

# 第零章 Perl 前言

- ↓ 第零章 Perl 前言
  - ↓ 1. 追求幸福
  - ↓ 2. 这个版本有什么新东西
  - ↓ 3. 标准的发布版
  - ↓ 4. 在线文档
  - ↓ 5. 阅读标准手册页
  - ↓ 6. 搜索手册页
  - ↓ 7. 非 Perl 手册页
  - ↓ 8. 离线文档
  - ↓ 9. 其它资源
    - ↓ 9.1 Web 上的 Perl
    - ↓ 9.2 Usenet 新闻组
  - ↓ 10. 臭虫报告
  - ↓ 11. 这本书里用的习惯
  - ↓ 12. 感谢

## 1. 追求幸福

Perl 是一种能完成任务的语言。

当然, 如果你的工作就是写程序, 那么从理论上讲, 你可以使用任何“完整”的计算机语言来完成任务。但是从我们的经验来看, 计算机语言的区别很大程度上不在它能干什么, 而是在它做事情的时候是否简单。从一个极端来说, 那些所谓的第四代语言做某些事情的时候相当容易, 但是做其它一些事情的时候几乎不可能。从另外一个极端来看, 那些所谓的工业级的语言在做任何事情的时候都几乎一样困难。

Perl 是不同的语言。从一开始, Perl 就设计成可以把简单工作简单化, 同时又不失去处理困难问题能力的语言。

那什么是“简单工作”呢? 当然就是那些你每天都要干的。你需要一种可以很容易操作数字, 文本, 文件和目录, 计算机和网络, 特别是程序的语言。这种语言应该很容易运行外部的程序并且扫描这些程序的输出获取感兴趣的东西。而且它还应该很容易能把这些你感兴趣的东西交给其它程序做特殊的处理。当然, 这种语言还应该很容易在任何现代的操作系统上可以移植地编译和运行。

Perl 做到了上述所有需求, 甚至更多。

Perl 最初是当做一种 Unix 的胶水语言设计的, 但是她早就移植到大多数其它操作系统里了。因为 Perl 几乎可以在任何地方运行, 所以 Perl 可以说是当今最具有移植性的编程环境。要想写可移植的 C/C++ 程序, 你得在程序里加上一大堆 `#ifdef` 标签来区分不同的系统。要想写可移植的 Java 程序, 你必须理解每种新的 Java 实现的特质。要想写可移植的

shell, 你可能要记住每条命令在每种操作系统上的语法, 走运的时候你可能可以找到一些公共的东西。而要想写可移植的 Visual Basic 程序, 那么你只需要对“移植”有个更灵活的定义就可以了。

我们很高兴的是 Perl 避免了所有这些问题, 同时还保留了这些语言中的许多优点, 同时还有一些自己的特色。Perl 的特色来自许多方面: 它的特性集的工具, Perl 社区的创造性, 以及开源运动的大环境。不过, 许多这些特性都是混合的东西; Perl 的身世复杂, 它总是把事物看成是优点的不同方面, 而不是弱点。Perl 是“背黑锅我来”的语言。如果你觉得自己陷入一团乱麻之中, 非常渴望自由, 那么请使用 Perl。

Perl 是跨文化的。Perl 的爆炸性增长很大程度上是因为那些前 Unix 系统程序员的渴望, 他们希望从他们的“老家”带着尽可能多的东西。对于他们而言, Perl 是可移植的 Unix 文化蒸馏器, 是"此

路不通"的沙漠中的绿洲。从另外一个角度来看, Perl 还可以从另外一个方向运转: 在 Windows 上工作的 web 设计者通常会非常开心地发现他们的 Perl 程序可以不加修改地在 Unix 服务器上跑。

尽管 Perl 在系统程序员和 web 设计师中间非常流行, 但这只是因为是他们最早发现 Perl 的, Perl 可以用于更广泛的用途。从 Perl 最早的文本处理语言开始, 它已经发展成为一种非常复杂的, 通用的编程语言, 以及完整的开发环境, 包括调试器, 调节器, 交叉引用, 编译器, 库, 语法提示编辑器, 以及所有其它“真正”的编程语言所具有的所有挂勾, 只要你需要。当然这些东西都是让我们可能处理难的问题的东西, 而且很多其它语言也可以做到这一点。Perl 之所以成为 Perl 是因为它从来不会因为保持简单事情简单化而丢失其他方面的特性。

因为 Perl 既强大又好用, 所以它被广泛地用于日常生活的方方面面, 从宇航工程到分子生物学, 从数学到语言学, 从图形处理到文档处理, 从数据库操作到网络管理。很多人用 Perl 进行快速处理那些很难分析或转换的大批量数据, 不管你是处理 DNA 序列, 网页, 还是猪肚皮的未来都无所谓。实际上, 在 Perl 社区有一个笑话就是, 下次股市大崩盘就很有可能是呢个家伙写的脚本里头有臭虫造成的。(不过, 乐观点来看就是, 任何还在失业的股票分析师仍然有可以利用的技巧。)

Perl 的成功有许多原因。Perl 早在开源软件的名字出现之前就已经是一个成功的开源项目了。Perl 是自由的, 并将永远自由下去。你可以在任何合适的场合使用 Perl, 只需要遵守一个非常自由的版权就可以了。如果你在从事商业活动并且还想使用 Perl, 那么用就是了。你可以把 Perl 嵌入到你写的商业软件中而不需要支付任何费用也没有任何限制。如果你碰上一个 Perl 社区解决不了的问题, 那你也还有最后一招: 源程序本身。Perl 社区不会在“升级”的伪装下租给你它们的商业秘密。而且 Perl 社区也不会“停业”, 更不会让你孤立无援。

Perl 是自由软件这一点无疑对它是有帮助的。但这一条并不足以解释 Perl 现象, 因为许多自由软件包没有能繁荣起来。Perl 不仅自由; 而且好玩。人们觉得自己在 Perl 里可以有创造力, 因为它们有表达的自由: 他们可以选择是为计算机速度优化还是为程序员的速度优化, 是冗长还是简洁, 是选择可读性还是可维护性, 或者选择复用性, 移植性, 接受性和传授性等等。假如你进入一次模糊的 Perl 比赛, 甚至你还可以为模糊性做优化。

Perl 可以给予你所有这些自由, 因为它是一门有着分裂人格的语言。Perl 同时是很简单并且很富有的语言。Perl 从其它地方拿来好主意, 然后把它们安装到易用的框架里面。对于只是喜欢她的人来说, Perl 是实用抽取和报表语言 (Practical Extractoin and Report Language)。对那些热爱她的人而言, 她是变态电子垃圾制造者 (Pathologically Electric Rubbish Lister)。在少数人眼里, Perl 是毫无意义的重复练习。不过世界需要一点点冗余。精简主义者总是想把事物分隔开。而我们则总是企图把它们合并到一起。

Perl 之所以是简单的语言是有很多原因的。比如你用不着知道什么特殊的指令就可以编译 Perl 程序--只要把它当做批处理或者 shell 脚本执行就可以了。Perl 的类型和结构很容易使用和理解。Perl 对你的数据没有任何限制--你的字串和数组可以要多长就多长 (只要你有足够的内存), 而且它们都会自动增长。Perl 不会强迫你学习新的语法和语意, Perl 改从许多其它你已经熟悉的语言里 (比如 C, awk, BASIC 和 Python, 英文, 希腊语等) 借来语法。实际上, 任何程序员都可以从书写良好的 Perl 代码段中读懂它的含义。

最重要的是, 你不用先学习所有 Perl 的东西就可以开始写有用的程序。你可以写很小的 Perl 程序。你也可以象小孩那样写 Perl 程序, 我们保证不会笑话你。或者更准确地说是, 我们绝不会笑话小孩做事的创造性。Perl 里的许多观点都是从自然语言中借来的, 其中一条最好的观点就是只要你能把自己的意思表述清楚, 那么你就可以使用这些语言的一个子集。Perl 文化可以接受任何熟练程度的成员。我们不会在你背后放个语言警察。如果你的老板不炒你, 而且你的 Perl 脚本也能完成工作, 那么它就是“正确”的。

尽管 Perl 很简单, 但它仍然是一种特性很丰富的语言, 如果你想用那些特性的话, 那你就需要学习一些东西。这也是把难题变简单的学费。虽然你要想把所有 Perl 能做的事情吸收还需要一些时间, 但

到你需要这些功能的时候你就会非常开心地发现 Perl 已经可以做这些事情了。

由于 Perl 的继承性, 就算它只是用做数据归纳语言的时候也有丰富的特性, Perl 一开始就设计成可以浏览文件, 扫描大量文本并且生成动态数据以及打印出这些数据的良好格式化的报表。不过, 随后 Perl 就开始风行, 于是它就成了可以操作文件系统, 进程管理, 数据库管理, 进行 C/S 编程和安全编程, web 信息管理, 甚至可以进行面向对象和面向功能的编程的语言。而且这些功能并非只是在 Perl 这边, 每种新功能都和其它东西交流得很好, 别忘了 Perl 从一开始就是设计成胶水语言的。

而且 Perl 并不仅仅只能黏合它自己的特性。Perl 是设计成可以用模块扩展的语言。你可以用 Perl 快速设计, 编写, 调试和部署 Perl 应用, 并且你还可以在需要的时候很方便地扩展这些应用。你可以在其它语言里嵌入 Perl, 而且你也可以在 Perl 里嵌入其它语言。通过模块输入机制, 你可以把这些外部的扩展当做内置于 Perl 的特性。那些面向对象的外部库在 Perl 内部仍然保持面向对象的特征。

Perl 还在许多其它方面协助你。和严格的每次执行一条命令的命令文件和 shell 脚本不同的是, Perl 先把你的程序快速编译成一种内部格式。和其它任何编译器一样, 这个时候还进行各种优化, 同时把碰到的任何问题反馈给你。一旦 Perl 的编译器前端对你的程序表示满意了, 它就把这些中间代码交给解释器执行 (或者是给其它的能生成 C 或者字节码的模块后端)。听起来挺复杂, 不过 Perl 的编译器和解释器干这些活效率相当高, 我们的编译-运行-修改的过程几乎都是以秒计。再加上 Perl 的许多其他开发特性, 这种快速的角色转换很适合做快速原型设计。然后随着你的程序的成熟, 你可以逐步拧紧身上的螺母, 减少散漫增强纪律。如果你做得好, Perl 也能帮你这个忙。

Perl 还可以帮你写更安全的程序。除了其它语言提供的典型的安全接口之外, Perl 还通过一种跟踪数据的机制给你提供预防意外安全错误的保护, 这样就可以在灾害发生之前预防其发生。最后, Perl 还可以让你设置一个特殊的防护隔段运行那些来源不明的 Perl 代码, 以此来杜绝危险操作。

不过, 偏执一点儿说, Perl 帮你的大部分内容和 Perl 本身没有什么关系, 而是和使用 Perl 的人有关。坦率地说, Perl 社区的人们可以说是地球上最热心的人了。如果 Perl 运动里面有那么一点点宗教色彩的话, 那么这就是它的核心了。Larry 希望 Perl 社区像一小片天堂那样运转, 目前看来他的愿望基本上是实现了。我们也请你为此做出自己的努力。

不管你是想拯救地球, 还是觉得新鲜, 或是你老板命令你学习, 这本书都将告诉你一些基本的和复杂的东西。虽然没有故意教你写程序, 但是如果你观察力足够强的话, 你还是能找到一些编程的艺术以及一些编程的科学。我们鼓励你培养下面三条程序员的优点: 懒惰, 急躁, 和傲慢。看书的同时, 我们也希望你能从中找到一些有趣的地方 (以及其它一些更有趣的地方)。如果这样还不能让你保持清醒, 那么就不停提醒自己学习 Perl 可以增加你的简历的分量。坚持读下去吧。

## 2. 这个版本有什么新东西

---

几乎全部都是新的。

即使是那些以前版本中的相当不错的地方 (我们得说有相当不少这样的地方), 我们也带着一些目的做了大量的修改。首先, 我们希望增加本书对那些有着非计算机科学背景的人士们更有吸引力。我们对读者已知的东西做出的假设更少了。同时, 我们把许多演示做得更生动, 以防那些已经有一定了解的朋友们看着看着就睡着了。

其次, 我们希望展示 Perl 本身开发的最新进展。在这个问题上, 我们丝毫不因表现我们当前的工作为耻, 即使我们展现给你的仍然是试验性的东西。尽管 Perl 的核心部分已经是久经考验的战士了, 但是有些开发中的试验性部分仍然偶尔会非常烫手。在这里我们要老实告诉你, 我们认为在线文档比我们在这里描述的东西更可靠。Perl 是蓝领阶层的语言, 所以我们不会因为把铁锹叫成铁铲而觉得脸红。

第三, 我们希望你能更自如地往来与书中的各个章节, 因此, 我们把这个版本分裂成了更小, 更连贯

的章节并且把它们重新组织成有意义的部分。下面是新版本的布局:

#### 第一部分, 概述

迈出第一步总是最困难的部分。这一部分向你展现了 **Perl** 的基本概念, 我们尽可能地把门槛放得更低一些。这一部分不是完整的教程, 只是一个快速入门, 这样便可以满足所有人的需要。你可以看看“离线文档”一节获取那些可能更适合你的学习习惯的书。

#### 第二部分, 活生生的细节

这一部分包含对这门语言各个层次抽象的深入的讨论, 从数据类型, 变量, 正则表达式, 到子过程, 模块和对象。在这里你可以对这门语言如何工作有更好的了解, 同时还可以得到一些好的程序设计的经验。(如果你从来没有用过带模式匹配的语言, 那么我们就得对你另眼相看了。)

#### 第三部分, 作为技术的 Perl

你用 **Perl** 本身可以干很多事情, 但这一部分带你到一个更高的境界。在这里你可以学到如何将 **Perl** 和那些你的计算机里的东西联系起来, 从处理 **Unicode**, 进程间通讯, 以及多线程, 到编译, 调用, 调试和调谐 **Perl**, 最后是写自己的 **C** 或 **C++** 的外部扩展(或者是任何你喜欢的现有 **API**)。Perl 会非常开心地和你的计算机里的任何接口, 或者是互联网上任何其他计算机进行交谈, 只要你允许。

#### 第四部分, Perl 文化

我们都明白每种文化都有语言, 但是 **Perl** 社区都明白一种语言一定有自己的文化。在这个部分里, 我们把 **Perl** 编程当作一种人类活动, 嵌入在现实世界的人群当中。我们会谈到你怎样才能改善和好人坏人相处的方法。我们还会聊一些可以让你自己变得更好的方法以及如何令你的程序对其他人更有益的东西。

#### 第五部分, 参考资料

在这里我们把所有你想按字母顺序查找的章节放了进来, 这些内容包括特殊变量和函数, 以及标准模块和用法。词汇表对那些不熟悉计算机科学的俚语的人会特别有用。比如, 如果你不知道“**pragma**”(用法)的含义, 那么你就可以直接把它找出来。(当然, 如果你不知道“**is**”的含义, 那么我们可就不能帮你忙了。)

## 3. 标准的发布版

---

如今的操作系统供应商把 **Perl** 当作系统的标准部件销售。在我们写这些的时候, **AIX**, **BeOS**, **BSDI**, **Debian**, **DG/UX**, **DYNIX/ptx**, **FressBSD**, **IRIX**, **LynxOS**, **Mac OS X**, **OpenBSD**, **OS390**, **RedHat**, **SINIX**, **Slackware**, **Solaris**, **SuSE**, 和 **Tru64** 的标准版本中都带着 **Perl** 一起发布。有些公司在独立的 CD 上提供 **Perl** 发布, 或者是通过各自的客户服务组提供。第三方公司为多种不同的操作系统提供预编译的 **Perl** 发布, 比如 [ActiveState<sup>2</sup>](#), 这些平台包括 **Microsoft**。

即使你的系统提供商把 **Perl** 当作标准版本提供, 有时候你也想自己编译和安装 **Perl**。这样你就知道自己有最新的版本, 并且你还可以选择把库文件和文档安装到哪里。你还可以选择是否需要一些扩展的功能, 比如线程支持, 大文件, 或者许多通过 **-D** 命令行选项可以获取的底层调试选项。(用户层的 **Perl** 调试器总是可以用的。)

下载 **Perl** 源程序工具箱的最简单的方法可能是把你的 **web** 浏览器定位到 **Perl** 的主页 [www.perl.com](#), 在那里你可以找到很明确的下载信息, 以及一些预编译的二进制文件。(那些平台把 **C** 编译器放错了地方。)

你还可以直接跑到 CPAN（在第二十二章，CPAN，描述），方法是访问 <http://www.perl.com/CPAN> 或者 <http://www.cpan.org>。如果你觉得太慢（因为访问它们的人

特别多），那么你应该找一个离你比较近的镜像。下面这几个 URL 只是世界上的少数几个 CPAN 的镜像，现在镜像数目已经超过一百了：

<http://www.funet.fi/pub/languages/perl/CPAN/>

<ftp://ftp.funet.fi/pub/languages/perl/CPAN/>

<ftp://ftp.cs.colorado.edu/pub/perl/CPAN/>

<ftp://ftp.cise.ufl.edu/pub/perl/CPAN/>

<ftp://ftp.perl.org/pub/perl/CPAN/>

<http://www.perl.com/CPAN-local>

<http://www.cpan.org/>

<http://www.perl.org/CPAN/>

<http://www.cs.uu.nl/mirror/CPAN/>

<http://CPAN.pacific.net.hk/>

这个列表中的头两个（就是在 funet.fi 的两个），指向 CPAN 仓库的主站点。其中 **MIRRORED.BY** 文件包含所有其他 CPAN 站点的列表，因此你可以只把这个文件拿过来然后选择你最喜欢的镜像。这些站里有些是 FTP，其他的是 HTTP（在一些公司的防火墙背后，这两种方式是有些不同的）。<http://www.perl.com/CPAN> 分路器会尝试替你选择。不过你稍后可以改变选择。

在你把源程序下载下来并且解压缩之后，你应该阅读 **README** 和 **INSTALL** 文件，学习如何制作 Perl。同时可能还会有 **INSTALL.platform** 文件，你也可以阅读一下，看看和你的平台相关的信息。

如果你的平台碰巧是某种 Unix，那么你用来抓取，配置，制作和安装 Perl 的命令可能是如下几条，首先，你必须选择抓取源程序的命令。你可以用 ftp 来抓取：`% ftp`  
<ftp://ftp.funet.fi/pub/languages/perl/CPAN/src/latest.tar.gz>

（在这里，你还是可以选择离你比较近的 CPAN 镜像。）如果你不能用 ftp，那么你也可以用浏览器或者命令行工具从 web 上下载：`% wget`  
<http://www.funet.fi/pub/languages/perl/CPAN/src/latest.tar.gz>

解包，配置，制作和安装：`% tar xzf latest.tar.gz` 或者先 `gunzip`，然后 `tar xf`。`% cd perl-5.6.0` 或者是 `5.*` 之类的数字 `% sh Configure -des` 假设缺省的回答 `% make test && make install` 安装通常要求超级用户这些工作要用到传统的 C 开发环境，所以如果你没有 C 编译器，那么你就不能编译 Perl。参阅 CPAN 的 **ports** 目录，获取各个平台捆绑 Perl 的最新信息（以及平台的版本），还有你是否可以获取标准的源程序工具箱，或者你是否需要一个特定移植等信息。

## 4. 在线文档

---

Perl 大量的在线文档是和标准版本一起发布的。（离线文档见下一节。）额外的文档显示你是否从 CPAN 安装了模块。

当我们在本书中提到“Perl 手册页”的时候，我们说的就是这些在你的计算机里保存的 Perl 手册页。手册页的说法只是表示一个文件包含文档的传统说法——你用不着拿 Unix 里面的 man 程序才能看它。你甚至可以把 Perl 的手册页安装成 HTML 格式，尤其是在非 Unix 平台上。

Perl 的在线手册页分成了几个独立的节，这样你就可以很容易找到你要的东西而用不着翻阅几百篇文本。因为顶层的手册页就是叫 perl，所以 Unix 命令 man perl 会带你到那里去。

=====footnote begins===== 如果你还是拿到了一个巨大无比的手册页，那么你看到的可能是老的版本 4 的手册页。检查你的 MANPATH 看看有没有问题。（用 perldoc perl 命令看看如何以 perl -V:man.dir 的输出为基础找出应该如何设置你的 MANPATH。） =====footnote ends=====

随后的页面指导你参考更准确的页面。比如，man perlre 将为 Perl 的正则表达式显示手册页。perldoc 命令经常在那些无法使用 man 命令的系统上用。在 Macs 上，你就需要使用 Shuck 程序。你的 Perl 版本还有可能提供了 HTML 格式或者系统本机格式的 Perl 手册。请你的系统管理员来检查一下--除非你自己就是系统管理员。

## 5. 阅读标准手册页

开始的时候（当然是 Perl 开始的时候，在 1987 年左右），perl 手册页是一个简单的文档，如果打印出来大约 24 页纸。比如，它的正则表达式一节只有两段长。（如果你知道 egrep 地话那么也足够了。）从某个角度来看，自从那以后几乎所有东西都改变了。包括标准文档，各种工具，平台相关的移植信息，和许多标准模块等，现在我们有大约 1,500 页文档分布在许多独立的手册页中。（而且这些甚至还不包括你安装的任何 CPAN 模块，那里还有不少。）

不过从另外一个角度来看，也没有改变什么东西：它们仍然是 perl 手册页。而且也仍然是一个从头开始的好地方。区别是一旦你开始看了，那你就不能只停留在那里。Perl 的文档不再是棉花工厂了，而是有几百家商店的超级市场。当你走进大门的时候，你需要先知道自己在哪里，在哪个商店或者是在哪个柜台，这样你才能找到自己想买的东西。当然，一旦你熟悉了这个大市场，那么你就能直接跑到相关商店了。

下面是几个你会看到的商店招牌：

手册页	内容
perl	有些什么 perl 手册页
perldata	数据类型
perlsyn	语法
perlop	操作符和优先级
perlre	正则表达式
perlvar	预定义变量
perlsub	子过程
prelfunc	内建函数
perlmod	如何令 Perl 模块工作
perlref	参考手册
perlobj	对象
perlipc	进程间通讯
perlrun	如何运行 Perl 命令，以及命令行开关
perldebug	调试
perldiag	诊断信息

这些只是摘录的一小部分，但却是重要的部分。如果你想看看操作符，那么 `perlop` 比较合适看看。如果你想找找预定义的变量，那么最好看看 `perlvar`。如果你看到了一条自己不明白的诊断信息，可以看看 `perldiag`。等等。

有一些标准 Perl 手册是常见问题（FAQ）。它们被分割成了下面九个不同的页面：

手册页	内容
<code>perlfaq1</code>	关于 Perl 的通用信息
<code>perlfaq2</code>	获取和学习 Perl
<code>perlfaq3</code>	编程工具
<code>perlfaq4</code>	数据操作
<code>perlfaq5</code>	文件和格式
<code>perlfaq6</code>	正则表达式
<code>perlfaq7</code>	通用 Perl 语言信息
<code>perlfaq8</code>	系统交互
<code>perlfaq9</code>	网络

有些手册页包含平台相关信息：

手册页	内容
<code>perlamiga</code>	Amiga 移植
<code>perlcygwin</code>	Cygwin 移植
<code>perldos</code>	MS-DOS 移植
<code>perlhpux</code>	HP-UX 移植
<code>perlmachten</code>	Power MachTen <sup>2</sup> 移植
<code>perlos2</code>	OS/2 移植
<code>perlos390</code>	OS/390 移植
<code>perlvms</code>	DEC VMS 移植
<code>perlwin32</code>	MS-Windows 移植

（又见第二十五章，可移植的 Perl，以及我们早先谈到的 CPAN 的 `port` 获取移植信息。）

## 6. 搜索手册页

---

没人让你只是为了找一个小小的问题就把 1,500 页手册都读一遍。俗话说你不能对一棵死树做 `grep *`

```
[ I ]grep[ R ]* =====footnote begins===== 别忘了我们还有词汇表。
=====footnote ends=====
```

除了从大多数文档查看程序继承过来的搜索能力之外，到了 Perl 5.6.1 的时候，每个主要的 Perl 手册页都有自己的搜索和显示能力。你可以用手册页名字进行搜索，并且给 Perl 一个正则表达式（见第五章，模式匹配）作为搜索模式：

```
% perlop comma
```

```
% perlfunc split
```

```
% perlvar ARGV
```

`% perldiag 'assigned to typeglob'` 如果你对文档里的东西并不十分清楚, 那么你可以扩大你的搜索。比如, 搜索所有的 FAQ, 使用 `perlfaq` 命令 (它本身也是个手册页):

```
% perlfaq round
```

`perltoc` 命令 (自己也是一个手册页) 搜索所有手册页收集的目录:

```
% perltoc typeglob perl5005delta: Undefined value assigned to typeglob perldata:
Typeglobs and Filehandles perldiag: Undefined value assigned to typeglob 或者搜索全部
Perl 在线文档, 包括所有头, 描述, 和例子, 对于任何字串的实例, 使用 perlhhelp 命令: %
perlhhelp CORE::GLOBAL
```

参阅 `perldoc` 手册页获取细节。

## 7. 非 Perl 手册页

---

当我们谈到非 Perl 文档的时候, 就象 `getitimer(2)`, 它指的是 Unix 程序员手册第二章的 `getitimer` 手册页。

**=====footnote begins=====** 第二章一般来说只包含对操作系统的直接调用。(通常称做"系统调用")。不过, 不同的系统会在哪些函数用系统调用来实现以及那些用 C 库实现上有所不同, 所以你也可能会在第三章里找到 `getitimer`。

**=====footnote ends=====** 象 `getitimer` 这样的系统调用的手册页可能在非 Unix 平台上无法找到, 不过一般说来可能都没问题, 因为这个时候你也不能用 Unix 系统调用。如果你真的需要 Unix 命令, 系统调用或者库函数的文档, 那么许多组织以及把它们的手册页放到了 web 上-- 比如在 [AltaVista<sup>2</sup>](#) 上找一下 `crypt(3)+manual` 就能找到好多。

虽然顶层的 Perl 手册页通常安装到标准 `man` 目录的第一章, 但是在本书中我们通常还是把 (1) 省掉了。你应该可以很方便地找到它们, 因为它们都叫 "perl某某"。

## 8. 离线文档

---

如果你想多学些 Perl 的东西, 下面是一些我们推荐的相关出版物: [ I ]Perl 5 Pocket Reference [ R ], 3d ed. , by Johan Vromans (O'Reilly, 2000)。这本小册子里面有很方便的 Perl 的快速参考。

[ I ]Perl Cookbook[ R ], by Tom Christiansen and Nathan Torkington (O'Reilly, 1998)。这是你手里现在在这本书的伴侣读物。

[ I ]Elements of Programming with Perl[ R ], by Andrew L. Johnson (Manning, 1999)。这本书教那些非程序员从零开始编程, 而且是用 Perl。

[ I ]Learning Perl[ R ], 2d ed. , by Randal Schwartz and Tom Christiansen (O'Reilly, 1997)。这本书教那些 Unix 系统管理员和程序员在 70% 的时间里要用到的 30% 的基本 Perl 内容。Erik Olson 把这本书重新定位到了在 Microsoft 系统上的程序员; 叫 [ I ]Learning Perl for Win32 Systems[ R ]。

[ I ]Perl: The Programmer's Companion[ R ], by Nigel Chapman (Wiley, 1997)。这本好书主要是给专业计算机科学家和程序员看的, 并没有考虑平台。它迅速并完整地介绍了 Perl。

[ I ]Mastering Regular Expressions[ R ], by Jeffrey Friedl (O'Reilly, 1997)。尽管它没有包含最新版本的 Perl 里面的正则表达式, 但它对任何想学习和了解正则表达式的内部工作的人都是无

价之宝。

[ I ]Object Oriented Perl[ R ], by Damian Conway (Manning, 1999)。对与新老 OO 程序员而言, 这本书解释了在 Perl 里写强大的对象系统的普通和高级的技巧。

[ I ]Mastering Algorithms with Perl[ R ], by Jon Orwant, Jarkko Hietaniemi, and John Macdonald (O'Reilly, 1999)。计算机算法课上所有有用的信息, 不过没有让人痛苦的证明。此书包含图形, 文本, 集合等领域的许多基本并且实用的算法。

[ I ]Writing Apache Modules with Perl and C[ R ], by Lincoln Stein and Doug MacEachern<sup>2</sup> (O'Reilly, 1999)。这本书指导你如何给 Apache 写一个扩展其能力的模块, 特别是使用涡轮加速的 mod\_perl 做快速的 CGI 脚本等。

[ I ]The Perl Journal[ R ], edited by Jon Orwant。这本季刊是程序员写给程序员看的, 包括编程技巧, 编程方法, 新闻等等。

还有许多其它的 Perl 书籍和出版物, 毫无疑问的是我们肯定会漏掉一些很不错的。(当然, 我们没有提到那些差劲的。)

除了上面列出的与 Perl 相关的出版物以外, 我们还推荐下面的书。它们和 Perl 没有直接关系, 但是做参考, 做参考以及从中找到闪光点还是很不错的。

[ I ]The Art of Computer Programming[ R ], by Donald Knuth, vol. 1, [ I ]Fundamental Algorithms[ R ]; vol. 2, [ I ]Seminumerical Algorithms[ R ]; and vol. 3, [ I ]Sorting and Searching[ R ] (Addison-Wesley, 1998)。

[ I ]Introduction to Algorithms[ R ], by Cormen, Leiserson, and Rivest (MIT Press and McGraw<sup>2</sup>-Hill, 1990)。 [ I ]Algorithms in C: Fundamental Data Structures, Sorting, Searching[ R ], 3d ed. , by Robert Sedgewick (Addison-Wesley, 1997)。

[ I ]The Elements of Programming Style[ R ], by Kernighan and Plauger (Prentiss-Hall, 1988)。

[ I ]The Unix Programming Environment[ R ], by Kernighan and Pike (Prentiss-Hall, 1984)。

[ I ]POSIX Programmer's Guide[ R ], by Donald Lewine (O'Reilly, 1991)。

[ I ]Advanced Programming in the UNIX Environment[ R ], by W. Richard Stevens (Addison-Wesley, 1992)。

[ I ]TCP/IP Illustrated[ R ], vols. 1-3, by W. Richard Stevens, (Addison-Wesley, 1994-1996)。

[ I ]The Lord of the Rings[ R ] by J. R. R. Tolkien (most recent printing: Houghton Mifflin, 1999)。

## 9. 其它资源

---

互联网是最宝贵的发明, 我们大家都仍然在发掘它的更多的潜在用途。

### 9.1 Web 上的 Perl

Perl 主页 <http://www.perl.com/>。

里面包括 Perl 世界的新闻以及源程序和移植版本, 专栏文章, 文档, 会议安排等等等等。

还有 Perl 贩子的网页: <http://www.perl.org>

## 9.2 Usenet 新闻组

Perl 的新闻组非常棒, 如果你不知道该到哪里找 Perl 的信息。你的第一站应该是 `comp.lang.perl.moderated`, 它是一个修改过的, 低流量的新闻组, 包括信息宣布和技术讨论。因为是经过修改的, 所以这个组相当易读。

高流量的 `comp.lang.perl.misc` 组讨论关于 Perl 的所有事情, 从技术问题到 Perl 哲学, 从 Perl 游戏到 Perl 诗歌。和 Perl 本身一样, `comp.lang.perl.misc` 就是干事的, 没有什么问题会被认为太愚蠢。

=====**footnote begins**===== 当然, 有些问题太愚蠢, 因此没人愿意回答。(特别是那些在在线手册页和 FAQ 里有的东西。如果你能更快地自己找到答案, 为什么要到新闻组里问呢?)

=====**footnote ends**=====

`comp.lang.perl.tk` 组讨论如何在 Perl 里使用流行的 TK 工具箱。`comp.lang.perl.modules` 组讨论 Perl 模块的开发和使用, 是获取复用程序最好的地方。当你阅读到这里的时候可能还有其他 `comp.lang.perl.某某` 的组, 你可以找找。

如果你不是用普通的新闻阅读器访问 Usenet, 而是用网络浏览器, 那么你要在新闻组名字前面前缀一个 "news:"。(前提是你有一个新闻组服务器。)另外, 如果你用 Alta Vista 或者 Deja 的新闻搜索功能, 那儿你要声明搜索 "\*perl\*" 做新闻组搜索。(译注: 很不幸, Deja 倒了, 不过 google.com 卖了 Deja 的所有数据, 所以可能您得去 google.com 搜索了。)

还有一些其它新闻组你可能也会看看, 至少如果你写 CGI 的话就会去 [ I ] `comp.infosystems.www.authoring.cgi`[ R ]。虽然严格说它不是 Perl 组, 但是在那里讨论的大多数程序都是用 Perl 写的。与 web 相关的 Perl 问题到这个组比较合适, 除非你是在 Apache 里用 `mod_perl`, 这个时候你可能就要看 `comp.infosystems.www.servers.uinx` 了。

## 10. 臭虫报告

---

你很难得有机会找到不是你的程序里的臭虫, 如果真的找到 Perl 的虫子, 那么你应该把它缩减为一个最小的测试环境, 然后用 `perlbug` 程序写一个报告。参阅 <http://bugs.perl.org> 获取详情。

## 11. 这本书里用的习惯

---

一些习惯自己就成了一个大的章节。编码风格在第二十四章, 普通实践里的“风格编程”讨论。我们的用语习惯在词汇表里给出(我们的语言)。

在本书中使用了下列字体习惯。

斜体用于 URL, 手册页, 路径名, 和程序。在文本中第一次出现的新术语也用斜体。这些术语中有很多都有不同的含义, 我们在词汇表里给出那些定义。

定宽字体用于例子和普通文本, 显示任何文本代码。数据值是用带引号(" ")的定宽字体代表的, 这些引号不是数值的一部分。

定宽粗体用于命令行开关。这样就可以把一些开关和操作符区分开, 比如警告开关 "-w" 和文件测试操作符 "-w"。这种字体还用于表示你输入的文本。

定宽斜体用于一般的代码项, 在这些项里你必须替换成特定的值。

我们有许多例子，大部分都应该是一个大程序的一部分。有些例子是完整的程序，你应该能看得出来，因为他们以后 `#!` 开头。我们的长程序几乎都是用：

```
#!/usr/bin/perl
```

开头的。还有一些其它我们在命令行上敲的东西。

我们用 `%` 表示常见的 `shell` 提示符：

```
% perl -e 'print "Hello, world.\n" Hello, world.'
```

这个风格是标准的 `Unix` 命令行的代表，单引号是“全部包围”的形式。其它系统里的引号包围和通配符习惯有所不同。比如，许多在 `MS-DOS` 和 `VMS` 里的命令行代换需要的是双引号而不是单引号，比如在你需要组合命令里面带空格的参数或者通配符的时候。

## 12. 感谢

---

我们在这里要向下面这些我们私下里臭骂过的技术审稿人公开致谢：

Todd Miller, Sharon Hopkins Rauenzahn, Rich Rauenzahn, Paul Marquess, Paul Grassie, Nathan Torkington, Johan Vromans, Jeff Haemer, Gurusamy Sarathy, Gloria Wall, Dan Sugalski, 和 Abigail。

我们还要特别感谢 Tim O'Reilly（和他的出版社）鼓励我们写这些读者可能爱看的東西。

我们愿意听你的意见

我们已经尽可能测试和核实了本书里面的所有信息，但是你还是可能发现有些特性改变了（或者是我们搞错了！）。如果这样，请让我们知道你找到的任何错误，以及你对未来版本的建议，写信给：

O'Reilly & Associates, Inc. 101 Morris Street Sebastopol, CA 95472 1-800-998-9938  
(in the US or Canada) 1-707-829-0515 (international/local) 1-707-829-0104 (fax)

你也可以发电子信息。要想参加 O'Reilly 的邮递列表或者拿一些目录，请给 [info@oreilly.com](mailto:info@oreilly.com) 发邮件。要询问技术问题或者对本书做评论，请发邮件给 [bookquestions@oreilly.com](mailto:bookquestions@oreilly.com)。

我们有个网站用于这本书，在那里我们有勘误和其他一些和 Perl 相关的信息：

<http://www.oreilly.com/catalog/ppperl3>

在这里你还可以找到本书的所有例子代码，这样你就不用象我们这样敲进去了。

- 
- Set MYTITLE = Perl 编程第三版, 第一章 前言

Revision: r1.3 - 18 Aug 2005 - 08:19 - [TingYu](#)

[Perl](#) > [PerlProgramming3](#) > [P3Preface](#)

---

版权 © 1999-2006 归这里所有作者。PostgreSQL 的中文文档版权归何伟平所有。  
向为这里贡献想法,文章的人致敬 [PostgreSQL 中文网](#)  
[反馈意见](#)