Perl进阶（影印版）

第二版
涵盖Perl 5.14

Intermediate Perl

O'REILLY®
東南大學出版社

Randal L. Schwartz,
brian d foy & Tom Phoenix 著

第二版

Perl进阶（影印版）

Intermediate Perl

Randal L. Schwartz, brian d foy, Tom Phoenix

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

Perl 进阶：第 2 版：英文 /(美)施瓦茨（Schwartz, R.L.），(美)福瓦（Foy, B.D.),(美)菲尼克斯（Phoenix, T.）著 . —影印本 . —南京：东南大学出版社，2013.1

书名原文：Intermediate Perl, 2E

ISBN 978-7-5641-3888-2

I. ① P… II. ①施… ②福… ③菲… 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（2012）第 273571 号

江苏省版权局著作权合同登记

图字：10-2011-409 号

Perl 进阶 第二版 （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：扬中市印刷有限公司
开　　本：787 毫米 × 980 毫米　16 开本
印　　张：24.75
字　　数：485 千字
版　　次：2013 年 1 月第 1 版
印　　次：2013 年 1 月第 1 次印刷
书　　号：ISBN 978-7-5641-3888-2
定　　价：58.00 元（册）

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

Foreword

Perl's object-oriented mechanism is classic prestidigitation. It takes a collection of Perl's existing non-OO features such as packages, references, hashes, arrays, subroutines, and modules, and then–with nothing up its sleeve–manages to conjure up fully functional objects, classes, and methods. Seemingly out of nowhere.

That's a great trick. It means you can build on your existing Perl knowledge and ease your way into OO Perl development, without first needing to conquer a mountain of new syntax or navigate an ocean of new techniques. It also means you can progressively fine-tune OO Perl to meet your own needs, by selecting from the existing constructs the one that best suits your task.

But there's a problem. Since Perl co-opts packages, references, hashes, arrays, subroutines, and modules as the basis of its OO mechanism, to use OO Perl you already need to understand packages, references, hashes, arrays, subroutines, and modules.

And there's the rub. The learning curve hasn't been eliminated; it's merely been pushed back half a dozen steps.

So then: how are you going to learn everything you need to know about non-OO Perl so you can start to learn everything you need to know about OO Perl?

This book is the answer. In the following pages, Randal draws on two decades of using Perl, and four decades of watching *Gilligan's Island* and *Mr. Ed*, to explain each of the components of Perl that collectively underpin its OO features. And, better still, he then goes on to show exactly how to combine those components to create useful classes and objects.

So if you still feel like Gilligan when it comes to Perl's objects, references, and modules, this book is just what the Professor ordered.

And that's straight from the horse's mouth.

—Damian Conway, May 2003

Preface

Almost 20 years ago (nearly an eternity in Internet time), Randal Schwartz wrote the first edition of *Learning Perl*. In the intervening years, Perl itself has grown substantially from a "cool" scripting language used primarily by Unix system administrators to a robust object-oriented programming language that runs on practically every computing platform known to mankind, and maybe some that aren't.

Throughout its six editions, *Learning Perl* remained about the same size, around 300 pages, and continued to cover much of the same material to remain compact and accessible to the beginning programmer. But there is much more to learn about Perl.

Randal called the first edition of this book *Learning Perl Objects, References, and Modules*, and we renamed its update *Intermediate Perl*, but we like to think of it as just *Learning More Perl*. This is the book that picks up where *Learning Perl* leaves off. We show how to use Perl to write larger programs.

As in *Learning Perl*, we designed each chapter to be small enough to read in just an hour or so. Each chapter ends with a series of exercises to help you practice what you've just learned, and the answers are provided in the appendix for your reference. And, like *Learning Perl*, we've developed the material in this book for use in a teaching environment.

Unless we note otherwise, everything in this book applies equally well to Perl on any platform, whether that is Unix, Linux, Windows ActivePerl from ActiveState, Strawberry Perl, or any other modern implementation of Perl. To use this book you just need to be comfortable with the material in *Learning Perl* and have the ambition to go further.

After you finish this book, you will have seen most of the core Perl language concepts that you'll need. The next book in the series is *Mastering Perl*, which focuses on applying what you already know to writing effective and robust Perl applications as well as managing the Perl software development life cycle.

At any point in your Perl career, you should also have *Programming Perl*, the (mostly) definitive bible of the language.

Structure of This Book

There are three major sections of this book. The first section deals with references, which are the keys to complex data structures as well as to object-oriented programming. The second section introduces objects and how Perl implements object-oriented programming. The third and last section deals with Perl's module structure, testing, and the community infrastructure for distributing our work.

You should read this book from front to back, stopping to do the exercises. Each chapter builds on preceding chapters, and we'll assume that you know the material from those chapters as we show new topics.

Chapter 1, Introduction
: An introduction to the material.

Chapter 2, Using Modules
: Use Perl's core modules as well as modules from other people. We're going to show you how to create your own modules later in the book, but until we do you can still use modules you already have.

Chapter 3, Intermediate Foundations
: Pick up some intermediate Perl skills you'll need for the rest of the book.

Chapter 4, Introduction to References
: Introduce a level of redirection to allow the same code to operate on different sets of data.

Chapter 5, References and Scoping
: Learn how Perl manages to keep track of pointers to data, and read an introduction to anonymous data structures and autovivification.

Chapter 6, Manipulating Complex Data Structures
: Create, access, and print arbitrarily deep and nested data structures including arrays of arrays and hashes of hashes.

Chapter 7, Subroutine References
: Capture behavior as an anonymous subroutine that you create dynamically and execute later.

Chapter 8, Filehandle References
: Store filehandles in scalar variables that you can easily pass around your program or store in data structures.

Chapter 9, Regular Expression References
: Compile regular expressions without immediately applying them, and use them as building blocks for larger patterns.

Chapter 10, Practical Reference Tricks
: Sorting complex operations, the *Schwartzian Transform*, and working with recursively defined data.

Chapter 11, Building Larger Programs
: Build larger programs by separating code into separate files and namespaces.

Chapter 12, Creating Your Own Perl Distribution
: Create a Perl distribution as your first step toward object-oriented programming.

Chapter 13, Introduction to Objects
: Work with classes, method calls, inheritance, and overriding.

Chapter 14, Introduction to Testing
: Start to test your modules so you find problems with the code as you create it.

Chapter 15, Objects with Data
: Add per instance data, including constructors, getters, and setters.

Chapter 16, Some Advanced Object Topics
: Use multiple inheritance, automatic methods, and references to filehandles.

Chapter 17, Exporter
: How `use` works, how we can decide what to export, and how we can create our own import routines.

Chapter 18, Object Destruction
: Add behavior to an object that is going away, including object persistence.

Chapter 19, Introduction to Moose
: Moose is an object framework available on CPAN.

Chapter 20, Advanced Testing
: Test complex aspects of code and metacode things such as documentation and test coverage.

Chapter 21, Contributing to CPAN
: Share your work with the world by uploading it to CPAN.

Appendix, Exercise Answers
: Where to go to get answers.

Conventions Used in This Book

The following typographic conventions are used in this book:

`Constant width`
: Used for function names, module names, filenames, environment variables, code snippets, and other literal text

Italics
: Used for emphasis and for new terms where they are defined

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book 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: "*Intermediate Perl* by Randal L. Schwartz, brian d foy, and Tom Phoenix. Copyright 2012 Randal L. Schwartz, brian d foy, and Tom Phoenix, 978-1-449-39309-0."

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/int-perl-2e

To comment or ask technical questions about this book, send an 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

From Randal. In the preface of the first edition of *Learning Perl*, I acknowledged the Beaverton McMenamin's Cedar Hills Pub[1] just down the street from my house for the "rent-free booth-office space" while I wrote most of the draft on my Powerbook 140. Well, like wearing your lucky socks every day when your favorite team is in the playoffs, I wrote nearly all of this book (including these words) at the same brewpub, in hopes that the light of success of the first book will shine on me twice. (As I update this preface for the second edition, I can see that my lucky socks do indeed work!)

This McM's has the same great local microbrew beer and greasy sandwiches, but they've gotten rid of my favorite pizza bread, replacing it with new items like marionberry cobbler (a local treat) and spicy jambalaya. (And they added two booths, and put in some pool tables.) Also, instead of the Powerbook 140, I'm using a Titanium Powerbook, with 1,000 times more disk space, 500 times more memory, and a 200-times-faster CPU running a real Unix-based operating system (OS X) instead of the limited MacOS. I also uploaded all of the draft sections (including this one) over my 144K cell-phone modem and emailed them directly to the reviewers, instead of having to wait to rush home to my 9600-baud external modem and phone line. How times have changed!

So, thanks once again to the staff of the McMenamin's Cedar Hills Pub for the booth space and the hospitality.

1. *http://www.mcmenamins.com/*

Like the previous editions of *Learning Perl*, I also owe much of what I'm saying here and how I'm saying it to the students of Stonehenge Consulting Services who have given me immediate and precise feedback (by their glazed eyes and awkwardly constructed questions) when I was exceeding the "huh?" factor threshold. With that feedback over many dozens of presentations, I was able to keep refining and refactoring the materials that paved the way for this book.

Speaking of which, those materials started as a half-day "What's new in Perl 5?" summary commissioned by Margie Levine of Silicon Graphics, in addition to my frequently presented onsite four-day Llama course (targeted primarily for Perl Version 4 at the time). Eventually, I got the idea to beef up those notes into a full course and enlisted fellow Stonehenge presenter Joseph Hall for the task. (He's the one that selected the universe from which the examples are drawn.) Joseph developed a two-day course for Stonehenge in parallel with his excellent *Effective Perl Programming* book (Addison-Wesley Professional), which we then used as the course textbook (until now).

Other Stonehenge instructors have also dabbled a bit in the "Packages, References, Objects, and Modules" course over the years, including Chip Salzenberg and Tad McClellan. But the bulk of the recent changes have been the responsibility of my senior trainer Tom Phoenix, who has been "Stonehenge employee of the month" so often that I may have to finally give up my preferred parking space.

Tom Phoenix contributed most exercises in this book and a timely set of review notes during my writing process, including entire paragraphs for me to just insert in place of the drivel I had written. We work well as a team, both in the classroom and in our joint writing efforts. It is for this effort that we've acknowledged Tom as a coauthor, but I'll take direct blame for any parts of the book you end up hating; none of that could have possibly been Tom's fault.

And last but not least, a special thanks to brian d foy, who shepherded this book into its second revision, and wrote most of the changes between the previous edition and this edition.

A book is nothing without a subject and a distribution channel, and for that I must acknowledge longtime associates Larry Wall and Tim O'Reilly. Thanks guys, for creating an industry that has paid for my essentials, discretionary purchases, and dreams for nearly 20 years.

And, as always, a special thanks to Lyle and Jack for teaching me nearly everything I know about writing and convincing me that I was much more than a programmer who might learn to write; I was also a writer who happened to know how to program. Thank you.

And to you, the reader of this book, for whom I toiled away the countless hours while sipping a cold microbrew and scarfing down a piece of incredible cheesecake, trying to avoid spilling on my laptop keyboard: thank you for reading what I've written. I

sincerely hope I've contributed (in at least a small way) to your Perl proficiency. If you ever meet me on the street, please say hi.[2] I'd like that. Thank you.

From brian. I have to thank Randal first, since I learned Perl from the first edition of *Learning Perl*, and learned the rest teaching the Llama and Alpaca courses for Stonehenge Consulting. Teaching is often the best way to learn.

The most thanks has to go to the Perl community, the wonderfully rich and diverse group of people who have made it a pleasure to work with the language and make the tools, websites, and modules that make Perl so useful. Many people have contributed indirectly to this book through my other work and discussions with them. There are too many to list, but if you've ever done anything with Perl with me, there's probably a little of you in this book.

From Tom. First of all, thanks to the entire team at O'Reilly for helping us to bring this book to fruition.

Thanks to my Stonehenge coworkers and the students I've worked with over the years, and the people I've assisted on Usenet. Your ideas and suggestions have greatly improved this material.

Especially deep thanks to my coauthor Randal for giving me freedom to explore teaching this material in varied ways.

To my wife Jenna Padbury, thanks for being a cat person, and everything thereafter.

From all of us. Thanks to our reviewers for providing comments on the draft of this book. Tom Christiansen did an amazing job not only correcting every technical problem he found, but also improving our writing quite a bit. This book is much better for it. David Golden, a fellow PAUSE admin and CPAN toolchain hacker, helped quite a bit in straightening out the details of the module release process. Several of the Moose crowd, including Stevan Little, Curtis "Ovid" Poe, and Jesse Luehrs, kindly helped with that chapter. Sawyer X, the current maintainer of `Module::Starter`, helped tremendously as we developed those parts of the book.

Thanks also to our many students who have let us know what parts of the course material have needed improvement over the years. It's because of you that we're all so proud of it today.

Thanks to the many Perl Mongers who have made us feel at home as we've visited your cities. Let's do it again sometime.

And finally, our sincerest thanks to our friend Larry Wall, for having the wisdom to share his really cool and powerful toys with the rest of the world so that we can all get our work done just a little bit faster, easier, and with more fun.

2. And yes, you can ask a Perl question at the same time. I don't mind.

Table of Contents

Foreword xi

Preface xiii

1. Introduction 1
What Should You Know Already? 2
strict and warnings 2
Perl v5.14 3
A Note on Versions 4
What About All Those Footnotes? 4
What's With the Exercises? 4
How to Get Help 5
What If I'm a Perl Course Instructor? 5
Exercises 6

2. Using Modules 7
The Standard Distribution 7
Exploring CPAN 8
Using Modules 9
Functional Interfaces 10
Selecting What to Import 11
Object-Oriented Interfaces 12
A More Typical Object-Oriented Module: Math::BigInt 12
Fancier Output with Modules 13
What's in Core? 14
The Comprehensive Perl Archive Network 15
Installing Modules from CPAN 16
CPANminus 17
Installing Modules Manually 17
Setting the Path at the Right Time 18
Setting the Path Outside the Program 21

Extending @INC with PERL5LIB 21
Extending @INC on the Command Line 22
local::lib 22
Exercises 23

3. Intermediate Foundations 25
List Operators 25
List Filtering with grep 26
Transforming Lists with map 28
Trapping Errors with eval 29
Dynamic Code with eval 31
The do Block 32
Exercises 33

4. Introduction to References 35
Doing the Same Task on Many Arrays 35
PeGS: Perl Graphical Structures 37
Taking a Reference to an Array 38
Dereferencing the Array Reference 41
Getting Our Braces Off 42
Modifying the Array 43
Nested Data Structures 44
Simplifying Nested Element References with Arrows 45
References to Hashes 47
Checking Reference Types 50
Exercises 52

5. References and Scoping 53
More than One Reference to Data 53
What If That Was the Name? 54
Reference Counting and Nested Data Structures 55
When Reference Counting Goes Bad 57
Creating an Anonymous Array Directly 59
Creating an Anonymous Hash 61
Autovivification 63
Autovivification and Hashes 66
Exercises 68

6. Manipulating Complex Data Structures 71
Using the Debugger to View Complex Data 71
Viewing Complex Data with Data::Dumper 75
Other Dumpers 77
Marshalling Data 78

Storing Complex Data with Storable 80
YAML 85
JSON 85
Using the map and grep Operators 86
Applying a Bit of Indirection 86
Selecting and Altering Complex Data 88
Exercises 90

7. Subroutine References 91
Referencing a Named Subroutine 91
Anonymous Subroutines 96
Callbacks 97
Closures 98
Returning a Subroutine from a Subroutine 100
Closure Variables as Inputs 103
Closure Variables as Static Local Variables 104
state Variables 105
Finding Out Who We Are 107
Enchanting Subroutines 108
Dumping Closures 111
Exercise 112

8. Filehandle References 115
The Old Way 115
The Improved Way 116
Filehandles to Strings 118
Processing Strings Line by Line 119
Collections of Filehandles 120
IO::Handle and Friends 121
IO::File 121
IO::Scalar 122
IO::Tee 123
IO::Pipe 124
IO::Null and IO::Interactive 125
Directory Handles 126
Directory Handle References 126
Exercises 127

9. Regular Expression References 129
Before Regular Expression References 129
Precompiled Patterns 131
Regular Expression Options 132
Applying Regex References 132

Regexes as Scalars 133
Build Up Regular Expressions 136
Regex-Creating Modules 137
Using Common Patterns 137
Assembling Regular Expressions 139
Exercises 140

10. Practical Reference Tricks 141
Fancier Sorting 141
Sorting with Indices 143
Sorting Efficiently 144
The Schwartzian Transform 145
Multilevel Sort with the Schwartzian Transform 147
Recursively Defined Data 147
Building Recursively Defined Data 149
Displaying Recursively Defined Data 152
Avoiding Recursion 153
The Breadth-First Solution 154
Exercises 156

11. Building Larger Programs 159
The Cure for the Common Code 159
Inserting Code with eval 160
Using do 161
Using require 163
The Problem of Namespace Collisions 164
Packages as Namespace Separators 165
Scope of a Package Directive 167
Packages and Lexicals 168
Package Blocks 169
Exercises 170

12. Creating Your Own Perl Distribution 173
Perl's Two Build Systems 173
Inside Makefile.PL 174
Inside Build.PL 175
Our First Distribution 176
h2xs 176
Module::Starter 177
Custom Templates 178
Inside Your Perl Distribution 178
The META File 180
Adding Additional Modules 181

Inside a Module 182
Plain Ol' Documentation 184
Pod Command Paragraphs 185
Pod Paragraphs 186
Pod Formatting Codes 186
Checking the Pod Format 187
The Module Code 187
Module Building Summary 188
Creating a Module::Build Distribution 188
Creating a ExtUtils::Makemaker Distribution 189
Exercises 189

13. Introduction to Objects 191
If We Could Talk to the Animals. . . 191
Introducing the Method Invocation Arrow 193
The Extra Parameter of Method Invocation 194
Calling a Second Method to Simplify Things 195
A Few Notes About @ISA 197
Overriding the Methods 198
Starting the Search from a Different Place 200
The SUPER Way of Doing Things 200
What to Do with @_ 201
Where We Are 201
Our Barnyard Summary 202
Exercises 203

14. Introduction to Testing 205
Why Should We Test? 205
The Perl Testing Process 206
Test Anywhere Protocol 206
The Art of Testing 208
A Test Example 209
The Test Harness 210
The Standard Tests 211
Checking that Modules Compile 212
The Boilerplate Tests 213
The Pod Tests 216
Adding Our First Tests 217
Measuring Our Test Coverage 220
Subroutine Coverage 221
Statement Coverage 221
Branch Coverage 221
Conditional Coverage 222

Exercises 222

15. Objects with Data 225
A Horse Is a Horse, of Course of Course—Or Is It? 225
Invoking an Instance Method 227
Accessing the Instance Data 228
How to Build a Horse 228
Inheriting the Constructor 229
Making a Method Work with Either Classes or Instances 230
Adding Parameters to a Method 230
More Interesting Instances 231
A Horse of a Different Color 232
Getting Our Deposit Back 233
Don't Look Inside the Box 234
Faster Getters and Setters 235
Getters that Double as Setters 236
Restricting a Method to Class Only or Instance Only 236
Exercise 237

16. Some Advanced Object Topics 239
UNIVERSAL Methods 239
Testing Our Objects for Good Behavior 240
The Last Resort 242
Using AUTOLOAD for Accessors 243
Creating Getters and Setters More Easily 244
Multiple Inheritance 246
Exercises 247

17. Exporter 249
What use Is Doing 249
Importing with Exporter 250
@EXPORT and @EXPORT_OK 251
Grouping with %EXPORT_TAGS 252
Custom Import Routines 254
Exercises 256

18. Object Destruction 257
Cleaning Up After Ourselves 257
Nested Object Destruction 259
Beating a Dead Horse 262
Indirect Object Notation 263
Additional Instance Variables in Subclasses 265
Using Class Variables 267