative. Our mathematicians mistrust your figures. Why should we entrust our money into your hands? What do you know that we do not?"

"For one thing," he replied, "I know how to balance an egg on its point without outside support. Do you?" And with that, the entrepreneur reached into his satchel and delicately withdrew a fresh hen's egg. He handed over the egg to the financial tycoons, who passed it amongst themselves trying to carry out the simple task. At last they gave up. In exasperation they declared, "What you ask is impossible! No man can balance an egg on its point."

So the entrepreneur took back the egg from the annoyed businessmen and placed it upon the fine oak table, holding it so that its point faced down. Lightly but firmly, he pushed down on the egg with just enough force to crush in its bottom about half an inch. When he took his hand away, the egg stood there on its own, somewhat messy, but definitely balanced. "Was that impossible?" he asked.

"It's just a trick," cried the businessmen. "Once you know how, anyone can do it."

"True enough," came the retort. "But the same can be said for anything. Before you know how, it seems an impossibility. Once the way is revealed, it's so simple that you wonder why you never thought of it that way before. Let me show you that easy way, so others may easily follow. Will you trust me?"

Eventually convinced that this entrepreneur might possibly have something to show them, the skeptical venture capitalists funded his project. From the tiny Andalusian port of Palos de Moguer set forth the *Niña*, the *Pinta*, and the *Santa María*, led by an entrepreneur with a slightly broken egg and his own ideas: Christopher Columbus.

Many have since followed.

Approaching a programming problem can be like balancing Columbus's egg. If no one shows you how, you may sit forever perplexed, watching the egg—and your program—fall over again and again, no closer to the Indies than when you began. This is especially true in a language as idiomatic as Perl.

This book isn't meant to be a complete reference book for Perl. Keeping a copy of *Programming Perl* handy will let you look up exact definitions of operators, keywords, functions, pragmata, or modules. Alternatively, every Perl installation comes with a voluminous collection of searchable, online reference materials. If those aren't where you can easily get at them, see your system administrator if you have one, or consult the documentation section at *http://www.perl.com*.

Neither is this book meant to be a bare-bones introduction for programmers who have never seen Perl before. That's what *Learning Perl*, a kinder and gentler introduction to Perl, is designed for. (If you're on a Microsoft system, you might prefer the *Learning Perl for Win32 Systems* version.)

Instead, this is a book for learning *more* Perl. Neither a reference book nor a tutorial book, *Perl Cookbook* serves as a companion book to both. It's for people who already know the basics but are wondering how to mix all those ingredients together into a complete program. Spread across 22 chapters and more than 400 focused topic areas affectionately called recipes, this task-oriented book contains thousands of solutions to everyday challenges encountered by novice and journeyman alike.

We tried hard to make this book useful for both random and sequential access. Each recipe is self-contained, but has a list of references at the end should you need further information on the topic. We've tried to put the simpler, more common recipes toward the front of each chapter and the simpler chapters toward the front of the book. Perl novices should find that these recipes about Perl's basic data types and operators are just what they're looking for. We gradually work our way through topic areas and solutions more geared toward the journeyman Perl programmer. Now and then we include material that should inspire even the master Perl programmer.

Each chapter begins with an overview of that chapter's topic. This introduction is followed by the main body of each chapter, its recipes. In the spirit of the Perl slogan of TMTOWTDI, *There's more than one way to do it*, most recipes show several different techniques for solving the same or closely related problems. These recipes range from short-but-sweet solutions to in-depth mini-tutorials. Where more than one technique is given, we often show costs and benefits of each approach.

As with a traditional cookbook, we expect you to access this book more or less at random. When you want to learn how to do something, you'll look up its recipe. Even if the exact solutions presented don't fit your problem exactly, they'll give you ideas about possible approaches.

Each chapter concludes with one or more complete programs. Although some recipes already include small programs, these longer applications highlight the chapter's principal focus and combine techniques from other chapters, just as any real-world program would. All are useful, and many are used on a daily basis. Some even helped us put this book together.

What's in This Book

Spread over five chapters, the first portion of the book addresses Perl's basic data types. Chapter 1, *Strings*, covers matters like accessing substrings, expanding function calls in strings, and parsing comma-separated data; it also covers Unicode strings. Chapter 2, *Numbers*, tackles oddities of floating-point representation, placing commas in numbers, and pseudo-random numbers. Chapter 3, *Dates and Times*, demonstrates conversions between numeric and string date formats and using timers. Chapter 4, *Arrays*, covers everything relating to list and array manipulation, including finding unique elements in a list, efficiently sorting lists, and randomizing them. Chapter 5, *Hashes*, concludes the basics with a demonstration of the most useful data type, the associative array. The chapter shows how to access a hash in insertion order, how to sort a hash by value, how to have multiple values per key, and how to have an immutable hash.

Chapter 6, *Pattern Matching*, includes recipes for converting a shell wildcard into a pattern, matching letters or words, matching multiple lines, avoiding greediness, matching nested or recursive patterns, and matching strings that are close to but not exactly what you're looking for. Although this chapter is one of the longest in the book, it could easily have been longer still—every chapter contains uses of regular expressions. It's part of what makes Perl Perl.

The next three chapters cover the filesystem. Chapter 7, *File Access*, shows opening files, locking them for concurrent access, modifying them in place, and storing filehandles in variables. Chapter 8, *File Contents*, discusses storing filehandles in variables, managing temporary files, watching the end of a growing file, reading a particular line from a file, handling alternative character encodings like Unicode and Microsoft character sets, and random access binary I/O. Finally, in Chapter 9, *Directories*, we show techniques to copy, move, or delete a file, manipulate a file's timestamps, and recursively process all files in a directory.

Chapters 10 through 13 focus on making your program flexible and powerful. Chapter 10, *Subroutines*, includes recipes on creating persistent local variables, passing parameters by reference, calling functions indirectly, crafting a switch statement, and handling exceptions. Chapter 11, *References and Records*, is about data structures; basic manipulation of references to data and functions are demonstrated. Later recipes show how to create elaborate data structures and how to save and restore these structures from permanent storage. Chapter 12, *Packages, Libraries, and Modules*, concerns breaking up your program into separate files; we discuss how to make variables and functions private to a module, customize warnings for modules, replace built-ins, trap errors loading missing modules, and use the *h2ph* and *h2xs* tools to interact with C and C++ code. Lastly, Chapter 13, *Classes, Objects, and Ties*, covers the fundamentals of building your own object-based module to create user-defined types, complete with constructors, destructors, and inheritance. Other recipes show examples of circular data structures, operator overloading, and tied data types.

The next two chapters are about interfaces: one to databases, the other to users. Chapter 14, *Database Access*, includes techniques for manipulating DBM files and querying and updating databases with SQL and the DBI module. Chapter 15, *Interactivity*, covers topics such as clearing the screen, processing command-line switches, single-character input, moving the cursor using *termcap* and *curses*, thumbnailing images, and graphing data.

The last portion of the book is devoted to interacting with other programs and services. Chapter 16, *Process Management and Communication*, is about running other programs and collecting their output, handling zombie processes, named pipes, signal management, and sharing variables between running programs. Chapter 17, *Sockets*, shows how to establish stream connections or use datagrams to create low-level networking applications for client-server programming. Chapter 18, *Internet Services*, is about higher-level protocols such as mail, FTP, Usenet news, XML-RPC, and SOAP. Chapter 19, *CGI Programming*, contains recipes for processing web forms, trapping their errors, avoiding shell escapes for security, managing cookies, shopping cart techniques, and saving forms to files or pipes. Chapter 20, *Web Automation*, covers non-interactive uses of the Web, such as fetching web pages, automating form submissions in a script, extracting URLs from a web page, removing HTML tags, finding fresh or stale links, and parsing HTML. Chapter 21, *mod_perl*, introduces mod_perl, the Perl interpreter embedded in Apache. It covers fetching form parameters, issuing redirections, customizing Apache's logging, handling authentication, and advanced templating with Mason and the Template Toolkit. Finally, Chapter 22, *XML* is about the ubiquitous data format XML and includes recipes such as validating XML, parsing XML into events and trees, and transforming XML into other formats.

What's New in This Edition

The book you're holding is thicker than its previous edition of five years ago—about 200 pages thicker. New material is spread across more than 80 entirely new recipes plus over 100 existing recipes that were substantially updated since the first edition. You'll also find two new chapters: one on mod_perl, Perl's interface to the popular Apache web server; the other on XML, an increasingly important standard for exchanging structured data.

Growth in this book reflects growth in Perl itself, from Version 5.004 in the first edition to v5.8.1 in this one. Syntactic changes to the core language are nevertheless comparatively few. Some include the spiffy `our` keyword to replace the crufty `use vars` construct for declaring global variables, fancier forms of `open` to disambiguate filenames with strange characters in them, and automatic allocation of anonymous filehandles into undefined scalar variables. We've updated our solutions and code examples to reflect these changes where it made sense to make use of the new features.

Several of Perl's major subsystems have been completely overhauled for improved functionality, stability, and portability. Some of these are relatively isolated, like the subsystems for threading (see Recipe 17.14) and for safe signals (see Recipe 16.17). Their applications are usually confined to systems programming.

More sweeping are the changes to Perl and to this book that stem from integrated support for Unicode characters. The areas most profoundly affected are strings (now with multibyte characters) and I/O (now with stackable encoding layers), so Chapters 1 and 8 include new introductory material to orient you to these sometimes confusing topics. These chapters also provide the bulk of recipes dealing with those specific topics, but this fundamental shift touches many more recipes throughout the book.

Another growth area for this book and Perl has been the welcome proliferation of many highly used and highly useful modules now released standard with the Perl core. Previously, these modules had to be separately located, downloaded, configured, built, tested, and installed. Now that they're included in the standard distribution, that's all taken care of when installing Perl itself.

Some new core modules are really pragmas that alter Perl's compilation or runtime environment, as demonstrated in Recipes like 1.21 ("Constant Variables"), 12.3 ("Delaying use Until Runtime"), and 12.15 ("Customizing Warnings"). Some are programmer tools to aid code development and debugging, like modules shown in Recipes 11.11 ("Printing Data Structures"), 11.13 ("Storing Data Structures to Disk"), 11.15 ("Coping with Circular Data Structures Using Weak References"), and 22.2 ("Parsing XML into a DOM Tree"). Others augment basic operations available on core data types, like those shown in Recipes 2.1 ("Checking Whether a String Is a Valid Number"), 4.13 ("Finding the First List Element That Passes a Test"), 4.18 ("Randomizing an Array"), 5.3 ("Creating a Hash with Immutable Keys or Values"), 8.7 ("Randomizing All Lines"), and 11.15 ("Coping with Circular Data Structures Using Weak References"). Finally, the networking modules have at last made their way into the core distribution, as seen throughout Chapter 18. We've probably not seen the last of this inward migration of modules.

Platform Notes

This book was developed using Perl release v5.8.1. That means major release 5, minor release 8, and patch level 1. We tested most programs and examples under BSD, Linux, and SunOS, but that doesn't mean they'll work only on those systems. Perl was *designed* for platform independence. When you use Perl as a general-purpose programming language, employing basic operations like variables, patterns, subroutines, and high-level I/O, your program should work the same everywhere that Perl runs—which is just about everywhere. The first two-thirds of this book uses Perl for general-purpose programming.

Perl was originally conceived as a high-level, cross-platform language for systems programming. Although it has long since expanded beyond its original domain, Perl continues to be heavily used for systems programming, both on its native Unix systems and elsewhere. Most recipes in Chapters 14 through 18 deal with classic systems programming. For maximum portability in this area, we've mainly focused on open systems as defined by the Portable Operating System Interface (POSIX), which includes nearly every form of Unix and numerous other systems as well. Most recipes should run with little or no modification on any POSIX system.

You can still use Perl for systems programming work even on non-POSIX systems by using vendor-specific modules, but these are not covered in this book. That's because they're not portable—and to be perfectly forward, because we have no such systems at our disposal. Consult the documentation that came with your port of Perl for any proprietary modules that may have been included. The *perlport*(1) manpage is a good start; its SEE ALSO section points to per-platform documentation, such as *perlmacos*(1) and *perlvms*(1).

But don't worry. Many recipes for systems programming should work on non-POSIX systems as well, especially those dealing with databases, networking, and web interaction. That's because the modules used for those areas hide platform dependencies. The principal exception is those few recipes and programs that rely upon multitasking constructs, notably the powerful `fork` function, standard on POSIX systems, but seldom on others. Mac OS X now supports `fork` natively, however, and even on Windows systems Perl now emulates that syscall remarkably well.

When we needed structured files, we picked the convenient Unix */etc/passwd* database; when we needed a text file to read, we picked */etc/motd*; and when we needed a program to produce output, we picked *who*(1). These were merely chosen to illustrate the principles—the principles work whether or not your system has these files and programs.

Other Books

If you'd like to learn more about Perl, here are some related publications that we (somewhat sheepishly) recommend:

Programming Perl, by Larry Wall, Tom Christiansen, and Jon Orwant; O'Reilly & Associates (Third Edition, 2000). This book is indispensable for every Perl programmer. Coauthored by Perl's creator, this classic reference is the authoritative guide to Perl's syntax, functions, modules, references, invocation options, and much more.

Mastering Algorithms with Perl, by Jon Orwant, Jarkko Hietaniemi, and John Macdonald; O'Reilly & Associates (2000). All the useful techniques from a CS algorithms course, but without the painful proofs. This book covers fundamental and useful algorithms in the fields of graphs, text, sets, and more.

Mastering Regular Expressions, by Jeffrey Friedl; O'Reilly & Associates (Second Edition, 2002). This book is dedicated to explaining regular expressions from a practical perspective. It not only covers general regular expressions and Perl patterns well, it also compares and contrasts these with those used in other popular languages.

Object Oriented Perl, by Damian Conway; Manning (1999). For beginning as well as advanced OO programmers, this book explains common and esoteric techniques for writing powerful object systems in Perl.

Learning Perl, by Randal Schwartz and Tom Phoenix; O'Reilly & Associates (Third Edition, 2001). A tutorial introduction to Perl for folks who are already programmers and who are interested in learning Perl from scratch. It's a good starting point if this book is over your head. Erik Olson refurbished this book for Windows systems, called *Learning Perl for Win32 Systems*.

Programming the Perl DBI, by Tim Bunce and Alligator Descartes; O'Reilly & Associates (2000). The only book on Perl's relational database interface, by the author of the DBI module.

CGI Programming with Perl, by Scott Guelich, Shishir Gundavaram, and Gunther Birznieks; O'Reilly & Associates (Second Edition, 2000). This is a solid introduction to the world of CGI programming.

Writing Apache Modules with Perl and C, by Lincoln Stein and Doug MacEachern; O'Reilly & Associates (1999). This guide to web programming teaches you how to extend the capabilities of the Apache web server, especially using the turbocharged mod_perl for fast CGI scripts and via the Perl-accessible Apache API.

Practical mod_perl, by Stas Bekman and Eric Cholet; O'Reilly & Associates (2003). A comprehensive guide to installing, configuring, and developing with mod_perl. This book goes into corners of mod_perl programming that no other book dares to touch.

The mod_perl Developer's Cookbook, by Geoff Young, Paul Lindner, and Randy Kobes; SAMS (2002). Written in a similar style to the Cookbook you hold in your hand, this book belongs on every mod_perl developer's desk. It covers almost every task a mod_perl developer might want to perform.

Beyond the Perl-related publications listed here, the following books came in handy when writing this book. They were used for reference, consultation, and inspiration.

The Art of Computer Programming, by Donald Knuth, Volumes I-III: "Fundamental Algorithms," "Seminumerical Algorithms," and "Sorting and Searching"; Addison-Wesley (Third Edition, 1998).

Introduction to Algorithms, by Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest; MIT Press and McGraw-Hill (1990).

Algorithms in C, by Robert Sedgewick; Addison-Wesley (1992).

The Art of Mathematics, by Jerry P. King; Plenum (1992).

The Elements of Programming Style, by Brian W. Kernighan and P.J. Plauger; McGraw-Hill (1988).

The UNIX Programming Environment, by Brian W. Kernighan and Rob Pike; Prentice-Hall (1984).

POSIX Programmer's Guide, by Donald Lewine; O'Reilly & Associates (1991).

Advanced Programming in the UNIX Environment, by W. Richard Stevens; Addison-Wesley (1992).

TCP/IP Illustrated, by W. Richard Stevens, et al., Volumes I–III; Addison-Wesley (1992-1996).

HTML: The Definitive Guide, by Chuck Musciano and Bill Kennedy; O'Reilly & Associates (Third Edition, 1998).

Official Guide to Programming with CGI.pm, by Lincoln Stein; John Wiley & Sons (1997).

Web Client Programming with Perl, by Clinton Wong; O'Reilly & Associates (1997).

The New Fowler's Modern English Usage, edited by R.W. Burchfield; Oxford (Third Edition, 1996).

Conventions Used in This Book

Programming Conventions

We give lots of examples, most of which are pieces of code that should go into a larger program. Some examples are complete programs, which you can recognize because they begin with a #! line. We start nearly all of our longer programs with:

```
#!/usr/bin/perl -w
use strict;
```

or else the newer:

```
#!/usr/bin/perl
use strict;
use warnings;
```

Still other examples are things to be typed on a command line. We've used % to show the shell prompt:

```
% perl -e 'print "Hello, world.\n"'
Hello, world.
```

This style represents a standard Unix command line, where single quotes represent the "most quoted" form. Quoting and wildcard conventions on other systems vary. For

example, many command-line interpreters under MS-DOS and VMS require double quotes instead of single ones to group arguments with spaces or wildcards in them.

Typesetting Conventions

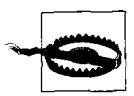
The following typographic conventions are used in this book:

Bold
: is used exclusively for command-line switches. This allows one to distinguish for example, between the **-w** warnings switch and the `-w` filetest operator.

Italic
: is used for URLs, manpages, pathnames, and programs. New terms are also italicized when they first appear in the text.

`Constant Width`
: is used for function and method names and their arguments; in examples to show text that you enter verbatim; and in regular text to show literal code.

`Constant Width Bold Italic`
: is used in examples to show output produced.



Indicates a warning or caution.

Documentation Conventions

The most up-to-date and complete documentation about Perl is included with Perl itself. If typeset and printed, this massive anthology would use more than a thousand pages of printed paper, greatly contributing to global deforestation. Fortunately, you don't have to print it out, because it's available in a convenient and searchable electronic form.

When we refer to a "manpage" in this book, we're talking about this set of online manuals. The name is purely a convention; you don't need a Unix-style `man` program to read them. The *perldoc* command distributed with Perl also works, and you may even have the manpages installed as HTML pages, especially on non-Unix systems. Plus, once you know where they're installed, you can grep them directly.* The HTML version of the manpages is available on the Web at *http://www.perl.com/CPAN/doc/manual/html/*.

When we refer to non-Perl documentation, as in "See *kill*(2) in your system manual," this refers to the *kill* manpage from section 2 of the *Unix Programmer's Manual* (system calls). These won't be available on non-Unix systems, but that's probably okay,

* If your system doesn't have *grep*, use the *tcgrep* program supplied at the end of Chapter 6.

because you couldn't use them there anyway. If you really do need the documentation for a system call or library function, many organizations have put their manpages on the Web; a quick search of Google for *crypt(3) manual* will find many copies.

We'd Like to Hear from You

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (which may in fact resemble bugs). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

O'Reilly & Associates, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the U.S. or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (FAX)

You can also send us messages electronically. To be put on the mailing list or request a catalog, send email to:

info@oreilly.com

To ask technical questions or comment on the book, send email to:

bookquestions@oreilly.com

There is a web site for the book, where we'll list errata and plans for future editions. Here you'll also find source code for the book's examples available for download so you don't have to type them in yourself. You can access this page at:

http://www.oreilly.com/catalog/perlckbk2/

For more information about this book and others, see the O'Reilly web site:

http://www.oreilly.com/

Acknowledgments for the First Edition

This book wouldn't exist but for a legion of people standing, knowing and unknowing, behind the authors. At the head of this legion would have to be our editor, Linda Mui, carrot on a stick in one hand and a hot poker in the other. She was great.

As the author of Perl, Larry Wall was our ultimate reality check. He made sure we weren't documenting things he was planning to change and helped out on wording and style.* If now and then you think you're hearing Larry's voice in this book, you probably are.

* And footnotes.

Larry's wife, Gloria, a literary critic by trade, shocked us by reading through every single word—and actually liking most of them. Together with Sharon Hopkins, resident Perl Poetess, she helped us rein in our admittedly nearly insatiable tendency to produce pretty prose sentences that could only be charitably described as lying somewhere between the inscrutably complex and the hopelessly arcane, eventually rendering the meandering muddle into something legible even to those whose native tongues were neither PDP-11 assembler nor Mediæval Spanish.

Our three most assiduous reviewers, Mark-Jason Dominus, Jon Orwant, and Abigail, have worked with us on this book nearly as long as we've been writing it. Their rigorous standards, fearsome intellects, and practical experience in Perl applications have been of invaluable assistance. Doug Edwards methodically stress-tested every piece of code from the first seven chapters of the book, finding subtle border cases no one else ever thought about. Other major reviewers include Andy Dougherty, Andy Oram, Brent Halsey, Bryan Buus, Gisle Aas, Graham Barr, Jeff Haemer, Jeffrey Friedl, Lincoln Stein, Mark Mielke, Martin Brech, Matthias Neeracher, Mike Stok, Nate Patwardhan, Paul Grassie, Peter Prymmer, Raphaël Manfredi, and Rod Whitby.

And this is just the beginning. Part of what makes Perl fun is the sense of community and sharing it seems to engender. Many selfless individuals lent us their technical expertise. Some read through complete chapters in formal review. Others provided insightful answers to brief technical questions when we were stuck on something outside our own domain. A few even sent us code. Here's a partial list of these helpful people: Aaron Harsh, Ali Rayl, Alligator Descartes, Andrew Hume, Andrew Strebkov, Andy Wardley, Ashton MacAndrews, Ben Gertzfield, Benjamin Holzman, Brad Hughes, Chaim Frenkel, Charles Bailey, Chris Nandor, Clinton Wong, Dan Klein, Dan Sugalski, Daniel Grisinger, Dennis Taylor, Doug MacEachern, Douglas Davenport, Drew Eckhardt, Dylan Northrup, Eric Eisenhart, Eric Watt Forste, Greg Bacon, Gurusamy Sarathy, Henry Spencer, Jason Ornstein, Jason Stewart, Joel Noble, Jonathan Cohen, Jonathan Scott Duff, Josh Purinton, Julian Anderson, Keith Winstein, Ken Lunde, Kirby Hughes, Larry Rosler, Les Peters, Mark Hess, Mark James, Martin Brech, Mary Koutsky, Michael Parker, Nick Ing-Simmons, Paul Marquess, Peter Collinson, Peter Osel, Phil Beauchamp, Piers Cawley, Randal Schwartz, Rich Rauenzahn, Richard Allan, Rocco Caputo, Roderick Schertler, Roland Walker, Ronan Waide, Stephen Lidie, Steven Owens, Sullivan Beck, Tim Bunce, Todd Miller, Troy Denkinger, and Willy Grimm.

And let's not forget Perl itself, without which this book could never have been written. Appropriately enough, we used Perl to build endless small tools to help produce this book. Perl tools converted our text in pod format into *troff* for displaying and review and into FrameMaker for production. Another Perl program ran syntax checks on every piece of code in the book. The Tk extension to Perl was used to build a graphical tool to shuffle around recipes using drag-and-drop. Beyond these, we also built innumerable smaller tools for tasks like checking RCS locks, finding

duplicate words, detecting certain kinds of grammatical errors, managing mail folders with feedback from reviewers, creating program indices and tables of contents, and running text searches that crossed line boundaries or were restricted to certain sections—just to name a few. Some of these tools found their way into the same book they were used on.

Tom

Thanks first of all to Larry and Gloria for sacrificing some of their European vacation to groom the many nits out of this manuscript, and to my other friends and family—Bryan, Sharon, Brent, Todd, and Drew—for putting up with me over the last couple of years and being subjected to incessant proofreadings.

I'd like to thank Nathan for holding up despite the stress of his weekly drives, my piquant vegetarian cooking and wit, and his getting stuck researching the topics I so diligently avoided.

I'd like to thank those largely unsung titans in our field—Dennis, Linus, Kirk, Eric, and Rich—who were all willing to take the time to answer my niggling operating system and *troff* questions. Their wonderful advice and anecdotes aside, without their tremendous work in the field, this book could never have been written.

Thanks also to my instructors who sacrificed themselves to travel to perilous places like New Jersey to teach Perl in my stead. I'd like to thank Tim O'Reilly and Frank Willison first for being talked into publishing this book, and second for letting time-to-market take a back seat to time-to-quality. Thanks also to Linda, our shamelessly honest editor, for shepherding dangerously rabid sheep through the eye of a release needle.

Most of all, I want to thank my mother, Mary, for tearing herself away from her work in prairie restoration and teaching high school computer and biological sciences to keep both my business and domestic life in smooth working order long enough for me to research and write this book.

Finally, I'd like to thank Johann Sebastian Bach, who was for me a boundless font of perspective, poise, and inspiration—a therapy both mental and physical. I am certain that forevermore the Cookbook will evoke for me the sounds of BWV 849, now indelibly etched into the wetware of head and hand.

Nat

Without my family's love and patience, I'd be baiting hooks in a 10-foot swell instead of mowing my lawn in suburban America. Thank you! My friends have taught me much: Jules, Amy, Raj, Mike, Kef, Sai, Robert, Ewan, Pondy, Mark, and Andy. I owe a debt of gratitude to the denizens of Nerdsholm, who gave sound technical advice and introduced me to my wife (they didn't give me sound technical

advice on her, though). Thanks also to my employer, Front Range Internet, for a day job I don't want to quit.

Tom was a great co-author. Without him, this book would be nasty, brutish, and short. Finally, I have to thank Jenine. We'd been married a year when I accepted the offer to write, and we've barely seen each other since then. Nobody will savour the final full-stop in this sentence more than she.

Acknowledgments for the Second Edition

We would like to thank our many tech reviewers, who gave generously of their time and knowledge so that we might look better. Some were formal reviewers who painstakingly plodded through endless drafts and revisions, while others were casual comrades roped into reading small excerpts related to their own particular expertise or interest. The bugs you don't find in this book are thanks to them. Those you do find were probably introduced after they reviewed it.

Just a few of these selfless people were Adam Maccabee Trachtenberg, Rafael Garcia-Suarez, Ask Björn Hansen, Mark-Jason Dominus, Abhijit Menon-Sen, Jarkko Hietaniemi, Benjamin Goldberg, Aaron Straup Cope, Tony Stubblebine, Michel Rodriguez, Nick Ing-Simmons, Geoffrey Young, Douglas Wilson, Paul Kulchenko, Jeffrey Friedl, Arthur Bergman, Autrijus Tang, Matt Sergeant, Steve Marvell, Damian Conway, Sean M. Burke, Elaine Ashton, Steve Lidie, Ken Williams, Robert Spier, Chris Nandor, Brent Halsey, Matthew Free, Rocco Caputo, Robin Berjon, Adam Turoff, Chip Turner, David Sklar, Mike Sierra, Dave Rolsky, Kip Hampton, Chris Fedde, Graham Barr, Jon Orwant, Rich Bowen, Mike Stok, Tim Bunce, Rob Brown, Dan Brian, Gisle Aas, and Abigail.

We'd also like to thank our patient and persistent editor, Linda Mui, who ran serious risk of getting herself committed as she tried to wrestle "the final edits" from us.

Tom

I would like to thank Larry Wall for making the programming world (and several others) a better place for all of us, Nathan for documenting the undocumented, and our editor, Linda Mui, for her indefatigable patience at herding her author cats of the Schrödinger clan ever onward. This book would not exist but for all three of them.

I would especially like to thank someone who is no longer here to read these words in print, words he would otherwise himself have shepherded: O'Reilly's longtime editor-in-chief and my friend, Frank Willison, gone from us two years now. His many erudite epistles are a thing of legend, carefully crafted treasures more dear to any writer than finest gold. Over our years of working together, Frank was a constant source of personal inspiration and encouragement. His easygoing cheer and charm, his broad learning and interests, and his sparkling wit—sometimes subtle,