TURING 图灵程序设计丛书

Addison Wesley

Effective Perl Programming
Ways to Write Better, More Idiomatic Perl Second Edition

Perl 高效编程
（第2版）

Joseph N. Hall
[美] Joshua A. McAdams 著
brian d foy
盛春 王晖 张东亮 蒋永清 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Effective Perl Programming
Ways to Write Better, More Idiomatic Perl Second Edition

Perl高效编程
（第2版）

Joseph N. Hall
[美] Joshua A. McAdams 著
brian d foy
盛春 王晖 张东亮 蒋永清 译

人民邮电出版社
北京

图书在版编目（CIP）数据

Perl高效编程 : 第2版 / (美) 霍尔 (Hall, J. N.) , (美) 麦克亚当斯 (McAdams, J. A.) , (美) 福瓦 (Foy, B. D.) 著 ; 盛春等译. -- 北京 : 人民邮电出版社, 2011.5
(图灵程序设计丛书)
书名原文: Effective Perl Programming: Ways to Write Better, More Idiomatic Perl, Second Edition
ISBN 978-7-115-25046-9

Ⅰ. ①P… Ⅱ. ①霍… ②麦… ③福… ④盛… Ⅲ. ①Perl语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2011)第053111号

内 容 提 要

本书是Perl编程领域的“圣经级”著作。它提供了一百多个详实的应用案例，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程序员来说，本书也是必备的技术参考。

图灵程序设计丛书

Perl 高效编程（第2版）

◆ 著 [美] Joseph N. Hall Joshua A. McAdams brian d foy
译 盛 春 王 晖 张东亮 蒋永清
责任编辑 朱 巍
执行编辑 王一枝

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：20.75
字数：516千字 2011年5月第1版
印数：1－3 000册 2011年5月北京第1次印刷

著作权合同登记号 图字：01-2010-5083号

ISBN 978-7-115-25046-9

定价：65.00元

读者服务热线：(010)51095186转604 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

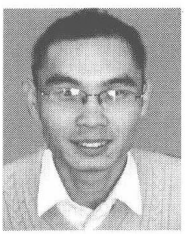
译 者 简 介

盛春

大二开始自学 Perl 编程，在通读完《Perl 5 详解》后利用暑期打工的机会，专职于 Perl 语言的 CGI 开发。毕业后的工作也一直以 Perl 语言作为主要开发工具，目前就职于思科 IronPort 的邮件及 Web 安全部门，负责中文反垃圾邮件规则的开发和维护以及内部工具和报告的开发。早年曾为 PerlChina 筹建社区站点，翻译过数篇技术文章。2007 年和蒋永清合作翻译《Perl Testing 程序高手秘笈》，2009 年与蒋永清、王晖合作翻译《Perl 语言入门（第 5 版）》，2010 年再度合作，翻译了即将出版的《使用 Perl 实现系统管理自动化》以及这本《Perl 高效编程》。此外，在 2009 年还主持翻译了开源图书《Pro Git》。

王晖

《Perl 语言入门（第 5 版）》及《使用 Perl 实现系统管理自动化》的合译者。接触到 Perl 是在 2000 年，很快喜欢上了这门语言和它的社区，熟悉 Linux/ Unix，在盛春的影响下成为 Mac 用户，目前和盛春一样服务于 IronPort 的邮件及 Web 安全部门，大部分工作都是在 FreeBSD 下使用 Perl 来完成的，包括数据分析、采集、系统管理及 Web 应用等。希望国内能有更多的 Perl 爱好者及基于 Perl 的应用出现。



张东亮

2004 年接触并喜欢上了 Perl，对正则表达式相逢恨晚。建有个人博客“我爱正则表达式”，专用来记录 Perl 等脚本语言中正则表达式的学习心得和应用技巧。目前服务于 IronPort 的邮件及 Web 安全部门，负责维护反垃圾邮件/反病毒系统和内部网络的建设。

蒋永清

1997 年在一台收银机上开始了 Linux 的自学，2002 年夏开始接触 Perl 编程，2003 年开始参与 PerlChina 技术论坛。2004 年至 2009 年完成了数十篇技术文章和两本 Perl 书籍的翻译。2010 年起在北京从事互联网维护工作，随后翻译完成了另外两本 Perl 书籍。目前和家人、孩子、朋友一起在北京生活。

推 荐 序

十年前，当我开始学习 Perl 的时候，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 这本书的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。不瞒您说，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这本书的第 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

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法。

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，那应该另写一本书）。

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了。另一些主题，

---

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）。

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*。

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

许多人帮我审阅第 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出他们的贡献。

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 以及 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蹓过弯儿。

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、编辑以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市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 的感激光用言语是不足以表达的。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[①]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。

① The Perl Journal 由 Jon Orwant 创立，该杂志已成为联系 Perl 社区的纽带和讨论 Perl 发展的前沿阵地。——编者注

还有许多朋友在其他方面提供了帮助，Nick Orlans、Chris Ice 和 Alan Piszcz 都耗费了大量时间用于阅读初稿。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

引　言

“学习某种语言的基础是一回事，但是知道如何有效使用这个语言进行程序设计则完全是另一回事”，这是 Scott Meyers 在 *Effective C++* 的简介中所说的，这句话对 Perl 来说也同样适用。

Perl 是一门非常高级的语言，可以称为 VHLL（Very High Level Language）。它集成了许多高级语言的功能，比如正则表达式、网络管理和进程管理，而且它的语法也是非常“人性化”的。对于文本处理来说，Perl 比其他常见的计算机语言要好用得多，可以说它是这个领域最好的语言。Perl 对于 Unix 系统管理来说也是极为高效的脚本工具，同时也是 UNIX CGI 脚本开发的最佳选择。Perl 还支持面向对象编程、模块化设计、跨平台开发、嵌入式开发，具有高度的可扩展性。

这本书适合你吗

本书假设你已经有了一些 Perl 开发经验。如果你正准备开始学习 Perl，那么这本书对你来说可能有些早。我们的目标是让你成为一个好的 Perl 程序员，而不只是一个普通的 Perl 程序员。

这本书算不上参考手册，虽然我们确实希望你常常把它放在案头。书中涉及的许多主题比较繁杂，因而我们难以一一深入介绍。不过我们会尝试告诉你最基本的理念，而它们在大多数情况下会很有用。不过这些都还只是敲门砖，你如果真的对此感兴趣的话，还需要更深入地研究一番才行。所以，你仍然需要不时地翻阅一下 Perl 文档，当然也包括附录中提到的那些书。

Perl 有许多值得学的地方

你若读过 Perl 的入门教材，或者学完了相关的课程，那就可以开始写点程序了。不过 Perl 的创造者 Larry Wall 喜欢把这时候写出来的程序称为“咿呀学语”。很多的 Perl 程序都可以归到这一类，它们简单、直白，而且冗长。这不是什么坏事，你可以用适合自己的任何风格来编写 Perl 程序。

但到了某个阶段，你可能会不满于这样的咿呀学语，而希望学习更简洁、更独特的写法。这本书就是为这样的你准备的。它的目标就是让你能写出流畅且表达能力强的 Perl 程序。为此我们会从以下几方面给你一些建议。

- 知识，或者称为“小技巧”。许多很复杂的 Perl 任务其实都有更简单的实现方法，甚至完全有可能只用短短几条语句来实现。Perl 高效编程的秘诀在于积累足够的经验，学会那些做事情的“正确”方法。首先，在看到你认为好的解决方案后，可以用它来解决自己的

问题。其次，当你对什么方案才能算好有了自己的认识后，你也可以自己创造新的方案，然后激扬文字，指出它到底好在哪里。

- 如何使用 CPAN。CPAN 如今已经成了吸引人们学习 Perl 的主要原因。使用其中超过 5G 的源代码、许多重要的编程框架以及各式常用库程序接口，能让你利用他人的成果快速完成许多任务。CPAN 也由此使得常规 Perl 编程任务得以简化。就像学习其他语言那样，你的技能就体现在充分利用已有的成果上。
- 如何解决问题。你可能已从其他语言学会了许多分析问题和调试错误的技巧。这本书通过展示多个问题及其 Perl 解决方案，教你如何使用 Perl 解决问题，同时也通过展示如何高效开发和改善程序，教你如何解决使用 Perl 时遇到的问题。
- 风格。本书主要通过示例来展现 Perl 语言的惯用风格，并教你写出更简洁优雅的 Perl 程序。如果简洁优雅不是你的目标，你至少能了解到如何避免写出某些笨拙的架构。另外，你也能学着品评自己或他人的程序。
- 如何进一步成长。这本书不可能涵盖你想知道的所有东西。虽然我们力图写一本高阶的 Perl 编程资料，但是要想真的涉及所有的高阶主题实在不切实际，因为哪怕只是罗列个大概也可能需要上千页的篇幅。所以本书的主要目标是让你具备成为高级 Perl 程序员的潜质，也就是说使你具备自我提升的能力，这包括如何找到需要的资源，如何不断通过实践来学习，以及如何评断自己的水平。

我们力图把这本书写成一本鼓励思考的书。大多数示例都有不少精微之处，我们会着重阐释那些比较复杂的，而简单的就不再赘言了，我们希望读者自己去领会。有时候我们会特别介绍某段示例代码的一个方面，而对其他部分则略过不表，但是大体上会使其尽可能简单，而又不影响你的理解。如果一时读不懂你也不必紧张，Perl 是一门独特的语言，和你学过的大多数语言都不一样，所以有时候必须得反复练习才能掌握某些窍门。学习的过程固然辛苦，但其间也蕴含着各种各样的乐趣和意想不到的收获。

Perl 的世界

Perl 是一门卓越的语言。在我们看来，它是最成功的模块化编程语言。

事实上，Perl 模块具有“软件芯片”的美名，因为它们常见于各类软件中。（在这里，软件等价于集成电路，是一种可以用于各类应用，而不必理解其内在工作原理的部件。）原因有很多，其中最重要的是因为存在集中协作的模块库 CPAN，它降低了我们在竞争、不兼容功能实现上所消耗的精力。参阅附录可以了解更多资源。

Perl 内建了最基础但足够充分的模块化和面向对象编程框架。由于其中缺乏严格的对象访问限制，Perl 可以写出非常自然简洁的代码。软件行业似乎有个自然法则，那就是最有用的功能却往往来自使用框架和模板最少的开发环境。而 Perl 正是以某种颠覆的方式实现了对这一“规则和惯例”的支持。

Perl 还提供了卓越的跨平台支持。而它之所以适合作为 UNIX 平台的系统管理工具，也正是

因为它把 UNIX 各个版本之间的差别尽可能地隐藏起来了。你能编写跨平台的 Shell 脚本吗？能，但实在太复杂了。大多数人应该不会进行这样的尝试。你能编写跨平台的 Perl 脚本吗？能，很简单。Perl 程序也可以很好地在 UNIX 和 Windows、VMS 等诸多其他平台间移植。

作为一名 Perl 程序员，你能得到一些世界上最好的资源支持。因为你可以获得所有可用模块的源代码，同时还有语言本身的源代码。你如果觉得自己找出代码缺陷太慢，还可以在网上获得 24/7 的技术支持。如果你不满意免费的技术支持，那么还可以考虑购买商业技术支持。

最终，你有了一门特立独行的语言。Perl 十分流畅，至少在目前几种脚本语言里，Perl 最具表达能力，甚至能达到随心所欲的境界（Do What I Mean，可简称为 DWIM）。这个思路或许有些唬人，却揭示了计算机技术的真正发展进程，它超越了时钟频率、硬盘空间和内存大小等物理性能指标的提升。

术语

通常来说，Perl 语言用到的术语和其他语言中的大致相同，当然一些术语也会有独特的含义。随着 Perl 的发展，其中一些术语已经渐渐不用了，同时也有一些新的术语生成。

通常我们说到语言本身的时候用的是 Perl（P 大写），而说到解释器和源程序的时候用的是 `perl`。大多数时候我们不会特别讨论解释器，所以通常你看到的是 Perl。

操作符（operator）在 Perl 中是一个不需要圆括号的动词（当然它的参数要放在括号中）。

列表操作符（list operator）则是一个标识符，后跟一个以逗号分隔的元素列表：

```
print "Hello", chr(44), " world!\n";
```

Perl 中的**函数**（function）是一个标识符，后跟一对圆括号，其中可以列出所有参数：

```
print("Hello", chr(44), " world!\n");
```

现在你可能看出来了：在 Perl 中列表操作符和函数很类似。实际上，两种语法有相同的功能。一般来说，我们会尽可能使用操作符这个词来描述 Perl 的内置函数，如 `print` 和 `open`，只会偶尔使用函数这个称呼，这两者之间没有本质区别。

在 Perl 中，子程序就叫**子程序**（subroutine），当然称它为“函数”、“操作符”甚至“过程”都是可以接受的。但是请注意，Perl 中“函数”的用法与数学中定义的不同，一些计算机科学家对这个词的滥用颇有微词。

所有的 Perl **方法**（method）都是遵从某种约定的子程序。这些约定既不是 Perl 要求的，也不被正式承认。不过，把这种调用称为“方法调用”还是比较合适的，因为 Perl 为了支持面向对象编程而实现了这种特殊的语法。方法和普通子程序的区别在于：方法的作者期望你按方法调用的语法来呼叫它。

Perl 的**标识符**（identifier）类似于 C 语言，是以字母或者下划线开始，后面跟着一个以上字母、数字或下划线的词。标识符用来命名 Perl 变量。另外，Perl 变量就是组合了某些特定符号的标识符，比如 `$a` 或 `&func`。

虽然我们并不是刻意要使用 Perl 内部的称呼，不过确实可以把 Perl 里面那些具有特别句法意义的标识符称为**关键词**（keyword），比如 `if` 和 `while`。相比之下，其他具有“函数”或“操作符”语法的标识符，比如 `print` 和 `oct`，可以称为**内置函数**（built-in）。

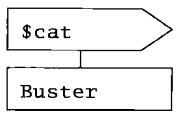
左值（lvalue）是指可以出现在赋值符号左边的值。这是这个词的一般意思，不过对 Perl 来说，某些特别的结构也可以作为左值，比如 `substr` 操作符。

使某个变量**本地化**（localizing）意味着使它的作用域局限在某个代码块或文件的“范围”内。特殊变量必须用 `local` 操作符来实现本地化。而普通变量则可以用 `my` 或者 `local` 来本地化（参考第 4 章的条款 43）。事实上，这是 Perl 的一个设计失误，Larry Wall 当年其实想用另一个名字来代替 `local` 的，但目前只能这样沿用下去了。有时候为了有所区别，我们会明确地说“用 `my` 来本地化”。

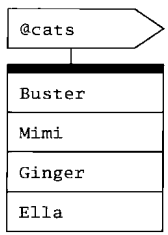
图解

在本书中，我们会使用 Joseph 的 PEGS（Perl Graphical Structure，Perl 图形化结构）来阐明数据结构。这种图示方法非常简单易懂，这里大致介绍一下。

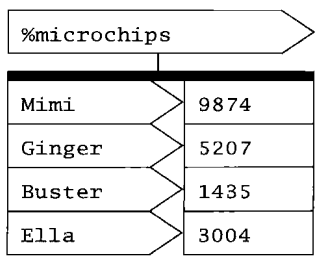
变量是带有名字的值。变量名出现在值的上方，用一侧尖端的方框表示。标量值使用简单的方框来表示：



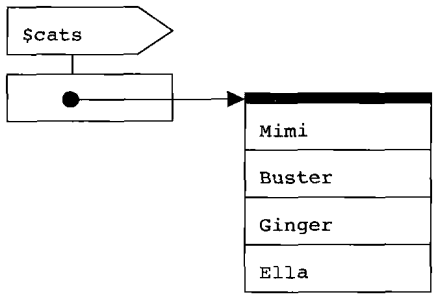
数组和列表采用相似的图形。其中数值用最上方有粗条的栈表示：



散列则在栈名旁列出相应的值：



引用的画法和 LISP 语言曾使用的图形化方法相似，使用点号加上箭头的方式表示：



以上就是数据结构的所有基本图示方法。

Perl 风格

从这本书中你应该还能领会到一些好的 Perl 编程风格。

当然了，所谓风格也是个人的喜好和考虑。我们不认为自己掌握的是最佳风格，不过还是希望读者会看到一种鲜活、实用而高效的编程风格。不过，有时候为了增加数据的可读性，我们也会暂时牺牲在风格方面的追求。基本上，本书的风格是遵从 **perlstyle** 文档的。

此外，考虑到书中篇幅对代码特有的限制，我们会特别注意行的长度，避免列出那种长篇大段（需要好几页）的代码，并尽可能避免采用那些太过冗长、枯燥乏味的代码作为例子。我们要求每个例子都能突出一两个重点，以避免使读者在代码中迷失。因此，你可能会发现它们不一定遵从最佳实践的要求。

在某些例子中，我们可能会为了突出重点而略过一些细节。在某些代码中，我们使用“...”来代表被省略的代码。（不过，到了本书面世的时候，你可能会发现这个“...”也成了一个合法的语句。因为 Perl 5.12 引入了 yadda yadda 操作符，这使得该语句能成功通过编译，仅在执行时会产生运行时错误。但这是编写桩代码的一种好方法。）

有些例子需要特殊版本的 Perl 才能运行。这里我们如果不特别声明的话，就是需要 Perl 5.8，这个版本稍微有点老，但却非常成熟。如果用到了 Perl 5.10 的特性，那么我们会在例子的第一行注明（例如第 1 章的条款 2）：

```
use 5.010;
```

另外，我们不会介绍 Perl 的开发版本，它们的小版本号一般为奇数，比如 Perl 5.009 或者 Perl 5.011。我们介绍某个特性的时候，会列出引入它的首个稳定版本。

并非每段代码都能通过 `warnings` 或者 `strict` 的约束（见条款 3）。我们建议所有的 Perl 程序员都遵从这两条最基本的约束。不过如果举例时总是提前声明这些约束，也可能会让读者抓不住我们所要论述的重点，因而我们舍弃了这种做法。如果合适，我们会尽可能遵守这些约束，但有时候从大段程序中抽取出来的代码并不一定和原来的程序一样严谨。

我们一般也会尽量减少标点符号的使用（见条款 18）。当然我们不是要鼓励大家尝试“Perl 高

尔夫”[1]，这个游戏关心的是程序最短能写成什么样。我们只是试图去除那些不必要的字符，而更多使用空白，这样能突出真正重要的部分，而不是那些死板的模式。

最后，我们尽可能选取有实用价值的代码。虽然并非所有例子都特别有用，但我们已尽可能采用某些实际应用程序中的代码。

组织结构

前两章主要是为了之后的章节做铺垫，使得读者能够逐渐适应较复杂的编程，而后的内容就会比较有挑战性。阅读过程中，请随时查看本书的目录和 Perl 文档（可以访问 http://perldoc.perl.org/）。

在第 2 版中，我们重新组织了书的结构。我们把第 1 版中某些条款分解成了多个新的条款，同时也对某些进行了合并或删除，因为其他的书中可能已经涉及了这些主题。附录则列出了一些你可能会用到的资源。

这本书的内容并非仅限于这些纸张之上，我们同时还架设了一个网站（http://effectiveperl-programming.com/）。那里你会发现更多与本书有关的信息，有些是我们遗漏的，有些是还未来得及写入本书的，还有些是其他 Perl 相关的有用话题。

① Perl 高尔夫（Perl Golf）是一种不定时举行的 Perl 程序设计游戏，以程序编码最短者获胜，就像高尔夫球中最少杆者获胜一样，详细可参考：http://perlgolf.sourceforge.net/。——编者注