

Perl 5 开发指南

(第二版)

李俊海 等译

[美] Stephen Asbury, Mike Glover,
Aidan Humphreys, Ed Weiss, 著
Jason Mathews, Selena Sol

解决200个
Perl和CGI
编程问题
的诀窍



- 在CGI程序中实现环境变量
相关客户机，服务器和资源
- 优化Perl和CGI代码
- 测试UNIX口令
- 将CGI库程序转换成模块
- 编写基于套接字服务器程序
以改进CGI性能

PERL 5 HOW-TO

SECOND EDITION



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

URL: <http://www.phei.com.cn>

内 容 提 要

Perl 5 第二版提供的资源将帮助程序员节约大量的编程时间。本书提供了详细地解决 200 个 Perl 和 CGI 编程步骤,其中包括问题的提出、解决的方法、可能的错误和全部源程序。

本书共分二十四章,涵盖了 Perl 5 的全部特征和 CGI 编程技巧,同时本书附带的光盘中包括了 Perl 的资料和程序库,以及书中程序源码,并且包括 Perl 手册在线文档。适合 Perl 的程序员及对此感兴趣者使用。

Authorized translation from the English language edition published by Waite Group Press, an imprint of Macmillan Computer Publishing U.S.A.

Copyright © 09/03/97

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

SIMPLIFIED CHINESE language edition published by Publishing House of Electronics Industry, China.

Copyright ©2000

本书中文简体专有翻译出版权由美国 Macmillan Computer Publishing 下属的 Waite Group Press 授予电子工业出版社。该专有出版权受法律保护。

图书在版编目(CIP)数据

Perl 5 开发指南(第二版)/(美)阿斯伯里(Asbury.S.)等著;李俊海译. - 北京:电子工业出版社,2000.8

ISBN 7-5053-5099-4

I. P... II. ①阿... ②李... III. Perl 语言-程序设计-指南 IV. TP312

中国版本图书馆 CIP 数据核字(2000)第 65955 号

书 名:Perl 5 开发指南(第二版)

原 书 名:PERL5 HOW-TO SECOND EDITION

著 者:[美]Stephen Ashury, Mike Glover, Aidan Humphreys, Ed Weiss, Jason Mathews, Selena Sol

译 者:李俊海 等

责任编辑:张 琛

特约编辑:李 威

排版制作:电子工业出版社计算机排版室监制

印 刷 者:北京天竺颖华印刷厂

出版发行:电子工业出版社 URL: <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036

经 销:各地新华书店

开 本:787×1092 1/16 印张:45 字数:1152 千字 附光盘:1 张

版 次:2000 年 8 月第 2 版 2000 年 11 月第 2 次印刷

书 号: ISBN 7-5053-5099-4
TP·2539

印 数:3000 册 定价:65.00 元(含光盘)

版权贸易合同登记号 图字:01-97-1932

凡购买电子工业出版社的图书,如有缺页、倒页、脱页、所附磁盘或光盘有问题者,请向购买书店调换;若书店售缺,请与本社发行部联系调换。电话 68279077

序 言

本书的主要内容

本书的内容是关于 Larry wall 发明的 Perl 编程语言。自 Perl 发布开始,它就成为非常流行的编程语言。这其中的背景原因很简单:Perl 是一种功能强大的编程语言,它涵盖了 UNIX 程序员所面临的大部分功能,如程序脚本、工具和 C 语言编程;与此同时,Perl 提供了丰富的语法,使它可以很容易地解决复杂问题。当然,这类丰富的语法所带来的是 Perl 语言本身在刚开始接触时会造成一定的麻烦。这就需要耐心和不断地学习实践,但你会发现 Perl 值得这样做。

使 Perl 如此流行的另一个原因是 Web 的蓬勃发展,程序员们使用 Perl 创建 CGI 程序用以驱动 Web 站点。在这本 Perl 5 的新版本中,我们加入了有关 CGI 的大量章节以及关于用 Perl 编写 CGI 程序脚本方面问题的讨论。

请大家注意,本书不是关于 Perl 和 CGI 的入门读物,编写本书的目的是要对实际应用中的问题提供有效的 Perl 解决方法。我们希望本书中包含的对问题的解决方法能帮助读者解决所有问题或作为解决问题的良好起点。

问答的方式

本书第二版采用的是问答方式,注重的是如何一步步地以解决实际问题来介绍 Perl 编程。每节的内容都是按照涉及的问题及相关的解决办法的统一格式来编写的,同时还有采用这种解决办法的总结和为什么要采用这种解决办法的注释,以使大家应用代码方便。本书涉及的所有代码在本书附带的光盘上均可找到。

本书是写给谁的

本书是写给所有编写 Perl 程序或脚本的程序员们。因为每节的内容都已按问题的复杂程度分级,所以读者可以按照自己的水平选择适合的内容。本书提供的高级技术可供有经验的 Perl 程序员参考,所以本书可作为您整个 Perl 学习和编程过程的全程参考。

使用本书需要什么

为使用本书,需要一台安装了 Perl 的计算机。用户可按照附录 A 提供的方法获得 Perl。如果要进行 CGI 编程,那么还需要一台 Web 服务器。当然也需要在上面安装 Perl。

光盘中的内容

光盘中包含三项内容:第一是书中各章节的例程代码,这些程序默认 Perl 已经安装在机器的 /usr/local/bin 目录中,读者可以拷贝、编辑这些程序。第二是有关 CPAN(Comprehensive Perl Archive Network)的内容,其中包括很多 Perl 功能扩展相关模块,还有 Perl 解释程序的源

码。最后光盘中还有很多有用的工具和 Perl 相关资源,包括源码和可执行程序。

光盘的安装

尽管 Perl 支持 DOS、Windows 3.X 和 Mac 操作系统,但本书所附的光盘还是对 Windows 95 Windows NT4.0 以及 UNIX 做了安装优化,所以如果读者使用的不是这几种操作系统,那么你将不能享受光盘提供的所有先进性能。

这下面的步骤可以帮助你硬盘上安装光盘中的内容,其中源码的内容只有 600KB,而 CPAN 文档的内容则有 200MB。

- 对于 Windows 95 和 Windows NT 4.0 操作系统

1. 将光盘放入光盘驱动器,如果你的机器有不只一个光盘驱动器,那么选择第一个。
2. 选择 Start 菜单 Run 功能,键入:

```
d: \ wsetup.exe
```

选择 OK 按钮,当然如果你的光盘驱动器不是 d:,选择适合的盘符就是了。

3. 按照屏幕提示完成安装。

你也可以用浏览器通过打开在光盘中的 00WHOIS.HTML 页浏览 CPAN 文档,而无需将它安装在硬盘上,该页面在 \CPAN\AVTHORS 目录中。

- 对于 UNIX 操作系统

因为 UNIX 系统有很多不同之处,安装过程不尽相同、请同时参考系统管理员和 man 手册页。

为安装源码,请按照下列步骤安装:

1. 将光盘插入光盘驱动器。
2. Mount 该光盘的一般命令如下:

```
mount [option]/dev/cdrom/mountpoint < ENTER >
```

3. 在本地文件系统中创建目录,比如为 Perl5ht。

```
mkdir/perl5ht < ENTER >
```

4. 进入该文件目录:

```
cd/Perl5ht < ENTER >
```

5. 拷贝源码文件,如果全部拷贝:

```
CP-r/mountpoint/source/* . < ENTER >
```

如果只拷贝某章内容:

```
CP-r/mountpoint/source/chapter??. < ENTER >
```

为安装 Perl,请按照下列步骤安装:

- 1、2、3 步骤与前面相同。

4. 为解压缩 Perl 源文件,需要使用 GNU 版本的 ZCat 或 gunzip,使用的命令为:

```
ZCat/mountpoint/archives/perl5.tgz|tarxvf - . < ENTER>
```

或

```
ungzip perl5.tgz < ENTER>
```

```
tar xvf perl5.004.tar < ENTER>
```

5. 遵循安装文件中的提示。

本书由李俊海、王涛、刘诚诚、籍顺心、刘彦丽等翻译,由于时间仓促,译者水平有限,若有翻译错误及不恰当之处,敬请读者指正。

目 录

第 1 章 Perl 基础	(1)
1.1 标量数据类型	(1)
1.2 数组	(2)
1.3 相关数组	(5)
1.4 访问	(8)
1.5 常规表达式	(14)
1.6 数值和字符串操作符	(20)
1.7 控制语句	(24)
1.8 子程序、包和模块	(29)
1.9 变量定位	(34)
1.10 特殊变量	(35)
1.11 CGI	(38)
第 2 章 创建 Perl 和 CGI 程序	(40)
2.1 如何在 UNIX 环境下将 Perl 程序变成可执行文件	(40)
2.2 在 DOS 环境下将 Perl 程序变成可执行文件	(42)
2.3 将 Perl 程序变成 DOS 命令	(45)
2.4 在 UNIX 环境下安装 CGI 程序	(48)
2.5 在 Windows NT 环境下安装 CGI 程序	(51)
2.6 在 Windows 3.1 环境下安装 CGI 程序	(53)
2.7 如何完成统一的命令行解析	(54)
2.8 处理复杂的命令行	(58)
第 3 章 文件操作	(61)
3.1 检查文件是否存在	(61)
3.2 从文件中读取数据	(63)
3.3 向文件中写入数据	(66)
3.4 向已建文件中加入数据	(69)
3.5 删除文件	(71)
3.6 查看文件的操作权限	(73)
3.7 改变文件的操作权限	(76)
3.8 获取文件的基本名(basename)	(78)
3.9 获取文件的目录名(dirname)	(79)
3.10 列出目录下的所有文件	(81)
3.11 查看目录树的内容	(83)
3.12 创建目录树	(86)
3.13 删除目录树	(87)

3.14	用一个通用扩展名给一组文件改名	(89)
3.15	随机获取文件	(91)
3.16	无缓冲输出	(94)
3.17	一个本地句柄	(96)
3.18	将文件句柄传递给函数	(100)
第 4 章	标准 CGI 输出	(104)
4.1	选择输出类型	(104)
4.2	初始化 CGI 程序的输出	(106)
4.3	输出本地文件访问	(109)
4.4	输出一个完整的文档 URL	(112)
4.5	输出一个本地文件	(116)
4.6	输出动态创建的 HTML	(119)
第 5 章	环境变量和命令	(126)
5.1	读取和设置环境变量	(126)
5.2	获得客户发出的 CGI 请求信息	(128)
5.3	获得服务器发出的 CGI 请求信息	(130)
5.4	获得当前的 CGI 请求	(132)
5.5	确定命令是否在的 PATH 中	(136)
5.6	从另一程序中读取输入	(141)
5.7	将输出发送给其他程序	(144)
第 6 章	高级控制结构	(146)
6.1	遍历一个列表	(146)
6.2	循环使用一个相关数组	(148)
6.3	退出循环	(151)
6.4	跳到下一循环(iteration)	(153)
6.5	使用多重循环(iterators)	(156)
6.6	使用 switch 表达式	(160)
第 7 章	用户输入	(165)
7.1	从键盘读入一行数据	(166)
7.2	从键盘读入单个字符	(167)
7.3	读入口令但不回显	(172)
7.4	转换混合输入	(175)
7.5	为 CGI 的 GET 请求读入数据	(177)
7.6	为 CGI 的 POST 请求读入数据	(182)
7.7	从请求表中读取数据	(187)
7.8	对请求表中的数据解码	(193)
7.9	存储请求表中的数据	(198)
7.10	从命令行读取传递给程序的数据	(203)

7.11	同时支持 GET 和 POST 请求类型	(205)
7.12	如何解释同一关键字的多个值	(211)
第 8 章	匹配、过滤和格式变换	(219)
8.1	在一组文件中代替一个字符串	(219)
8.2	匹配带有“/”的路径名称	(221)
8.3	找到符合部分一般表达的参考数据	(222)
8.4	匹配多行方式	(224)
8.5	改组重排文件	(225)
8.6	将 DOS 文本文件转换成 UNIX 文本文件	(226)
8.7	修改字符串的内容	(227)
8.8	文件名的扩展代字符“~”	(229)
8.9	用标准时间格式打印当前时间	(231)
第 9 章	用 Perl 生成报表	(234)
9.1	如何在报表中把字段对齐	(234)
9.2	如何把长的字段放在多行上输出	(237)
9.3	如何给报表添加标题	(239)
9.4	如何把变量内容放在报表的顶端	(242)
9.5	如何在报表中添加页脚	(244)
9.6	如何在多个报表输出格式之间切换	(248)
第 10 章	动态输出时 HTML 文件的操作	(252)
10.1	如何在语法检查时分解 HTML 成为标识和主体部分	(252)
10.2	如何对表设置 action 或 request 方法	(264)
10.3	如何找到输入项并且决定它们的类型	(269)
10.4	如何改变正文的值或者大小、把正文隐藏以及改变口令输入项	(274)
10.5	如何管理复选按钮的状态	(279)
10.6	如何管理单选按钮的状态	(285)
10.7	如何改变值或者中文区域的大小	(290)
10.8	如何管理选择列表中的可选项	(294)
10.9	如何对定制 HTML 命令提供支持	(302)
10.10	如何把使用当前格式处理的数据插入到一个已经存在的 HTML 文件中	(305)
10.11	如何把一个超文本链接插入到已经存在的 HTML 文件中	(309)
10.12	如何把一个选择列表插入到 HTML 文件中	(312)
10.13	如何把提交按钮插入到 HTML 文件中	(316)
第 11 章	DBM 文件	(320)
11.1	使用 Perl 创建 DBM 文件	(320)
11.2	显示某个 DBM 的内容	(323)
11.3	修改 DBM 文件中的记录	(327)

11.4	删除 DBM 文件中的记录	(331)
11.5	清空一个 DBM 文件	(335)
11.6	合并两个 DBM 文件	(337)
11.7	如何在 CGI 程序中存取 DBM 文件	(341)
第 12 章	程序自动化、CGI、测试和保密	(349)
12.1	如何做:自动 ftp	(349)
12.2	如何自动地注册到某个远程的计算机	(355)
12.3	如何测试 CGI 程序而不用浏览器和服务 器	(362)
12.4	如何使用 Web 服务器而不是浏览器来测试 CGI 程序	(364)
12.5	如何使用 Web 服务器和浏览器来测试 CGI 程序	(365)
12.6	如何测试和诊断 CGI 程序	(367)
12.7	如何避免常见的 CGI 程序设计错误	(368)
12.8	如何避免常见的安全隐患	(370)
第 13 章	进程间通信	(375)
13.1	用 Perl 程序创建子进程	(375)
13.2	用管道给执行进程发送数据	(378)
13.3	创建监护进程	(382)
13.4	用相同输入执行不同 CGI 程序	(385)
13.5	用不同输入执行不同 CGI 程序	(389)
13.6	用 Perl 程序发送 E-mail	(394)
第 14 章	客户机-服务器和网络程序设计	(402)
14.1	创建 Internet Domain Socket	(402)
14.2	创建基于 TCP 的 client 程序	(406)
14.3	创建基于 UDP 的 client 程序	(409)
14.4	创建并发非死锁 client 程序	(411)
14.5	创建 server socket	(415)
14.6	创建基于 socket 的网络 Server 程序	(418)
14.7	用 CGI 程序直接给 client 发送 HTTP	(424)
第 15 章	函数、库、软件包、模块	(430)
15.1	通过引用传递变量	(430)
15.2	将多个数组传递给一个函数	(435)
15.3	从函数返回多个变量	(438)
15.4	创建和使用软件包	(443)
15.5	创建和使用库	(448)
15.6	创建和使用模块	(451)
15.7	创建 POD 文件	(455)
15.8	将 CGI 库转化为模块	(459)
第 16 章	处理异步事件	(468)

16.1	处理 Perl 中的信号	(468)
16.2	利用信号与运行进程通信	(471)
16.3	让进程等待一个事件	(473)
16.4	创建超时进程	(475)
16.5	调度基于时间的事件	(477)
16.6	巧妙处理异常	(482)
第 17 章	数据结构	(485)
17.1	创建二叉树	(485)
17.2	处理嵌套表	(491)
17.3	创建多叉树	(501)
第 18 章	排序、查寻和修改	(509)
18.1	给数组排序	(509)
18.2	使排序后的数组元素唯一	(512)
18.3	对非量化数据类型排序	(514)
18.4	在数组中查找一个元素	(516)
18.5	确定数组中是否存在相同元素	(518)
18.6	将相关数组按值排序	(519)
18.7	创建递归子程序	(521)
第 19 章	特殊文件处理	(524)
19.1	处理非文本化编码文件	(524)
19.2	处理压缩文件	(527)
19.3	文件加密	(529)
19.4	从二进制文件析取文本文件	(530)
19.5	处理以太网信息包	(532)
19.6	利用 Perl 由日志产生统计表	(542)
19.7	利用 Perl 为 Web 站点创建主页	(550)
第 20 章	UNIX 系统管理	(557)
20.1	读取口令文件	(557)
20.2	不利用口令发现所有用户	(560)
20.3	列出用户所属的所有工作组	(562)
20.4	产生随机口令	(564)
20.5	测试 UNIX 口令	(566)
20.6	检查用户所有权和权限	(567)
20.7	确定何时文件系统将溢出	(572)
20.8	确定文件比给定年龄大或小	(574)
20.9	确定文件比给定大小大或小	(578)
20.10	比较两棵目录树	(581)
20.11	用 NCSA 服务器对用户进行安全验证	(584)

第 21 章 性能	(588)
21.1 如何完成记录执行日志	(588)
21.2 编译时捕捉潜在的错误	(591)
21.3 编写可移植的 Perl 程序	(593)
21.4 剖析 Perl 代码	(595)
21.5 优化 Perl 和 CGI 代码	(598)
第 22 章 Perl 调试工具	(602)
22.1 使用 Perl 调试器	(602)
22.2 调试包含子程序的 Perl scripts	(608)
22.3 在 Perl script 中设置和取消断点	(610)
22.4 用普通命令别名设置调试器	(612)
22.5 用调试器交互地执行 Perl 命令	(613)
第 23 章 面向对象的程序设计	(615)
23.1 产生一个类	(615)
23.2 产生一个对象	(619)
23.3 从类继承	(623)
23.4 重载父类方法	(628)
23.5 产生类变量	(631)
23.6 直接调用类方法	(642)
23.7 安装使用在 Web 上发现的 Perl 5 CGI 模块	(646)
第 24 章 扩充 Perl 5	(647)
24.1 使用 Perl script h2xs	(647)
24.2 让 Perl 理解数据类型	(651)
24.3 把一个引用变成 char* *	(656)
24.4 扩充 Perl 使之包含函数	(660)
24.5 从一个函数返回多个值	(666)
24.6 让 Perl 变量取消已分配的变量	(672)
24.7 在 XSUB 程序中设定缺省的参数值	(676)
24.8 产生变长参数列表	(684)
24.9 在 Perl 中产生回调函数	(690)
24.10 把扩充编译到 Perl 中	(697)
附录 A Perl 和 CGI 的 Internet 资源	(701)
附录 B CGI 环境变量	(705)
附录 C HTML 制表元素	(707)

第 1 章 Perl 基础

Perl 的流行有很多原因,其中之一是由于用 Perl 编程十分容易,Perl 是解释型的,可以大大减少开发时间。如果不考虑编译过程,程序员可以在相对较少的时间里创建相对较大的、较复杂的程序。但是,这样也有一种消极的影响,即有些程序员不认真学习语言,盲目图快的开发造成劣质的低效代码。程序越复杂,理解语义和语法语言就越重要。

这一章将介绍 Perl 编程语言的语义和语法。我们的目的是给开始使用 Perl 的程序员一个良好的基础,以便使他们能创建有效的 Perl 程序。请记住,这是一本指南性的读物,所以如果你是一个 Perl 的新手,就需要仔细研究一下本章的其他部分,以便对 Perl 5 及其特点有一个全面的了解。对于熟悉 Perl 4 的人可以浏览一下本章对 Perl 5 的介绍,因为其中包括一些 Perl 4 没有的特征。

全书中都用到标量文本和数组文本的提法。其中标量文本的意思是调用了一个函数,并且返回一个标量类型或单值类型值。下面的例子给出了用标量文本调用名为 `scalarContextFunction` 的函数的格式:

```
$returnValue = scalarContextFunction();
```

调用 `scalarContextFunction` 的结果存储在一个叫作 `$returnValue` 的标量型变量中。标量型变量是 Perl 最基础的数据类型。如果是用数组文本调用函数,函数将返回一个列表,下面的例子给出了用数组文本调用函数的格式:

```
@returnValue = arrayContextFunction();
```

变量 `@returnValue` 是一个数组,包含调用 `arrayContextFunction` 函数的返回元素。

有些函数,如 `Values` 和 `keys` 函数,既可以用标量文本,也可以用数组文本调用。这意味着这些函数有多重个性。例如,当 `keys` 函数用标量格式调用时,它将返回给定相关数组成员的数目;当 `keys` 函数用数组格式调用时,它将返回给定相关数组中所有值的列表。

1.1 标量数据类型

标量数据类型是 Perl 中最基本的数据存储形式。一个标量型变量可以描述一个串值或一个数字类型值。实际上,Perl 有三种表达形式,在每种表达形式中解释一个标量类型值。它们分别为:串表达形式、数字表达形式和杂项表达形式,这部分内容将在 1.4 节讨论。

Perl 对数值和字串几乎是同样对待的。在 Perl 程序中,定义或分配一个标量型变量可以通过创建一个标量型变量并分配给它一个值来完成。创建变量和命名一样容易,在创建标量

型变量时,变量名前将前缀一个美元符号(\$),赋值可通过等号(=)和一个常量完成。举个例子,如下三行定义了三个标量型变量:

```
$name = "Gizmo";  
$age = 3;  
$height = 4.5;
```

\$name 包含一个串值, \$age 包含一个整型值, \$height 包含一个浮点式十进制数。当标量型变量被赋值时,赋值的语法将帮助 Perl 的解释程序决定变量的类型。如果变量的值用单或双引号引出,Perl 会将变量认成串值;如果没有引号,Perl 就要决定变量是串值还是数字值,就像下面的 Perl 程序所示:

```
#!/usr/local/bin/perl -w  
  
$firstName = Gizmo;  
$lastName = "Senegal";  
$age = "3";
```

如果执行这段程序,将得到一个警告:没有引号的串值。

```
Unquoted string (tm)Gizmo(tm) may clash with future reserved word at  
bareword.pl line 3
```

警告提示 Gizmo 可以成为将来的保留字。比如函数名,如果有一个函数名为 Gizmo,它将改变赋值表达式。请注意变量 \$age 的赋值, \$age 变量的命名使用了“引号”,这意味着 Perl 将把这个标量型变量初始化为串值而非数字值。尽管这样的操作是允许的,但这将导致混乱的编程风格,所以应该回避。

1.2 数组

Perl 有一种实际上是一组标量类型值的数据结构,这一结构通常被看成是一个数组或是一个表。Perl 的数组可被当作一个简单表、一个栈或一个复杂数据结构的框架。本节将讨论 Perl 的数组以便使程序员明白如何获得,并用不同方式使用它们。

Perl 的标量数组的定义有几种方法。其中一般是定义一个空数组,如下面的例子中定义了一个叫 @myList 的数组:

```
@myList = ();
```

注意:数组定义中有一个前缀符号(@),它将告诉 Perl,我们要一个数组而不是一个标量值。

Perl 不总是需要在使用前定义变量或数组。上面的例子只是说明,如何定义一个空数组

及如何清空一个现有的数组。

一、将数组用作索引表

用数组索引表通常的方法是在创建时给数组中的所有变量赋值。下例将一年中的 12 个月赋值给数组 @months。

```
@months = qw(JUNK Jan Feb March April May June July Aug Sept Oct Nov Dec);
```

注意:位置标记 JUNK 和关键字 qw。数组从索引 0 开始,JUNK 用来作位置标记。这样我们就可以从索引 1 开始访问月份得到 January(五月)。`$months[0]`由 JUNK 表示,这样 Jan 可由 `$months[1]`得到。June 在 `$months[6]`中,而 Dec 在 `$months[12]`中,qw 关键字是由 Perl 引入的。下面两行 Perl 代码给 @array 列表赋值的结果是一样的。

```
@array = qw(a b c d e);  
@array = ("a","b","c","d","e");
```

关键字 qw 是用来从字符串摘取单个字的缩写形式。在上面的例子中,单词即是月份的名字,对它执行 qw 的结果就是它们存于 \$months 数组中,如果数组不能一次赋值,可以单独给每个数组成分赋值。例如,可以用下面的方式给 \$month 数组命名:

```
$months[0] = "JUNK";  
$months[1] = "Jan";  
$months[2] = "Feb";  
... $months[12] = "Dec";
```

省略号是为了简洁,\$months 数组中的其他成员也应以同样的方式赋值。上面的代码将 \$months[0]赋值为“JUNK”, \$months[1]赋值为“Jan”, \$months[2]赋值为“Feb”,依此类推。

注意:当给数组成员直接赋值时,应该用 \$ 字符,而不是 @ 字符。数组前的字符 \$ 告诉 Perl,赋值的是数组的一个成员而非全部数组。

从数组中提取元素有很多方式,其中最为常用的方式就是直接通过数组索引,下例就表示如何直接索引一个数组的内容。

```
#!/usr/local/bin/perl -w  
my @months = qw(JUNK Jan Feb March April May June July Aug Sept Oct NovDec);  
  
for ($x=0; $x <= $#months; $x++)  
{  
    print "Index[ $x] = $months[ $x] \n";  
}
```

\$#months 是 Perl 的一种习惯表达,它可以告诉程序员一个数组最大的下标值。如果 \$#months 返回 -1,则说明这个数组是空的。

执行上述程序,输出如下:

```
Index[0] = JUNK
Index[1] = Jan
Index[2] = Feb
Index[3] = March
Index[4] = April
Index[5] = May
Index[6] = June
Index[7] = July
Index[8] = Aug
Index[9] = Sept
Index[10] = Oct
Index[11] = Nov
Index[12] = Dec
```

二、将数组作栈使用

能用多种方式在数组中存储信息,其中一种是将数组按栈使用。下面使用 `push` 函数将元素推入到数组的顶部:

```
push (@myList, "Hello");
push (@myList, "World!");
```

上面的例子将两个字符串“Hello”和“World!”推入数组变量 `@myList` 中。因为我们使用了 `push` 函数,数组变量 `@myList` 被看成后入先出栈(LIFO)。一个 LIFO 栈的工作很像咖啡屋中的盘子堆。盘子被堆到盘子堆顶,而其他盘子被压下;当盘子被取走时,它将被从堆顶取走,而其他的盘子移向堆顶。图 1-1 用盘子堆模拟表示了一个 LIFO 栈。

从栈顶取走元素使用 `pop` 函数。使用上面的例子,如果对 `@myList` 调用 `pop` 函数,因为它是最后一个进栈的数据,“World!”值将被返回。下面的例子将说明如何把数组当栈使用。

```
#!/usr/local/bin/perl -w
push (@myList, "Hello");
push (@myList, "World!");
push (@myList, "How");
push (@myList, "Are");
push (@myList, "You?");

while ($index = pop(@myList))
{
    print "Popping off stack: $index\n";
}
```

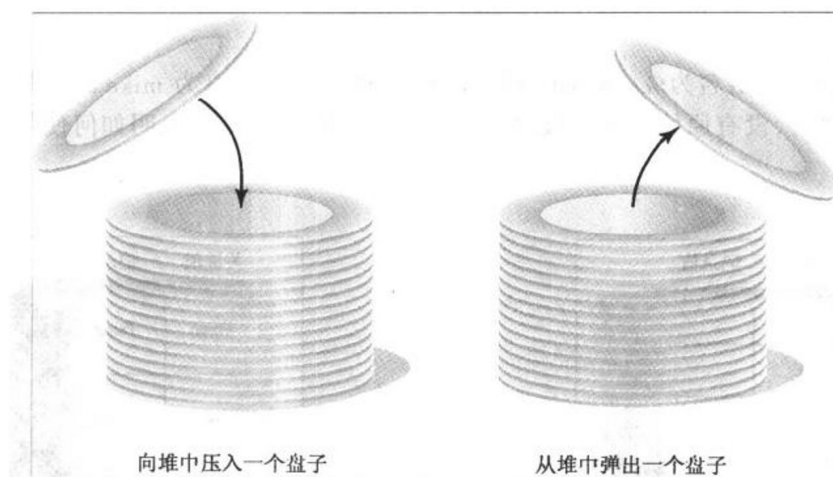


图 1-1 LIFO 栈图

执行上面的代码,输出如下:

```
Popping off stack; You?  
Popping off stack; Are  
Popping off stack; How  
Popping off stack; World!  
Popping off stack; Hello
```

数组成分按相反的顺序被弹出,这就是 LIFO 栈的效果。当一个元素被弹出,这个数据项实际上就被删除了。一旦所有的成分都被移出,栈就空了。

1.3 相关数组

相关数组是按串值索引而不是按整数索引的。图 1-2 列出了一个标准表的成分。图 1-3 列出了一个相关数组的样子。为了把问题讲得更清楚一些,我们将比较一个一般数组和一个相关数组。如果有一个命名为 `@scalarArray` 的数组并想打印输出这个数组的第一个成分,应该使用下面的语法:

```
print $scalarArray[0];
```

相关数组不像标量数组,它们不具有一般的顺序,没有第一个。这是因为相关数组是按字符串索引的,元素不是按预定的顺序排列的。要想从相关数组中抽出一个值,必须知道键值。如果知道相关数组 `%associativeArray` 的一个键值,并想打印出来,就应该用下面的语法:

```
print $associativeArray{'mike'};
```

这个例子将打印出名为%associativeArray 相关数组中关键值为 mike 的变量值。很多像 C 这样的编程语言,没有原始的相关数组,而 Perl 是个例外,本节将说明如何使用 Perl 的标量型相关数组。

索引	记录值
0	
1	
2	
3	
4	
5	
6	

图 1-2 标准表的成分

关键值	记录
Gizmo	Parrot
Elmo	Budgie
Timmy	Cat
Fegus	Dog

图 1-3 相关数组的例子

一、创建一个相关数组

Perl 的标量型相关数组可用很多方法建立,通常的方法是声明一个空的相关数组,就像下面的例子:

```
%cities = ();
```

Perl 中的变量和相关数组在使用前不需要提前定义。上面的例子给出了如何定义一个空的相关数组,或如何将一个现存的数组清空。上例创建了一个名为%cities 的标量型的、空的相关数组,该数组用于对应一组城市名以及它们各自位于加拿大的地址名。

注意:定义标量型一般数组和标量型相关数组的区别是,一般数组由@符号表示;而相关数组由%符号表示。

与一般数组一样,相关数组一次将所有成分赋值。下面的 Perl 代码给相关数组%cities 的三个记录赋值:

```
%cities = ("Toronto" => "East", "Calgary" => "Central", "Vancouver" => "West");
```

操作符“=>”与逗号“,”等价;它们的主要作用就是建立一个可见的连接对。所以,下面的两行的意思就一样了。

```
%cities = ("Toronto" => "East", "Calgary" => "Central", "Vancouver" => "West");
%cities = ("Toronto", "East", "Calgary", "Central", "Vancouver", "West");
```