Advanced Perl Programming

高级Perl编程

O'REILLY®
中国电力出版社

Sriram Srinivasan 著
Perlish 译

高级Perl编程

Sriram Srinivasan 著

Perlish 译

O'REILLY®

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

中国电力出版社

图书在版编目（CIP）数据

高级 Perl 编程 /（美）斯里尼瓦桑（Srinivasan, S.）编著；Perlish 译 . - 北京：中国电力出版社，2001. 2

书名原文：Advanced Perl Programming

ISBN 7-5083-0512-4

Ⅰ. 高 … Ⅱ. ①斯 … ② P… Ⅲ. Perl 语言 - 程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2001）第 02373 号

北京市版权局著作权合同登记

图字：01-2000-3956 号

书　　名 / 高级 Perl 编程

书　　号 / ISBN 7-5083-0512-4

责任编辑 / 刘江

封面设计 / Ellie Volckhausen，Hanna Dyer，张健

出版发行 / 中国电力出版社

地　　址 / 北京三里河路 6 号（邮政编码 100044）

经　　销 / 全国新华书店

印　　刷 / 北京市地矿印刷厂

开　　本 / 787 毫米 × 1092 毫米　16 开本　31 印张　480 千字

版　　次 / 2001 年 2 月第一版　2001 年 2 月第一次印刷

印　　数 / 0001-5000 册

定　　价 / 69.00 元（册）

译者序

对一种计算机语言优越性的衡量不是看有多少人在使用它，而是看人们是否喜欢它。Perl被程序员称为“瑞士军刀”，有着众多坚定的支持者和拥护者，是一件为专业人员打造的利器。许多复杂应用在Perl程序员看来可以轻松搞定，极高的编程效率是他们常常面带优越感的原因。Perl博大精深，拥有自己的文化。Perl有着融合其他优秀语言的长处。

本书在Perl社团中被昵称为“Panther Book(黑豹书)”，是众多Perl黑客必读书中的一本（其他几本还包括《Programming Perl》，《Perl Cookbook》等，也即将由中国电力出版社陆续出版）。它描述了Perl的体系结构，涵盖了许多其他有关Perl的图书所没有涉及到的极有价值的专题，在每个专题结尾还将Perl与Python、Tcl、Java和C/C++语言进行了颇有意趣的比较，可以使你对Perl本身的理解上升到一个崭新的高度。作者对Perl的精通令人钦佩。针对每一个专题，作者不是单纯的讲述如何使用该语言去做，而是精辟的阐述了这样做的原因和内部机理。你从这本书中获得的不仅仅是对Perl的透彻理解，还能学到许多许多编程思想，和深入挖掘计算技术的方法和指导。译者本人就获益匪浅。希望将这本有价值的图书奉献给国内的Perl程序员们，能促进国内Perl社团的发展，让更多的Perl黑客有机会深入理解Perl，为CPAN做出自己的贡献。

由于时间仓促，译文中难免会有不妥甚至错误的地方，希望读者批评指正。

—— *Perlish*

2001年元旦

目录

前言 1

第一章 数据引用与匿名存储 17

对已有变量的引用 19
使用引用 26
嵌套数据结构 32
引用的查询 34
符号引用 35
内部工作细节 36
其他语言中的引用 41
相关资源 42

第二章 实现复杂的数据结构 43

用户定义数据结构 44
例子：矩阵 45
教授，学生与课程 49
颁奖 53

格式化打印工具 56
相关资源 60

第三章 Typeglob 和符号表 61
Perl 变量，符号表和作用域 62
Typeglob 66
Typeglob 与引用 71
文件句柄，目录句柄及打印格式 73

第四章 子例程引用与闭包 77
子例程引用 78
使用子例程引用 80
闭包 83
闭包的应用 86
和其他语言的比较 92
相关资源 93

第五章 Eval 94
字符串形式：表达式计算 95
代码块形式：例外处理 97
注意你的引号 100
应用 Eval 来进行表达式计算 101
应用 Eval 来提高运行效率 103
在超时中应用 Eval 110
其他语言中的 Eval 112
相关资源 114

第六章 模块 115
包的基本知识 116
包与文件 118

包的初始化与销毁 120
私有性 121
符号的导入 123
包的嵌套 126
自动加载 127
存取符号表 128
与其他语言的比较 130

第七章 面向对象编程 133
面向对象简介 133
Perl 中的对象 135
UNIVERSAL 151
习惯的更新 153
与其他面向对象语言的对比 157
相关资源 159

第八章 面向对象：下面的几步 161
高效的属性存储 161
代理 174
关于继承 175
相关资源 178

第九章 绑定 179
标量变量的绑定 180
数组的绑定 183
散列表的绑定 186
文件句柄的绑定 188
例子：对变量的监控 189
与其他语言的比较 194

第十章 持续性 196
有关持续性的问题 197
流式数据 199
面向记录的方案 202
关系数据库 205
相关资源 212

第十一章 对象持续性的实现 214
适配器介绍 216
设计注意事项 219
实现 226
相关资源 236

第十二章 使用套接字进行网络编程 238
网络计算入门 238
Socket API 和 IO::Socket 240
同时处理多个客户端 243
现实世界中的服务器 249
IO 对象和文件句柄 250
预编译的客户端模块 252
相关资源 254

第十三章 网络计算：RPC 的实现 255
Msg:消息传递工具包 255
远程过程调用（RPC） 270
相关资源 276

第十四章 使用 Tk 进行用户界面编程 278
对 GUI，Tk 和 Perl/Tk 的介绍 279
开始使用 Perl/Tk 280

组件之旅 283
布局管理 303
定时器 307
事件联编 307
事件循环 310
相关资源 312

第十五章 GUI 实例：Tetris 313
有关 Tetris 的介绍 314
设计 315
实现 316

第十六章 GUI 实例：Man 页面查看器 324
Man 与 perlman 325
实现 326
相关资源 334

第十七章 模板驱动的代码生成 335
有关代码生成的问题 335
Jeeves 的例子 339
Jeeves 概述 344
Jeeves 的实现 346
规格语法分析器样例 355
相关资源 357

第十八章 扩展 Perl：第一课 359
编写一个扩展：概述 360
例子：Perl 与分形计算 364
SWIG 的功能 368
XS 的功能 371

自由度 375
分形介绍 376
相关资源 380

第十九章 Perl的嵌入：简单的方式 381
为什么要嵌入？ 381
解释器嵌入概述 383
例子 386
增加扩展 390
相关资源 391

第二十章 Perl的内部工作 392
阅读源代码 393
体系结构 394
Perl的值类型 402
堆栈与消息传递协议 429
内涵丰富的扩展 436
简单的嵌入式API 448
未来展望 451
相关资源 453

附录一 Tk组件参考 455

附录二 语法概要 472

词汇表 481

前言

错误，就像稻草，漂浮在水面上；
搜寻珍珠的人必须潜入水下。

—— 约翰·德莱登（译注1）

《一切为了爱·序曲》

写这本书有两个目的：一是使你成为Perl编程专家，一是在更广泛的意义上，为你补充编写应用系统的技能与工具。本书讲述了Perl语言的高级特性，教你学习perl解释器的工作原理，以及诸如网络计算、用户界面、对象的持续存储（persistence）与代码生成等现代计算技术。

从这本书中，你不仅仅会涉及Perl语言的语法和各种模块的API（应用编程接口），你同样要花费相当的时间来处理现实情况中的问题，如在远程过程调用中避免死锁，在平面文件（flat file）或数据库的数据存储方式之间进行平滑的切换。同时，你将熟练使用诸如运行时（run-time）计算、嵌套数据结构、对象以及闭包（closure）一类的技巧。

在你读这本书以前，你需要了解Perl的基本知识，实际上只需要其中关键的一小部分就可以了。你必须熟知其基本数据类型（标准变量、数组和散列表），正则表达式，子例程，基本控制结构（if，while，unless，for，foreach），文件的输入输出，以及一些标准变量如@ARGV和$_等。如果你对这些不甚了解的话，我向你推荐Randal Schwartz与Tom Christiansen所著的优秀辅导教程《Learning Perl》第二版（译注2）。

译注1：约翰·德莱登（1631－1700），英国著名诗人。

译注2：中文版《Perl语言入门》已由中国电力出版社出版，请访问www.infopower.com.cn。

这本书，特别是这段序言，将详细阐述我的两条理念。

第一条理念是，对于处理那些典型的大型应用系统工程而言，同时采用两种语言的方案，也就是将Perl，Visual Basic，Python或Tcl这样的脚本语言，同系统编程语言（C，C++，Java）结合起来的方案最为合适。脚本语言没有严格的编译时（compile-time）类型检查，拥有高级数据结构（例如Perl中作为基本数据类型的散列表，C语言中就没有这种东西），一般没有附加的编译与链接过程。系统编程语言则往往更加贴近操作系统，具有精细的数据类型（如C语言中就有short，int，long，unsigned int，float，double等等，而Perl中却只有标量变量类型），通常要比解释语言运行速度快。Perl拥有多种编程语言的许多特性。作为脚本语言它表现出色，然而它也提供低级的访问操作系统API的功能，其速度要比Java快许多（这是指本书英文版出版时的情况），而且需要的话还可以进行编译。

讨论脚本语言与系统编程语言的差异是一个有争议的话题。但是实践中我却因此获益匪浅。这一点将在最后三章中着重阐述。

我相信这两类语言没有一种具备足够的特性，可以独立处理复杂应用项目的开发。而且我希望能够将前面提到的结合两类语言的方案以Perl和C/C++为例阐述清楚。即便是你选择使用其他语言，如果本书中所讲述的设计思想和教训对你有所帮助的话，那么就像孩子们常说的，“这样真值，这太棒了”。

我的第二条理念就是：要想部署更为有效的应用系统，仅熟知编程语言的语法是不够的。你必须更进一步了解语言的内部机制，必须扎实的掌握诸如网络、用户界面、数据库等技术领域的知识（尤其是那些超出特定语言功能库的知识）。

让我们来更详细的阐述一下这两条理念。

脚本编程语言

我的职业生涯始于用汇编语言来创建整个应用系统。那时我不时担心的是如何节省100个字节的空间和优化删除一条指令。后来C和PL/M改变了我的世界观，使自己有机会从工程项目的生命周期（life-cycle）和最终用户如何使用等方面，从整体上把握应用系统。对于中断服务例程而言，当运行效率是最首要的需求时，

我依旧使用汇编语言（回首过去，我曾怀疑过PL/M编译器产生的汇编指令是否能比我手工编制的更为出色，我的虚荣心使我始终无法承认这一点）。

我参与的应用系统需求越来越复杂；除了要处理图形用户界面、事务、安全、网络无关和异构平台的问题，我又开始着手进行诸如航空调度和网络管理一类问题的软件体系结构设计。我自己的工作效率却要比应用系统的更成问题。虽然面向对象技术使我在设计上更有效率，而系统实现语言C++，还有那些功能库和工具并没有帮助我提高编程水平。我仍旧需要处理诸如为动态数组、元数据、文本操纵和内存管理等创建应用框架之类的底层问题。不幸的是，能够很好的处理此类问题的计算环境如Eiffel，Smalltalk和NeXT系统对我的组织而言并不实用。你也许现在可以理解，为什么我会成为支持选择Java语言作为应用开发语言的嗓门嘶哑的啦啦队长了。尽管这还不是最终的结果。

近来我渐渐发现自己忽略了软件生命周期中的两大时间黑洞。在设计阶段，有时你要想清楚的理解问题，就需要创建“电子情节串联图版”（也就是原型）。在该软件完成后，用户通常很会对他们所看到的一切挑三拣四。这就意味着即便是基于窗体的简单界面也会被不时的修改，不停的产生新的需求报告。于是，那些急于求成的开发人员就希望在该软件的开发一完成就进入另一个项目。这里就是脚本语言的用武之地。它能提供快速的代码修改，动态用户界面，绝佳的文本处理功能，运行时计算和良好的数据库与网络连通性。最重要的是，它们不需要细致的程序员的精心呵护。你可以将注意力放在如何使应用更以用户为中心，而不是如何用Xlib库（注1）来画饼图上。

显而易见，单独使用脚本语言来开发复杂的应用并不可行。你仍需要那些诸如运行效率、精细的数据结构和类型安全之类的特性。（这一点在多个程序员处理同一个问题时尤为重要。）这也是我热心支持将脚本语言同C/C++（当性能达到实际应用水平时，Java也是一种选择）结合起来使用的原因。许多人从这种基于组件（component-based）形式的开发中获益匪浅，这是一种用C语言来书写组件代码，然后使用脚本语言将组件连接起来的方法。你去问一下那些众多的Visual Basic，PowerBuilder，Delphi，Tcl和Perl程序员们就知道了。噢，对了，还有微软的Office和Emacs用户们。

注1：X Window函数库。有人曾经提到过，X Window编程，就好像使用罗马数字算出一个数的平方根！

要寻找对使用脚本编程更为详尽而雄辩的、根据切身体会描述的文章（这里不提它们的争议性），你可以读一读John Ousterhout博士（注2）的文章，地址在http://www.scriptics.com/people/john.ousterhout。

想要更真切的体味一下这种论断的话，请试用一下在上面那个地址中提到的Netscape的Tcl插件，看一看Tcl小应用程序（Tclets）的代码，你会发现用它来解决一些简单问题是多么的简洁。一个包含用户界面的计算器小应用程序只花费了100行代码，恐怕写同样功能的Java小应用程序的代码量不会低于800行，而且远没有前者灵活。

为什么要选择Perl语言

那么为什么要选择使用Perl而不是Visual Basic, Tcl或Python呢？

尽管Visual Basic在Wintel（注3）的PC上是一种优秀的选择，但是它无法在其他平台上运行，因此对我来说，它就不是一种实际可行的选择。

Tcl会使我更频繁的使用C语言，主要就是因为数据和代码结构的原因。Tcl的性能对我来说并不是一种关键因素。因为我通常考虑到这一实际情况，所以只用它来书写那些对性能要求不高的部分。我向你推荐Brian Kernighan（译注3）的文章“Tcl/Tk在科学与工程可视化中的应用经验谈”中有关Tcl和Visual Basic的论述。它的获取地址为*http://inferno.bell-labs.com/cm/cs/who/bwk*。

大多数的Tcl用户基本上都离不开Tk用户界面工具包，我也不例外。在Perl中同样可以使用Tk，因此我可以在自己首选的语言中使用另一种语言环境中的最为优秀的特性。

注2：Tcl（Tool Command Language，工具命令语言，发音为“tickle”）的发明者。

注3：Wintel：微软视窗操作系统Windows与Intel微处理器的组合。以后我将使用“PC”这个字眼来表示这种特定的组合，如果是指Linux和Mac机我会明确提出。

译注3：他是C语言的创造者之一。

我是个不折不扣的Python崇拜者。这是一种由Guido van Rossum开发的脚本语言（可以查看相关网址http://www.python.org/）。Python语言拥有清晰的语法结构，良好的面向对象机制和线程安全，拥有大量的功能库及与C语言的完美接口。我之所以更倾向于Perl（相对Python而言）是出于实用而不是工程上的原因。在工程方面，Perl语言在文本处理上快速而无可替代。其语言高度专业化，也就意味着比一般用其他语言编写出的代码更为紧凑。后者或许不是种好事，这要看你持何种观点（尤其对于Python程序员来说）。所有这些特性使Perl成为一种构造工具的优秀编程语言。（请参考第十七章“模板驱动的代码生成”。）在其他方面，Python则拥有更多的优势。我建议你进行认真的分析。Mark Lutz的《Programming Python》（O'Reilly公司1996年出版）一书对Python语言和功能库进行了很好的描述。

从实用角度讲，你当地的书店和报纸上的招工信息已经表明了Perl语言的流行性。基本上说，这就意味着更容易雇佣到Perl程序员，或是人们可以很快的学习这种语言。我敢打赌，95%的程序员甚至从没听说过Python语言（译注4）。不幸的是，这种情况是真的。

把玩这些语言并得出你自己的结论是必要的；毕竟，前面的一些观察都带有我个人的色彩。正如Byron Langenfeld所说的：“很少有人能不经实地测量而发现别人的错误。”本书将把Tcl、Python、Java与Perl在具体特性上做一比较，我要强调的是，对语言和开发工具的选择，从来没有定论，也不是非此即彼的，我还要说明，大多数时候，你可以使用其中一种，也可以使用另一种。

我必须要了解什么？

要想在应用系统中更有效的使用Perl语言，你必须熟知以下三方面信息：

- 语言的语法及语言所提供的专业用语。
- Perl解释器本身，这是为了书写Perl脚本程序的C扩展模块，或是将Perl解释器嵌入到你的C/C++应用系统中。

译注4：这是1997年本书首次出版时的情况，今天，Python已经成了热门语言。

- 诸如网络、用户界面、万维网和持续性存储等相关技术问题。

图0-1描述了本书中要谈论的主题。每条所列的主要内容都进行了进一步细分。余下的章节描述了每个话题的简单介绍以及详细论述的章节。它们是按照主题而不是本书中章节出现的先后次序来组织的。

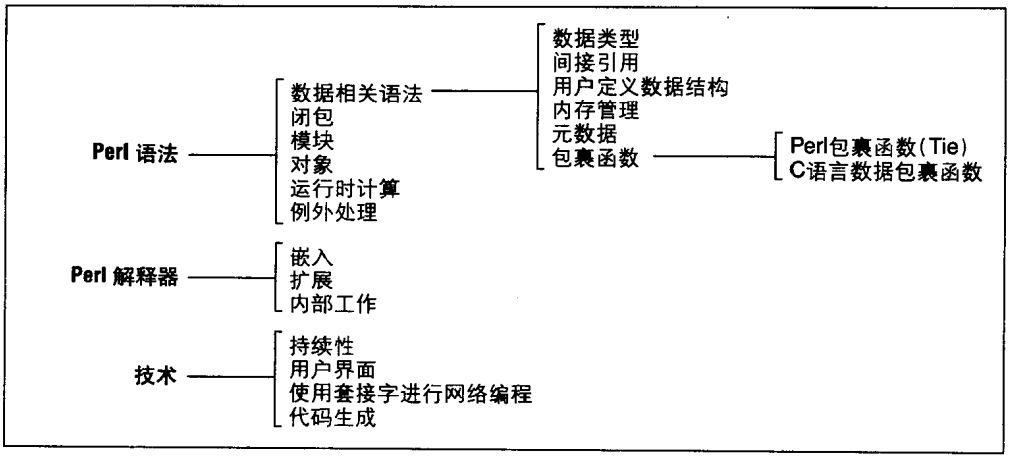

图 0-1 本书涵盖的主题分类

语言语法

指针（pointer）或引用（reference）可以使你创建非常复杂的数据结构。Perl语言支持引用，而且还允许你无须进行烦琐的声明，即可直接书写代码，因此成为一种功能极其强大的编程语言。例如，你只用一行代码就能创建由数组的散列表所组成的数组（注4）。第一章“数据引用与匿名存储”，向你介绍了引用的概念以及Perl内部的内存管理机理。第二章“实现复杂的数据结构”通过几个实际的例子练习使用前面章节讲述的语法内容。

Perl支持对子例程的引用和一种称为闭包的强大结构。LISP程序员对此可能比较熟悉，这实际上是一种没有名字的子例程，通过上下文环境来交换信息。这种机

注4: 我们以后将把索引列表/数组（indexed list/array）称为“数组”，而将关联数组（associative array）称作“散列表（hash）”以避免混淆。

制及其相关专业用语将在第四章“子例程引用与闭包”中予以澄清并加以很好的利用。

引用只是一种间接存取的方式。标量变量中可以包含指向C数据结构的嵌入式指针。该内容放在第二十章“Perl的内部工作”中讲述。Tie代表着另外一种形式的间接存取：所有Perl的值在创建、存取或释放时都可以有选择的触发特定的Perl子例程。这些要在第九章“绑定”中讨论。

文件句柄、目录句柄和格式都不能算是基本的数据类型；它们之间不能相互赋值或作为参数传递，而且你不能够为它们创建局部的版本。在第三章“Typeglob和符号表”中我们将学习这些机制处于首要位置的原因，以及如何通过迂回的方式来获得这种机制。这一章中重点讨论一种数据类型typeglob，（它在某种程度上是隐含的）以及它的内部表示。理解这些内容对于获取解释器状态信息（元数据）以及创建方便的别名来说至关重要。

现在让我们来讨论与Perl数据类型并不直接相关的语言问题。

Perl支持包括异步例外（从信号处理过程中产生用户定义的例外的能力）在内的例外处理。实际上，eval除被用来进行运行时计算外，还应用在例外的捕获上。在第五章“Eval”中将对这两方面貌似不同而实际相关的话题进行讲解。

第六章“模块”中将详细讲述Perl对模块化编程的支持，其中包括的特性如运行时联编（也就是只有在运行时才决定调用哪个过程），继承（Perl透明的使用来自于另外一个类的子例程的能力），还有自动加载（捕获对不存在函数的调用并进行相应的动作）。第七章“面向对象编程”中将使模块的概念从逻辑上更进一层：不仅从功能库用户的观点体现模块的重用性，而且还从增加库接口的开发者的角度加以体现。

Perl支持运行时计算：也就是把字符串看作小段Perl程序，动态的计算它们的值。第五章中引入关键字eval，并举出一些例子来描述如何使用这种特性，但是它真正的重要性还要在后面的章节中详细讨论。它将被应用在诸如SQL查询的计算（第十一章“对象持续性的实现”），代码生成（第十七章“模板驱动的代码生成”）及动态生成对象属性的存取函数上（第八章“面向对象：下面的几步”）。