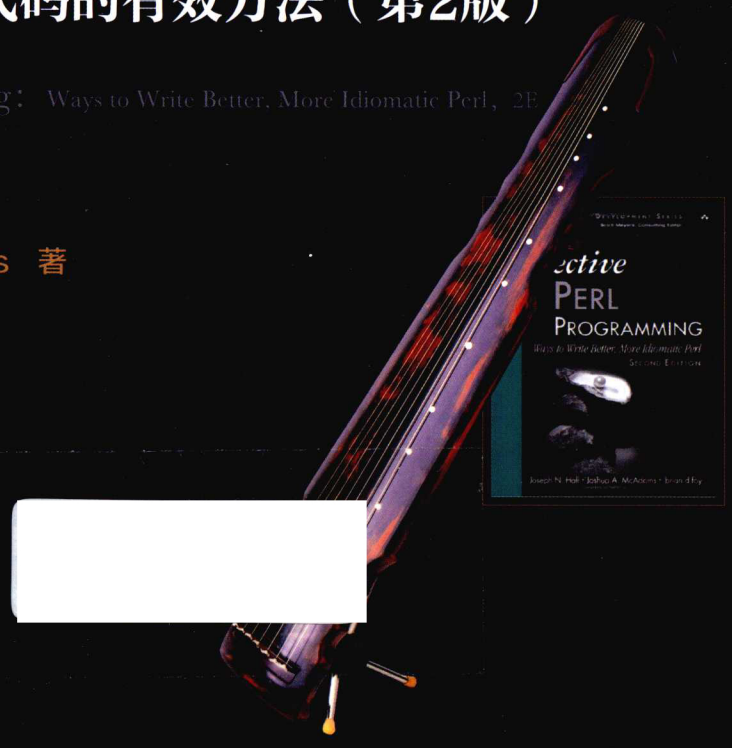
Broadview
www.broadview.com.cn
PEARSON
· 原味精品书系 ·

Effective Perl（英文版）

编写高质量Perl代码的有效方法（第2版）

Effective Perl Programming: Ways to Write Better, More Idiomatic Perl, 2E

[美] Joseph N. Hall
Joshua A. McAdams 著
brian d foy



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

· 原味精品书系 ·

Effective Perl（英文版）

编写高质量Perl代码的有效方法（第2版）

Effective Perl Programming: Ways to Write Better, More Idiomatic Perl, 2E

[美] Joseph N. Hall
Joshua A. McAdams 著
brian d foy

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内容简介

本书是 Perl 编程领域的“圣经”级著作。它提供了 100 多个翔实的应用案例，足以涵盖编程过程中经常遇到的方方面面，由此详细阐释出各种高效且简洁的写法。本书第 1 版曾畅销十年之久，而在第 2 版中不仅修正了前版存在的一些问题，更与时俱进地引入了许多 Perl 领域的新主题，使内容更加完善丰富，也更具实用性。

本书为初级 Perl 程序员铺就了一条通往高阶之路，而对高级 Perl 程序员而言，也是必备的技术参考书。

版权贸易合同登记号　图字：01-2015-6093

图书在版编目（CIP）数据

Effective Perl：编写高质量 Perl 代码的有效方法：第 2 版 =Effective Perl Programming: Ways to Write Better, More Idiomatic Perl: 2nd Edition：英文 /（美）霍尔（Hall,J.N.），（美）麦克亚当斯（McAdams,J.A.），（美）福瓦（foy,B.D.）著. — 北京：电子工业出版社，2016.4
（原味精品书系）
ISBN 978-7-121-27268-4
I. ① E…II. ① 霍… ② 麦… ③ 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（2015）第 227489 号

责任编辑：张　玲
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980 1/16　　印张：30.75　　字数：590 千字
版　　次：2016 年 4 月第 1 版
印　　次：2016 年 4 月第 1 次印刷
定　　价：89.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

推荐序

十年前，当我开始学习 Perl 时，我认为自己对这门语言已经了解得很多了——没错，对这门语言本身，我确实知道得很多。而我所不知道的，则是那些真正赋予 Perl 力量的惯用方法和其他灵活的语法结构。尽管不用它们也能写出绝大多数程序，但不掌握这些，则意味着自己的知识结构还不够完善，自己的工作效率也远远达不到理想状态。

我是幸运的，因为我得到了本书的第 1 版。不过，那本书从来没有机会停留在我的书架上，它一直都在我的包里，一有空我就会打开来读一段。

Joseph N. Hall 的这本书内容编排简单得让人爱不释手，每一段内容虽短，但都饱含智慧，而且讲得十分明白透彻。不瞒你说，我们免费的 Perl Tips 电子报（http://perltraining.com.au/tips/）正是受了本书的启发才创刊的，这份电子报一直致力于探讨 Perl 及其社区的发展。

对于一门语言来说，十年意味着很大的变化，而社区对语言的认知则有更大的变化。因此，让我非常高兴的不仅是听到这本书的第 2 版即将出版的消息，更重要的是这个新版本出自 Perl 社区最杰出的两位成员之手。

不用说，brian 对 Perl 的全心投入是有目共睹的。他不仅写了很多 Perl 语言方面的书，还负责出版一份杂志（*The Perl Review*），并且维护着 Perl 官方网站中的 FAQ（常见问题解答），另外他在众多 Perl 及编程语言社区一直享有盛誉。

而 Josh 则以他运营的著名播客网站 Perlcast 闻名，他从 2005 年就开始在这个网站中以音频形式播放 Perl 新闻了。Josh 总能找到那些著名的、有趣的人，对他们进行采访，这使他自己快速积累了大量知识，也让我对他羡慕不已。

总之，能向亲爱的读者朋友推荐本书的第 2 版，我感到荣幸之至。希望它能让你真正掌握这门语言的精髓，就像当年第 1 版对我的启蒙那样。

Paul Fenwick

Perl Training Australia 总裁

前言

再版前言

很多 Perl 程序员都是通过本书的第 1 版启蒙的。在 1998 年 Addison-Wesley 出版第 1 版的时候，整个世界似乎都在使用 Perl。当时 .com 大潮正在兴起，所有懂点 HTML 的人都能找到程序员的工作。而这些人一旦开始编程，就需要迅速提升自己的技能。本书和其他两本“圣经级”著作 *Programming Perl*[1]、*Learning Perl*[2] 基本上是这些新程序员的必读书。

当时市面上还有不少其他的 Perl 书籍。如今的编程学习者应该很难想象当时美国书店的情况，那时候的书店中有数十米的书架摆放的都是编程书，而大多数都是关于 Java 和 Perl 的。如今的书店则只在一个小角落里摆放编程书，每种语言往往只会有几本书，而且大多数的书在上架后的半年内就会被其他书取代。

尽管如此，本书还是畅销了十年之久。这要归功于 Joseph Hall 对 Perl 编程哲学的深刻理解和他本人的过人智慧。毕竟这本书主要讨论的是 Perl 编程思想，他在第 1 版中给出的建议直到现在都还非常实用。

不过，如今 Perl 的世界和 1998 年相比已经有了很大的变化，值得提倡的理念也更多了。CPAN（Comprehensive Perl Archive Network，Perl 综合典藏网）仅仅经过几年的发展，如今已经成了 Perl 最吸引人的特性。人们已经发现了许多更新更好的编程方式，而且这十年来业界积累了更多使用 Perl 的经验，也催生了很多新的最佳实践和惯用技巧。

自从本书第 1 版面世以来，Perl 本身也有了很大的变化。第 1 版存在于从 Perl 4 到 Perl 5 的过渡时期，当时大家仍然在广泛使用 Perl 4 的一些古老特性。在这个新版本中，我们基本上消除了这些差异。现在只有一个 Perl，那就是 Perl 5（本书不讨论 Perl 6，因为它值得另写一本书）。

现代 Perl 已经能够支持 Unicode（而不仅仅是 ASCII），因此你也应该适应这一点，我们为这个主题专门设置了一章。几年来，在 Michael Schwern 的推动之下，Perl 已经成为被测试最多的语言，几乎每一个模块都非常稳定。Perl 粉丝们怀念的“蛮荒时代”已经成为历史。今天，即使是快速原型的开发也可以同时考虑测试。如果你开发的是企业应用，那么你应

1 Larry Wall、Tom Christiansen 及 Jon Orwant 合著的 *Programming Perl, Third Edition*（O'Reilly Media，2000）。

2 Randal L. Schwartz、Tom Phoenix 及 brian d foy 合著的 *Learning Perl, Fifth Edition*（O'Reilly Media，2008）。

该好好看一看我们针对测试给出的建议。如果你是一位正则表达式高手，那么你一定想了解最新的 Perl 正则特性，本书将介绍其中那些最常用的。

Perl 仍然在成长中，新的主题还在不断涌现。有些主题本身就值得用一本书的篇幅来介绍，比如 Moose 这个“后现代”的 Perl 面向对象框架，因而本书也就不勉为其难地去涵盖它们了。另一些主题，比如 POE（Perl Object Environment，Perl 对象环境）、对象关系映射器，还有 GUI 工具包等也都因为同样的原因而没有办法在本书中详细介绍。不过，我们已经计划再写一本 More Effective Perl，到时候可能会涵盖更多的内容。

最后，Perl 的各种文档和专著较以前也丰富多了。虽然本书会尽可能多地讲到我们认为你应该知道的内容，但如果市面上已经有了详细讨论某些内容的书，我们自然也就不必置喙了，附录会推荐其他一些 Perl 图书，这样做无疑也给更深入地讨论当前本书中的这些主题留出了余地。

Joseph N. Hall、Joshua A. McAdams 和 brian d foy

第 1 版前言

我曾经写过大量 C 和 C++ 代码。在专注于 Perl 之前，我参与的一个重点项目是实现一门脚本语言，该语言能用来画圈、计算概率和生成 FrameMaker 格式的书。这个语言用了大约 5 万行与平台无关的 C++ 代码，其中有不少相当有趣的特性。应该说，这个项目还是非常有意思的。

它耗费了我两年的时间。

对我来说，大多数有趣的 C 或者 C++ 项目都需要数月乃至数年的时间才能完成。这是因为那些有趣的功能往往比较复杂，所以需要耗费很多开发时间。不过在换为一门高级语言之后，只要 3 个月的时间，我也可以把原先一大堆平淡无奇的想法变成有趣的项目。

这是我最初对 Perl 感兴趣的原因之一。吸引我的是 Perl 这个脚本语言中强大的字符串处理、正则表达式和流程控制能力。这正是那些 C 和 C++ 程序员（面对时间紧凑的项目时）最需要的特性。于是我学了 Perl，而且实实在在地喜欢上了它。这要归功于一个相关项目，我在该项目中负责实现处理文本文件的功能：获取一个程序的输出，格式化之后再发给另一个程序处理。我用 Perl 只花了不到一天就实现了这个功能，而如果使用其他语言，则恐怕需要几天甚至几周时间。

为何要写这本书

我一直都想成为一个作家。小时候我就迷上了科幻小说。我一直热衷于此，有时候一天能读三本，甚至还试着自己写（但写得不好）。之后的 1985 年，我参加了在密歇根州东兰辛市（East Lansing）举办的 Clarion 科幻小说作家培训班。随后一年左右的时间里，我偶尔会写一些短篇小说，不过从来没有发表过。后来，上课和工作占用的时间越来越多，我最终也就打消了写科幻小说的念头。不过，我还是坚持写作，只不过写的都是技术文档、教程、方案和文件。当然，我这些年来还陆续接触了好几个技术作者。

其中一个就是 Randal Schwartz。我在一个工程项目中聘用他给我帮忙达一年之久（这是我第一次做技术主管，也是我第一次管理软件开发类的项目，相信大多数认识 Randal 的人能猜到这点）。后来他选择离职去教 Perl，过了一段时间我也去了。

在这段时间中，我对写作的兴趣更浓厚了。在 C++、Perl、Internet 和 World Wide Web 等这些热门领域打拼了很多年，我觉得应该把其中一些有趣的东西写下来。应用和教授 Perl 的经验也让我的这个想法越来越强烈。我盼望着能写一本书，把自己日积月累的各种 Perl 技巧和反反复复遇到的陷阱汇集起来。

1996 年 5 月，在圣何塞的一次开发者大会上，我和 Keith Wollman 有了一次交谈。当时并没有谈到我想写书，我们只是讨论了哪些好的题材可以写成书。当谈到 Perl 的时候，他问我："你觉得一本名叫 *Effective Perl* 的书会不会受欢迎呢？"这个书名打动了我。要知道，Scott Meyers 的 *Effective C++* 是我最喜欢的一本 C++ 著作，而给该系列写一本 Perl 的书显然是个好主意。

Keith 的话始终在我耳边回响。过了一段时间，我在 Randal 的帮助下写了一个选题报告，而 Addison-Wesley 公司批准了这个选题。

接下来，好戏开场了。我开始没日没夜地写作，常常在电脑跟前一坐就是 12 个小时，除了用 FrameMaker 写作，还在 Perl 5 Porters 邮件列表中不厌其烦地问了不少的问题，查阅了几十本书和手册，编写了很多很多段 Perl 代码，也喝了很多很多罐健怡可乐和百事可乐。在查阅资料时，偶尔还会发现一些曾被自己忽略的最基础的 Perl 知识。就这样过了一段时间，本书的第一稿诞生了。

这本书是我的一个尝试，希望借此与大家分享我在学习 Perl 的过程中收获的经验和乐趣。最后，非常感谢你花时间阅读，希望这本书对你有价值，也能让你感到乐在其中。

Joseph N. Hall
亚利桑那州钱德勒市
1998 年

致谢

第 2 版致谢

许多人帮我们审阅了第 2 版，指出了我们忽略的一些问题。对此，我们要感谢 Abigail、Patrick Abi Salloum、Sean Blanton、Kent Cowgill、Bruce Files、Mike Fraggasi、Jarkko Hietaniemi、Slaven Rezic、Andrew Rodland、Michael Stemle 和 Sinan Ünür。书中的某些地方也会直接指明他们的贡献。

另外有些人则为我们做了更多，他们几乎针对书中的每一章都提出了问题。正是因为他们的努力，书中许多错误才会在付梓前得以消灭。他们是 Elliot Shank、Paul Fenwick 及 Jacinta Richardson。若还有错误，或许就只能归咎于我们没有管好自己的猫，这个调皮的小家伙一定是在我们不曾注意的时候到键盘上遛过弯儿。

Joseph N. Hall、Joshua A. McAdams 和 brian d foy

第 1 版致谢

这本书写来不易。我自认已经倾尽全力了，但如果不是得到了许多程序员、作者、编辑及其他专业人员的帮助肯定会更加艰难。我衷心感谢所有为本书面世而贡献了宝贵时间和精力的人。

Chip Salzenberg 和 Andreas “MakeMaker” König 帮我修正了不少程序漏洞，使得本书更加精练。对 Chip 的感激是无法用言语表达的。我对程序的关注实在太少了，向 Chip 致敬！

Perl 5 Porters 邮件列表工作组也起到了很大的作用，他们为我解答了不少问题。其中尤其要感谢的是 Jeffrey Friedl、Chaim Frenkel、Tom Phoenix、Jon Orwant（以 *The Perl Journal* 杂志闻名）和 Charlie Stross。

Randal Schwartz 是畅销书作者、教师和 Just Another Perl Hacker 的发起人，他也是我最主要的技术审稿人。所以如果你发现书中有任何问题，可以直接给他写邮件（开玩笑的）。非常感谢 Randal，因为他在这本书上付出了许多时间和精力。

感谢 Larry Wall 创造了 Perl，而他本人也解答了我很多的疑问，并且在一些地方提出了建议。

能和 Addison-Wesley 在这个项目上合作，我觉得非常荣幸。这里遇到的每个人都善良而乐于助人，特别要感谢 Kim Fryer、Ben Ryan、Carol Nelson 和 Keith Wollman。

还有许多朋友在其他方面提供了帮助，Nick Orlans、Chris Ice 和 Alan Piszcz 都耗费了大量时间阅读初稿。我的几位现任及前任老板 Charlie Horton、Patrick Reilly 和 Larry Zimmerman 则提供了许多灵感，也给了我很大鼓励。

另外，我在写作过程中尽可能坚持原创，但仍不可避免地受到 Perl 在线手册和 *Programming Perl* 等资料的影响。殊途同归，我已尽力用最有创意的方法来阐述，但很多情形下那些有关 Perl 的经典诠释是难以超越的。

非常感谢 Jeff Gong，他总是帮我“骚扰”电话公司，从而让我的 T-1 线路保持畅通。Jeff 总是懂得如何让客户开心。

非常感谢高尔夫这项运动，击球的简单动作能够让我保持清醒，并帮我排解压力。同样地，还要感谢《猎户座之王》和《文明 II》两款游戏。

最后，我必须感谢 Donna，我的未婚妻和终生伴侣，她也是专业的程序员。没有她的支持、鼓励和爱，这本书就无法写成。

Joseph N. Hall
1998 年

关于作者

Joseph N. Hall，自称“电脑神童”，一路玩着得州仪器的可编程计算器和配有 4KB 内存的 Radio Shack TRS-80 Model 1 长大。14 岁时第一次教授计算机课程。Joseph 拥有北卡罗来纳州立大学计算机科学学士学位，自 1984 年起开始以编程为生。他从 1987 年起开始使用 UNIX 和 C，自 1993 年以来一直在使用 Perl。他的兴趣涉及软件工具和编程语言、钢琴和电子琴，以及高尔夫。

Joshua A. McAdams 活跃于 Perl 社区已近 5 年。他创办了 *Perlcast*，主持过两届在芝加哥的 YAPC::NA，他为 Chicago.pm 举办会议，在世界各地的 Perl 会议上做过发言，是一位 CPAN 作者。这是他的图书处女作，不过此前已为 *The Perl Review* 和 Perl Advent Calendar 写过文章。至于日常工作，Josh 就职于 Google，在那里他的日常开发并不一定涉及 Perl，不过只要可能他就会使用。

brian d foy 是《Perl 语言入门（第 5 版）》和 *Intermediate Perl* 的合著者，以及《精通 Perl》的作者。他发起了第一个 Perl 用户组 New York Perl Mongers，出版了 *The Perl Review*，维护着部分 Perl 核心文档，是一名 Perl 讲师，并常在大会上发言。

目录

Introduction

"Learning the fundamentals of a programming language is one thing; learning how to design and write effective programs in that language is something else entirely." What Scott Meyers wrote in the Introduction to Effective C++ is just as true for Perl.

Perl is a Very High Level Language—a VHLL for the acronym-aware. It incorporates high-level functionality like regular expressions, networking, and process management into a context-sensitive grammar that is more "human," in a way, than that of other programming languages. Perl is a better text-processing language than any other widely used computer language, or perhaps any other computer language, period. Perl is an incredibly effective scripting tool for UNIX administrators, and it is the first choice of most UNIX CGI scripters worldwide. Perl also supports object-oriented programming, modular software, cross-platform development, embedding, and extensibility.

Is this book for you?

We assume that you already have some experience with Perl. If you're looking to start learning Perl, you might want to wait a bit before tackling this book. Our goal is to make you a better Perl programmer, not necessarily a new Perl programmer.

This book isn't a definitive reference, although we like to think that you'd keep it on your desktop. Many of the topics we cover can be quite complicated and we don't go into every detail. We try to give you the basics of the concepts that should satisfy most situations, but also serve as a starting point for further research if you need more. You will still need to dive into the Perl documentation and read some of the books we list in Appendix A.

There is a lot to learn about Perl.

Once you have worked your way through an introductory book or class on Perl, you have learned to write what Larry Wall, Perl's creator, fondly refers to as "baby talk." Perl baby talk is plain, direct, and verbose. It's not bad—you are allowed and encouraged to write Perl in whatever style works for you.

You may reach a point where you want to move beyond plain, direct, and verbose Perl toward something more succinct and individualistic. This book is written for people who are setting off down that path. *Effective Perl Programming* endeavors to teach you what you need to know to become a fluent and expressive Perl programmer. This book provides several different kinds of advice to help you on your way.

- **Knowledge, or perhaps, "Perl trivia."** Many complex tasks in Perl have been or can be reduced to extremely simple statements. A lot of learning to program effectively in Perl entails acquiring an adequate reservoir of experience and knowledge about the "right" ways to do things. Once you know good solutions, you can apply them to your own problems. Furthermore, once you know what good solutions look like, you can invent your own and judge their "rightness" accurately.
- **How to use CPAN.** The Comprehensive Perl Archive Network is modern Perl's killer feature. With over 5 gigabytes of Perl source code, major frameworks, and interfaces to popular libraries, you can accomplish quite a bit with work that people have already done. CPAN makes common tasks even easier with Perl. As with any language, your true skill is your ability to leverage what has already been done.
- **How to solve problems.** You may already have good analytical or debugging skills from your work in another programming language. This book teaches you how to beat your problems using Perl by showing you a lot of problems and their Perl solutions. It also teaches you how to beat the problems that Perl gives you, by showing how to efficiently create and improve your programs.
- **Style.** This book shows you idiomatic Perl style, primarily by example. You learn to write more succinct and elegant Perl. If succinctness isn't your goal, you at least learn to avoid certain awkward constructs. You also learn to evaluate your efforts and those of others.
- **How to grow further.** This book doesn't cover everything you need to know. Although we do call it a book on advanced Perl, not a whole lot of advanced Perl can fit between its covers. A real compendium of

advanced Perl would require thousands of pages. What this book is really about is how you can make yourself an advanced Perl programmer—how you can find the resources you need to grow, how to structure your learning and experiments, and how to recognize that you have grown.

We intend this as a thought-provoking book. There are subtleties to many of the examples. Anything really tricky we explain, but some other points that are simple are not always obvious. We leave those to stand on their own for your further reflection. Sometimes we focus on one aspect of the example code and ignore the surrounding bits, but we try to make those as simple as possible. Don't be alarmed if you find yourself puzzling something out for a while. Perl is an idiosyncratic language, and in many ways is very different from other programming languages you may have used. Fluency and style come only through practice and reflection. While learning is hard work, it is also enjoyable and rewarding.

The world of Perl

Perl is a remarkable language. It is, in our opinion, the most successful modular programming environment.

In fact, Perl modules are the closest things to the fabled "software ICs" (that is, the software equivalent of integrated circuits, components that can be used in various applications without understanding all of their inner workings) that the software world has seen. There are many reasons for this, one of the most important being that there is a centralized, coordinated module repository, CPAN, which reduces the amount of energy wasted on competing, incompatible implementations of functionality. See Appendix A for more resources.

Perl has a minimal but sufficient modular and object-oriented programming framework. The lack of extensive access-control features in the language makes it possible to write code with unusual characteristics in a natural, succinct form. It seems to be a natural law of software that the most-useful features are also the ones that fit existing frameworks most poorly. Perl's skeletal approach to "rules and regulations" effectively subverts this law.

Perl provides excellent cross-platform compatibility. It excels as a systems administration tool on UNIX because it hides the differences between