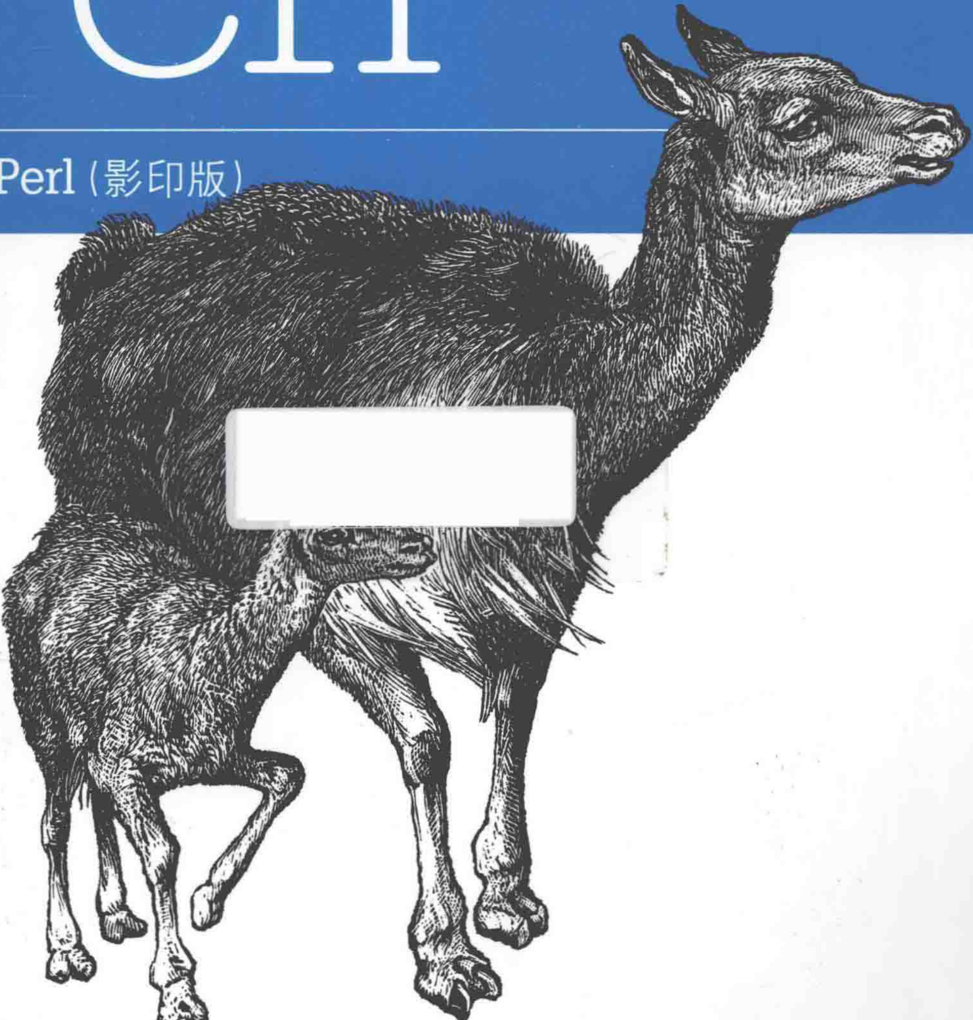
O'REILLY®

第2版

Mastering Perl

精通Perl（影印版）



東南大學出版社

brian d foy 著

第2版

精通Perl（影印版）

Mastering Perl

brian d foy 著

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo O'REILLY®

O'Reilly Media, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目（CIP）数据

精通 Perl :第 2 版:英文 / （美）福瓦（Foy, B.D.)著 . —影印本 . —南京：东南大学出版社，2014.10

书名原文：Mastering Perl, 2E

ISBN 978-7-5641-5002-0

I. ① 精… II. ①福… III. ① PERL 语言 – 程序设计 – 英文 IV. ① TP312

中国版本图书馆 CIP 数据核字（2014）第 115564 号

江苏省版权局著作权合同登记

图字：10-2013-372 号

精通 Perl 第 2 版（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼 2 号　　邮编：210096
出 版 人：江建中
网　　址：http://www.seupress.com
电子邮件：press@seupress.com
印　　刷：常州市武进第三印刷有限公司
开　　本：787 毫米 × 980 毫米　16 开本
印　　张：25
字　　数：490 千字
版　　次：2014 年 10 月第 1 版
印　　次：2014 年 10 月第 1 次印刷
书　　号：ISBN 978-7-5641-5002-0
定　　价：74.00 元

本社图书若有印装质量问题，请直接与营销部联系。电话（传真）：025-83791830

Preface

Mastering Perl is the third book in the series starting with *Learning Perl*, which taught you the basics of Perl syntax, progressing to *Intermediate Perl*, which taught you how to create reusable Perl software, and finally this book, which pulls everything together to show you how to bend Perl to your will. This isn't a collection of clever tricks, but a way of thinking about Perl programming so you integrate the real-life problems of debugging, maintenance, configuration, and other tasks you'll encounter as a working programmer. This book starts you on your path to becoming the person with the answers, and, failing that, the person who knows how to find the answers or discover the problem.

Becoming a Master

This book isn't going to make you a Perl master; you have to do that for yourself by programming a lot of Perl, trying a lot of new things, and making a lot of mistakes. I'm going to help you get on the right path, but the road to mastery is one of self-reliance and independence. As a Perl master, you'll be able to answer your own questions as well as those of others.

In the golden age of guilds, craftsmen followed a certain path, both literally and figuratively, as they mastered their craft. They started as apprentices and would do the boring bits of work until they had enough skill to become the more trusted journeyman. The journeyman had greater responsibility but still worked under a recognized master. When they had learned enough of the craft, the journeymen would produce a "master work" to prove their skill. If other masters deemed it adequately masterful, the journeyman became a recognized master himself.

The journeymen and masters also traveled (although people dispute if that's where the "journey" part of the name came from) to other masters, where they would learn new techniques and skills. Each master knew things the others didn't, perhaps deliberately guarding secret methods or doing it in a different way. Part of the journeymen's education was learning from more than one master.

Interactions with other masters and journeymen continued the master's education. He learned from those masters with more experience, and learned from himself as he taught journeymen, who also taught him as they brought skills they learned from other masters. A master never stops learning.

The path an apprentice followed affected what he learned. An apprentice who studied with more masters was exposed to many more perspectives and ways of teaching, all of which he could roll into his own way of doing things. Odd things from one master could be exposed, updated, or refined by another, giving the apprentice a balanced view on things. Additionally, although the apprentice might be studying to be a carpenter or a mason, different masters applied those skills to different goals, giving the apprentice a chance to learn different applications and ways of doing things.

Unfortunately, programmers don't operate under the guild system. Most Perl programmers learn Perl on their own (I'm sad to say, as a Perl instructor), program on their own, and never get the advantage of a mentor. That's how I started. I bought the first edition of *Learning Perl* and worked through it on my own. I was the only person I knew who had even heard of Perl, although I'd seen it around a couple of times. Most people used what others had left behind. Soon after that, I discovered *comp.lang.perl.misc* and started answering any question that I could. It was like self-assigned homework. My skills improved and I got almost instantaneous feedback, good and bad, and I learned even more Perl. I ended up with a job that allowed me to program Perl all day, but I was the only person in the company doing that. I kept up my homework on *comp.lang.perl.misc*.

I eventually caught the eye of Randal Schwartz, who took me under his wing and started my Perl apprenticeship. He invited me to become a Perl instructor with Stonehenge Consulting Services, and then my real Perl education began. Teaching, meaning figuring out what you know and how to explain it to others, is the best way to learn a subject. After a while of doing that, I started writing about Perl, which is close to teaching, although with correct grammar (mostly) and an editor to correct mistakes.

That presents a problem for *Mastering Perl*, which I designed to be the third book of a trilogy starting with *Learning Perl* and *Intermediate Perl*, both of which I've had a hand in. Each of those are about 300 pages, and that's what I'm limited to here. How do I encapsulate the years of my experience in such a slim book?

In short, I can't. I'll teach you what I think you should know, but you'll also have to learn from other sources. As with the old masters, you can't just listen to one person. You need to find other masters too, and that's also the great thing about Perl: you can do things in so many different ways. Some of these masters have written very good books, from this publisher and others, so I'm not going to duplicate those topics here, as I discuss in a moment.

What It Means to Be a Master

This book takes a different tone from *Learning Perl* and *Intermediate Perl*, which we designed as tutorial books. Those mostly cover the details of the Perl language and only delve a little into the practice of programming. *Mastering Perl*, however, puts more responsibility on you, the reader.

Now that you've made it this far in Perl, you're working on your ability to answer your own questions and figure out things on your own, even if that's a bit more work than simply asking someone. The very act of doing it yourself builds your experience and prevents you from annoying your coworkers with extra work.

Although I don't cover other languages in this book, like *Advanced Perl Programming, First Edition* did and *Mastering Regular Expressions* does, you should learn some other languages. This informs your Perl knowledge and gives you new perspectives, some that make you appreciate Perl more and others that help you understand its limitations.

And, as a master, you will run into Perl's limitations. I like to say that if you don't have a list of five things you hate about Perl and the facts to back them up, you probably haven't done enough Perl; see "My Frozen Perl 2011 Keynote" (*http://bit.ly/JDI5LC*). It's not really Perl's fault. You'll get that with any language. The mastery comes from knowing these things and still choosing Perl because its strengths outweigh the weaknesses for your application. You're a master because you know both sides of the problem and can make an informed choice that you can explain to others.

All of that means that becoming a master involves work, reading, and talking to other people. The more you do, the more you learn. There's no shortcut to mastery. You may be able to learn the syntax quickly, as in any other language, but that will be the tiniest portion of your experience. Now that you know most of Perl, you'll probably spend your time reading some of the "meta"-programming books that discuss the practice of programming rather than just slinging syntax. Those books will probably use a language that's not Perl, but I've already said you need to learn some other languages, if only to be able to read these books. As a master, you're always learning.

Becoming a master involves understanding more than you need to, doing quite a bit of work on your own, and learning as much as you can from the experience of others. It's not just about the code you write, because you have to deal with the code from many other authors too.

It may sound difficult, but that's how you become a master. It's worth it, so don't give up. Good luck!

Who Should Read This Book

I wrote this book as a successor to *Intermediate Perl*, which covered the basics of references, objects, and modules. I'll assume that you already know and feel comfortable with those features. Where possible, I make references to *Intermediate Perl* in case you need to refresh your skills on a topic.

If you're coming directly from another language and haven't used Perl yet, or have only used it lightly, you might want to skim *Learning Perl* and *Intermediate Perl* to get the basics of the language. Still, you might not recognize some of the idioms that come with experience and practice. I don't want to tell you not to buy this book (hey, I need to pay my mortgage!), but you might not get the full value I intend, at least not right away.

How to Read This Book

I'm not writing a third volume of "Yet More Perl Features." I want to teach you how to learn Perl on your own. I'm setting you on your own path to mastery, and as an apprentice you'll need to do some work on your own. Sometimes this means I'll show you where in the Perl documentation to get the answers (meaning I can use the saved space to talk about other topics).

You don't need to read the chapters in any particular order, and the material isn't cumulative. If there's something that doesn't interest you, you can probably safely skip it.

If you want to know more about a subject, check out the references I include at the end of each chapter.

What Should You Know Already?

I'll presume that you already know everything that we covered in *Learning Perl* and *Intermediate Perl*. By we, I mean coauthors Randal Schwartz, Tom Phoenix, and myself.

Most importantly, you should know these subjects, each of which imply knowledge of other subjects:

- Using Perl modules
- Writing Perl modules
- References to variables, subroutines, and filehandles
- Basic regular expression syntax and workings
- Object-oriented Perl

If I want to discuss something not in either of those books, I'll explain it in a bit more depth. Even if we did cover it in the previous books, I might cover it again just because it's that important.

What I Cover

After learning the basic syntax of Perl in *Learning Perl* and the basics of modules and team programming in *Intermediate Perl*, the next thing you need to learn are the idioms of Perl and the integration of the skills that you already have to create robust and scalable applications that other people can use without your help.

I'll cover some subjects you've seen in those two books, but in more depth. As we said in *Learning Perl*, we sometimes told white lies to simplify the details and to get you going as soon as possible without getting bogged down. Now it's time to get a bit dirty in the bogs.

Don't mistake my coverage of a subject for an endorsement, though. There are millions of Perl programmers in the world, and they all have their own way of doing things. Part of becoming a Perl master involves reading quite a bit of Perl even if you wouldn't write that Perl yourself. I'll endeavor to tell you when I think you shouldn't do something, but that's really just my opinion. As you strive to be a good programmer, you'll need to know more than you'll use. Sometimes I'll show things I don't want you to use, but I know you'll see in code from other people. Oh well, it's not a perfect world.

Not all programming is about adding or adjusting features in code. Sometimes it's pulling code apart to inspect it and watch it do its magic. Other times it's about getting rid of code that you don't need. The practice of programming is more than creating applications. It's also about managing and wrangling code. Some of the techniques I'll show are for analysis, not your own development.

What I Don't Cover

As I talked over the idea of this book with the editors, we decided not to duplicate the subjects more than adequately covered by other books. You need to learn from other masters too, and I don't really want to take up more space on your shelf than I really need. Ignoring those subjects gives me the double bonus of not writing those chapters and using that space for other things. You should already have read those other books anyway.

That doesn't mean that you get to ignore those subjects, though, and where appropriate I'll point you to the right book. In Appendix A, I list some books I think you should add to your library as you move toward Perl mastery. Those books are by other Perl masters, each of whom has something to teach you. At the end of most chapters I point you toward other resources as well. A master never stops learning.

Since you're already here, though, I'll just give you the list of topics I'm explicitly avoiding, for whatever reason: Perl internals, embedding Perl, threads, best practices, object-oriented programming, source filters, and dolphins. This is a dolphin-safe book.

Structure of This Book

Preface
: An introduction to the scope and intent of this book.

Chapter 1, Advanced Regular Expressions
: More regular expression features, including global matches, lookarounds, readable regexes, and regex debugging.

Chapter 2, Secure Programming Techniques
: Avoid some common programing problems with the techniques in this chapter, which covers taint checking and gotchas.

Chapter 3, Perl Debuggers
: A little bit about the Perl debugger, writing your own debugger, and using the debuggers others wrote.

Chapter 4, Profiling Perl
: Before you set out to improve your Perl program, find out where you should concentrate your efforts.

Chapter 5, Benchmarking Perl
: Figure out which implementations do better on time, memory, and other metrics, along with cautions about what your numbers actually mean.

Chapter 6, Cleaning Up Perl
: Wrangle Perl code you didn't write (or even code you did write) to make it more presentable and readable by using `Perl::Tidy` or `Perl::Critic`.

Chapter 7, Symbol Tables and Typeglobs
: Learn how Perl keeps track of package variables and how you can use that mechanism for some powerful Perl tricks.

Chapter 8, Dynamic Subroutines
: Define subroutines on the fly and turn the tables on normal procedural programming. Iterate through subroutine lists rather than data to make your code more effective and easy to maintain.

Chapter 9, Modifying and Jury-Rigging Modules
: Fix code without editing the original source so you can always get back to where you started.

Chapter 10, Configuring Perl Programs
: Let your users configure your programs without touching the code.

Chapter 11, Detecting and Reporting Errors
: Learn how Perl reports errors, how you can detect errors Perl doesn't report, and how to tell your users about them.

Chapter 12, Logging
: Let your Perl program talk back to you by using Log4perl, an extremely flexible and powerful logging package.

Chapter 13, Data Persistence
: Store data for later use in other programs, a later run of the same program, or to send as text over a network.

Chapter 14, Working with Pod
: Translate plain ol' documentation into any format that you like, and test it too.

Chapter 15, Working with Bits
: Use bit operations and bit vectors to efficiently store large data.

Chapter 16, The Magic of Tied Variables
: Implement your own versions of Perl's basic data types to perform fancy operations without getting in the user's way.

Chapter 17, Modules as Programs
: Write programs as modules to get all of the benefits of Perl's module distribution, installation, and testing tools.

Appendix A, Further Reading
: Explore these resources to continue your Perl education.

Appendix B, brian's Guide to Solving Any Perl Problem
: My popular step-by-step guide to solving any Perl problem. Follow these steps to improve your troubleshooting skills.

Conventions Used in This Book

The following typographic conventions are used in this book:

Italics
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *http://www.masteringperl.org/*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Mastering Perl, Second Edition*, by brian d foy (O'Reilly). Copyright 2014 brian d foy, 978-1-449-39311-3."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online (*www.safaribooksonline.com*) is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

> O'Reilly Media, Inc.
> 1005 Gravenstein Highway North
> Sebastopol, CA 95472
> 800-998-9938 (in the United States or Canada)
> 707-829-0515 (international or local)
> 707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://oreil.ly/mastering-perl-2e*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

Many people helped me during the year I took to write the first edition of this book. The readers of the *Mastering Perl* mailing list gave constant feedback on the manuscript and sent patches, which I mostly applied as is, including those from Andy Armstrong, David H. Adler, Renée Bäcker, Anthony R. J. Ball, Daniel Bosold, Alessio Bragadini, Philippe Bruhat, Katharine Farah, Shlomi Fish, Deyan Ginev, David Golden, Bob Goolsby, Ask Bjørn Hansen, Jarkko Hietaniemi, Joseph Hourcle, Adrian Howard, Offer Kaye, Stefan Lidman, Eric Maki, Joshua McAdams, Florian Merges, Jason Messmer, Thomas Nagel, Xavier Noria, Manuel Pégourié-Gonnard, Les Peters, Bill Riker, Yitzchak Scott-Thoennes, Ian Sealy, Sagar R. Shah, Alberto Simões, Derek B. Smith, Kurt Starsinic, Adam Turoff, David Westbrook, and Evan Zacks. Many more people submitted errata on the first edition. I'm quite reassured that their constant scrutiny kept me on the right path.

Tim Bunce provided gracious advice about the profiling chapter, which includes `DBI::Profile`, and Jeffrey Thalhammer updated me on the current developments with his `Perl::Critic` module (*http://www.metacpan.org/module/Perl::Critic*).

Perrin Harkins, Rob Kinyon, and Randal Schwartz gave the manuscript of the first edition a thorough beating at the end, and I'm glad I chose them as technical reviewers because their advice is always spot-on. For the second edition, the input of Matthew Horsfall and André Philipp were invaluable to me.

Allison Randal provided valuable Perl advice and editorial guidance on the project, even though she probably dreaded my constant queries. Several other people from O'Reilly helped; it takes much more than an author to create a book, so thank a random O'Reilly employee next time you see one.

Finally, I have to thank the Perl community, which has been incredibly kind and supportive over the many years that I've been part of it. So many great programmers and managers helped me become a better programmer, and I hope this book does the same for people just joining the crowd.

Table of Contents