

Linux Shell Scripting Cookbook Third Edition

Linux Shell 脚本攻略

(第3版)

【美】Clif Flynt 【印】Sarah Lakshman Shantanu Tushar 著
门佳 译

- 100多则立竿见影的shell脚本攻略
- 解决系统管理现实问题，实现繁琐任务自动化，轻松驾驭Linux操作系统



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS

作者简介

Clif Flynt

Tcl/Tk及Linux用户社区活跃分子，经常在技术会议和用户小组中发表演说。Noumena公司创始人，负责开发定制软件和举办培训课程。另著有*Tcl/Tk: A Developer's Guide*一书。

Sarath Lakshman

Linux天才程序员，开源软件及GNU/Linux活跃分子，SLYNUX(2005)开发者，*Linux For You*专栏作家。在Fedora、Pardus Linux、PiTiVi、Ubuntu 以及Google编程夏令营等项目中均做出了不可估量的贡献。

Shantanu Tushar

GNU/Linux用户，KDE社区重要贡献者，维护着Calligra Active（用于Tablets的KDE办公文档查看器）、Plasma Media Center以及Gluon Player。Shantanu坚信终有一天编程会变得无比轻松，每个人都会热衷于为计算机编写程序。

译者简介

门佳

GNU / Linux深度用户，喜欢溯本求源，挖掘技术背后的来龙去脉，对程序语言设计理论、编译技术、操作系统设计与实现、Web开发等领域均有涉猎，译著包括《TCP Sockets编程》《精通JavaScript（第2版）》《Linux命令行与shell脚本编程大全（第3版）》以及本书前两版等。

站在巨人的肩上

Standing on Shoulders of Giants



iTuring.cn

站在巨人的肩上
Standing on Shoulders of Giants



iTuring.cn

Linux Shell Scripting Cookbook Third Edition

Linux Shell 脚本攻略

(第3版)

【美】Clif Flynt 【印】Sarath Lakshman Shantanu Tushar 著
门佳 译

人民邮电出版社
北京

图书在版编目（CIP）数据

Linux Shell脚本攻略：第3版 / （美）克里夫·弗林特（Clif Flynt），（印）萨拉特·拉克什曼（Sarah Lakshman），（印）山塔努·图沙尔（Shantanu Tushar）著；门佳译。—北京：人民邮电出版社，2018.3
（图灵程序设计丛书）
ISBN 978-7-115-47738-5

I. ①L… II. ①克… ②萨… ③山… ④门… III. ①Linux操作系统—程序设计 IV. ①TP316. 89

中国版本图书馆CIP数据核字(2018)第003826号

内 容 提 要

本书结合丰富的实际案例介绍了如何利用 shell 命令实现与 Linux 操作系统的高效沟通，具体内容包括：各类日常任务以及如何利用 shell 命令更快速地解决问题；编写脚本从 Web 中挖掘数据并进行处理；在脚本中与简单的 Web API 进行交互；任务的执行及自动化；创建及维护文件和文件夹归档，利用 shell 进行压缩和加密。第 3 版讲解了最新的 Linux 发行版中加入的令人激动的新特性，帮助你完成从未想到过的功能。

本书适合 Linux 系统管理员和程序员阅读，是编写 shell 脚本的绝佳参考资料。

-
- ◆ 著 [美] Clif Flynt [印] Sarah Lakshman Shantanu Tushar
 - 译 门 佳
 - 责任编辑 朱 巍
 - 执行编辑 张海艳
 - 责任印制 周昇亮
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
 - 邮编 100164 电子邮件 315@ptpress.com.cn
 - 网址 <http://www.ptpress.com.cn>
 - 三河市君旺印务有限公司印刷
 - ◆ 开本：800×1000 1/16
 - 印张：25.5
 - 字数：603千字 2018年3月第1版
 - 印数：1-4 000册 2018年3月河北第1次印刷
 - 著作权合同登记号 图字：01-2017-6484号
-

定价：89.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京东工商广登字 20170147 号

前　　言

本书将向你展示如何驾驭Linux操作系统。书中描述了如何执行诸如文件查找这类常见任务，解释了复杂的系统管理工作，例如系统监控和调优，还讨论了网络、安全、应用分发以及云的应用。

普通用户会乐于看到重新格式化照片、下载视频和音频文件以及文件归档这些技巧。

高级用户可以从中找到复杂问题的解决攻略及详细讲解，比如备份、版本控制和网络分组嗅探。

系统管理员和集群管理员则能够学会利用容器、虚拟机和云来简化自己的工作。

本书内容

第1章：小试牛刀。本章讲解了命令行的用法、bash脚本的编写与调试，以及管道和shell配置。

第2章：命令之乐。本章介绍了一些可用于命令行或bash脚本中的常用Linux命令。另外还讲解了如何从文件中读取数据，按照名称、类型或日期查找文件以及进行文件比较。

第3章：以文件之名。本章讲解了文件的相关操作，其中包括文件的查找与比较、文本搜索、目录导航以及处理图像和视频文件。

第4章：让文本飞。本章讲解了如何使用正则表达式以及awk、sed和grep命令。

第5章：一团乱麻？没这回事！本章讲解了在不使用浏览器的情况下如何实现Web交互。另外还演示了如何利用脚本检查网站中的无效链接，以及下载及解析HTML数据。

第6章：仓库管理。本章介绍了如何使用Git和Fossil进行版本控制，跟踪变更以及维护历史记录。

第7章：B计划。本章讨论了传统的和现代的Linux备份工具。磁盘容量越大，你要备份的东西就越多。

第8章：无网不利。本章讲解了网络配置及排错、网络共享以及搭建VPN。

第9章：明察秋毫。本章会帮助你了解系统的运行细节，另外还讲解了如何跟踪磁盘及内存的使用情况、跟踪登录用户以及检查日志文件。

第10章：管理重任。本章讲解了如何管理任务、向用户发送信息、调度自动化任务、书写工作文档以及有效地使用终端。

第11章：觅迹寻踪。本章讲解了如何通过嗅探网络找出故障所在以及跟踪库和系统调用中的问题。

第12章：系统调优。本章帮助你理解如何提升系统性能，如何有效地使用内存、磁盘、I/O以及CPU。

第13章：在云端。本章讲解了何时以及如何利用容器、虚拟机和云来分发应用程序和共享数据。

阅读本书要求

本书中所讲到的攻略可以运行在任何安装了Linux操作系统的计算机上——无论是树莓派还是IBM大型机。

本书读者对象

无论你是新手还是经验老到的系统管理员，都可以从本书中受益。书中兼顾了基本工具和高级概念，除此之外，还有各种实用技巧。

小节

在本书中，你会发现有些标题频繁地出现（预备知识、实战演练、工作原理、补充内容以及参考）。

为了清晰地指明如何完成攻略，我们使用了下面这些小节。

预备知识

本节中给出了攻略的要求，讲述了实现该攻略所需要设置的软件或其他预备知识。

实战演练

本节包含了实现攻略所要完成的步骤。

工作原理

本节通常详细解释了实现步骤背后的原理。

补充内容

为了加深用户的理解，本节给出了有关攻略的一些扩展信息。

参考

本节提供了其他相关的信息源。

本书约定

本书用多种不同格式的文本来区分不同种类的信息。下面是各类格式的例子及其所代表的含义。

正文中的代码、用户输入会像这样显示：“shebang是一个文本行，其中#!位于解释器路径之前。”

代码块以如下形式显示：

```
$> env  
PWD=/home/clif/ShellCookBook  
HOME=/home/clif  
SHELL=/bin/bash  
# ... And many more lines
```

如果我们希望你注意代码块的某个部分，会使用粗体显示相关的代码行或条目：

```
$> env  
PWD=/home/clif/ShellCookBook  
HOME=/home/clif  
SHELL=/bin/bash  
# ... And many more lines
```

命令行输入或输出写成如下形式：

```
$ chmod a+x sample.sh
```

新术语和重要的词句显示为黑体。



警告或重要的提示出现在这里。



建议和窍门则会以这种方式出现。

读者反馈

十分欢迎读者提供反馈意见。我们想知道你对本书的看法：喜欢哪些部分，不喜欢哪些部分。这些反馈对于协助我们编写出真正对读者有所裨益的书至关重要。

你只需要向feedback@packtpub.com发送电子邮件，并在邮件标题中注明书名即可。

如果你在某方面有所专长并且愿意参与图书编写或出版，请参阅我们的作者指南www.packtpub.com/authors。

客户支持

现在你已经拥有了这本由Packt出版的图书，为了让此书尽可能地物有所值，我们还为你提供了其他诸多方面的服务。

下载示例代码

你可以在<http://www.packtpub.com>下载本书的示例代码。如果你是在其他地方购买的本书的英文版，可以访问<http://www.packtpub.com/support>并注册，示例代码将用电子邮件发送给你。

按照以下步骤下载代码：

- (1) 使用电子邮件地址和密码登录或注册；
- (2) 将鼠标指针放在页面顶部的**SUPPORT**标签上；
- (3) 点击**Code Downloads & Errata**；
- (4) 在**Search**框中输入书名；
- (5) 选择你要下载代码的书；
- (6) 从下拉菜单中选择书本的购买途径；
- (7) 点击**Code Download**。

你也可以进入Packt Publishing的网站，点击书籍页面上的**Code Files**按钮来下载代码文件。在**Search**栏中输入书名就可以访问到该页面。注意，你需要先登录你的Packt账户。

代码文件下载好之后，使用最新版的解压缩软件提取其中的文件：

- WinRAR / 7-Zip (Windows)
- Zipeg / iZip / UnRarX (Mac)
- 7-Zip / PeaZip (Linux)

本书的配套代码也可在GitHub上找到：

<https://github.com/PacktPublishing/Linux-Shell-Scripting-Cookbook-Third-Edition>。

其他书籍的代码和视频可以在这里找到：<https://github.com/PacktPublishing/>。任意挑选吧！

下载本书的彩色图片

我们还为你提供了含有本书中彩色截图/图示的PDF文件。这些彩色的图片有助于你理解书中的内容。可以从下面的链接下载：

https://www.packtpub.com/sites/default/files/downloads/LinuxShellScriptingCookbookThirdEdition_ColorImages.pdf

勘误

尽管我们已经竭尽全力确保本书内容准确，但错误终难避免。如果你发现了书中的任何错误，无论是出现在正文中还是代码中的，我们都非常乐于见到你将错误提交给我们。这样不仅能够减少其他读者的困惑，还能帮助我们改进本书后续版本的质量。如果需要提交勘误，请访问<http://www.packtpub.com/submit-errata>，选择相应的书名，单击**Errata Submission Form**链接，就可以开始输入详细的勘误信息了。^①一旦勘误得到确认，我们将接受你的提交，同时勘误内容也将被上传到我们的网站，或者被添加到对应书目勘误区的现有勘误表中。

要查看图书当前的勘误，可以进入<https://www.packtpub.com/books/content/support>，在搜索栏中输入相应的书名。在**Errata**下就会出现之前提交过的勘误信息。

举报盗版

各种媒体在Internet上一直饱受版权侵害的困扰。Packt坚持严格保护版权和授权。如果你在网上发现我社图书的任何形式的盗版，请立即为我们提供地址或网站名称，以便我们采取进一步

^① 读者也可登录图灵社区本书主页 (ituring.com.cn/book/2439) 提交反馈意见、勘误以及下载本书示例代码。

——编者注

请将疑似侵权的网站链接发送至copyright@packtpub.com。

非常感谢你对保护作者知识产权所做的工作，我们将竭诚为读者提供有价值的内容。

疑难解答

如果你对本书的某方面抱有疑问，请通过questions@packtpub.com联系我们，我们会尽力为你解决。

电子书

扫描如下二维码，即可购买本书电子版。



目 录

第1章 小试牛刀	1
1.1 简介	1
1.2 在终端中显示输出	2
1.2.1 预备知识	2
1.2.2 实战演练	4
1.2.3 工作原理	6
1.2.4 补充内容	6
1.3 使用变量与环境变量	7
1.3.1 预备知识	7
1.3.2 实战演练	8
1.3.3 补充内容	9
1.4 使用函数添加环境变量	11
1.4.1 实战演练	11
1.4.2 工作原理	12
1.5 使用 shell 进行数学运算	12
1.6 玩转文件描述符与重定向	14
1.6.1 预备知识	14
1.6.2 实战演练	15
1.6.3 工作原理	17
1.6.4 补充内容	17
1.7 数组与关联数组	19
1.7.1 预备知识	19
1.7.2 实战演练	20
1.7.3 补充内容	20
1.8 别名	21
1.8.1 实战演练	21
1.8.2 补充内容	22
1.9 采集终端信息	23
1.9.1 预备知识	23
1.9.2 实战演练	23
1.10 获取并设置日期及延时	24
1.10.1 预备知识	24
1.10.2 实战演练	25
1.10.3 工作原理	26
1.10.4 补充内容	27
1.11 调试脚本	27
1.11.1 实战演练	28
1.11.2 工作原理	29
1.11.3 补充内容	29
1.12 函数和参数	29
1.12.1 实战演练	30
1.12.2 补充内容	31
1.13 将一个命令的输出发送给另一个命令	33
1.13.1 预备知识	33
1.13.2 实战演练	34
1.13.3 补充内容	34
1.14 在不按下回车键的情况下读入 n 个字符	35
1.15 持续运行命令直至执行成功	36
1.15.1 实战演练	36
1.15.2 工作原理	37
1.15.3 补充内容	37
1.16 字段分隔符与迭代器	37
1.16.1 预备知识	38
1.16.2 实战演练	38
1.17 比较与测试	40
1.18 使用配置文件定制 bash	43
第2章 命令之乐	46
2.1 简介	46

2 目 录

2.2	用 cat 进行拼接	46
2.2.1	实战演练	46
2.2.2	补充内容	47
2.3	录制并回放终端会话	49
2.3.1	预备知识	49
2.3.2	实战演练	49
2.3.3	工作原理	50
2.4	查找并列出文件	50
2.4.1	预备知识	50
2.4.2	实战演练	50
2.4.3	补充内容	51
2.5	玩转 xargs	58
2.5.1	预备知识	59
2.5.2	实战演练	59
2.5.3	工作原理	60
2.5.4	补充内容	60
2.6	用 tr 进行转换	63
2.6.1	预备知识	63
2.6.2	实战演练	64
2.6.3	工作原理	64
2.6.4	补充内容	65
2.7	校验和与核实	67
2.7.1	预备知识	68
2.7.2	实战演练	68
2.7.3	工作原理	68
2.7.4	补充内容	69
2.8	加密工具与散列	71
2.9	行排序	72
2.9.1	预备知识	72
2.9.2	实战演练	72
2.9.3	工作原理	73
2.9.4	补充内容	73
2.10	临时文件命名与随机数	76
2.10.1	实战演练	77
2.10.2	工作原理	77
2.11	分割文件与数据	77
2.11.1	工作原理	78
2.11.2	补充内容	78
2.12	根据扩展名切分文件名	80
2.12.1	实战演练	80
2.12.2	工作原理	80
2.13	多个文件的重命名与移动	82
2.13.1	预备知识	82
2.13.2	实战演练	82
2.13.3	工作原理	83
2.14	拼写检查与词典操作	84
2.14.1	实战演练	84
2.14.2	工作原理	84
2.15	交互输入自动化	85
2.15.1	预备知识	86
2.15.2	实战演练	86
2.15.3	工作原理	86
2.15.4	补充内容	87
2.16	利用并行进程加速命令执行	88
2.16.1	实战演练	88
2.16.2	工作原理	89
2.16.3	补充内容	89
2.17	检查目录以及其中的文件与子目录	89
2.17.1	预备知识	90
2.17.2	实战演练	90
	第 3 章 以文件之名	92
3.1	简介	92
3.2	生成任意大小的文件	92
3.3	文本文件的交集与差集	94
3.3.1	预备知识	94
3.3.2	实战演练	94
3.3.3	工作原理	96
3.4	查找并删除重复文件	97
3.4.1	预备知识	97
3.4.2	实战演练	97
3.4.3	工作原理	98
3.5	文件权限、所有权与粘滞位	99
3.5.1	实战演练	101
3.5.2	补充内容	102
3.6	将文件设置为不可修改	104
3.6.1	预备知识	104
3.6.2	实战演练	104

3.7 批量生成空白文件	105	3.18.2 实战演练	126
3.7.1 预备知识	105		
3.7.2 实战演练	105		
3.8 查找符号链接及其指向目标	106	第4章 让文本飞	128
3.8.1 实战演练	106	4.1 简介	128
3.8.2 工作原理	107	4.2 使用正则表达式	128
3.9 枚举文件类型统计信息	107	4.2.1 实战演练	129
3.9.1 预备知识	107	4.2.2 工作原理	131
3.9.2 实战演练	107	4.2.3 补充内容	131
3.9.3 工作原理	108	4.3 使用 grep 在文件中搜索文本	132
3.10 使用环回文件	109	4.3.1 实战演练	132
3.10.1 实战演练	110	4.3.2 补充内容	134
3.10.2 工作原理	111	4.4 使用 cut 按列切分文件	138
3.10.3 补充内容	111	4.4.1 实战演练	138
3.11 生成 ISO 及混合型 ISO 文件	113	4.4.2 补充内容	139
3.11.1 预备知识	113	4.5 使用 sed 替换文本	140
3.11.2 实战演练	113	4.5.1 实战演练	140
3.11.3 补充内容	114	4.5.2 补充内容	141
3.12 查找并修补文件差异	115	4.6 使用 awk 进行高级文本处理	144
3.12.1 实战演练	115	4.6.1 预备知识	144
3.12.2 补充内容	117	4.6.2 实战演练	144
3.13 使用 head 与 tail 打印文件的 前 10 行和后 10 行	117	4.6.3 工作原理	144
3.14 只列出目录的各种方法	120	4.6.4 补充内容	146
3.14.1 预备知识	120	4.7 统计特定文件中的词频	149
3.14.2 实战演练	120	4.7.1 预备知识	149
3.14.3 工作原理	120	4.7.2 实战演练	150
3.15 在命令行中使用 pushd 和 popd 实现快速定位	121	4.7.3 工作原理	150
3.15.1 预备知识	121	4.7.4 参考	151
3.15.2 实战演练	121	4.8 压缩或解压缩 JavaScript	151
3.15.3 补充内容	122	4.8.1 预备知识	151
3.16 统计文件的行数、单词数和字符数	122	4.8.2 实战演练	152
3.17 打印目录树	123	4.8.3 工作原理	152
3.17.1 预备知识	123	4.8.4 参考	153
3.17.2 实战演练	123	4.9 按列合并多个文件	153
3.17.3 补充内容	124	4.9.1 实战演练	154
3.18 处理视频与图像文件	125	4.9.2 参考	154
3.18.1 预备知识	125	4.10 打印文件或行中的第 n 个单词或列	154
		4.10.1 实战演练	154
		4.10.2 参考	155
		4.11 打印指定行或模式之间的文本	155

4.11.1	预备知识	155
4.11.2	实战演练	155
4.11.3	参考	156
4.12	以逆序形式打印行	156
4.12.1	预备知识	156
4.12.2	实战演练	156
4.12.3	工作原理	157
4.13	解析文本中的电子邮件地址和 URL	157
4.13.1	实战演练	157
4.13.2	工作原理	158
4.13.3	参考	158
4.14	删除文件中包含特定单词的句子	158
4.14.1	预备知识	158
4.14.2	实战演练	159
4.14.3	工作原理	159
4.14.4	参考	159
4.15	对目录中的所有文件进行文本替换	159
4.15.1	实战演练	160
4.15.2	工作原理	160
4.15.3	补充内容	160
4.16	文本切片与参数操作	160
4.16.1	实战演练	160
4.16.2	参考	161
第 5 章	一团乱麻？没这回事！	162
5.1	简介	162
5.2	Web 页面下载	162
5.2.1	预备知识	162
5.2.2	实战演练	163
5.2.3	工作原理	163
5.2.4	补充内容	164
5.3	以纯文本形式下载页面	165
5.3.1	预备知识	165
5.3.2	实战演练	165
5.4	cURL 入门	166
5.4.1	预备知识	166
5.4.2	实战演练	166
5.4.3	工作原理	167
5.4.4	补充内容	167
5.4.5	参考	169
5.5	从命令行访问未读的 Gmail 邮件	170
5.5.1	实战演练	170
5.5.2	工作原理	170
5.5.3	参考	171
5.6	解析网站数据	171
5.6.1	实战演练	171
5.6.2	工作原理	172
5.6.3	参考	172
5.7	图片爬取器及下载工具	172
5.7.1	实战演练	172
5.7.2	工作原理	173
5.7.3	参考	174
5.8	网页相册生成器	174
5.8.1	预备知识	175
5.8.2	实战演练	175
5.8.3	工作原理	176
5.8.4	参考	176
5.9	Twitter 命令行客户端	176
5.9.1	预备知识	177
5.9.2	实战演练	177
5.9.3	工作原理	178
5.9.4	参考	179
5.10	通过 Web 服务器查询单词含义	179
5.10.1	预备知识	179
5.10.2	实战演练	180
5.10.3	工作原理	180
5.10.4	参考	180
5.11	查找网站中的无效链接	181
5.11.1	预备知识	181
5.11.2	实战演练	181
5.11.3	工作原理	182
5.11.4	参考	182
5.12	跟踪网站变动	182
5.12.1	预备知识	182
5.12.2	实战演练	182
5.12.3	工作原理	184
5.12.4	参考	184
5.13	发送 Web 页面并读取响应	184

5.13.1 预备知识	184
5.13.2 实战演练	185
5.13.3 参考	186
5.14 从 Internet 下载视频	186
5.14.1 预备知识	186
5.14.2 实战演练	186
5.14.3 工作原理	186
5.15 使用 OTS 汇总文本	187
5.15.1 预备知识	187
5.15.2 实战演练	187
5.15.3 工作原理	187
5.16 在命令行中翻译文本	187
5.16.1 预备知识	188
5.16.2 实战演练	188
5.16.3 工作原理	188
第 6 章 仓储管理	189
6.1 简介	189
6.2 创建新的 git 仓库	190
6.2.1 预备知识	190
6.2.2 实战演练	190
6.2.3 工作原理	190
6.3 克隆远程 git 仓库	191
6.4 使用 git 添加与提交变更	191
6.5 使用 git 创建与合并分支	192
6.5.1 预备知识	193
6.5.2 实战演练	193
6.5.3 工作原理	193
6.5.4 补充内容	193
6.6 分享工作成果	194
6.7 推送分支	196
6.8 检查 git 仓库状态	197
6.8.1 实战演练	197
6.8.2 工作原理	198
6.9 查看 git 历史记录	198
6.10 查找 bug	198
6.10.1 实战演练	199
6.10.2 补充内容	199
6.11 快照标签	200
6.12 提交信息规范	201
6.13 使用 fossil	202
6.13.1 预备知识	202
6.13.2 实战演练	202
6.14 创建新的 fossil 仓库	202
6.14.1 实战演练	203
6.14.2 工作原理	203
6.14.3 补充内容	203
6.15 克隆远程 fossil 仓库	204
6.15.1 实战演练	204
6.15.2 工作原理	204
6.16 打开 fossil 项目	204
6.16.1 实战演练	204
6.16.2 工作原理	205
6.16.3 补充内容	205
6.17 使用 fossil 添加与提交变更	205
6.17.1 实战演练	205
6.17.2 补充内容	206
6.18 使用 fossil 分支与 fork	206
6.18.1 实战演练	207
6.18.2 工作原理	207
6.18.3 补充内容	208
6.19 使用 fossil 分享工作成果	208
6.19.1 实战演练	208
6.19.2 工作原理	209
6.20 更新本地 fossil 仓库	209
6.21 检查 fossil 仓库状态	209
6.22 查看 fossil 历史记录	210
第 7 章 B 计划	215
7.1 简介	215
7.2 使用 tar 归档	215
7.2.1 预备知识	215
7.2.2 实战演练	216
7.2.3 工作原理	216
7.2.4 补充内容	216
7.2.5 参考	220
7.3 使用 cpio 归档	221
7.3.1 实战演练	221

7.3.2 工作原理	221
7.4 使用 gzip 压缩数据	222
7.4.1 实战演练	222
7.4.2 补充内容	223
7.4.3 参考	225
7.5 使用 zip 归档及压缩	225
7.5.1 实战演练	225
7.5.2 工作原理	226
7.6 更快的归档工具 pbzip2	226
7.6.1 预备知识	226
7.6.2 实战演练	226
7.6.3 工作原理	227
7.6.4 补充内容	227
7.7 创建压缩文件系统	228
7.7.1 预备知识	228
7.7.2 实战演练	228
7.7.3 补充内容	229
7.8 使用 rsync 备份系统快照	229
7.8.1 实战演练	229
7.8.2 工作原理	230
7.8.3 补充内容	231
7.9 差异化归档	232
7.9.1 实战演练	232
7.9.2 工作原理	232
7.10 使用 fsarchiver 创建全盘镜像	233
7.10.1 预备知识	233
7.10.2 实战演练	233
7.10.3 工作原理	234
第 8 章 无网不利	235
8.1 简介	235
8.2 网络设置	236
8.2.1 预备知识	236
8.2.2 实战演练	236
8.2.3 补充内容	237
8.2.4 参考	241
8.3 ping!	241
8.3.1 实战演练	241
8.3.2 补充内容	242
8.4 跟踪 IP 路由	244
8.5 列出网络中所有的活动主机	245
8.5.1 预备知识	245
8.5.2 实战演练	245
8.5.3 工作原理	245
8.5.4 补充内容	246
8.5.5 参考	247
8.6 使用 SSH 在远程主机上执行命令	247
8.6.1 预备知识	247
8.6.2 实战演练	248
8.6.3 补充内容	249
8.7 在远程主机上执行图形化命令	250
8.7.1 实战演练	250
8.7.2 参考	251
8.8 通过网络传输文件	251
8.8.1 预备知识	251
8.8.2 实战演练	251
8.8.3 补充内容	252
8.8.4 参考	254
8.9 连接无线网络	254
8.9.1 预备知识	254
8.9.2 实战演练	254
8.9.3 工作原理	255
8.9.4 参考	256
8.10 实现 SSH 的无密码自动登录	256
8.10.1 预备知识	256
8.10.2 实战演练	256
8.11 使用 SSH 实现端口转发	257
8.11.1 实战演练	257
8.11.2 补充内容	258
8.12 在本地挂载点上挂载远程驱动器	259
8.12.1 预备知识	259
8.12.2 实战演练	259
8.12.3 参考	259
8.13 分析网络流量与端口	259
8.13.1 预备知识	259
8.13.2 实战演练	260
8.13.3 工作原理	260

8.13.4 补充内容	261
8.14 测量网络带宽	261
8.15 创建套接字	262
8.15.1 预备知识	262
8.15.2 实战演练	262
8.15.3 补充内容	263
8.15.4 工作原理	263
8.16 搭建网桥	264
8.16.1 预备知识	264
8.16.2 实战演练	264
8.17 Internet 连接共享	265
8.17.1 预备知识	265
8.17.2 实战演练	265
8.17.3 工作原理	266
8.18 使用 iptables 架设简易防火墙	266
8.18.1 实战演练	267
8.18.2 工作原理	267
8.18.3 补充内容	268
8.19 创建虚拟私有网络	268
8.19.1 预备知识	268
8.19.2 实战演练	269
第 9 章 明察秋毫	274
9.1 简介	274
9.2 监视磁盘使用情况	274
9.2.1 预备知识	275
9.2.2 实战演练	275
9.2.3 补充内容	275
9.3 计算命令执行时间	279
9.3.1 实战演练	279
9.3.2 工作原理	281
9.4 收集登录用户、启动日志及启动 故障的相关信息	281
9.4.1 预备知识	282
9.4.2 实战演练	282
9.5 列出 1 小时内占用 CPU 最多的 10 个进程	284
9.5.1 预备知识	284
9.5.2 实战演练	284
9.5.3 工作原理	285
9.5.4 参考	286
9.6 使用 watch 监视命令输出	286
9.6.1 实战演练	286
9.6.2 补充内容	286
9.7 记录文件及目录访问情况	287
9.7.1 预备知识	287
9.7.2 实战演练	287
9.7.3 工作原理	287
9.8 使用 syslog 记录日志	288
9.8.1 预备知识	288
9.8.2 实战演练	289
9.8.3 参考	289
9.9 使用 logrotate 管理日志文件	290
9.9.1 预备知识	290
9.9.2 实战演练	290
9.9.3 工作原理	290
9.10 通过监视用户登录找出入侵者	291
9.10.1 预备知识	291
9.10.2 实战演练	291
9.10.3 工作原理	293
9.11 监视远程磁盘的健康情况	293
9.11.1 预备知识	293
9.11.2 实战演练	294
9.11.3 工作原理	295
9.11.4 参考	295
9.12 确定系统中用户的活跃时段	295
9.12.1 预备知识	296
9.12.2 实战演练	296
9.12.3 工作原理	297
9.13 电源使用情况的测量与优化	297
9.13.1 预备知识	298
9.13.2 实战演练	298
9.14 监视磁盘活动	298
9.14.1 预备知识	298
9.14.2 实战演练	299
9.15 检查磁盘及文件系统错误	299
9.15.1 预备知识	299
9.15.2 实战演练	299

9.15.3 工作原理	300	10.10 读写 SQLite 数据库	326
9.16 检查磁盘健康情况	300	10.10.1 预备知识	326
9.16.1 预备知识	301	10.10.2 实战演练	327
9.16.2 实战演练	301	10.10.3 工作原理	327
9.16.3 工作原理	303	10.10.4 补充内容	327
9.17 获取磁盘统计数据	303	10.11 读写 MySQL 数据库	328
9.17.1 预备知识	303	10.11.1 预备知识	329
9.17.2 实战演练	303	10.11.2 实战演练	329
9.17.3 工作原理	304	10.11.3 工作原理	332
9.17.4 补充内容	304	10.12 用户管理脚本	333
第 10 章 管理重任	305	10.12.1 实战演练	333
10.1 简介	305	10.12.2 工作原理	335
10.2 收集进程信息	305	10.13 图像文件的批量缩放及格式转换	336
10.2.1 预备知识	305	10.13.1 预备知识	336
10.2.2 实战演练	306	10.13.2 实战演练	336
10.2.3 工作原理	307	10.13.3 工作原理	338
10.2.4 补充内容	307	10.13.4 参考	339
10.2.5 参考	312	10.14 终端截图	339
10.3 which、whereis、whatis 与 file	312	10.14.1 预备知识	339
10.4 杀死进程以及发送和响应信号	313	10.14.2 实战演练	340
10.4.1 预备知识	313	10.15 集中管理多个终端	340
10.4.2 实战演练	313	10.15.1 预备知识	340
10.4.3 补充内容	314	10.15.2 实战演练	341
10.5 向用户终端发送消息	316	第 11 章 觅迹寻踪	342
10.5.1 预备知识	316	11.1 简介	342
10.5.2 实战演练	317	11.2 使用 tcpdump 跟踪分组	342
10.6 /proc 文件系统	318	11.2.1 预备知识	342
10.7 收集系统信息	319	11.2.2 实战演练	343
10.8 使用 cron 进行调度	321	11.2.3 工作原理	345
10.8.1 预备知识	321	11.3 使用 ngrep 查找分组	346
10.8.2 实战演练	321	11.3.1 预备知识	346
10.8.3 工作原理	322	11.3.2 实战演练	346
10.8.4 补充内容	323	11.3.3 工作原理	347
10.9 数据库的形式及用法	324	11.3.4 补充内容	347
10.9.1 预备知识	324	11.4 使用 ip 跟踪网络路由	347
10.9.2 实战演练	325	11.4.1 预备知识	347
10.9.3 补充内容	325	11.4.2 实战演练	348
		11.4.3 工作原理	349

11.5 使用 strace 跟踪系统调用	349
11.5.1 预备知识	350
11.5.2 实战演练	350
11.5.3 工作原理	351
11.6 使用 ltrace 跟踪动态库函数	352
11.6.1 预备知识	352
11.6.2 实战演练	352
11.6.3 工作原理	353
11.6.4 补充内容	353
第 12 章 系统调优.....	355
12.1 简介	355
12.2 识别服务	356
12.2.1 预备知识	356
12.2.2 实战演练	357
12.2.3 补充内容	359
12.3 使用 ss 收集套接字数据	360
12.3.1 预备知识	360
12.3.2 实战演练	360
12.3.3 工作原理	361
12.4 使用 dstat 收集系统 I/O 使用情况	362
12.4.1 预备知识	362
12.4.2 实战演练	362
12.4.3 工作原理	363
12.4.4 补充内容	363
12.5 使用 pidstat 找出资源占用大户	364
12.5.1 预备知识	364
12.5.2 实战演练	364
12.5.3 工作原理	364
12.6 使用 sysctl 调优 Linux 内核	365
12.6.1 预备知识	365
12.6.2 实战演练	365
12.6.3 工作原理	366
12.6.4 补充内容	366
12.7 使用配置文件调优 Linux 系统.....	366
12.7.1 预备知识	367
12.7.2 实战演练	367
12.7.3 工作原理	367
12.8 使用 nice 命令更改调度器优先级.....	367
12.8.1 实战演练	367
12.8.2 工作原理	368
12.8.3 补充内容	368
第 13 章 在云端.....	369
13.1 简介	369
13.2 使用 Linux 容器.....	370
13.2.1 预备知识	370
13.2.2 实战演练	371
13.2.3 工作原理	377
13.3 使用 Docker.....	377
13.3.1 预备知识	377
13.3.2 实战演练	378
13.3.3 工作原理	381
13.4 在 Linux 中使用虚拟机	381
13.4.1 预备知识	381
13.4.2 实战演练	382
13.5 云端的 Linux	382
13.5.1 预备知识	382
13.5.2 实战演练	384
13.5.3 补充内容	384

第1章

小试牛刀



本章内容

- 在终端中显示输出
- 使用变量与环境变量
- 使用函数添加环境变量
- 使用shell进行数学运算
- 玩转文件描述符与重定向
- 数组与关联数组
- 别名
- 采集终端信息
- 获取并设置日期及延时
- 调试脚本
- 函数和参数
- 将一个命令的输出发送给另一个命令
- 在不按下回车键的情况下读入 n 个字符
- 持续运行命令直至执行成功
- 字段分隔符与迭代器
- 比较与测试
- 使用配置文件定制 bash

1.1 简介

起初，计算机从卡片或磁带中读入程序并生成单个报表。没有操作系统，也没有图形化显示器，甚至连交互式提示符都没有。

到了20世纪60年代，计算机开始支持使用交互式终端（通常是电传打字设备或高级打字机）来调用命令。

当贝尔实验室为全新的Unix操作系统创建了交互式用户界面之后，计算机便拥有了一项独有的特性。它可以从文本文件（称为shell脚本）中读取并执行命令，就好像这些命令是在终端中输入的一样。

这种能力是生产力上的一次巨大飞跃。程序员们再也不用输入一堆命令来执行一系列操作，只需要把这些命令保存在文件中，随后轻敲几次按键运行这个文件就可以了。shell脚本不仅节省了时间，而且清楚明白地表明了所执行的操作。

Unix刚开始只支持一种交互式shell，它是由Stephen Bourne所编写的**Bourne Shell (sh)**。

1989年，GNU项目的Brian Fox吸收了大量其他用户界面的特性，编写出了一种全新的shell：**Bourne Again Shell (bash)**。bash shell与Bourne Shell完全兼容，同时又增添了一些来自csh、ksh等的功能。

随着Linux成为最流行的类Unix操作系统实现，bash shell也变成了Unix和Linux中既成事实的标准shell。

本书关注的是Linux和bash。即便如此，书中的大部分脚本都可以运行在使用了bash、sh、ash、dash、ksh或其他sh风格shell的Linux和Unix系统中。

本章将带领读者熟悉shell环境并演示一些基本的shell特性。

1.2 在终端中显示输出

用户是通过终端会话同shell环境打交道的。如果你使用的是基于图形用户界面的系统，这指的就是终端窗口。如果没有图形用户界面（生产服务器或SSH会话），那么登录后你看到的就是shell提示符。

在终端中显示文本是大多数脚本和实用工具经常需要执行的任务。shell可以使用多种方法和格式显示文本。

1.2.1 预备知识

命令都是在终端会话中输入并执行的。打开终端时会出现一个提示符。有很多方法可以配置提示符，不过其形式通常如下：

```
username@hostname$
```

或者也可以配置成root@hostname #，或者简单地显示为\$或#。

\$表示普通用户，#表示管理员用户root。root是Linux系统中权限最高的用户。

以root用户（管理员）的身份直接使用shell来执行任务可不是个好主意。因为如果shell具备较高的权限，命令中出现的输入错误有可能造成更严重的破坏，所以推荐使用普通用户（shell会在提示符中以\$来表明这种身份）登录系统，然后借助sudo这类工具来运行特权命令。使用sudo <command> <arguments>执行命令的效果和root一样。



shell脚本通常以shebang^①起始：

```
#!/bin/bash
```

shebang是一个文本行，其中#!位于解释器路径之前。/bin/bash是Bash的解释器命令路径。bash将以#符号开头的行视为注释。脚本中只有第一行可以使用shebang来定义解释该脚本所使用的解释器。

脚本的执行方式有两种。

(1) 将脚本名作为命令行参数：

```
bash myScript.sh
```

(2) 授予脚本执行权限，将其变为可执行文件：

```
chmod 755 myScript.sh  
./myScript.sh.
```

如果将脚本作为bash的命令行参数来运行，那么就用不着使用shebang了。可以利用shebang来实现脚本的独立运行。可执行脚本使用shebang之后的解释器路径来解释脚本。

使用chmod命令赋予脚本可执行权限：

```
$ chmod a+x sample.sh
```

该命令使得所有用户可以按照下列方式执行该脚本：

```
$ ./sample.sh      #./表示当前目录
```

或者

```
$ /home/path/sample.sh      #使用脚本的完整路径
```

内核会读取脚本的首行并注意到shebang为#!/bin/bash。它会识别出/bin/bash并执行该脚本：

```
$ /bin/bash sample.sh
```

当启动一个交互式shell时，它会执行一组命令来初始化提示文本、颜色等设置。这组命令来自用户主目录中的脚本文件~/.bashrc（对于登录shell则是~/.bash_profile）。Bash shell还维护了一个历史记录文件~/.bash_history，用于保存用户运行过的命令。

① shebang这个词其实是两个字符名称（sharp-bang）的简写。在Unix的行话里，用sharp或hash（有时候是mesh）来称呼字符“#”，用bang来称呼惊叹号“！”，因而shebang合起来就代表了这两个字符。详情请参考：[http://en.wikipedia.org/wiki/Shebang_\(Unix\)](http://en.wikipedia.org/wiki/Shebang_(Unix))。（注：书中脚注均为译者注。）



~表示主目录，它通常是/home/user，其中user是用户名，如果是root用户，则为/root。登录shell是登录主机后创建的那个shell。但登录图形化环境（比如GNOME、KDE等）后所创建的终端会话并不是登录shell。使用GNOME或KDE这类显示管理器登录后并不会读取.profile或.bash_profile（绝大部分情况下不会），而使用ssh登录远程系统时则会读取.profile。shell使用分号或换行符来分隔单个命令或命令序列。比如：

```
$ cmd1 ; cmd2
```

这等同于：

```
$ cmd1
$ cmd2
```

注释部分以#为起始，一直延续到行尾。注释行通常用于描述代码或是在调试期间禁止执行某行代码^①：

```
# sample.sh - echoes "hello world"
echo "hello world"
```

现在让我们继续讨论基本特性。

1.2.2 实战演练

echo是用于终端打印的最基本命令。

默认情况下，echo在每次调用后会添加一个换行符：

```
$ echo "Welcome to Bash"
Welcome to Bash
```

只需要将文本放入双引号中，echo命令就可以将其中的文本在终端中打印出来。类似地，不使用双引号也可以得到同样的输出结果：

```
$ echo Welcome to Bash
Welcome to Bash
```

实现相同效果的另一种方式是使用单引号：

```
$ echo 'text in quotes'
```

这些方法看起来相似，但各有特定的用途及副作用。双引号允许shell解释字符串中出现的特殊字符。单引号不会对其做任何解释。

思考下面这行命令：

^① shell不执行脚本中的任何注释部分。

```
$ echo "cannot include exclamation - ! within double quotes"
```

命令输出如下：

```
bash: !: event not found error
```

如果需要打印像!这样的特殊字符，那就不要将其放入双引号中，而是使用单引号，或是在特殊字符之前加上一个反斜线(\)：

```
$ echo Hello world !
```

或者

```
$ echo 'Hello world !'
```

或者

```
$ echo "Hello world \!"      #将转义字符放在前面
```

如果不使用引号，我们无法在echo中使用分号，因为分号在Bash shell中用作命令间的分隔符：

```
echo hello; hello
```

对于上面的命令，Bash将echo hello作为一个命令，将hello作为另外一个命令。

在下一条攻略中将讨论到的变量替换不会在单引号中执行。

另一个可用于终端打印的命令是printf。该命令使用的参数和C语言中的printf函数一样。例如：

```
$ printf "Hello world"
```

printf命令接受引用文本或由空格分隔的参数。我们可以在printf中使用格式化字符串来指定字符串的宽度、左右对齐方式等。默认情况下，printf并不会自动添加换行符，我们必须在需要的时候手动指定，比如在下面的脚本中：

```
#!/bin/bash
#文件名：printf.sh

printf "%-5s %-10s %-4s\n" No Name Mark
printf "%-5s %-10s %-4.2f\n" 1 Sarath 80.3456
printf "%-5s %-10s %-4.2f\n" 2 James 90.9989
printf "%-5s %-10s %-4.2f\n" 3 Jeff 77.564
```

可以得到如下格式化的输出：

No	Name	Mark
1	Sarath	80.35
2	James	91.00
3	Jeff	77.56

1.2.3 工作原理

%s、%c、%d和%f都是格式替换符 (format substitution character)，它们定义了该如何打印后续参数。%-5s指明了一个格式为左对齐且宽度为5的字符串替换 (-表示左对齐)。如果不指明-，字符串就采用右对齐形式。宽度指定了保留给某个字符串的字符数量。对Name而言，其保留宽度是10。因此，任何Name字段的内容都会被显示在10字符宽的保留区域内，如果内容不足10个字符，余下的则以空格填充。

对于浮点数，可以使用其他参数对小数部分进行舍入 (round off)。

对于Mark字段，我们将其格式化为%-4.2f，其中.2指定保留两位小数。注意，在每行的格式字符串后都有一个换行符 (\n)。

1.2.4 补充内容

使用echo和printf的命令选项时，要确保选项出现在命令中的所有字符串之前，否则Bash会将其视为另外一个字符串。

1. 在echo中转义换行符

默认情况下，echo会在输出文本的尾部追加一个换行符。可以使用选项-n来禁止这种行为。echo同样接受双包含转义序列的双引号字符串作为参数。在使用转义序列时，需要使用echo -e "包含转义序列的字符串"这种形式。例如：

```
echo -e "1\t2\t3"
1 2 3
```

2. 打印彩色输出

脚本可以使用转义序列在终端中生成彩色文本。

文本颜色是由对应的色彩码来描述的。其中包括：重置=0，黑色=30，红色=31，绿色=32，黄色=33，蓝色=34，洋红=35，青色=36，白色=37。

要打印彩色文本，可输入如下命令：

```
echo -e "\e[1;31m This is red text \e[0m"
```

其中\e[1;31m是一个转义字符串，可以将颜色设为红色，\e[0m将颜色重新置回。只需要将31换成想要的色彩码就可以了。

对于彩色背景，经常使用的颜色码是：重置=0，黑色=40，红色=41，绿色=42，黄色=43，蓝色=44，洋红=45，青色=46，白色=47。

要设置彩色背景的话，可输入如下命令：

```
echo -e "\e[1;42m Green Background \e[0m"
```

这些例子中包含了一些转义序列。可以使用man console_codes来查看相关文档。

1.3 使用变量与环境变量

所有的编程语言都利用变量来存放数据，以备随后使用或修改。和编译型语言不同，大多数脚本语言不要求在创建变量之前声明其类型。用到什么类型就是什么类型。在变量名前面加上一个美元符号就可以访问到变量的值。shell定义了一些变量，用于保存用到的配置信息，比如可用的打印机、搜索路径等。这些变量叫作环境变量。

1.3.1 预备知识

变量名由一系列字母、数字和下划线组成，其中不包含空白字符。常用的惯例是在脚本中使用大写字母命名环境变量，使用驼峰命名法或小写字母命名其他变量。

所有的应用程序和脚本都可以访问环境变量。可以使用env或printenv命令查看当前shell中所定义的全部环境变量：

```
$> env  
PWD=/home/clif/ShellCookBook  
HOME=/home/clif  
SHELL=/bin/bash  
# ..... 其他行
```

要查看其他进程的环境变量，可以使用如下命令：

```
cat /proc/$PID/environ
```

其中，`PID`是相关进程的进程ID（`PID`是一个整数）。

假设有一个叫作gedit的应用程序正在运行。我们可以使用pgrep命令获得gedit的进程ID：

```
$ pgrep gedit  
12501
```

那么，你就可以执行以下命令来查看与该进程相关的环境变量：

```
$ cat /proc/12501/environ  
GDM_KEYBOARD_LAYOUT=usGNOME_KEYRING_PID=1560USER=slylinuxHOME=/home/slylinux
```

注意，实际输出的环境变量远不止这些，只是考虑到页面篇幅的限制，这里删除了不少内容。



特殊文件/proc/PID/environ是一个包含环境变量以及对应变量值的列表。每一个变量以name=value的形式来描述，彼此之间由null字符（\0）分隔。形式上确实不太易读。

要想生成一份易读的报表，可以将cat命令的输出通过管道传给tr，将其中的\0替换成\n：

```
$ cat /proc/12501/environ | tr '\0' '\n'
```

1.3.2 实战演练

可以使用等号操作符为变量赋值：

```
varName=value
```

varName是变量名，value是赋给变量的值。如果value不包含任何空白字符（例如空格），那么就不需要将其放入引号中，否则必须使用单引号或双引号。



注意，var = value不同于var=value。把var=value写成var = value是一个常见的错误。两边没有空格的等号是赋值操作符，加上空格的等号表示的是等量关系测试。

在变量名之前加上美元符号（\$）就可以访问变量的内容。

```
var="value" #将"value"赋给变量var
echo $var
```

也可以这样写：

```
echo ${var}
```

输出如下：

```
value
```

我们可以在printf、echo或其他命令的双引号中引用变量值：

```
#!/bin/bash
#文件名:variables.sh
fruit=apple
count=5
echo "We have $count ${fruit}(s)"
```

输出如下：

```
We have 5 apple(s)
```

因为shell使用空白字符来分隔单词，所以我们需要加上一对花括号来告诉shell这里的变量名是fruit，而不是fruit(s)。

环境变量是从父进程中继承而来的变量。例如环境变量HTTP_PROXY，它定义了Internet连接应该使用哪个代理服务器。

该环境变量通常被设置成：

```
HTTP_PROXY=192.168.1.23:3128
export HTTP_PROXY
```

export命令声明了将由子进程所继承的一个或多个变量。这些变量被导出后，当前shell脚本所执行的任何应用程序都会获得这个变量。shell创建并用到了很多标准环境变量，我们也可以导出自己的环境变量。

例如，PATH变量列出了一系列可供shell搜索特定应用程序的目录。一个典型的PATH变量包含如下内容：

```
$ echo $PATH
/home/slynx/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/games
```

各目录路径之间以:分隔。\$PATH通常定义在/etc/environment、/etc/profile或~/.bashrc中。

如果需要在PATH中添加一条新路径，可以使用如下命令：

```
export PATH="$PATH:/home/user/bin"
```

也可以使用

```
$ PATH="$PATH:/home/user/bin"
$ export PATH
$ echo $PATH
/home/slynx/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/games:/home/user/bin
```

这样，我们就将/home/user/bin添加到了PATH中。

另外还有一些众所周知的环境变量：HOME、PWD、USER、UID、SHELL等。

 使用单引号时，变量不会被扩展(expand)，仍依照原样显示。这意味着\$ echo '\$var'会显示\$var。

但如果变量\$var已经定义过，那么\$ echo "\$var"会显示出该变量的值；如果没有定义过，则什么都不显示。

1.3.3 补充内容

shell还有很多内建特性。下面就是其中一些。

1. 获得字符串的长度

可以用下面的方法获得变量值的长度：

```
length=${#var}
```

考虑这个例子：

```
$ var=12345678901234567890  
$ echo ${#var}  
20
```

length就是字符串所包含的字符数。

2. 识别当前所使用的shell

可以通过环境变量SHELL获知当前使用的是哪种shell：

```
echo $SHELL
```

也可以用

```
echo $0
```

例如：

```
$ echo $SHELL  
/bin/bash
```

执行echo \$0命令也可以得到同样的输出：

```
$ echo $0  
/bin/bash
```

3. 检查是否为超级用户

环境变量UID中保存的是用户ID。它可以用于检查当前脚本是以root用户还是以普通用户的身份运行的。例如：

```
If [ $UID -ne 0 ]; then  
    echo Non root user. Please run as root.  
else  
    echo Root user  
fi
```

注意，[实际上是一个命令，必须将其与剩余的字符串用空格隔开。上面的脚本也可以写成：

```
If test $UID -ne 0:  
then  
    echo Non root user. Please run as root.  
else
```

```
echo Root user
fi
```

root用户的UID是0。

4. 修改Bash的提示字符串 (`username@hostname:~$`)

当我们打开终端或是运行shell时，会看到类似于`user@hostname:/home/$`的提示字符串。不同的GNU/Linux发布版中的提示字符串及颜色各不相同。我们可以利用`PS1`环境变量来定义主提示字符串。默认的提示字符串是在文件`~/.bashrc`中的某一行设置的。

- 查看设置变量`PS1`的那一行：

```
$ cat ~/.bashrc | grep PS1
PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
```

- 如果要修改提示字符串，可以输入：

```
slynux@localhost: ~$ PS1="PROMPT>"      #提示字符串已经改变
PROMPT> Type commands here.
```

- 我们可以利用类似于`\e[1;31`的特定转义序列来设置彩色的提示字符串（参考1.2节的内容）。

还有一些特殊的字符可以扩展成系统参数。例如：`\u`可以扩展为用户名，`\h`可以扩展为主机名，而`\w`可以扩展为当前工作目录。

1.4 使用函数添加环境变量

环境变量通常保存了可用于搜索可执行文件、库文件等的路径列表。例如`$PATH`和`$LD_LIBRARY_PATH`，它们通常看起来像这样：

```
PATH=/usr/bin; /bin
LD_LIBRARY_PATH=/usr/lib; /lib
```

这意味着只要shell执行应用程序(二进制文件或脚本)时，它就会首先查找`/usr/bin`，然后查找`/bin`。

当你使用源代码构建并安装程序时，通常需要为新的可执行文件和库文件添加特定的路径。假设我们要将`myapp`安装到`/opt/myapp`，它的二进制文件在`/opt/myapp/bin`目录中，库文件在`/opt/myapp/lib`目录中。

1.4.1 实战演练

这个例子展示了如何将新的路径添加到环境变量的起始部分。第一个例子利用我们目前所讲过的知识来实现，第二个例子创建了一个函数来简化修改操作。本章随后会讲到函数。

```
export PATH=/opt/myapp/bin:$PATH
export LD_LIBRARY_PATH=/opt/myapp/lib; $LD_LIBRARY_PATH
```

PATH和LD_LIBRARY_PATH现在看起来应该像这样：

```
PATH=/opt/myapp/bin:/usr/bin:/bin
LD_LIBRARY_PATH=/opt/myapp/lib:/usr/lib; /lib
```

我们可以在.bashrc文件中定义如下函数，简化路径添加操作：

```
prepend() { [ -d "$2" ] && eval $1=\"\$2':'\$\$1\" && export $1; }
```

该函数用法如下：

```
prepend PATH /opt/myapp/bin
prepend LD_LIBRARY_PATH /opt/myapp/lib
```

1.4.2 工作原理

函数prepend()首先确认该函数第二个参数所指定的目录是否存在。如果存在，eval表达式将第一个参数所指定的变量值设置成第二个参数的值加上：（路径分隔符），随后再跟上第一个参数的原始值。

在进行添加时，如果变量为空，则会在末尾留下一个：。要解决这个问题，可以对该函数再做一些修改：

```
prepend() { [ -d "$2" ] && eval $1=\"\$2\$\\{\$1:+':'\$\$1\\}\\" && export $1 ; }
```

在这个函数中，我们引入了一种shell参数扩展的形式：



`$\{parameter:+expression\}`

如果parameter有值且不为空，则使用expression的值。

通过这次修改，在向环境变量中添加新路径时，当且仅当旧值存在，才会增加：。

1.5 使用 shell 进行数学运算

Bash shell使用let、(())和[]执行基本的算术操作。工具expr和bc可以用来执行高级操作。

实战演练

(1) 可以像为变量分配字符串值那样为其分配数值。这些值会被相应的操作符视为数字。

```
#!/bin/bash
no1=4;
no2=5;
```

(2) let命令可以直接执行基本的算术操作。当使用let时，变量名之前不需要再添加\$，例如：

```
let result=no1+no2
echo $result
```

let命令的其他用法如下：

□ 自加操作

```
$ let no1++
```

□ 自减操作

```
$ let no1--
```

□ 简写形式

```
let no+=6
let no-=6
```

它们分别等同于let no=no+6和let no=no-6。

□ 其他方法

操作符[]的使用方法和let命令一样：

```
result=$[ no1 + no2 ]
```

在[]中也可以使用\$前缀，例如：

```
result=$[$no1 + 5 ]
```

也可以使用操作符(())。出现在(())中的变量名之前需要加上\$：

```
result=$(( no1 + 50 ))
```

expr同样可以用于基本算术操作：

```
result=`expr 3 + 4`
result=$(expr $no1 + 5)
```

以上这些方法不支持浮点数，只能用于整数运算。

(3) bc是一个用于数学运算的高级实用工具，这个精密的计算器包含了大量的选项。我们可以借助它执行浮点数运算并使用一些高级函数：

```
echo "4 * 0.56" | bc
2.24
no=54;
result=`echo "$no * 1.5" | bc`
echo $result
81.0
```

bc可以接受操作控制前缀。这些前缀之间使用分号分隔。

- 设定小数精度。在下面的例子中，参数scale=2将小数位个数设置为2。因此，bc将会输出包含两个小数位的数值：

```
echo "scale=2;22/7" | bc  
3.14
```

- 进制转换。用bc可以将一种进制系统转换为另一种。来看看下面的代码是如何在十进制与二进制之间相互转换的：

```
#!/bin/bash  
用途：数字转换  
no=100  
echo "obase=2;$no" | bc  
1100100  
no=1100100  
echo "obase=10;ibase=2;$no" | bc  
100
```

- 计算平方以及平方根。

```
echo "sqrt(100)" | bc #Square root  
echo "10^10" | bc #Square
```

1.6 玩转文件描述符与重定向

文件描述符是与输入和输出流相关联的整数。最广为人知的文件描述符是stdin、stdout和stderr。我们可以将某个文件描述符的内容重定向到另一个文件描述符中。下面展示了一些文件描述符操作和重定向的例子。

1.6.1 预备知识

在编写脚本的时候会频繁用到标准输入（stdin）、标准输出（stdout）和标准错误（stderr）。脚本可以使用大于号将输出重定向到文件中。命令产生的文本可能是正常输出，也可能是错误信息。默认情况下，正常输出（stdout）和错误信息（stderr）都会显示在屏幕上。我们可以分别为其指定特定的文件描述符来区分两者。

文件描述符是与某个打开的文件或数据流相关联的整数。文件描述符0、1以及2是系统预留的。

- 0 —— stdin（标准输入）。
- 1 —— stdout（标准输出）。
- 2 —— stderr（标准错误）。

说明：

- 1、本PDF仅仅是书籍的样章
- 2、书籍版权归著者和出版社所有
- 3、本PDF未经版权方允许不能在网上传播
- 4、如有需要，请购买正版实体图书
- 5、实在有必要获取完整版PDF

用于非商业用途或者个人研究学习使用

请联系QQ：3508171164 谢谢！！