Apress®

Expert
Shell Scripting

Shell 脚本专家指南

[美] Ron Peters 著
李晓群 付弘宇 译

这里囊括了所有你迫切希望解决、
却又无处发问的shell脚本问题。

人民邮电出版社
POSTS & TELECOM PRESS

Shell 脚本专家指南

[美] Ron Peters 著
李晓群 付弘宇 译

人民邮电出版社
北京

图书在版编目（CIP）数据

Shell脚本专家指南 / （美）彼得（Peters,R.）著；
李晓群，付弘宇译. -- 北京：人民邮电出版社，2010.8
ISBN 978-7-115-23080-5

Ⅰ. ①S… Ⅱ. ①彼… ②李… ③付… Ⅲ. ①
UNIX操作系统－程序设计 Ⅳ. ①TP316.81

中国版本图书馆CIP数据核字(2010)第096165号

版 权 声 明

Shell 脚本专家指南

◆ 著 [美] Ron Peters
译 李晓群 付弘宇
责任编辑 刘映欣

◆ 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 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：14.75
字数：314 千字 2010 年 8 月第 1 版
印数：1－3 500 册 2010 年 8 月北京第 1 次印刷

著作权合同登记号 图字：01-2009-2886 号
ISBN 978-7-115-23080-5

定价：39.00 元

读者服务热线：(010)67132705 印装质量热线：(010)67129223
反盗版热线：(010)67171154

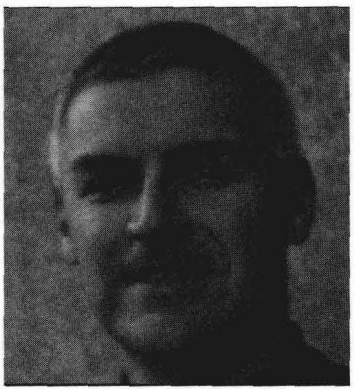
内 容 提 要

本书旨在为 Linux、Unix 以及 OS X 系统管理员提供短小精悍且功能强大的 shell 实现解决方案，教会读者如何使用现有调试器调试 shell 脚本。

全书分为 3 个部分：脚本技术基础、系统交互和高级技术、有用的脚本实例。主要内容包括如何使小到中型的系统管理任务自动化，分析系统数据并编辑配置文件，使用 bash 和 ksh 等编写 Linux、Unix 和 OS X 应用程序的脚本文件等。

本书面向中高级的 shell 程序员，以及需要解决日常问题的系统管理员，但假定读者能够读懂一般的 shell 代码。

作 者 简 介



Ron Peters 在过去的 15 年中大部分时间都在做系统管理员的工作。他曾是 Intel 公司的高级管理员，在那些日子里，他总是每周 7 天、每天 24 小时不停地工作，他也曾是一个专用于设计工作的大型计算机集群的首席管理员。他现在是 Columbia 运动服装公司的 Linux/UNIX 系统管理员。他喜欢和家人在一起，喜欢修理他的道奇 Challenger 汽车，还喜欢玩美式壁球。

致　谢

对于世界上的大多数事情，我们都要依赖他人的帮助来完成。本书也不例外。这个项目的规模超出了我参加过的大多数项目，我不可能独立完成。

首先也是最重要的，我要感谢上帝赐予我生命，以及我拥有的技能，让我可以写这本书。我也要感谢我的妻子 Kathleen、我的两个儿子 Austin 和 Grant，他们忍受了我在计算机前耗费的无数个日日夜夜。

我想对两位 Brian 表达我的感激之情：感谢 Brian Grell 给我提供了思路，我们讨论过的很多内容都包含在本书之中；感谢 Brian Culp 通读了整本书，保证我集中表达了想要表达的东西，他也提出了一些适当的问题，保证了本书内容的清晰。

最后，我想感谢所有帮忙删掉本书中那些“Englilsh 文字”（http://www.homestarrunner.com/sbemail64.swf）的编辑们。

技术审校简介

Brian Culp 已经在信息服务行业工作了 20 年。这期间他曾在初创阶段的小公司工作过，也曾在一家顶级 IT 公司工作十几年，分别做过 UNIX 系统管理员、项目经理、电子商务网站管理员和方案开发人员的工作。他对 UNIX shell 脚本程序的开发和使用总是为了特定的目的，用来解决某个技术问题。他希望读者能利用本书中描述的脚本和方法来创建自己需要的解决问题工具包。

前　言

我在读书期间学到了编程的基本知识，学会了如何根据例子写 shell 脚本。

我曾经认识许多系统管理员或其他*NIX 系统相关人员并和他们合作过，说到管理系统、与环境交互或是编写脚本几方面，他们每个人都有自己的一些技巧。和这些人交谈、互动是很有益的，因为你总是能学到一些东西，可以充实自己的技巧储备。我决定把这些年来学到的所有有用的 shell 脚本编程和交互技术集中起来，汇编成一本好的参考手册。实际上，我在写这本书时也加入了自己对这些技术的一些观点。因为我不可能记住本书中包含的所有内容，我在写到各方面技术时会不时地查些资料。我希望这本书会成为一系列高级参考手册的第一本，这个系列可以不断增长壮大。

读者可能看到过大量的 shell 脚本编程图书和网络资源，它们可以帮助掌握 shell 脚本编程技术。许多 shell 编程书都写得很好，涵盖的内容很广。

本书的主要目的是把一部分最基本的工具、代码片段和高于基本水平的脚本组织起来。我的设想是编一本菜谱，里面是一些并不广为人知的菜肴（技术），以及对我而言很有用的、较高级的算法。

本书包含读者可以直接使用的脚本，以及用于演示某个特定算法的脚本示例。

本书也演示了几个在命令行环境下可能很有用的复杂命令，并试图对脚本进行裁剪，使它们可以应用于多个层次。但在大多数情况下都几乎或完全没有错误检查，因为这不一定是这个特定脚本的重点。读者必须准备好进行某些修改，以适应所在的本地环境。

本书的成因

我的朋友 Brian Culp 和我作为 UNIX 系统管理员，曾共事多年。我们每过一段时间就会在写某个脚本的时候遇到问题。这时，我们当中的一个就会停下工作，走到另一个身边说："你那有完成某某工作的代码吗？"回答可能是否定的，这时我们就开始讨论怎样解决这个问题，或者提出几种不同的解决方案。但更多时候可能是说："唔，是，我记得在一个脚本里做过这样的事，在某某系统上做了某某工作。我找找看。"然后细心操作几次 grep 命令，答案就到手了。

尽管找到需要的答案是很好的结果，但这并没有最有效地利用我们的时间。从有个地方可以存储和组织我们（当然也包括其他程序员）的创作精华，并把它们很好地整理为文档，

到写一本这方面的书，这只需要前进一小步。即使可以从网络资源中快速搜索特定的代码，有很多时候我还是想只要从书架上抽出一本书来查。当需要解决特定的某个问题时，并不一定明确知道要找什么，因此也无法确定要在网上搜索什么。

这在一定程度上说明了我这种时候的窘境。我的家人和朋友都认为我是一个计算机大牛，但我心里很明白，有很多程序员比我更擅长 shell 编码。我主要想把在我作为一位系统管理员的职业生涯中觉得非常有用的 shell 脚本代码进行收集、整理，添加说明，并把这些与他人分享。

本书的目标读者

这本书的目标读者是从中级到高级的 shell 程序员，因此我没有讲解基本的程序结构。如果读者想了解这方面的内容，应该去搜索附录 C 内列举的资源。

这并不是说这本书对初学者没有用处，它同样可以作为对传统的 shell 脚本编码指南的一个补充参考。但是，作为第二语言来学习英语和学习如何用英语来表达讽刺，两者有很大的不同。本书就如同学习表达讽刺，它假定读者已经具有一定的 shell 编码基础。

本书非常详细地讲解了脚本是如何并且为什么编写为当前的形式，也说明了如何避免某些问题。在我的学习过程中发现，在很多学习资源里总是对问题进行模糊处理，而不强调清晰表达，而本书的写作力求清楚明了，宁可啰唆也不会语焉不详。读者可以把本书的很多章节看作是带有超详细注释的 shell 脚本。

本书分为 3 个部分：“基本脚本编程技术”、“系统交互与高级技术”及“有用的脚本”。本书中各章大多是独立的，但可能在一些细节之处引用其他章的内容。

问题和想法

我已经竭尽全力测试本书所含的代码，以保证它们能正确执行。但是，对于如此大规模的工作量，即便有一些人已经通读过这本书，还是可能会有错误。我想知道书中出现了哪些错误，同时更重要的是，我想知道任何可以用于本书修订版中的想法和脚本示例。请发到我的邮箱：rbpeters@peterro.com。

目　录

第 1 部分　基本脚本编程技术

第 1 部分
基本脚本编程技术

第 1 章　shell 脚本的错误检测

虽然这本书不是一本关于“如何编写脚本”的手册，但是还是要介绍一下编写好脚本的基本知识，如代码调试。代码调试是代码编写中非常重要的一部分。无论你是多么训练有素，或者编写代码的技术多么高超，你的代码都会出错，有时是语义错误，有时是逻辑错误。其中句法错误是最容易解决的，因为句法错误会在代码运行时表现出来。而逻辑错误跟踪起来则比较困难，因为包含逻辑错误的代码在运行时不会出现错误，但是运行结果却和程序的设计预期不一致。随着你编程技巧的不断提高，你写的代码也会越来越复杂，这类逻辑错误就会越难以发现和处理。

因为编写没有错误的代码几乎是不可能的，所以需要一些技术来帮助完成代码、发现代码错误以及修改和整理代码。本章将介绍几种我一直使用的、帮助我从脚本文件的内部活动中提取细节的代码调试方法。这些技术保证代码运行结果是符合预期的，并且演示为了完成既定任务，需要对那部分代码做更多的工作。

1.1　shell 跟踪选项

第一种技术是使用 set 命令，这种方法执行起来最简单，而且能提供大量逻辑处理和脚本内部变量值的细节。使用 set 命令其实就是使用 shell 选项来显示脚本运行时的冗余输出。set 命令的一个功能就是打开和关闭 shell 中的各种选项。在这种情况下，设置的选项是-x 或 xtrace，这样在运行脚本时，除了正常的输出之外，还会显示代码运行前每一行代码的扩展命令和变量。有了这些新增加的输出，就可以更方便地观察脚本的运行情况，确定出现错误的代码行。

如果脚本文件中加入了命令“set –x”，那么在 set 命令之后执行的每一条命令以及加在命令行中的任何参数（包括变量和变量的值）都会显示出来。每一行之前都会加上加号（+），提示它是跟踪输出的标识。在子 shell 中执行的 shell 跟踪命令会加两个加号即“++”。

为了演示“set -x”命令的作用，先看如下脚本文件：

```
#!/bin/sh
#set -x
echo -n "Can you write device drivers? "
read answer
answer='echo $answer | tr [a-z] [A-Z]'
if [ $answer = Y ]
then
   echo "Wow, you must be very skilled"
else
   echo "Neither can I, I'm just an example shell script"
fi
```

请注意，“set –x”那一行被注释掉了，当我们在名为 example 的文件中输入上述脚本并运行该文件时，就可以得到期望的结果。

```
$ ./example
Can you write device drivers? y
Wow, you must be very skilled
```

或如下所示：

```
$ ./example
Can you write device drivers? n
Neither can I, Im just an example shell script
```

如果“set –x”那一行没有注释掉，则运行结果如下所示：

```
$ ./example
+ echo -n 'Can you write device drivers? '
Can you write device drivers? + read answer
y
++ tr '[a-z]' '[A-Z]'
++ echo y
+ answer=Y
+ '[' Y = Y ']'
+ echo Wow, you must be very skilled
Wow, you must be very skilled
```

或如下所示：

```
$ ./example
+ echo -n 'Can you write device drivers? '
Can you write device drivers? + read answer
n
++ echo n
++ tr '[a-z]' '[A-Z]'
+ answer=N
+ '[' N = Y ']'
+ echo Neither can I, Im just an example shell script
```

```
Neither can I, Im just an example shell script
```

以上输出就是对脚本运行的一个冗余跟踪。注意，那些不带加号的行是未启用跟踪时脚本的正常输出。正如你所看到的，这种类型的跟踪对于确定脚本运行时变量的值，以及代码中条件判断语句的执行顺序是非常有用的。

shell 的一些选项可以引发这个输出结果的稍许变动，它对错误检测也是有用的。例如，shell 的-v 选项就可以打开冗余模式，和将脚本代码输出（和它正在执行一样）到标准错误文件（经常被缩写成 stderr）中。更特别的是，对于 shell 脚本，执行每一行代码和脚本的其他输出都会被发送到 stderr 文件中（关于文件处理的更多内容参见第 9 章）。下面是上述脚本文件加上“-v”选项之后的输出：

```
$ ./example
echo -n "Can you write device drivers? "
Can you write device drivers? read answer
y
answer='echo $answer | tr [a-z] [A-Z]'
echo $answer | tr [a-z] [A-Z]if [ $answer = Y ]
then
   echo "Wow, you must be very skilled"
else
   echo "Neither can I; I'm just an example shell script"
fi
Wow, you must be very skilled
```

或如下所示：

```
$ ./example
echo -n "Can you write device drivers? "
Can you write device drivers? read answer
n
answer='echo $answer | tr [a-z] [A-Z]'
echo $answer | tr [a-z] [A-Z]if [ $answer = Y ]
then
   echo "Wow, you must be very skilled"
else
   echo "Neither can I; I'm just an example shell script"
fi
Neither can I; I'm just an example shell script
```

用 xtrace（-x）选项可以确认代码是否按照设计预期运行，如果只想看看正在运行的脚本文件的代码（相对于变量的扩展值），那么 shell 中的冗余选项-v 是非常有用的。用“set -xv”同时设置两个选项，可以同时看到两种类型的输出结果，只是不方便浏览。

对于逻辑错误和句法错误的检测，冗余和 xtrace 选项各有所长。和所有 shell 中的选项一样，它们可以打开或者关闭。关闭和打开选项的句法是相反的，除了如前面所述，使用如-x 的减号（-）来打开一个选项，也可以使用加号，例如+x 来关闭某个选项。这样就可以从某个点关闭选