LINUX 101 HACKS

Practical Examples to Build a Strong Foundation in Linux

Ramesh Natarajan

www.thegeekstuff.com

译者序

《Linux 101 Hacks》是哈工大 IBM 技术俱乐部 08 年新人暑假培训中学习 Linux 基本系统管理知识的参考文献，该书中的很多技巧对于初学者提高系统管理的工作效率很有帮助。考虑到很多初学者因为个人程度或者习惯的原因，更喜欢中文版的文档，因此在开学后，IBM 俱乐部组织所有新人对《Linux 101 Hacks》进行了翻译。从大二到大三的七名同学参与了翻译工作，他们是：朱涛，陶克路，白亚龙，王珑珑，谭振声，黄大伟，许之晟。 本书主审是哈尔滨工业大学计算机学院吴晋老师，同时由李志，熊飞，张智，韩文祥，王澍，张祖羽协助完成部分章节的审校，他们牺牲了宝贵的考试复习和工作时间对译版的初稿进行了细致的批注和修改，让我们受益颇多，在此向他们表示感谢。本书基于原版模板制作，最后的统稿以及排版由朱涛完成。

本书的翻译工作得到了作者 Ramesh Natarajan 和浮图开放实验室的很大支持，在此向他们表示感谢。

本书所有 Hacks 都已在 Ubuntu 9.04 环境下测试通过。本书仅供学习交流之用，整个翻译过程存在错误与疏漏不可避免。如果书中存在任何问题，请用以下方式联系我们：

作者网站： http://www.thegeekstuff.com/

原书网站： http://linux.101hacks.com/

译者网站： blog.footoo.org blog.cliffwoo.com

译者联系方式： Pigsy.Beard@gmail.com cliffwoo@gmail.com

目录

引文

```
"世界上只有 10 种人:理解二进制的人，不理解二进制的人，以及理解格雷码的人"
-- Geek
```

本书中共介绍了101条技巧，以此来帮助你打下坚实的Linux系统管理的基础。每条技巧中都有与之对应的示例来帮助你理解。

这本书包含 13 个章节，前六章的技巧是基于我以前发表的日志整理过来的。后面六章是刚刚发表的。

第十三章“额外技巧”，包含 10 个新加的技巧。我们会在后续的版本中增添更多的技巧。

你还可以在如下网站阅读此书——http://linux.101hacks.com/

关于作者

我是 Ramesh Natarajan, The Geek Stuff 博客以及本书的作者。

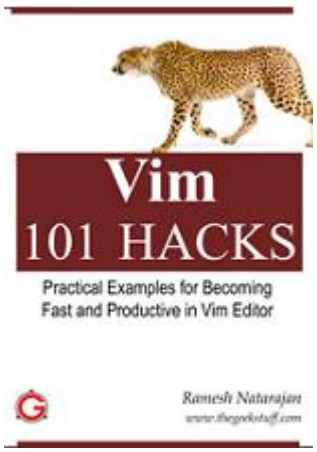
我精通多门编程语言，而 C 是我的最爱。我还曾从事过一些基础架构方面的工作，如 Linux 系统管理，数据库管理，网络，硬件和存储（EMC）。

我还开发过一个免费，简单，可靠的密码管理系统——Password Dragon，它可以 Windows, Linux, Max 下运行。

同时我也是《vim 101 hacks》的作者——thegeekstuff.com/vim-101-hacks-ebook/

如果对本书有意见反馈，请通过这个 thegeekstuff.com/contac 我联系。

电子书《Vim 101 Hacks》



如果你喜欢电子书《Linux 101 Hacks》，考虑一下购买我的另一本书——《Vim 101 Hacks》。

如果你在 Linux/Unix 环境上花很多时间，阅读该书会让你更好的掌握 Vim。

如果你因为 Vim 编辑器不是很直观、使用起来枯燥无趣而决定推迟学习 Vim——那么看了本书后，你将不再孤单。

《Vim 101 Hacks》是一本可下载并包含 101 个基于 Vim 最新特性示例的电子书，通过学习它，你将会更快更高效地使用 Vim 编辑器。

这本电子书中提及的每种技巧都非常的明了而且易于理解。这些实用的技巧会向你展示如何使用 Vim 编辑器的某一种特性。

购买电子书 《Vim 101 hacks》
http://www.thegeekstuff.com/vim-101-hacks-ebook/

版权和声明

前言

又一本技巧的合集？是的！如果你刚完成初级系统管理课程或者想寻求一种能更有效的完成工作的方法，学习这本“Linux 101 Hacks”电子书是个不错的开始。这些实用的技巧写的很好，简练且易读。

真的不错-我会把它推荐给我的学生。

--Prof. Dr. Fritz Mehner, FH Südwestfalen, Germany

（包含 bah-support vim 插件内的 Vim 插件作者）

版本

版本号	日期	修订版
1.0	2009 年 2 月 12 日	第一版
1.1	2010 年 1 月 13 日	增加了第十三章“额外技巧”

在此处下载最新版本。

第一章：强大的 CD 命令技巧

cd 在 Linux（在 Unix 中也是一样）中使用频率很高的一个命令。这一章中提到的 6 中 cd 命令的使用技巧，将会立刻提高你的工作效率，让你更加轻松地在命令行下切换目录。

技巧 1：CDPATH：设置 cd 命令的基目录

如果你经常使用CD命令进入某一个特定目录下的子目录，你可以将CDPATH设置为该目录，当使用 cd 进入其下的子目录时，则不需要键入该目录名,示例如下：

```
 [ramesh@dev-db ~]# pwd
/home/ramesh

[ramesh@dev-db ~]# cd mail
-bash: cd: mail: No such file or directory

[注：以上试图进入当前目录下的 mail 子目录]

[ramesh@dev-db ~]# export CDPATH=/etc
[ramesh@dev-db ~]# cd mail
/etc/mail

[注：以上为进入/etc 下的 mail 目录，而不是当前目录下的 mail 目录]

[ramesh@dev-db /etc/mail]# pwd
/etc/mail
```

如果希望设置永久有效，可以将 export CDPATH=/etc 写入 ~/.bash_profile 。

与环境变量PATH相似的是，你也可以在CDPATH加入多个目录项，每个目录项以冒号分隔，示例如下：例如：

```
export CDPATH=.:~:/etc:/var
```

这个技巧在如下情况中使用相当广泛：

- Oracle的数据库管理员（DBAs）经常要在ORACLE的家目录（$ORACLE_HOME）下工作，就可以将CDPATH变量设置为Oracle的目录。
- Linux/Unix系统管理员经常要在/etc目录下工作，就可以将CDPATH变量设置成/etc
- 开发者要经常在工程目录下工作，如/home/projects，就可以将CDPATH变量设置为 /home/projects
- 普通用户经常要在他们的主目录下进行子目录之间的切换，就可以设置CDPATH变量

为~（这是Linux/Unix中主目录的简单写法）

技巧 2：使用 cd 和 alias 命令快速返回上级目录

当你在一个多级的目录下希望返回上级目录时，经常需要使用 cd ../../ ,至于需要多少个../,完全由你所需要返回的目录级数而定，如下例：

```
# mkdir -p
/tmp/very/long/directory/structure/that/is/too/deep

# cd /tmp/very/long/directory/structure/that/is/too/deep

# pwd
/tmp/very/long/directory/structure/that/is/too/deep

# cd ../../../../

# pwd
/tmp/very/long/directory/structure
```

你可以使用以下四种方法来替代 cd ../../../../ ,以便能快速的向上返回四级目录。：

方法一： 用“..n”退回到高层目录

在下面的方法中，“..4”用来退回 4 级目录，“..3”退回 3 级目录，“..2”退回 2 级目录。将以下的几个别名写入到~/.bashrc（别的 Linux/Unix 可能是~/.bash_profile）中，之后重新登入即可。

```
alias ..="cd .."
alias ..2="cd ../.."
alias ..3="cd ../../.."
alias ..4="cd ../../../.."
alias ..5="cd ../../../../.."
```

```
# cd /tmp/very/long/directory/structure/that/is/too/deep

# ..4
[注：使用..4 向上返回四级目录]

# pwd
/tmp/very/long/directory/structure/
```

方法二：仅使用 "." 返回上级目录

在以下的例子中，.....(5 个点）用于向上返回四级目录。使用 5 个点来代表返回四级目录其实很容易记忆，当你输入前来那个点时，意味着返回上级目录，然后每输入一个点，代表又向上返回一级目录。所以使用....(4 个点)代表返回 3 级目录，2 个点代表返回一级目录。为了使.....(5 个点)可以正常使用，将以下别名加入./.bash_profile 文件，并重新登入。

```
alias ..="cd .."
alias ...="cd ../.."
alias ....="cd ../../.."
alias .....="cd ../../../.."
alias ......="cd ../../../../.."
```

```
# cd /tmp/very/long/directory/structure/that/is/too/deep
# .....
[注:使用.....（5 个点）意味着向上返回四级目录]
# pwd
/tmp/very/long/directory/structure/
```

方法三：用 cd 命令加上连续的”点”实现退回高层目录

在以下的例子中，使用 cd.....(cd 后接 5 个点)代表向上返回四级目录。当你输入前两个点，意味着你想返回上级目录，然后每输入一个点就意味着再向上返回一级目录，所以使用 5 个点来向上返回 4 级目录是很容易记忆的。使用 cd ....（cd 后接 4 个点）向上返回 3 级目录，cd ...(cd 后接 3 个点)向上返回 2 级目录。为了使 cd .....(5 个点)可以正常使用，将以下别名加入./.bash_profile 文件，并重新登入即可。

```
alias cd..="cd .."
alias cd...="cd ../.."
alias cd....="cd ../../.."
alias cd.....="cd ../../../.."
alias cd......="cd ../../../../.."
```

```
# cd /tmp/very/long/directory/structure/that/is/too/deep

# cd.....
[ 注：用 cd.....退回四层目录]

# pwd
/tmp/very/long/directory/structure
```

方法四：用 cd 后跟数字退回高层目录（译者注：原著中为 Method 5）

在下面的例子中，用“cd4”（cd 后跟数字 4）来退回 4 层目录。

```
alias cd1="cd .."
alias cd2="cd ../.."
alias cd3="cd ../../.."
```

```
alias cd4="cd ../../../.."
alias cd5="cd ../../../../.."
```

技巧 3：用一条语句执行“mkdir”和“cd”命令

有时候当你需要新建一个目录后，希望马上进入到这个目录中去（cd），你可以按照下面的方法去做。

```
# mkdir -p /tmp/subdir1/subdir2/subdir3

# cd /tmp/subdir1/subdir2/subdir3

# pwd
/tmp/subdir1/subdir2/subdir3
```

如果将 mkdir 和 cd 操作在一个单独的命令中实现会不会很棒呢？试着将如下代码加入./.bash_profile 并重新登录。

```
$ vi .bash_profile

function mkdircd () { mkdir -p "$@" && eval cd "\"\$$#\""; }
```

好了，现在看一下吧，mkdir 和 cd 已经可以在一个命令行中实现了。

```
# mkdircd /tmp/subdir1/subdir2/subdir3

[注：该命令在创建目录后并自动切换到该目录下]

# pwd
/tmp/subdir1/subdir2/subdir3
```

技巧 4：用“cd -”在最近访问的两个目录之间切换

你可以用“cd -”在最近访问的两个目录之间进行切换，如下所示：

```
# cd /tmp/very/long/directory/structure/that/is/too/deep

# cd /tmp/subdir1/subdir2/subdir3

# cd -

# pwd
/tmp/very/long/directory/structure/that/is/too/deep
```

```
# cd -

# pwd
/tmp/subdir1/subdir2/subdir3

# cd -

# pwd
/tmp/very/long/directory/structure/that/is/too/deep
```

技巧 5：用“dirs”、“pushd”、“popd”来操作目录栈

你可以将目录压入目录栈，也可以稍后将该目录弹出。在随后的示例中，将会用到以下三个命令：

* dirs:显示目录栈
* pushd:将目录压入目录栈
* popd:将目录弹出目录栈

dirs 命令显示当前所在目录及目录栈中的内容。即使目录栈为空，dirs 也会显示当前所在的目录，示例如下：

```
# popd
-bash: popd: directory stack empty

# dirs
~

# pwd
/home/ramesh
```

如何使用 pushd 和 popd 呢？让我们先创建一些临时的目录然后把他们压入目录堆栈中去。

```
# mkdir /tmp/dir1
# mkdir /tmp/dir2
# mkdir /tmp/dir3
# mkdir /tmp/dir4

# cd /tmp/dir1
# pushd .
```

```
# cd /tmp/dir2
# pushd .

# cd /tmp/dir3
# pushd .

# cd /tmp/dir4
# pushd .

# dirs
/tmp/dir4 /tmp/dir4 /tmp/dir3 /tmp/dir2 /tmp/dir1
[注:显示的第一个目录名总会是当前所在的目录，而不是显式压入堆栈里的内容]
```

此时，目录堆栈就包含以下内容：

```
/tmp/dir4
/tmp/dir3
/tmp/dir2
/tmp/dir1
```

最后被压入的目录会在堆栈的顶端。当你执行“popd”，系统就会跳转到栈顶的目录中，并且从堆栈中把它清除掉。正如上面所展示的，最后压入堆栈的是/tmp/dir4，所以当进行一次 popd 时，就会跳转到/tmp/dir4 目录下同时从堆栈中将它移除。如下所示：

```
# popd
# pwd
/tmp/dir4

[注:执行完上 popd 命令之后，目录栈包括以下目录
/tmp/dir3
/tmp/dir2
/tmp/dir1]

# popd
# pwd
/tmp/dir3

[注:执行完上 popd 命令之后，目录栈包括以下目录]
/tmp/dir2
/tmp/dir1]

# popd
# pwd
```

```
/tmp/dir2

[注:执行完上 popd 命令之后，目录栈包括以下目录
/tmp/dir1]

# popd
# pwd
/tmp/dir1

[注:执行完上 popd 命令之后，目录栈包括以下目录]

# popd
-bash: popd: directory stack empty
```

技巧 6：用“shopt -s cdspell”自动纠正“cd”命令的目录名输入错误

使用"shopt -s cdspell"可以自动修正 cd 时拼写错误的目录名。如果你在输入时经常犯些错误，这个命令是很有用的。详见以下示例：

```
# cd /etc/mall
-bash: cd: /etc/mall: No such file or directory

# shopt -s cdspell

# cd /etc/mall

# pwd
/etc/mail
[注：当我错误的把 mail 敲成了 mall，用这个命令 mall 就自动被换成了 mail]
```

第二章：日期操作

技巧 7：设置系统日期和时间

可以使用下列命令修改系统时间

```
# date {mmddhhmiyyyy.ss}
```

mm——代表月份

dd——代表日期

hh——代表 24 小时制的小时

mi——代表分钟

yyyy——代表年

ss——代表秒

例如，将系统日期设置为 2008 年 01 月 31 日，下午 10：19：53

命令为：

```
# date 013122192009.53
```

你也可以按照下面的办法来设置：

```
# date 013122192009.53
# date +%Y%m%d -s "20090131"
# date -s "01/31/2009 22:19:53"
# date -s "31 JAN 2009 22:19:53"
# date set="31 JAN 2009 22:19:53"
```

如果只设置时间：

命令：

```
# date +%T -s "22:19:53"
# date +%T%p -s "10:19:53PM"
```

技巧 8：设置硬件日期和时间

在设置系统日期和时间之前，先要确定操作系统时间已经被设置恰当。方法见：技巧 7。

设置硬件时间要依赖于操作系统时间，具体方法如下：

```
# hwclock -systohc
```

```
# hwclock --systohc --utc
```

不加任何参数使用 hwclock，可以查看当前的硬件日期和时间。

```
# hwclock
```

查看 clock 文件，确认是否设置了 UTC(译者注:协调世界时)：

```
#cat /etc/default/rcS
UTC=yes
```

在其他一些版本的 Linux（如 RebHat）中可以这样查看：

```
# cat /etc/sysconfig/clock

ZONE="America/Los_Angeles"
UTC=false
ARC=false
```

技巧 9：用特定格式显示当前时间

以下的方法可以用各种不同的格式来显示当前时间：

```
$ date
Thu Jan 1 08:19:23 PST 2009

$ date --date="now"
Thu Jan 1 08:20:05 PST 2009

$ date --date="today"
Thu Jan 1 08:20:12 PST 2009

$ date --date='1970-01-01 00:00:01 UTC +5 hours' +%s
18001

$ date '+Current Date: %m/%d/%y%nCurrent Time:%H:%M:%S'
Current Date: 01/01/09
Current Time:08:21:41

$ date +"%d-%m-%Y"
01-01-2009

$ date +"%d/%m/%Y"
01/01/2009
```

```
$ date +"%A,%B %d %Y"
Thursday,January 01 2009
```

以下是 date 命令的不同的格式选项，各选项所代表含义如下：

- %D 日期(月/日/年)
- %d 一个月中的第几天 (01..31)
- %m 月份 (01..12)
- %y 年份的后两位 (00..99)
- %a 当前语言下星期的缩写 (Sun..Sat)
- %A 当前语言下星期的全拼 (Sunday..Saturday)
- %b 当前语言下月份的缩写 (Jan..Dec)
- %B 当前语言下的月份的全称 (January..December)
- %H 24 小时制小时 (00..23)
- %I 12 小时制小时 (01..12)
- %Y 年份(1970…)

技巧 10：显示过去的日期和时间

如下方法可以用来显示过去的日期和时间：

```
$ date --date='3 seconds ago'
Thu Jan 1 08:27:00 PST 2009

$ date --date="1 day ago"
Wed Dec 31 08:27:13 PST 2008

$ date --date="1 days ago"
Wed Dec 31 08:27:18 PST 2008

$ date --date="1 month ago"
Mon Dec 1 08:27:23 PST 2008

$ date --date="1 year ago"
Tue Jan 1 08:27:28 PST 2008

$ date --date="yesterday"
Wed Dec 31 08:27:34 PST 2008

$ date --date="10 months 2 day ago"
Thu Feb 28 08:27:41 PST 2008
```

技巧 11：显示未来的日期和时间

如下的方法可以用来展示未来的日期和时间：

```
$ date
Thu Jan 1 08:30:07 PST 2009

$ date --date='3 seconds'
Thu Jan 1 08:30:12 PST 2009

$ date --date='4 hours'
Thu Jan 1 12:30:17 PST 2009

$ date --date='tomorrow'
Fri Jan 2 08:30:25 PST 2009

$ date --date="1 day"
Fri Jan 2 08:30:31 PST 2009

$ date --date="1 days"
Fri Jan 2 08:30:38 PST 2009

$ date --date="2 days"
Sat Jan 3 08:30:43 PST 2009

$ date --date='1 month'
Sun Feb 1 08:30:48 PST 2009

$ date --date='1 week'
Thu Jan 8 08:30:53 PST 2009

$ date --date="2 months"
Sun Mar 1 08:30:58 PST 2009

$ date --date="2 years"
Sat Jan 1 08:31:03 PST 2011

$ date --date="next day"
Fri Jan 2 08:31:10 PST 2009

$ date --date="-1 days ago"
Fri Jan 2 08:31:15 PST 2009
```

```
$ date --date="this Wednesday"
Wed Jan 7 00:00:00 PST 2009
```

第三章：SSH（Secure SHell）客户端命令

技巧 12：查看 SSH 客户端版本

有的时候需要确认一下 SSH 客户端及其相应的版本号。使用 ssh -V 命令可以得到版本号。需要注意的是，Linux 一般自带的是 OpenSSH:

下面的例子即表明该系统正在使用 OpenSSH：

```
$ ssh -V
OpenSSH_3.9p1, OpenSSL 0.9.7a Feb 19 2003
```

下面的例子表明该系统正在使用 SSH2：

```
$ ssh -V
ssh: SSH Secure Shell 3.2.9.1 (non-commercial version) on
i686-pc-linux-gnu
```

技巧 13：用 SSH 登录到远程主机

当你第一次使用 ssh 登录远程主机时，会出现没有找到主机密钥的提示信息。输入"yes"后，系统会将远程主机的密钥加入到你的主目录下的 .ssh/hostkeys 下，这样你就可以继续操作了。示例如下：

```
localhost$ ssh -l jsmith remotehost.example.com
Host key not found from database.
Key fingerprint:
xabie-dezbc-manud-bartd-satsy-limit-nexiu-jambl-title-jarde-tuxum
You can get a public key's fingerprint by running
% ssh-keygen -F publickey.pub
on the keyfile.
Are you sure you want to continue connecting (yes/no)? Yes
Host key saved to
/home/jsmith/.ssh2/hostkeys/key_22_remotehost.example.com.pub
host key for remotehost.example.com, accepted by jsmith Mon May 26
2008 16:06:50 -0700
jsmith@remotehost.example.com password:
remotehost.example.com$
```

因为远程主机的密钥已经加入到 ssh 客户端的已知主机列表中，当你第二次登陆远程主机时，只需要你输入远程主机的登录密码即可。

```
localhost$ ssh -l jsmith remotehost.example.com
jsmith@remotehost.example.com password:
remotehost.example.com$
```

由于各种原因，可能在你第一次登陆远程主机后，该主机的密钥发生改变，你将会看到一些警告信息。出现这种情况，可能有两个原因：

- 系统管理员在远程主机上升级或者重新安装了SSH服务器
- 有人在进行一些恶意行为，等等。

在你输入“yes”之前呢，最佳的选择或许是联系你的系统管理员来分析为什么会出现主机验证码改变的信息，核对主机验证码是否正确。

```
localhost$ ssh -l jsmith remotehost.example.com
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-
middle attack)!
It is also possible that the host key has just been changed.
Please contact your system administrator.
Add correct host key to
“/home/jsmith/.ssh2/hostkeys/key_22_remotehost.example.com.pub”
to get rid of this message.
Received server key’s fingerprint:
xabie-dezbc-manud-bartd-satsy-limit-nexiu-jambl-title-arde-tuxum
You can get a public key’s fingerprint by running
% ssh-keygen -F publickey.pub
on the keyfile.
Agent forwarding is disabled to avoid attacks by corrupted servers.
Are you sure you want to continue connecting (yes/no)? yes
Do you want to change the host key on disk (yes/no)? yes
Agent forwarding re-enabled.
Host key saved to
/home/jsmith/.ssh2/hostkeys/key_22_remotehost.example.com.pub
host key for remotehost.example.com, accepted by jsmith Mon May 26
2008 16:17:31 -0700
jsmith @remotehost.example.com’s password:
remotehost$
```

技巧 14：调试 SSH 客户端会话

当 ssh 连接出现问题时，我们需要通过查看调试信息来定位这些错误。一般来讲使用 v 选项(注意：是小写的 v)，即可查看调试信息。

没有 SSH 客户端调试信息的例子：

```
localhost$ ssh -l jsmith remotehost.example.com
```

```
warning: Connecting to remotehost.example.com failed: No address
associated to the name
```

包含 ssh 调试信息的例子：

```
locaclhost$ ssh -v -l jsmith remotehost.example.com
debug: SshConfig/sshconfig.c:2838/ssh2_parse_config_ext:
Metaconfig parsing stopped at line 3.
debug: SshConfig/sshconfig.c:637/ssh_config_set_param_verbose:
Setting variable 'VerboseMode' to 'FALSE'.
debug: SshConfig/sshconfig.c:3130/ssh_config_read_file_ext: Read 17
params from config file.
debug: Ssh2/ssh2.c:1707/main: User config file not found, using
defaults. (Looked for '/home/jsmith/.ssh2/ssh2_config')
debug: Connecting to remotehost.example.com, port 22… (SOCKS not used)
warning: Connecting to remotehost.example.com failed: No address
associated to the name
[译者注：很多命令中，v 选项对应的英文是 verbose，也就是详细的信息的意思。]
```

当你使用 ssh 从本机登录到远程主机时，你可能希望切换到本地做一些操作，然后再重新回到远程主机。这个时候，你不需要中断 ssh 连接，只需要按照如下步骤操作即可：

技巧 15：用 SSH 退出符切换 SSH 会话

这个技巧非常实用。尤其是远程登陆到一台主机 A，然后从 A 登陆到 B，如果希望在 A 上做一些操作，还得再开一个终端，很是麻烦。

当你使用 ssh 从本机登录到远程主机时，你可能希望切换到本地做一些操作，然后再重新回到远程主机。这个时候，你不需要中断 ssh 连接，只需要按照如下步骤操作即可：

当你已经登录到了远程主机时，你可能想要回到本地主机进行一些操作，然后又继续回到远程主机。在这种情况下，没有必要断开远程主机的会话，你可以用下面的办法来完成：

1. 登入远程主机：

```
localhost$ ssh -l jsmith remotehost
```

2. 已连接远程主机：

```
remotehost$
```

3. 要临时回到本地主机，输入退出符号：“~”与“Control-Z”组合。

当你输入“~”你不会立即在屏幕上看到，当你按下<Control-Z>并且按回车之后才一起显示。如下，在远程主机中以此输入“~<Control-Z>”

```
remotehost$ ~^Z
[1]+ Stopped ssh -l jsmith remotehost

localhost$
```

4. 现在你已经退回到了本地主机，ssh 远程客户端会话就在 UNIX 后台中运行，你可以向下面那样查看它：

```
localhost$ jobs
[1]+ Stopped ssh -l jsmith remotehost
```

5. 你可以将后台运行的 ssh 会话进程切换到前台，重新回到远程主机，而无需输入密码

```
localhost$ fg %1
ssh -l jsmith remotehost

remotehost$
```

技巧 16：用 SSH 退出字符会话，显示信息

要想取得一些关于当前会话有用的信息，可以按以下方式完成。不过这只能在 SSH 2 客户端上使用。

登录到远程服务器

```
localhost$ ssh -l jsmith remotehost
```

如下所示，在远程服务器上，输入 ssh 退出字符~并输入 s。这样会显示出很多有关当前 ssh 连接的有用信息

```
remotehost$ [注:当你在命令行上输入~s 时，它是不可见的.]

     remote host: remotehost
     local host: localhost
     remote version: SSH-1.99-OpenSSH_3.9p1
     local version: SSH-2.0-3.2.9.1 SSH Secure Shell
(non-commercial)
     compressed bytes in: 1506
     uncompressed bytes in: 1622
     compressed bytes out: 4997
     uncompressed bytes out: 5118
```

```
    packets in: 15
    packets out: 24
    rekeys: 0
    Algorithms:
    Chosen key exchange algorithm: diffie-hellman-group1-sha1
    Chosen host key algorithm: ssh-dss
    Common host key algorithms: ssh-dss,ssh-rsa
    Algorithms client to server:
    Cipher: aes128-cbc
    MAC: hmac-sha1
    Compression: zlib
    Algorithms server to client:
    Cipher: aes128-cbc
    MAC: hmac-sha1
    Compression: zlib

localhost$
```

了解更多关于 SSH 的信息

提醒一下，有关 SSH 公钥认证请参考 openSSH 和 SSH2 程

第四章：重要的 linux 命令

技巧 17：grep 命令

grep 命令是用来在文件中查找含有一些特定文本。 grep 有许多参数，它是一个非常有用的命令

```
语法：grep [选项] pattern [文件名]
```

在一个文件中，如何查找所有可以匹配关键字的内容？

在下面这个例子中，grep 在“/etc/password/” 文件下查找所有含有“John” 的行并全部显示出来

```
# grep John /etc/passwd

jsmith:x:1082:1082:John Smith:/home/jsmith:/bin/bash
jdoe:x:1083:1083:John Doe:/home/jdoe:/bin/bash
```

参数-v 会显示出所有不包含匹配文本的内容。下面的例子中，显示了所有在“/etc/password”下不含有“John”的内容。

注：在“/etc/password”下有几行不包含“John”，只有第一行显示了

```
# grep -v John /etc/passwd

jbourne:x:1084:1084:Jason Bourne:/home/jbourne:/bin/bash
```

在一个特定的文件中有多少行包含指定的匹配内容呢？

在下面这个例子中，显示了在“/etc/password/”下含有“John”的行的总数

```
# grep -c John /etc/passwd
2
```

用-cv 也可以得到不含有“John”的行的总数

```
# grep -cv John /etc/passwd
39
```

如何在查找时忽略大小写

利用“-i”，搜索的时候会忽略大小写

```
# grep -i john /etc/passwd
```

```
jsmith:x:1082:1082:John Smith:/home/jsmith:/bin/bash
jdoe:x:1083:1083:John Doe:/home/jdoe:/bin/bash
```

如何在所有的子目录下执行相应的查找？

可以利用“-r”来完成。在下面的例子中，在“/home/users/”的子目录下忽略大小写，查找“John”，这会以“文件名：匹配的内容”形式显示。也可以利用参数“-l”，只显示文件名。

```
# grep -ri john /home/users

/home/users/subdir1/letter.txt:John, Thanks for your contribution.
/home/users/name_list.txt:John Smith
/home/users/name_list.txt:John Doe

# grep -ril john /root

/home/users/subdir1/letter.txt
/home/users/name_list.txt
```

技巧 18：find 命令

find 命令是一个在 UNIX 文件系统中查找文件的常用命令，可以进行很多条件查找。让我们来看一些 find 命令的实例。

```
语法: find 路径 约束条件
```

如何查找在文件名中含有指定关键字的文件？

下面这条命令查找在“/etc”目录下所有文件名中含有“mail”的文件。

```
# find /etc -name "*mail*"
```

如何查找文件大小超过指定值的文件？

下面这个命令会列出系统中所有大于 100M 的文件

```
# find / -type f -size +100M
```

如何查找在最近几天没有被修改过的文件？

下面这条命令会列出在当前目录下在最近 60 天没有被修改过文件

```
# find . -mtime +60
```

如何查找在最近几天被修改的文件？

下面这条命令会列出在当前目录下在最近 2 天被修改过文件

```
# find . -mtime -2
```

如何删除扩展名为.tar.gz 并且大于 100M 的压缩文件？

当你不想意外删除文件时，那么当执行下列命令要小心点。最好的方法是利用“ls -l”去执行下列相同命令以确保当执行 rm 命令时，你知道什么文件会被删除。

```
# find / -type f -name *.tar.gz -size +100M -exec ls -l {} \;

# find / -type f -name *.tar.gz -size +100M -exec rm -f {} \;
```

如何对最近几天没有被修改的文件进行存档？

下面的命令查找到在“/home/jsmith”目录下最近 60 天没有被修改的文件并且在“/tmp”下创建一个格式为“ddmmyyyy_archive.tar”的归档文件

```
# find /home/jsmith -type f -mtime +60 | xargs tar -cvf /tmp/`date
'+%d%m%Y'_archive.tar`
```

提醒一下：你可以使用“midnight commander”上进行有关文件的一些操作，它是 UNIX 字符界面下一款非常强大的文件管理器。

技巧 19：禁止标准输出和错误信息的输出

当我们调试 shell 脚本的时候，我们往往不希望看到标准输出和标准错误的信息。我们可以使用/dev/nulll 来禁止标准错误的信息。

将标准输出重定向到/dev/null

当你调试 shell 脚本的时候不想看输出，只想看错误信息的时候，这个命令会非常有用。

```
# cat file.txt > /dev/null

# ./shell-script.sh > /dev/null
```

用“2>/dev/null”禁止错误输出

当你只想看标准输出，而不想看错误信息的时候，这个命令会非常的有用。

```
# cat invalid-file-name.txt 2> /dev/null

# ./shell-script.sh 2> /dev/null
```

技巧 20：join 命令

join 命令可基于两个文件的共同项，把他们合并起来。

在下例中，我们有两个文件，分别是 employee.txt 和 salary，他们拥有共同的项-employee-id。所以我们基于 employee-id 项，将两个文件合并起来。

```
$ cat employee.txt

100 Jason Smith
200 John Doe
300 Sanjay Gupta
400 Ashok Sharma

$ cat bonus.txt

100 $5,000
200 $500
300 $3,000
400 $1,250

$ join employee.txt bonus.txt

100 Jason Smith $5,000
200 John Doe $500
300 Sanjay Gupta $3,000
400 Ashok Sharma $1,250
```

技巧 21：改变字符的大小写

将一个文件内容全部转换成大写字母

```
$ cat employee.txt

100 Jason Smith
200 John Doe
300 Sanjay Gupta
400 Ashok Sharma

$ tr a-z A-Z < employee.txt

100 JASON SMITH
```

```
200 JOHN DOE
300 SANJAY GUPTA
400 ASHOK SHARMA
```

将一个文件内容全部转换成小写字母

```
$ cat department.txt

100 FINANCE
200 MARKETING
300 PRODUCT DEVELOPMENT
400 SALES

$ tr A-Z a-z < department.txt

100 finance
200 marketing
300 product development
400 sales
```

技巧 22：xargs 命令

xargs 是一个非常强大的命令，可以取一个命令的输出作为另一个命令的参数。下面就是几个关于如何有效使用 xargs 命令的实例

1. 当你使用 rm 命令去删除很多的文件时，你可能会得到错误信息：“/bin/rm Argument list too long - Linux”。这时可以用 xargs 来避免这个问题

```
find ~ -name '*.log' -print0 | xargs -0 rm -f
```

获得/etc 下所有以.conf 结尾的文件。可以有多种方法获得如下结果。以下命令仅仅为了帮助大家理解如何使用 xargs. find 命令的输入结果一个接一个的传递给 xargs，作为 ls -l 的参数。

```
# find /etc -name "*.conf" | xargs ls -l
```

当你想下载一些 URL，这些 URL 都保存在一个文件里，你可以以如下的方式使用 xargs 命令

```
# cat url-list.txt | xargs wget -c
```

找出所有的.jpg 格式的图片，并将其归档。

```
# find / -name *.jpg -type f -print | xargs tar -cvzf images.tar.gz
```

将所有的 jpg 图片文件复制到外置的硬盘中

```
# ls *.jpg | xargs -n1 -i cp {} /external-hard-drive/directory
```

技巧 23：sort 命令

sort 命令可以对一个文件中的文本行进行排序。以下几个例子来演示如何使用 sort 命令，样例文本是雇员数据，格式如下：

employee_name: employee_id: department_name.

```
$ cat names.txt

Emma Thomas:100:Marketing
Alex Jason:200:Sales
Madison Randy:300:Product Development
Sanjay Gupta:400:Support
Nisha Singh:500:Sales
```

以升序对文本排序

```
$ sort names.txt

Alex Jason:200:Sales
Emma Thomas:100:Marketing
Madison Randy:300:Product Development
Nisha Singh:500:Sales
Sanjay Gupta:400:Support
```

以降序对文本排序

```
$ sort -r names.txt

Sanjay Gupta:400:Support
Nisha Singh:500:Sales
Madison Randy:300:Product Development
Emma Thomas:100:Marketing
Alex Jason:200:Sales
```

对一个使用冒号分隔的文件的第二项进行排序（也就是 employee_id）

```
$ sort -t: -k 2 names.txt

Emma Thomas:100:Marketing
```

```
Alex Jason:200:Sales
Madison Randy:300:Product Development
Sanjay Gupta:400:Support
Nisha Singh:500:Sales
```

对使用 tab 分隔的第三项进行排序(department_id),并去掉重复项

```
$ sort -t: -u -k 3 names.txt

Emma Thomas:100:Marketing
Madison Randy:300:Product Development
Alex Jason:200:Sales
Sanjay Gupta:400:Support
```

对 passwd 文件的第三项进行排序(userid)

```
$ sort -t: -k 3n /etc/passwd | more

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

基于 ip 地址对/etc/hosts 文件排序

```
$ sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n /etc/hosts

127.0.0.1 localhost.localdomain localhost
192.168.100.101 dev-db.thegeekstuff.com dev-db
192.168.100.102 prod-db.thegeekstuff.com prod-db
192.168.101.20 dev-web.thegeekstuff.com dev-web
192.168.101.21 prod-web.thegeekstuff.com prod-web
```

与其它命令组合在一起使用

```
ps -ef | sort : 对进程列表进行排序
ls -al | sort +4n : 使用升序对 ls -al 的输出以文件大小进行排序(第 5 项)
ls -al | sort +4nr : 使用降序对 ls -al 的输出以文件大小进行排序(第 5 项)
```

技巧 24：uniq 命令

uniq 命令可以去除排序过的文件中的重复行，因此 uniq 经常和 sort 合用。也就是说，为了使 uniq 起作用，所有的重复行必须是相邻的。以下是一些常见的例子。

当你有一个包含相同条目的雇员（employee）的文件，你可以以如下方式来删除相同的条目

```
$ sort namesd.txt | uniq
$ sort -u namesd.txt
```

如果你想知道有多少行是相同的，可以像下面这个做。以下例子中的第一列显示该行的重复数量。在本例中，以 Alex 和 Emma 开头的行，在文件中有两个重复行。

```
$ sort namesd.txt | uniq -c

      2 Alex Jason:200:Sales
      2 Emma Thomas:100:Marketing
      1 Madison Randy:300:Product Development
      1 Nisha Singh:500:Sales
      1 Sanjay Gupta:400:Support
```

3. 以下命令仅仅列出了相同的条目

```
$ sort namesd.txt | uniq -cd

      2 Alex Jason:200:Sales
      2 Emma Thomas:100:Marketing
```

技巧 25：cut 命令

cut 命令可以用来显示一个文本文件中特定的列或者其它命令的输出。

下面就是一些例子

显示一个以冒号分隔的文件中的第一列（employee_name）

```
$ cut -d: -f 1 names.txt

Emma Thomas
Alex Jason
Madison Randy
Sanjay Gupta
```

```
Nisha Singh
```

显示一个以冒号分隔的文件中的第一列和第三列

```
$ cut -d: -f 1,3 names.txt

Emma Thomas:Marketing
Alex Jason:Sales
Madison Randy:Product Development
Sanjay Gupta:Support
Nisha Singh:Sales
```

显示文件中每行的前八个字符

```
$ cut -c 1-8 names.txt Emma Tho

Alex Jas
Madison
Sanjay G
Nisha Si
```

将 cut 命令与其它命令组合起来的例子

- **Cut -d: -f1 /etc/passwd** #显示unix系统所有用户的姓名。
- **Free | tr -s ‘’| sed '/^Mem/!d' | cut -d" " -f2** #显示系统中的所有记忆文件

技巧 26：stat 命令

stat 命令那个可以用来查看文件或者文件系统的状态和属性。

显示一个文件或目录的属性

```
$ stat /etc/my.cnf

     File: `/etc/my.cnf'
     Size: 346 Blocks: 16 IO Block: 4096 regular file
Device: 801h/2049d Inode: 279856 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)
Access: 2009-01-01 02:58:30.000000000 -0800
Modify: 2006-06-01 20:42:27.000000000 -0700
Change: 2007-02-02 14:17:27.000000000 -0800
```

```
$ stat /home/ramesh

     File: `/home/ramesh'
     Size: 4096 Blocks: 8 IO Block: 4096 directory
Device: 803h/2051d Inode: 5521409 Links: 7
Access: (0755/drwxr-xr-x) Uid: ( 401/ramesh) Gid: ( 401/ramesh)
Access: 2009-01-01 12:17:42.000000000 -0800
Modify: 2009-01-01 12:07:33.000000000 -0800
Change: 2009-01-09 12:07:33.000000000 -0800
```

使用 -f 参数查看系统文件得某些属性

```
$ stat -f /

     File: "/"
     ID: 0 Namelen: 255 Type: ext2/ext3
Blocks: Total: 2579457 Free: 2008027 Available: 1876998 Size: 4096
Inodes: Total: 1310720 Free: 1215892
```

技巧 27：diff 命令

diff 命令可以用来比较两个文件并显示差异。但是输出结果不是很容易阅读。

```
语法：diff [参数] 文件 1   文件 2
```

与旧的文件比较时，新文件做了哪些修改？

执行 diff 命令时使用 w 选项，可以忽略空格。

一般的 diff 命令的输出是如下形式

o---上面的内容，表明了第一个文件（也就是name_list.txt）中的发生变化
o---下面的内容，表明了第二个文件（也就是name_list_new.txt）听发生的变化。

有关第一个文件中的内容以<开始，有关第二个文件的内容以>开始

diff -w name_list.txt name_list_new.txt

```
2c2,3
< John Doe
---
```

```
> John M Doe
> Jason Bourne
```

技巧 28：显示用户总的连接时间

ac 可以显示用户登录主机的时间信息。

目前登录用户的连接时间

使用 d 选项，可以将输出的时间按天来显示。在本例中，今天我已经登录系统超过 6 个小时了，在 12 月 1 日，我登录系统的时间大约是 1 个小时。

```
$ ac -d

Dec 1 total 1.08
Dec 2 total 0.99
Dec 3 total 3.39
Dec 4 total 4.50
Today total 6.10
```

所有用户的连接时间

就像下面所显示的，可以用“-p” 显示所有用户的连接时间。注意这显示了每一个用户的总的连接时间。

```
$ ac -p

     john                3.64
     madison          0.06
     sanjay           88.17
     nisha             105.92
     ramesh           111.42
     total 309.21
```

显示指定用户的连接时间

```
$ ac -d sanjay

Jul 2 total 12.85
Aug 25 total 5.05
Sep 3 total 1.03
Sep 4 total 5.37
```

```
Dec 24 total 8.15
Dec 29 total 1.42
Today total 2.95
```

第五章：PS1,PS2,PS3 和 PROMPT_COMMAND

技巧 29：PS1——默认提示符

如下所示，可以通过修改 Linux 下的默认提示符，使其更加实用。在下面的例子中，默认的 PS1 的值是“\s-\v\$”，显示出了 shell 的名称的版本。我们通过修改，可以使其显示用户名、主机名和当前工作目录。

```
-bash-3.2$ export PS1="\u@\h \w> "
ramesh@dev-db ~> cd /etc/mail
ramesh@dev-db /etc/mail>

[注：提示符修改为 "username@hostname current-dir>的形式]
```

本例中 PS1 使用的一些代码如下：

- \u -- 用户名
- \h -- 主机名
- \w -- 当前目录的完整路径。请注意当你在主目录下的时候，如上面所示只会显示～

注意，在 PS1 值之后有一个空格。从个人角度来讲，使用这个空格可以增加一定的可读性。

将 export PS1=”\u@\h \w>” 添加到.bash_profile 或者 .bashrc 中，则可以保证其永久有效。

```
ramesh@dev-db ~> vi ~/.bash_profile

ramesh@dev-db ~> vi ~/.bashrc

[注：将命令添加到以上任何一个即可]
```

有关”PS1”的详细用法可以参见下一章

技巧 30：PS2——再谈提示符

一个非常长的命令可以通过在末尾加“\”使其分行显示。多行命令的默认提示符是“>”。 我们可以通过修改 PS2 ，将提示符修改为”continue->” 。

```
ramesh@dev-db ~> myisamchk --silent --force --fast --update-state \
> --key_buffer_size=512M --sort_buffer_size=512M \
> --read_buffer_size=4M --write_buffer_size=4M \
> /var/lib/mysql/bugs/*.MYI
```

```
[注：这里使用 ">" 作为默认提示符]

ramesh@dev-db ~> export PS2="continue-> "
ramesh@dev-db ~> myisamchk --silent --force --fast --update-state \
continue-> --key_buffer_size=512M --sort_buffer_size=512M \
continue-> --read_buffer_size=4M --write_buffer_size=4M \
continue-> /var/lib/mysql/bugs/*.MYI

[注：与上面类似，用 "continue-> " 作提示符]
```

当用“\”使长命令分行显示，我发现非常易读。当然我也见过有的人不喜欢分行显示命令

技巧 31：PS3——Shell 脚本中使用 select 时的提示符

你可以像下面示范的那样，用环境变量 PS3 定制 shell 脚本的 select 提示：

不使用 PS3 的脚本输出：

```
ramesh@dev-db ~> cat ps3.sh
select i in mon tue wed exit
do
   case $i in
     mon) echo "Monday";;
     tue) echo "Tuesday";;
     wed) echo "Wednesday";;
     exit) exit;;
   esac
done

ramesh@dev-db ~> ./ps3.sh
1) mon
2) tue
3) wed
4) exit
#? 1
Monday
#? 4

[注：缺省的提示符是 #?]
```

使用 PS3 的脚本输出：

```
ramesh@dev-db ~> cat ps3.sh
PS3="Select a day (1-4): "
select i in mon tue wed exit
do
  case $i in
     mon) echo "Monday";;
     tue) echo "Tuesday";;
     wed) echo "Wednesday";;
     exit) exit;;
  esac
done

ramesh@dev-db ~> ./ps3.sh
1) mon
2) tue
3) wed
4) exit
Select a day (1-4): 1
Monday
Select a day (1-4): 4

[注：设置了 PS3 变量后，命令提示符已经成为 "Select a day (1-4):"]
```

技巧 32：PS4——PS4-“set -x"用来修改跟踪输出的前缀

如果你像下面那样在调试模式下的脚本中，PS4 环境变量可以定制提示信息：

没有设置PS4 时的shell脚本输出：

```
ramesh@dev-db ~> cat ps4.sh

set -x
echo "PS4 demo script"
ls -l /etc/ | wc -l
du -sh ~

ramesh@dev-db ~> ./ps4.sh
       ++ echo 'PS4 demo script'
PS4 demo script
++ ls -l /etc/
```

```
++ wc -l
243
++ du -sh /home/ramesh
48K     /home/ramesh

[注：当使用 sex -x 跟踪输出时的提示符为 ++]
```

设置PS4 后的脚本输出:

PS4 在 ps. sh 中定义了下面两个变量

- $0 显示当前的脚本名
- $LINENO 显示的当前的行号

```
ramesh@dev-db ~> cat ps4.sh

export PS4='$0.$LINENO+ '
set -x
echo "PS4 demo script"
ls -l /etc/ | wc -l
du -sh ~

ramesh@dev-db ~> ./ps4.sh
../ps4.sh.3+ echo 'PS4 demo script'
PS4 demo script
../ps4.sh.4+ ls -l /etc/
../ps4.sh.4+ wc -l
243
../ps4.sh.5+ du -sh /home/ramesh
48K     /home/ramesh

[注：使用 PS4 后使 "{script-name}.{line-number}+" 成为 set -x 的命令提示符]
```

技巧 33：PROMPT_COMMAND 环境变量

Bash 在显示 PS1 之前先执行 PROMPT_COMMAND 定义的内容：

```
ramesh@dev-db ~> export PROMPT_COMMAND="date +%k:%m:%S"

22:08:42
ramesh@dev-db ~>

[注：PROMPT_COMMAND 和 PS1 在不同行显示]
```

79B 如果你想让 PROMPT_COMMAND 和 PS1 在同一行显示，用 -n 参数即可：

```
ramesh@dev-db ~> export PROMPT_COMMAND="echo -n [$(date+%k:%m:%S)]"

[22:08:51]ramesh@dev-db ~>
[注：PROMPT_COMMAND 和 PS1 在同一行显示]
```

第六章：用功能强大的彩色终端快速使用 PS1

技巧 34：在提示符中显示用户名，主机名，当前目录

在 PS1 的例子中包含了以下三条信息：

- /u ——用户名
- /h ——主机名
- /W ——当前目录名

```
-bash-3.2$ export PS1="\u@\h \W> "

ramesh@dev-db ~> cd /etc/mail

ramesh@dev-db mail>
```

技巧 35：在提示符里显示当前时间

81B 在 PS1 定义的环境变量下可以以$(LINUX COMMAND)的方式执行 LINUX 命令，在下面的例子中用$(date)在提示中显示当前时间：

```
ramesh@dev-db ~> export PS1="\u@\h [\$(date+%k:%m:%S)]> "

ramesh@dev-db [11:09:56]>
```

你还可以用\t 使时间以 hh:mm:ss 格式显示：

```
ramesh@dev-db ~> export PS1="\u@\h [\t]> "

ramesh@dev-db [12:42:55]
```

83B 你还可以用\@使当前时间以 12 小时制显示：

```
ramesh@dev-db ~> export PS1="[\@] \u@\h> "
[04:12 PM] ramesh@dev-db>
```

技巧 36：任意命令的输出作为提示符

你可以在提示里显示任意 LINUX 命令的输出，下列命令显示了三项信息，以"|"做分隔：

- \! 历史命令的个数
- \h 主机名
- $kernel_version 用name -r 显示内核信息
- \$! 最后一条命令的状态

```
ramesh@dev-db ~> kernel_version=$(uname -r)

ramesh@dev-db ~> export PS1="\!|\h|$kernel_version|\$?> "

473|dev-db|2.6.25-14.fc9.i686|0>
```

技巧 37：改变提示符的前景颜色

用蓝色显示提示符中的用户名，主机名，和当前目录信息：

```
$ export PS1="\e[0;34m\u@\h \w> \e[m "

[注：亮蓝色]

$ export PS1="\e[1;34m\u@\h \w> \e[m "

[注：暗绿色]
```

- \e[——表示颜色的提示的起始位置
- x;ym ——表示颜色的代码。颜色的代码在下面有说明
- \e[m ——表示代颜色的提示的结束位置

颜色代码：

```
黑 0:30
蓝 0;34
绿 0;32
青 0;36
红 0;31
紫 0;35
棕 0;33
[注：用 1 代替 0，可以使之变成黑体]
```

将下面的内容写入~/.bashrc 或~/.bash_profile 可以使其永久生效

```
$ vi ~/.bash_profile

STARTCOLOR='\e[0;34m';
ENDCOLOR="\e[0m"
export PS1="$STARTCOLOR\u@\h \w> $ENDCOLOR"
```

技巧 38：改变提示符的背景色

在 PS1 变量用“/e{code}m”语句改变提示符的背景色：

```
$ export PS1="\e[47m\u@\h \w> \e[m "

[注：显示亮灰背景]
```

同时改变前景色和背景色。

```
$ export PS1="\e[0;34m\e[47m\u@\h \w> \e[m "

[注：显示亮蓝突出亮灰背景]
```

87B 在~/.bashrc 或~/.bash_profile 中添加如下语句使上面的关于前景色和和背景色的设置保存下来。

```
$ vi ~/.bash_profile
STARTFGCOLOR='\e[0;34m';
STARTBGCOLOR="\e[47m"
ENDCOLOR="\e[0m"
export PS1="$STARTFGCOLOR$STARTBGCOLOR\u@\h \w>
$ENDCOLOR"
```

尝试下下面的背景色，挑选一个你喜欢的：

- \e[40m
- \e[41m
- \e[42m
- \e[43m
- \e[44m
- \e[45m
- \e[46m
- \e[47m

技巧 39：在提示符中显示多种颜色

在~/bashrc 中加入下面的函数就可以在提示符中显示多种颜色。

```
function prompt {
  local BLUE="\[\033[0;34m\]"
  local DARK_BLUE="\[\033[1;34m\]"
  local RED="\[\033[0;31m\]"
  local DARK_RED="\[\033[1;31m\]"
  local NO_COLOR="\[\033[0m\]"
  case $TERM in
    xterm*|rxvt*)
      TITLEBAR='\[\033]0;\u@\h:\w\007\]'
      ;;
    *)
      TITLEBAR=""
      ;;
  esac
  PS1="\u@\h [\t]> "
  PS1="${TITLEBAR}\
  $BLUE\u@\h $RED[\t]>$NO_COLOR "
  PS2='continue-> '
  PS4='$0.$LINENO+ '
}
```

你可以重新登录或通过“source”命令就可以让上述函数生效。

```
$. ./.bash_profile

$ prompt

ramesh@dev-db [13:02:13]>
```

技巧 40：用 tput 改变提示符颜色

你可以在 PS1 中使用“tput”改变提示符颜色

```
$ export PS1="\[$(tput bold)$(tput setb 4)$(tput setaf
7)\]\u@\h:\w $ \[$(tput sgr0)\]"
```

tput 设置颜色的方法：

- tput setab [1-7] - 通过ANSI转义设置背景色
- tput setb [1-7] - 设置背景色
- tput setaf [1-7] - 通过ANSI转义符设置前景色
- tput setf [1-7] - 设置前景颜色

tput 设置文本的方法：

- tput bold - 设置粗体

- tput dim - 打开半光亮模式
- tput smul - 开始下划线
- tput rmul - 退出下划线模式
- tput rev - 打开反转模式
- tput smso - 进入突出显示模式
- tput rmso - 退出突出显示模式
- tput sgr0 - 关闭所有属性

tput 的颜色码：

- 0 - 黑
- 1 - 红
- 2 - 绿
- 3 - 黄
- 4 - 蓝
- 5 - 洋红
- 6 - 青
- 7 - 白

技巧 41：使用已有的 PS1 变量创建自己的提示符

使用下面的变量定制符合自己口味的提示符：

- \a ASCII响铃字符
- \d 以"星期 月 日"的格式显示日期 (例如，"Tue May 26")
- \D{format} - 格式传给strftime(3),返回的结果的传入到提示字符串中，空的格式会被本地指定的时间取代，需要括号。
- \e ASCII转义字符
- \h 将主机名作为提示符的第一部分
- \H 主机名
- \j 当前shell管理的作业数
- \l shell终端名称
- \n 换行
- \r 回车
- \s shell名，$0 的基础名(最后一个斜线的部分)
- \t 以 24-hour HH:MM:SS 格式显示当前时间
- \T 以 12-hour HH:MM:SS 格式显示当前时间
- \@ 以 12-hour am/pm格式显示当前时间
- \A 以 24-hour HH:MM 格式格式显示当前时间
- \u 当前用户的用户名
- \v bash的版本号(例如 2.00)
- \V bash发行版，版本号＋补丁号(例如 2.00.0)
- \w 当前工作目录,当工作目录是$HOME时，用波浪号表示
- \W 当前工作目录全路径，当工作目录是$HOME时，用波浪号表示
- \! 命令历史序号

- \# 当前命令的命令号
- \$ 如果有效的用户ID是 0，这个值是#，否则为$
- \nnn 字符相应的八进制数
- \\ 反斜杠
- \[非输出字符的开始标记,可以嵌入到提示的终端控制序列中
- \] 非输出字符的结束标记

技巧 42：通过 PS1 调用 bash shell 函数

你可以像下面那样在 PS1 中调用一个 bash shell 函数：

```
ramesh@dev-db ~> function httpdcount {
> ps aux | grep httpd | grep -v grep | wc -l
> }
ramesh@dev-db ~> export PS1="\u@\h [`httpdcount`]> "
ramesh@dev-db [12]>
[注：显示当前运行的 httpd 进程的总数]
```

为使其永久生效，可以将其加入./bash_profile 和 ./bashrc 中

```
$ vi .bash_profile
function httpdcount {
  ps aux | grep httpd | grep -v grep | wc -l
}
export PS1='\u@\h [`httpdcount`]> '
```

技巧 43：通过 PS1 中调用 shell 脚本

你可以在PS1中使用脚本。本例中在PS1中调用~/bin/totalfilesize.sh 脚本 ，该脚本可以计算当前目录下的所有文件大小的总和。

```
ramesh@dev-db ~> cat ~/bin/totalfilesize.sh
for filesize in $(ls -l . | grep "^-" | awk '{print
$5}')
do
  let totalsize=$totalsize+$filesize
done
echo -n "$totalsize"
ramesh@dev-db ~> export PATH=$PATH:~/bin
ramesh@dev-db ~> export PS1="\u@\h
[\$(totalfilesize.sh) bytes]> "

ramesh@dev-db [534 bytes]> cd /etc/mail

ramesh@dev-db [167997 bytes]>
```

```
[注：通过 PS1 调用 totalfilesize.sh 显示当前目录所有文件的总和]
```

第七章：归档和压缩

技巧 44：zip 命令基础

怎样用 zip 压缩多个文件呢？

```
语法: zip {.zip file-name} {file-names}
```

```
# zip var-log-files.zip /var/log/*
  adding: var/log/acpid (deflated 81%)
  adding: var/log/anaconda.log (deflated 79%)
  adding: var/log/anaconda.syslog (deflated 73%)
  adding: var/log/anaconda.xlog (deflated 82%)
  adding: var/log/audit/ (stored 0%)
  adding: var/log/boot.log (stored 0%)
  adding: var/log/boot.log.1 (deflated 40%)
  adding: var/log/boot.log.2 (deflated 42%)
  adding: var/log/boot.log.3 (deflated 40%)
  adding: var/log/boot.log.4 (deflated 40%)
```

如何递归地压缩一个目录及目录下的文件？

```
# zip -r var-log-dir.zip /var/log/
updating: var/log/ (stored 0%)
  adding: var/log/wtmp (deflated 78%)
  adding: var/log/scrollkeeper.log (deflated 94%)
  adding: var/log/rpmpkgs.3 (deflated 68%)
  adding: var/log/spooler (stored 0%)
  adding: var/log/cron.2 (deflated 90%)
  adding: var/log/spooler.1 (stored 0%)
  adding: var/log/spooler.4 (stored 0%)
  adding: var/log/httpd/ (stored 0%)
  adding: var/log/rpmpkgs.1 (deflated 68%)
  adding: var/log/anaconda.log (deflated 79%)
  adding: var/log/secure.2 (deflated 93%)
```

如何解压 a*.zip 的压缩包？

```
# unzip var-log.zip
Archive: var-log.zip
  inflating: var/log/acpid
```

```
  inflating: var/log/anaconda.log
  inflating: var/log/anaconda.syslog
  inflating: var/log/anaconda.xlog
   creating: var/log/audit/
```

用参数 v 查看解压缩过程中的详细信息：

```
# unzip -v var-log.zip

Archive: var-log.zip
 Length   Method    Size Ratio   Date   Time  CRC-32
Name
-------- ------ ------- -----    ----   ----  ------   ---
-
    1916 Defl:N       369 81% 02-08-08 14:27  e2ffdc0c
var/log/acpid
   13546 Defl:N      2900 79% 02-02-07 14:25  34cc03a1
var/log/anaconda.log
skip..
    7680 Defl:N       411 95%  12-30-08 10:55 fe876ee9
var/log/wtmp.1
   40981 Defl:N      7395 82%  02-08-08 14:28 6386a95e
var/log/Xorg.0.log
--------          ------- ---                           ---
----
41406991          2809229 93%                           56
files
```

如何在不解压一个压缩包的情况下看里面的文件呢？

```
# unzip -l var-log.zip

Archive: var-log.zip
  Length     Date   Time Name
 --------    ----   ---- ----
     1916 02-08-08 14:27 var/log/acpid
    13546 02-02-07 14:25 var/log/anaconda.log
..skip..
    40981 02-08-08 14:28 var/log/Xorg.0.log
    40981 02-08-07 14:56 var/log/Xorg.0.log.old
 --------                -------
 41406991                56 files
```

技巧 45：zip 高级用法

zip 命令提供了十个压缩等级：

- 等级 0 是最低等级，只做归档，不压缩
- 等级 1 压缩率低，但速度很快
- 等级 6 是默认的压缩等级
- 等级 9 的压缩率最高，但它耗时也多，除了大文件，我们一般推荐于用等级 9

下面的例子中我分别利用等级 0，等级 6，等级 9 压缩同样的一个目录，看看他们分别压缩后的大小：

```
# zip var-log-files-default.zip /var/log/*

# zip -0 var-log-files-0.zip /var/log/*

# zip -9 var-log-files-9.zip /var/log/*

# ls -ltr
-rw-r--r--   1 root       root    2817248 Jan 1 13:05
var-log-files-default.zip
-rw-r--r--   1 root       root   41415301 Jan 1 13:05
var-log-files-0.zip
-rw-r--r--   1 root       root    2582610 Jan 1 13:06
var-log-files-9.zip
```

技巧 46：zip 文件的密码保护

使用 zip 命令的 P 选项来加密 zip 文件

```
#zip -P mysecurepwd var-log-protected.zip /var/log/*
```

在 shell 脚本中使用上述选项来做后台工作是个不错的选择。但在使用交互式命令行时，你一般不会想让密码在 History 中可见。所以，这个时候就要使用下述的 e 选项来设定密码了。

```
#  zip -e var-log-protected.zip /var/log/*
Enter password:
Verify password:
updating: var/log/acpid (deflated 81%)
updating: var/log/anaconda.log (deflated 79%)
```

当你要解压缩一个有密码保护的压缩文件时，会被要求输入密码

```
#  unzip var-log-protected.zip
Archive: var-log-protected.zip
[var-log-protected.zip] var/log/acpid password:
```

技巧 47：检查 zip 文件的完整性

有时候你想要检查 zip 文件的完整性又不想解压它。这时可以使用如下所述的 t 选项

```
# unzip -t var-log.zip

Archive: var-log.zip
testing: var/log/acpid        OK
testing: var/log/anaconda.log OK
testing: var/log/anaconda.syslog   OK
skip...
testing: var/log/wtmp             OK
testing: var/log/wtmp.1           OK
testing: var/log/Xorg.0.log       OK
No errors detected in compressed data of var-log.zip.
```

技巧 48：tar 命令的基础知识

tar 命令(tape archive)用来将一批文件转成一个归档文件。

```
语法：tar  [选项][档案文件名称][需要归档的文件]
```

怎样将我的用户目录下所有文件和子目录创建一个备份文件？

下面的命令将在/tmp 目录下创建一个名为 my_home_directory.tar 的备份归档。此备份文件包含/home/jsmith 下所有文件和子目录。

选项 c，创建档案文件。

选项 v，verbose 模式，即在命令执行过程中显示更多信息。

选项 f，在命令中指出归档文件名。

```
#tar cvf /tmp/my_home_directory.tar /home/jsmith
```

我该怎么查看归档文件里包含哪些文件?

选项 t 会显示归档文件里面的所有文件

```
# tar tvf /tmp/my_home_directory.tar
```

怎样从档案文件提取所有文件？

使用选项 x 可以从档案文件中提取文件，下面的例子会释放档案文件的内容到当前目录

```
#tar xvf /tmp/my_home_directory.tar
```

怎样将 tar.gz 文件解压到指定目录？

```
#tar xvfz /tmp/my_home_directory.tar.gz -C /home/ramesh
```

技巧 49：在 tar 中使用 gzip, bzip2

怎样在 tar 中使用 gzip？

处理 tar.gz 格式的压缩文件时需要添加选项 z

```
#tar cvfz /tmp/my_home_directory.tar.gz /home/jsmith
#tar xvfz /tmp/my_home_directory.tar.gz
#tar tvfz /tmp/my_home_directory.tar.gz
[注：使用 gzip 要比 bzip2 快]
```

怎样在 tar 中使用 bzip2？

处理 tar.bzip2 格式的压缩文件时需要添加选项 j

```
#tar cvfj /tmp/my_home_directory.tar.bz2 /home/jsmith
#tar xvfj /tmp/my_home_directory.tar.bz2
#tar tvfj /tmp/my_home_directory.tar.bz2
[注：使用 bzip2 会获得比 gzip 高的压缩率]
```

第八章：history 命令

如果你经常使用命令行，那么有效的使用命令历史机制将会使效率获得极大提升。事实上，一旦你掌握了我在下面给出的 15 个例子，你就会发现使用命令行将更有乐趣

技巧 50：使用 HISTTIMEFORMAT 在历史中显示 TIMESTAMP

通常情况下，当你在命令行中键入 history 时，终端中将显示你刚输入的命令及其编号。如果出于审查命令的目的，和命令一起显示时间戳将会很有帮助，如下所示。

```
# export HISTTIMEFORMAT='%F %T '
# history | more
  1  2008-08-05 19:02:39 service network restart
  2  2008-08-05 19:02:39 exit
  3  2008-08-05 19:02:39 id
  4  2008-08-05 19:02:39 cat /etc/redhat-release

[注：你也可以设置 alias 语句来查看最近的历史命令]

  alias h1='history 10'
  alias h2='history 20'
  alias h3='history 30'
```

技巧 51：用 Ctrl ＋ R 搜索历史命令

我非常确信这应该是你最常使用的 history 特性，当你执行了一串相当长的命令之后，你只要用关键字搜索一下历史命令然后重新执行这条命令而不需要将整条命令再输一遍。方法是：按下 Ctrl + R 然后输入关键字。

在以下示例中，我搜索“red”，则显示以前的命令中含有”red”的命令“cat /etc/redhat-release”。

```
[注：在命令行提示符下按下 Ctrl＋R，终端将显示如下提示“reverse-i-search”]

(reverse-i-search)`red`: cat /etc/redhat-release
[注：当看到你要的命令后按回车键，就可以重新执行这条命令了]
# cat /etc/redhat-release
Fedora release 9 (Sulphur)
```

而有的时候你需要在执行一条历史命令之前编辑它.比如，你可以像下面那样搜索“httpd”，终端显示历史命令“service httpd stop”，选择它把“stop”改为“start”然后执行它

```
[注：在命令提示符下按 Ctrl+R ，将会显示提示符”reverse-i-search”]
(reverse-i-search)`httpd`: service httpd stop
```

```
[注：看到你想要的命令后按下左键或者右键,就可以在执行这条命令之前编辑它了]
# service httpd start
```

技巧 52：四种不同的方法快速执行之前的命令

有时出于某些原因你需要执行之前的命令，下面的四种方法可以用来重复最后执行的命令：

1. 用向上键（ up arrow ）查看上条命令，按回车执行。

2. 在命令行中输入!!并按回车。

3. 在命令行中输入!-1 并按回车。

4. 按 Ctrl+P 显示上条命令，按回车执行。

技巧 53：执行历史命令中的特定命令

在下面的例子中，如果你想再次执行第四条命令，执行！4 即可

```
# history | more
1  service network restart
2  exit
3  id
4  cat /etc/redhat-release
# !4
cat /etc/redhat-release
Fedora release 9 (Sulphur)
```

技巧 54：执行以特定字开头的历史命令

输入！和你要重新执行的命令的前几个字母。在下面的例子中，输入！ps，回车，执行历史命令中以“ps”开头的“ps aux | grep yp”

```
# !ps
ps aux | grep yp
root 16947  0.0  0.0   36516   1264   ?       Sl  13:10   0:00 ypbind
root 17503  0.0  0.0    4124    740   pts/0   S+ 19:19   0:00 grep yp
```

技巧 55：用 HISTSIZE 控制历史命令的总数

把下面的两行添加到.bash_profile 然后重新登录 bash（译者注：用 source .bash_profile 即可）查看有什么变化，在这个例子中，bash 命令历史记录中只能储存 450 条命令。

```
# vi ~/.bash_profile
HISTSIZE=450
```

```
HISTFLESIZE=450
```

技巧 56：使用 HISTFILE 改变历史文件名

默认情况下，命令历史被储存在.bash_history 文件中，把下面的一行添加到.bash_profile 文件中，重新登录 shell，则.commandline_warrior 文件将取代.bash_history 文件用来储存历史命令。你可以使用这个命令来追踪不同终端中执行的命令，届时只需要将不同终端中所执行的命令保存在不同的历史文件中即可。

```
# vi ~/.bash_profile
HISTFILE=/root/.commandline_warrior
```

技巧 57：使用 HISTCONTROL 来消除命令历史中的连续重复条目

在下面的例子中 pwd 被输入了三次，当你使用 history 的时候，你会看到这三条命令连续出现。设置 HISTCONTROL 为 ignoredups，来消除重复命令：

```
# pwd
# pwd
# pwd
# history | tail -4
44 pwd
45 pwd
46 pwd
47 history | tail -4
[注：上面 pwd 被执行了三次之后，历史中有三条 pwd 命令]
# export HISTCONTROL=ignoredups
# pwd
# pwd
# pwd
# history | tail -3
56 export HISTCONTROL=ignoredups
57 pwd
58 history | tail -4
[注：即使上面 pwd 被执行了三次，历史中也只有一条 pwd 命令]
```

技巧 58：使用 HISTCONTROL 在整个历史中去除重复命令

上面的 ignoredups 去除连续重复的命令，要消除整个命令历史中的重复命令，把 HISTCONTROL 设置成 erasedups

```
# export HISTCONTROL=erasedups
# pwd
# service httpd stop
```

```
# history | tail -3
38 pwd
39 service httpd stop
40 history | tail -3
# ls -ltr
# service httpd stop
# history | tail -6
35 export HISTCONTROL=erasedups
36 pwd
37 history | tail -3
38 ls -ltr
39 service httpd stop
40 history | tail -6
[注：之前在 pwd 后面的 service httpd stop 命令被除去了]
```

技巧 59：使用 HISTCONTROL 强制 history 忽略某条特定命令

在执行一条命令时，你可以将HISTCONTROL 设置为ignorespace 并在该命令前加上一个空格来指示history 忽略这条命令。可以预见的是，很多初级系统管理员将为他们能从history 中成功隐藏某条命令而振奋。

了解 ignorespace 的工作方式是很好。不过，实践中最好不要从 history 中去掩盖任何命令。

```
# export HISTCONTROL=ignorespace
# ls -ltr
# pwd
#  service httpd stop
[注：在 service 的前面加上一个空格，以便在命令历史中忽略它]
# history | tail -3
67 ls -ltr
68 pwd
69 history | tail -3
```

技巧 60：使用 c 选项清除所有的历史命令

有时候你或许想要清除之前的历史命令。而你又想让 history 继续工作：

```
# history -c
```

技巧 61：替换命令历史中的内容

当你搜索历史命令时，你可能希望执行一个与刚刚查找到的历史命令具有相同的参数的命令。

在本例中，vi 之后的"!!:$" 可以将前一个命令的参数做为当前命令的参数。

```
# ls anaconda-ks.cfg
anaconda-ks.cfg

# vi !!:$
vi anaconda-ks.cfg
```

在下面的例子中，紧跟在 vi 后的参数“!^”把上一个命令的第一个参数传递给当前命令。

```
# cp anaconda-ks.cfg anaconda-ks.cfg.bak
anaconda-ks.cfg

# vi !^
vi anaconda-ks.cfg
```

技巧 62：替换特定命令的特定参数

在下例中，!cp:2 在命令历史中查找以 cp 开头的命令，并将其第二个参数做为当前命令(ls -l)的参数。

```
# cp ~/longname.txt /really/a/very/long/path/long-filename.txt

# ls -l !cp:2
ls -l /really/a/very/long/path/long-filename.txt
```

在下面的例子中“!cp:$”寻找先前以 cp 为开头的命令，并把该条指令的最后一个参数(在这个例子中依旧是第二个参数）替换到“ls -l”中。

```
# ls -l !cp:$
ls -l /really/a/very/long/path/long-filename.txt
```

技巧 63：用 HISTSIZE 禁用 history

如果你想禁用 history，又不让 bash shell 记录你的命令，像下面那样把 HISTSIZE 设为 0 即可。

```
# export HISTSIZE=0
# history

# [注：执行 history 后没有显示任何信息]
```

技巧 64：用 HISTIGNORE 让 history 在存储时忽略某些指令

有时你不想在记录里看到诸如“pwd”，“ls”之类的基本指令，可以用 HISTIGNORE 忽略这些指令。

注意在 HISTIGNORE 中添加“ls”，只忽略“ls”不忽略“ls -l”。一定要准确的写出要忽略的的指令。

```
# export HISTIGNORE="pwd:ls:ls -ltr:"

# pwd

# ls

# ls -ltr

# service httpd stop

# history | tail -3
79 export HISTIGNORE="pwd:ls:ls -ltr:"
80 service httpd stop
81 history

[注: history 命令不显示 pwd 和 ls]
```

第九章：系统管理任务

技巧 65：用 fdisk 进行分区

当你在服务器上装上全新的硬盘后，你需要用类似 fdisk 这样的工具进行相应的分区。

fdisk 中相应的提供 5 种基本操作：

- n – 创建新分区
- d – 删除一个已有的分区
- p – 打印现有的分区表
- w – 把改动写入到分区表，也就是保存.
- q – 离开 fdisk

创建新分区

在下面的例子中，我创建了主分区 /dev/sda1

```
# fdisk /dev/sda
Device contains neither a valid DOS partition table, nor Sun, SGI or
OSF disklabel Building a new DOS disklabel. Changes will remain in
memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.
The number of cylinders for this disk is set to 34893.
There is nothing wrong with that, but this is larger than 1024, and
could in certain setups cause problems with:
 1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)
Warning: invalid flag 0x0000 of partition table 4 will be corrected
by w(rite)
Command (m for help): p
Disk /dev/sda: 287.0 GB, 287005343744 bytes
255 heads, 63 sectors/track, 34893 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
```

```
Partition number (1-4): 1
First cylinder (1-34893, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-34893, default 34893):
Using default value 34893
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

确认分区是否创建成功

```
# fdisk /dev/sda
The number of cylinders for this disk is set to 34893.
There is nothing wrong with that, but this is larger than 1024, and
could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)
Command (m for help): p
Disk /dev/sda: 287.0 GB, 287005343744 bytes
255 heads, 63 sectors/track, 34893 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/sda1 1 34893 280277991 83 Linux
Command (m for help): q
```

技巧 66：用 mke2fsk 格式化分区

在硬盘分区后，硬盘没有被格式化，因此，分区尚不能使用。在这个阶段，如果我们尝试去查看硬盘信息的话，系统将会给出以下的错误信息说明有效的超级块（superblock）不存在：

```
# tune2fs -l /dev/sda1

tune2fs 1.35 (28-Feb-2004)
tune2fs: Bad magic number in super-block while trying to open /dev/sda1
Couldn't find valid filesystem superblock.
```

可以使用 mke2fs 来格式化硬盘。

```
# mke2fs /dev/sda1
```

你也可以在 mke2fs 中使用如下参数 ：

- -m 0 : reserved-blocks-percentage –这个用来指示文件系统保留给根用户的块的比例。默认情况下是 5%。在下面的例子中，它设置为 0 。
- -b 4096 : block-size specified in bytes. 有效值为每个块 1024，2048 或 4096 字节。

```
# mke2fs -m 0 -b 4096 /dev/sda1
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
205344 inodes, 70069497 blocks
0 blocks (0.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=71303168
2139 block groups
32768 blocks per group, 32768 fragments per group
96 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
4096000, 7962624, 11239424, 20480000, 23887872
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 32 mounts or 180
days, whichever comes first. Use tune2fs -c or -i to override.
```

以上命令会创建一个 ext2 的文件系统。如果想创建 ext3 文件系统则用如下命令：

```
# mkfs.ext3 /dev/sda1
# mke2fs -j /dev/sda1
```

技巧 67：挂载分区

当一个分区被创建和格式化后，你可以把它挂载到一个挂载点上。

首先创建一个可以挂载分区的目录

```
# mkdir /home/database
```

挂载文件系统

```
# mount /dev/sda1 /home/database
```

要让系统在重启后自动地挂载文件系统，把下面的一项加入到 /etc/fstab

```
/dev/sda1 /home/database ext3 defaults 0 2
```

技巧 68：用 tune2fs 进行分区调整

如下，使用命令 tune2fs -l /dev/sda1 查看文件系统信息：

```
# tune2fs -l /dev/sda1
tune2fs 1.35 (28-Feb-2004)
Filesystem volume name: /home/database
Last mounted on: <not available>
Filesystem UUID: f1234556-e123-1234-abcd-bbbbaaaaae11
Filesystem magic number: 0xEF44
Filesystem revision #: 1 (dynamic)
Filesystem features: resize_inode filetype sparse_super
Default mount options: (none)
Filesystem state: not clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 1094912
Block count: 140138994
Reserved block count: 0
Free blocks: 16848481
Free inodes: 1014969
First block: 0
Block size: 2048
Fragment size: 2048
Reserved GDT blocks: 512
Blocks per group: 16384
Fragments per group: 16384
Inodes per group: 128
Inode blocks per group: 8
Filesystem created: Tue Jul 1 00:06:03 2008
Last mount time: Thu Aug 21 05:58:25 2008
Last write time: Fri Jan 2 15:40:36 2009
Mount count: 2
Maximum mount count: 20
Last checked: Tue Jul 1 00:06:03 2008
Check interval: 15552000 (6 months)
Next check after: Sat Dec 27 23:06:03 2008
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
First inode: 11
```

```
Inode size: 128
Default directory hash: tea
Directory Hash Seed: 12345829-1236-4123-9aaa-ccccc123292b
```

你也可以用 tune2fs 来调整 ex2/ext3 文件系统的参数。例如，如果你想改变文件系统卷名，你可以进行如下操作：

```
# tune2fs -l /dev/sda1 | grep volume
Filesystem volume name: /home/database
# tune2fs -L database-home /dev/emcpowera1
tune2fs 1.35 (28-Feb-2004)
# tune2fs -l /dev/sda1 | grep volume Filesystem volume name:
database-home
```

技巧 69：创建交换分区

如下例所示，创建一个交换文件：

```
# dd if=/dev/zero of=/home/swap-fs bs=1M count=512
512+0 records in
512+0 records out
# ls -l /home/swap-fs
-rw-r--r-- 1 root root 536870912 Jan 2 23:13 /home/swap-fs
```

用 mkswap 在 home/swap-fs 文件上面创建/一个 Linux 交换空间。

```
# mkswap /home/swap-fs
```

一旦文件被创建并被设定为 Linux 交换空间，就可以进行如下操作让交换空间变得可用起来：

```
# swapon /home/swap-fs
```

把下面的一行加入到 /etc/fstab ，然后重启，这时候交换分区就可以使用了！

```
/home/swap-fs swap swap defaults 0 0
```

技巧 70：创建新用户

添加一个新用户——基本用法

仅指定要添加的用户名

```
# useradd jsmith
```

用额外的参数添加一个新用户

你还可以使用以下的参数来使用 useradd：

- -c ：有关用户的描述。
- -e ：用户的过期时间，以 mm/dd/yy 格式显示。

```
# adduser -c "John Smith - Oracle Developer" -e 12/31/09 jsmith
Verify that the user got added successfully.
# grep jsmith /etc/passwd
jsmith:x:510:510:John Smith - Oracle
Developer:/home/jsmith:/bin/bash
```

更改用户密码

```
# passwd jsmith
Changing password for user jsmith.
New UNIX password:
BAD PASSWORD: it is based on a dictionary word
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

提示：确认你遵循这些最优方法为用户创建一个安全性好的密码。

如何确定 useradd 使用的默认值 ？

以下是当用户被创建时会用到的默认值：

```
# useradd -D

GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

技巧 71：创建新的组并将用户加入该组

创建一个新的开发者用户组

```
# groupadd developers
```

确认用户组被成功创建

```
# grep developer /etc/group
developers:x:511:
```

把用户添加到一个已经存在的用户组上

你不能用 useradd 去修改一个已经存在的用户，否则你将会得到如下的错误信息：

```
# useradd -G developers jsmith
useradd: user jsmith exists

# usermod -g developers jsmith
```

确认用户组是否被成功修改

```
# grep jsmith /etc/passwd
jsmith:x:510:511:Oracle Developer:/home/jsmith:/bin/bash

# id jsmith
uid=510(jsmith) gid=511(developers) groups=511(developers)

# grep jsmith /etc/group
jsmith:x:510:
developers:x:511:jsmith
```

技巧 72：在 OpenSSH 中设置 SSH 的无密码登陆

使用下例中 ssky-keygen 和 ssh-copy-id，仅需通过 3 个步骤的简单设置，你无需输入密码就能登录远程 Linux 主机。

ssh-keygen 创建公钥和密钥。 ssh-copy-id 把本地主机的公钥复制到远程主机的 authorized_keys 文件上。ssh-copy-id 也会给远程主机的用户主目录（home）和~/.ssh, 和 ~/.ssh/authorized_keys 设置合适的权限 。

步骤 1: 用 ssh-key-gen 在本地主机上创建公钥和密钥

```
jsmith@local-host$ ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/home/jsmith/.ssh/id_rsa):[Enter
key]
Enter passphrase (empty for no passphrase): [Press enter key]
Enter same passphrase again: [Pess enter key]
```

```
Your identification has been saved in /home/jsmith/.ssh/id_rsa.
Your public key has been saved in /home/jsmith/.ssh/id_rsa.pub.
The key fingerprint is:
33:b3:fe:af:95:95:18:11:31:d5:de:96:2f:f2:35:f9 jsmith@local-host
```

步骤 2: 用 ssh-copy-id 把公钥复制到远程主机上

```
jsmith@local-host$ ssh-copy-id -i ~/.ssh/id_rsa.pub remote-host

jsmith@remote-host's password:
Now try logging into the machine, with "ssh 'remote-host'", and check
in:
.ssh/authorized_keys to make sure we haven't added extra keys that
you weren't expecting.
[注: ssh-copy-id 把密钥追加到远程主机的 .ssh/authorized_key 上.]
```

步骤 3: 直接登录远程主机

```
jsmith@local-host$ ssh remote-host

Last login: Sun Nov 16 17:22:33 2008 from 192.168.1.2

[注: SSH 不会询问密码.]

jsmith@remote-host$ [注: 你现在已经登录到了远程主机上]
```

技巧 73: 与 ssh-agent 一起来使用 ssh-copy-id

与 ssh-add/ssh-agent 一起来使用 ssh-copy-id

当没有值传递给选项 i 或者当~/.ssh/identity.pub 不存在时, ssh-copy-id 会显示如下的错误信息:

```
jsmith@local-host$ ssh-copy-id -i remote-host
/usr/bin/ssh-copy-id: ERROR: No identities found
```

如果你已经用 ssh-add 将密钥加载到 ssh-agent 上, 那么 ssh-copy-id 会从 ssh-agent 那里得到密钥, 进而复制到远程主机上。也就是说, 你不给 ssh-copy-id 传递 i 选项时, 它会复制由 ssh-add -L 命令提供的密钥到远程主机上。

```
jsmith@local-host$ ssh-agent $SHELL

jsmith@local-host$ ssh-add -L
The agent has no identities.
```

```
jsmith@local-host$ ssh-add
Identity added: /home/jsmith/.ssh/id_rsa (/home/jsmith/.ssh/id_rsa)

jsmith@local-host$ ssh-add -L
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAsJIEILxftj8aSxMa3d8t6JvM79D
aHrtPhTYpq7kIEMUNzApnyxsHpH1tQ/Ow== /home/jsmith/.ssh/id_rsa

jsmith@local-host$ ssh-copy-id -i remote-host

jsmith@remote-host's password:
Now try logging into the machine, with "ssh 'remote-host'", and check
in: .ssh/authorized_keys to make sure we haven't added extra keys that
you weren't expecting.
[注：已经把 ssh-add -L 显示的密钥成功添加]
```

以下是用到 ssh-copy-id 时三个烦人的小地方：

1. 默认公钥：ssh-copy-id 以 ~/.ssh/identity.pub 为默认的公钥文件（那就是，当不给选项 i 传值的时候）。但是当我想用 id_dsa.pub，或id_rsa.pub，或 identity.pub 代替它作为默认密钥。也就是说如果它们中任意一个存在，它就应该复制到远程主机上。如果它们当中有两个或者三个存在，系统则会默认复制 identity.pub

2. 认证代理没有身份：当 ssh-agent 运行时，ssh-add -L 返回 "The agent has no identities"（也就是说没有密钥要加到 ssh-agent 上），ssh-copy-id 仍将会复制信息 "The agent has no identities" 到远程主机上的 authorized_keys 项中。

3. authorized_keys的重复项：我希望 ssh-copy-id 可以确认远程主机上 authorized_keys中重复项的有效性。如果你在本地主机上多次运行 ssh-copy-id，它将会一直地把同一个密钥追加到远程主机的authorized_keys文件上而不检查是否有重复项。即使有重复项也可以正常工作。但是，我更喜欢不凌乱的 authorized_keys文件。

技巧 74：crontab

使用 crontab 你可以在指定的时间执行一个 shell 脚本或者一系列 Linux 命令。例如系统管理员安排一个备份任务使其每天都运行

如何往 cron 中添加一个作业?

```
# crontab -e
```

```
0 5 * * * /root/bin/backup.sh
```

这将会在每天早上 5 点运行 /root/bin/backup.sh

Cron 各项的描述

以下是 crontab 文件的格式：

```
{minute} {hour} {day-of-month} {month} {day-of-week}
{full-path-to-shell-script}
```

- minute: 区间为 0 – 59
- hour: 区间为 0 – 23
- day-of-month: 区间为 0 – 31
- month: 区间为 1 – 12. 1 是 1 月. 12 是 12 月.
- Day-of-week: 区间为 0 – 7. 周日可以是 0 或 7.

Crontab 示例

1. 在 12:01 a.m 运行，即每天凌晨过一分钟。这是一个恰当的进行备份的时间，因为此时系统负载不大。

```
1 0 * * * /root/bin/backup.sh
```

2. 每个工作日(Mon – Fri) 11:59 p.m 都进行备份作业。

```
59 11 * * 1,2,3,4,5 /root/bin/backup.sh
```

下面例子与上面的例子效果一样：

```
59 11 * * 1-5 /root/bin/backup.sh
```

3. 每 5 分钟运行一次命令

```
*/5 * * * * /root/bin/check-status.sh
```

4. 每个月的第一天 1:10 p.m 运行

```
10 13 1 * * /root/bin/full-backup.sh
```

5. 每个工作日 11 p.m 运行。

```
0 23 * * 1-5 /root/bin/incremental-backup.sh
```

Crontab 选项

以下是 crontab 的有效选项:

- crontab -e : 修改 crontab 文件. 如果文件不存在会自动创建。
- crontab -l : 显示 crontab 文件。
- crontab -r : 删除 crontab 文件。

- crontab -ir : 删除 crontab 文件前提醒用户。

技巧 75：用 Magic SysRq 键实现 Linux 安全重启

Magic SysRq 是 Linux 内核中一个组合键（key combination），它允许用户可以忽略系统的状态而执行一些低级指令。

它常被用来从系统冻结状态中恢复过来或者重启电脑而不会破坏文件系统。组合键由 Alt+SysRq 再加上一个相应的键 组成。在大多数系统中 SysRq 键就是 Print Screen 键.

首先，你需要像下面那样使 SysRq 键生效。

```
echo "1" > /proc/sys/kernel/sysrq
```

SysRq 对应的组合键列表

以下是对 Alt+SysRq+commandkey 有效的 Commandkeys.

- ‘k’ – 杀死所有在当前虚拟控制台（virtual console）运行的进程 。
- ‘s’ – 尝试同步所有挂载的文件系统。
- ‘b’ – 不卸载分区也不做同步的情况下立即重启系统
- ‘e’ – 向除了init 外的所有进程发送 SIGTERM 信号。
- ‘m’ – 将当前的内存信息输出到控制台。
- ‘i’ – 向除了init 外的所有进程发送 SIGKILL 信号。
- ‘r’ – 把键盘从raw模式（一种被像X11 这样的程序调用的模式）转换为 XLATE模式。
- ‘s’ – 使所有挂载的文件系统同步。
- ‘t’ – 将当前任务的清单以及它们的信息输出到控制台。
- ‘u’ – 以只读方式重新挂载所有已挂载的文件系统。
- ‘o’ – 立刻关闭系统。
- ‘p’ – 将当前的寄存器和标志信息输出到控制台。
- ‘0-9’- 设置控制台的日志级别，从而控制输出到控制台的内核信息。
- ‘f’ – 将会调用 oom_kill 来杀死占用较多内存的进程。
- ‘h’ – 用来显示帮助说明。但是除了上述列出的键外其他键也可以打印出帮助说明。

我们也可以将该键 写入 /proc/sysrq-trigger 文件中。例如，要重启系统你可以运行下面的命令：

```
echo "b" > /proc/sysrq-trigger
```

用 Magic SysRq 键实现 Linux 的安全重启

如果要实现挂起状态中的 Linux 的安全重启，请按如下操作。这将可避免在下一次重启时进行文件系统检查和修复（fsck）。也就是，按住 Alt+SysRq，再按住如下加粗的字母。

- un**R**aw （从X11 获取键盘的控制权），
- t**E**rminate （向所有进程发出SIGTERM信号，使他们可以正常结束），
- **K**ill （向所有进程发出SIGKILL信号，强制这些进程立刻结束），
- **S**ync （将数据同步到磁盘），

- **U**nmount （卸载所有只读的文件系统），
- re**B**oot （重启系统）

第十章：Apachectl 和 Httpd 实例

在安装 Apache2 后,如果你想最大限度发挥 apachectl 和 httpd 的能力，你应该不再满足于使用启动，停止以及重新启动。

本章提供的 9 个实例将帮助你更有效地使用 apachectl 和 httpd。

Apachectl 做为 SysV 初始化脚本，将 start、stop、restart、status 做为参数。同时也可以做为前端程序，将参数传给 httpd。因此，你可以使用 apachectl 调用所有命令，你也可以通过调用 httpd 直接执行。

如果你没有安装 Apache，请参阅教程：H 从源码安装 apacheH 或 H 使用 yum 安装 LAMPH。

技巧 76：传递不同的 httpd.conf 文件给 apachectl

一般情况下你可以通过修改原始 httpd.conf 来尝试不同的 Apache 配置。如果出现错误，你可以撤销改变。我们不是在 httpd.conf 进行修改，而是把它复制为 httpd.conf.debug，将 httpd.conf.debug 做为新的配置文件传递给 给 Apache 以供测试用途，如例子所示使用选项 -f。

```
# apachectl -f conf/httpd.conf.debug

# httpd -k start -f conf/httpd.conf.debug

[注：如上所示，用 apachectl 或 httpd 都是可以的]

# ps -ef | grep http
root   25080     1  0 23:26 00:00:00 /usr/sbin/httpd -f
conf/httpd.conf.debug
apache 25099 25080  0 23:28 00:00:00 /usr/sbin/httpd -f
conf/httpd.conf.debug

[注：ps 显示运行中的 httpd 正在使用 httpd.conf.debug]
```

当你满意这个改变并且 Apache 正常工作时，你可以将这个配置文件覆盖原来的 httpd.conf，接着正常启动 Apache，如下所示:

```
# cp httpd.conf.debug httpd.conf

# apachectl stop

# apachectl start

# ps -ef | grep httpd
root     25114     1  0 23:28 00:00:00 /usr/sbin/httpd
```

```
-k start
daemon   25115 25114  0 23:28 00:00:00 /usr/sbin/httpd
-k start

[注：ps 表明正在运行的 httpd 使用默认的配置文件]
```

技巧 77：使用一个临时 DocumentRoot 而不修改 httpd.conf

当你想尝试不同的网站布局而又不想修改原来 DocumentRoot 文件时，这个技巧是非常有用的。拷贝你原来的 DocumentRoot directory(/var/www/html)到一个新的临时 DocumentRoot 目录下(例如/var/www/html/_debug)。你的所有改动在这个临时 DocumentRoot 目录（/var/www/html_debug）下，同时在启动 Apache 时使用 c 选项将这个临时目录做为 DocumentRoot，如下所示：

```
# httpd -k start -c "DocumentRoot /var/www/html_debug/"
```

如果你想使用原来的默认配置的 DocumentRoot(/var/www/html)，只需要重新启动 Apache 即可：。

```
# httpd -k stop
# apachectl start
```

技巧 78：暂时提高 Log 的级别

当进行调试时，你可以在不直接修改 httpd.conf 中 LogLevel 参数的情况下暂时地提升 LogLevel，具体操作如下所示，使用选项 e。在这个例子里，LogLevel 设置为调试。

```
# httpd -k start -e debug

[Sun Aug 17 13:53:06 2008] [debug] mod_so.c(246):
loaded module auth_basic_module
[Sun Aug 17 13:53:06 2008] [debug] mod_so.c(246):
loaded module auth_digest_module
```

能够传给选项 e 的值是：

- debug
- info
- notice
- warn
- error
- crit

- alert
- emerg

技巧 79：显示 Apache 内的模块

显示编译进 Apache 内的模块

```
# httpd -l

Compiled in modules:
core.c
prefork.c
http_core.c
mod_so.c
```

显示 Apache 加载的静态和动态模块

当你将选项 l 传给 httpd，它只会展示静态模块。若使用 M 选项将显示静态和分享模块，如下所示：

```
# httpd -M

Loaded Modules:
core_module (static)
mpm_prefork_module (static)
http_module (static)
so_module (static)
auth_basic_module (shared)
auth_digest_module (shared)
authn_file_module (shared)
authn_alias_module (shared)
Syntax OK
```

技巧 80：显示 httpd.conf 内所有可接受的指令

这就像 httpd 的扩展帮助一样，它将显示 httpd.conf 内所有指令及其相应的有效位置。对于特定的指令，它将显示所有可能的操作以及在 httpd.conf 的什么位置可以使用。当你想尽快知道一个特定的 Apache 指令时这非常有用。

```
# httpd -L

HostnameLookups (core.c)
“on” to enable, “off” to disable reverse DNS lookups,
```

```
or "double" to enable double-reverse DNS lookups
Allowed in *.conf anywhere

ServerLimit (prefork.c)
Maximum value of MaxClients for this run of Apache
Allowed in *.conf only outside <Directory>, <Files> or
<Location>

KeepAlive (http_core.c)
Whether persistent connections should be On or Off
Allowed in *.conf only outside <Directory>, <Files> or
<Location>

LoadModule (mod_so.c)
a module name and the name of a shared object file to
load it from
Allowed in *.conf only outside <Directory>, <Files> or
<Location>
```

技巧 81：验证被修改的 httpd.conf

使用选项 t 可以验证指定的 Apache 配置文件是否存在问题。在下例中，在 httpd.conf.debug 的 148 行有问题,mod_auth_basicso 的 so 前遗漏了一个”.”。

```
# httpd -t -f conf/httpd.conf.debug

httpd: Syntax error on line 148 of
/etc/httpd/conf/httpd.conf.debug:
Cannot load /etc/httpd/modules/mod_auth_basicso into
server:
/etc/httpd/modules/mod_auth_basicso: cannot open shared
object file: No such file or directory
```

当你修复后，它将显示 Syntax OK。

```
# httpd -t -f conf/httpd.conf.debug

Syntax OK
```

技巧 82：显示 httpd 的编译参数

使用选项 V（大写）显示 Apache 的版本号和所有 Apache 编译时使用的参数。

```
# httpd -V

Server version: Apache/2.2.9 (Unix)
Server built:   Jul 14 2008 15:36:56
Server's Module Magic Number: 20051115:15
Server loaded:  APR 1.2.12, APR-Util 1.2.12
Compiled using: APR 1.2.12, APR-Util 1.2.12
Architecture:   32-bit
Server MPM:     Prefork
threaded:     no
forked:     yes (variable process count)
Server compiled with….
-D APACHE_MPM_DIR="server/mpm/prefork"
-D APR_HAS_SENDFILE
-D APR_HAS_MMAP
-D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
-D APR_USE_SYSVSEM_SERIALIZE
-D APR_USE_PTHREAD_SERIALIZE
-D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
-D APR_HAS_OTHER_CHILD
-D AP_HAVE_RELIABLE_PIPED_LOGS
-D DYNAMIC_MODULE_LIMIT=128
-D HTTPD_ROOT="/etc/httpd"
-D SUEXEC_BIN="/usr/sbin/suexec"
-D DEFAULT_PIDLOG="logs/httpd.pid"
-D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
-D DEFAULT_LOCKFILE="logs/accept.lock"
-D DEFAULT_ERRORLOG="logs/error_log"
-D AP_TYPES_CONFIG_FILE="conf/mime.types"
-D SERVER_CONFIG_FILE="conf/httpd.conf"
```

如果你只想显示 Apache 版本号，使用选项 v(小写)，如下所示。

```
# httpd -v

Server version: Apache/2.2.9 (Unix)
Server built:   Jul 14 2008 15:36:56
```

技巧 83：根据需要加载一个指定模块

有时你可能不需要装载所有的 Apache 模块。例如,当你正在测试 LDAP 时，你可能只需要将 LDAP 相关的模块加载到 Apache。我们可以通过以下方式达到这个目的。

修改 httpd.conf 并添加一个名为 load-ldap（你可以自由命名)的 IfDefine 指令。

```
<IfDefine load-ldap>
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module
modules/mod_authnz_ldap.so
</IfDefine>
```

当你正在测试 LDAP 和希望加载 LDAP 相关模块，可以将 load-ldap 传递给选项 D，如下所示。

```
# httpd -k start -e debug -Dload-ldap -f
/etc/httpd/conf/httpd.conf.debug

[Sun Aug 17 14:14:58 2008] [debug] mod_so.c(246):
loaded module ldap_module
[Sun Aug 17 14:14:58 2008] [debug] mod_so.c(246):
loaded module authnz_ldap_module
[Note: Pass -Dload-ldap, to load the ldap modules into
Apache]

# apachectl start
[注：如果你不想装载模块 ldap，正常开启 Apache]
```

第十一章：Bash 脚本

技巧 84：.bash_*files 的执行顺序

下列文件的执行顺序是什么？

- /etc/profile
- ~/.bash_profile
- ~/.bashrc
- ~/.bash_login
- ~/.profile
- ~/.bash_logout

交互式登录 shell 的执行顺序

下面的伪代码将说明这些文件的执行顺序

```
execute /etc/profile
IF ~/.bash_profile exists THEN
    execute ~/.bash_profile
ELSE
    IF ~/.bash_login exist THEN
        execute ~/.bash_login
    ELSE
        IF ~/.profile exist THEN
            execute ~/.profile
        END IF
    END IF
END IF
```

当你从交互式 shell 中注销，以下是执行顺序：

```
IF ~/.bash_logout exists THEN
    execute ~/.bash_logout
END IF
```

请注意 /etc/bashrc 是通过~/.bashrc 执行，如下所示：

```
# cat ~/.bashrc
if [ -f /etc/bashrc ]; then
. /etc/bashrc
Fi
```

非登录交互式 shell 的 执行顺序

当你启动一个非登录交互式 shell, 下面是执行顺序

```
IF ~/.bashrc exists THEN
   execute ~/.bashrc
END IF
```

[注意：当一个非交互式 shell 启动，它会寻找环境变量 ENV，并执行环境变量 ENV 里的文件名变量。]

技巧 85：如何在 bash shell 中产生随机数

使用 bash 内置的$RANDOM 可以产生 0-32767 之间随机数，如下所示。

```
$ echo $RANDOM
22543

$ echo $RANDOM
25387

$ echo $RANDOM
647
```

技巧 86：调试一个脚本

想要调试一个脚本应当将 set -xv 写在脚本的顶部

不包含调试命令的脚本：

```
$ cat filesize.sh
#!/bin/bash
for filesize in $(ls -l . | grep "^-" | awk ‘{print
$5}’ )
do
  let totalsize=$totalsize+$filesize
done
echo "Total file size in current directory: $totalsize"
```

不包含调试命令下的脚本输出：

```
$ ./filesize.sh
Total file size in current directory: 652
```

包含调试命令的脚本:

脚本里添加了 set -xv 输出调试，如下所示。

```
$ cat filesize.sh
#!/bin/bash
set -xv
for filesize in $(ls -l . | grep "^-" | awk ‘{print
$5}’ )
do
  let totalsize=$totalsize+$filesize
done
echo "Total file size in current directory: $totalsize"
```

包含调试命令的脚本输出：

```
$ ./fs.sh
++ ls -l .
++ grep ‘^-’
++ awk ‘{print $5}’
+ for filesize in ‘$(ls -l . | grep "^-" | awk
‘\’ ’ {print $5}’ \’ ’ )’
+ let totalsize=+178
+ for filesize in ‘$(ls -l . | grep "^-" | awk
‘\’ ’ {print $5}’ \’ ’ )’
+ let totalsize=178+285
+ for filesize in ‘$(ls -l . | grep "^-" | awk
‘\’ ’ {print $5}’ \’ ’ )’
+ let totalsize=463+189
+ echo ‘Total file size in current directory: 652’
Total file size in current directory: 652
```

使用调试选项执行脚本

除了在脚本内给出 set -xv, 同样可以悬着如下执行脚本：

```
$ bash -xv filesize.sh
```

技巧 87：使用引号（Quoting）

没有特殊字符的 echo 语句

```
$ echo The Geek Stuff
The Geek Stuff
```

有特殊字符“；”的语句。分号在 bash 中是一种命令结束符。在接下来的例子里，“The Geek”作为 echo 的参数而“Stuff”被认为是另外一个的 Linux 命令，系统将显示“command not found“。

```
$ echo The Geek; Stuff
The Geek
-bash: Stuff: command not found
```

为了避免上述情况你可以在分号前面加一个“\”，这样就去掉了分号的特殊含义而像下面那样打印出来。

```
$ echo The Geek\; Stuff
The Geek; Stuff
```

单引号

当你想按字面逐字输出内容的话就用单引号。例如$HOSTNAME 这样的特殊变量也会被打成 $HOSTNAME 而不是 Linux 主机的名字。

```
$ echo ‘Hostname=$HOSTNAME ;  Current User=`whoami` ;
Message=\$ is USD’

Hostname=$HOSTNAME ;  Current User=`whoami` ;
Message=\$ is USD
```

双引号

当你想显示特殊变量的真正含义时就用双引号。

```
$ echo "Hostname=$HOSTNAME ;  Current User=`whoami` ;
Message=\$ is USD"
Hostname=dev-db ;  Current User=ramesh ; Message=$ is
USD
```

除了以下字符，双引号会去除所有字符的特殊含义：

- $ 参数替换
- ` 后引号（Backquotes）
- \$ 字面上的美元符
- \′ 字面上的反引号
- \" 插入双引号
- \\ 插入反斜杠

技巧 88：将数据文件的指定域读取到 shell 脚本中

这个例子说明了怎样在 shell 脚本中从数据文件读取特定的域（field）并进行操作。例如，假设文件 employees.txt 的格式是{employee-name}:{employee-id}:{department-name}，以冒号进行划分，如下所示。

```
$ cat employees.txt
Emma Thomas:100:Marketing
Alex Jason:200:Sales
Madison Randy:300:Product Development
Sanjay Gupta:400:Support
Nisha Singh:500:Sales
```

下面的 shell 脚本说明了如何从这个 employee.txt 文件中读取特定的域（field）。

```
$ vi read-employees.sh
#!/bin/bash
IFS=:
echo "Employee Names:"
echo "---------------"
while read name empid dept
do
  echo "$name is part of $dept department"
done < ~/employees.txt
```

赋予脚本可执行权限后执行该脚本

```
$ chmod u+x read-employees.sh

$ ./read-employees.sh
Employee Names:
---------------
Emma Thomas is part of Marketing department
Alex Jason is part of Sales department
Madison Randy is part of Product Development department
Sanjay Gupta is part of Support department
Nisha Singh is part of Sales department
```

第十二章：监控系统和性能

技巧 89：free 命令

free 命令可以显示所有关于系统物理（RAM）和交换空间的必要信息

```
语法：free [选项]
```

我的系统一共有多少 RAM？

在下面的例子里，这个系统总的物理内存是 1GB。显示的单位为 KB。

```
# free
      total   used    free   shared  buffers  cached
Mem: 1034624   1006696 27928  0       174136   615892
-/+ buffers/cache:     216668      817956
Swap:    2031608       0    2031608
```

包括 RAM 和交换空间，我的系统一共有多少内存？

下面的命令中：

- 选项m 以MB为单位来显示变量
- 选项t 显示物理和交换空间变量总和，即“Total”行
- 选项o 不显示上面例子中的 buffers/cache行

```
             total       used       free     shared    buffers     cached
Mem:          1010        983         27          0        170        601
Swap:         1983          0       1983
Total:        2994        983       2011
```

技巧 90：top 命令

top 命令显示各种系统性能评测标准的实时数据信息，主要包括 CPU 负载，内存使用，进程列表等。

```
语法：top [选项]
```

怎样查看我当前系统的状态包括 CPU 使用情况？

在命令行中不带任何参数执行 top 的时候，将会有如下输出。在你按“Contorl+c”或 q 从命令退出前，top 命令的输出保持显示实时数据

```
# top
top - 13:10:13 up 171 days, 20:21,  3 users,  load average: 0.01, 0.05, 0.00

Tasks: 194 total,   1 running, 193 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.6% us,  0.7% sy,  0.0% ni, 98.7% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   1034624k total,  1007420k used,    27204k free,   174540k buffers
Swap:  2031608k total,        0k used,  2031608k free,   615904k cached

PID   USER     PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
11912 apache    15   0 31828  13m  3916 S    1  0.2   0:46.35 httpd
19299 oracle    19   0  279m  18m   17m S    1  0.2   0:00.03 oracle
11398 jsmith    16   0  107m  28m  6404 S    0  0.4   0:03.07 perl
```

怎样理解上面 top 命令的输出内容呢？

- 第一行“top”，表示系统已经启动并运行了 171 天。
- 第二行“Tasks”，显示总的进程数，同时，分别显示处于运行状态，休眠状态，停止状态和僵尸状态的进程数
- 第三行“Cpu（s）”显示当前系统CPU利用率。在这个例子中，CPU是 98.7%空闲
- 第四行“Mem”和第五行“swap”提供内存信息，和free命令得到的信息一致
- 剩下的行显示系统上所有活动的进程，按CPU使用率进行排序，即使用CPU最多的进程被默认放在顶部

top 命令有一些有用的命令行选项和交互选项。我们来看看 top 命令的几个重要选项。

如何确定内存密集型进程（memory intensive processes）？

当 top 命令显示输出的时候，按 F，可以显示以下信息和所有可供排序的字段，按 n（按内存使用对进程排序）然后按 Enter，这样就会在顶部按内存使用排序显示进程。

```
  Current Sort Field: K for window1:Def
```

通过对应字母选择排序字段，按任意键返回

怎样添加其他的域到 top 的输出中（例如 CPU 时间）？

当 top 命令运行的时候，按 f 可以显示以下信息和所可供排序的字段，按 l，将 CPU 时间添加到输出顶部的显示栏

Current Fields: AEHIOQTWKNMbcdfgjplrsuvyzX for window 1:Def

通过对应字母切换字段，按任意键返回

如何获取一个运行进程的绝对路径和参数？

当 top 命令正在运行的时候，按 c，就可以在命令栏显示正在运行的进程的绝对路径，也就是显示/usr/local/apache2/bin/httpd 而不是 httpd

```
PID   USER    PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
11912 apache    15     0 31828  13m  3916 S          1          0.2    0:46.35
/usr/local/apache2/bin/httpd
```

怎样在 top 命令里查看各个 CPU？

当 top 命令运行时，按 1（数字 1）,可以显示这个机器上的各个 CPU 的性能数据，如下所示

```
top - 13:10:13 up 171 days, 20:21,  3 users,  load average: 0.01, 0.05, 0.00
Tasks: 194 total,   1 running, 193 sleeping,   0 stopped,   0 zombie
Cpu0  : 10.2% us,  2.6% sy,  0.0% ni, 86.8% id,  0.3% wa,  0.0% hi,  0.0% si
Cpu1  :  9.6% us,  8.0% sy,  0.0% ni, 82.4% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu2  :  1.3% us,  1.3% sy,  0.0% ni, 95.0% id,  2.3% wa,  0.0% hi,  0.0% si
Cpu3  :  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   1034624k total,  1007420k used,    27204k free,   174540k buffers
Swap:  2031608k total,        0k used,  2031608k free,   615904k cached
```

技巧 91：ps 命令

ps 命令（进程状态）可以显示所有活动的进程的快照信息

```
语法: ps [选项]
```

如何显示系统上所有正在运行的进程？

如下所示，用"ps aux"：

```
# ps aux | more
    USER       PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
    root         1    0.0     0.0 2044 588 ?     Ss   Jun27
```

```
0:00 init [5]
    apache 31186       0.0    1.6 23736 17556 ? S     Jul26
0:40 /usr/local/apache2/bin/httpd
    apache 31187       0.0    1.3 20640 14444 ? S     Jul26
0:37 /usr/local/apache2/bin/httpd
```

你也可以使用"ps -ef | more"得到一个相似的输出

显示进程树

你可以使用 ps axuf 或 ps -ejH 以树的形式显示这些进程，树状结构有助于直接观察进程及其父进程。为了看起来更简洁明了，下面输出有一些列被删减了。

```
# ps axuf
root Oct14  0:00 /opt/VRTSralus/bin/beremote
root Oct14  0:00 \_ /opt/VRTSralus/bin/beremote
root Oct14  0:00      \_ /opt/VRTSralus/bin/beremote
root Oct14  0:00      \_ /opt/VRTSralus/bin/beremote
root Oct14  0:01      \_ /opt/VRTSralus/bin/beremote
root Oct 14 0:00      \_ /opt/VRTSralus/bin/beremote
root Dec03  0:01 /usr/local/sbin/sshd
root Dec22  1:08 /usr/local/sbin/sshd
root 23:35  0:00 \_ /usr/local/sbin/sshd
511  23:35  0:00      \_ -bash
511                      \_ ps axuf
```

[注：你也可以使用 pstree 命令来以树的结构显示进程]

查看指定用户的进程

以下命令显示 linux 用户 oracle 拥有的所有进程

```
$ ps U oracle
PID TTY      STAT TIME COMMAND
5014 ?        Ss   0:01 /oracle/bin/tnslsnr
7124 ?        Ss   0:00 ora_q002_med
8206 ?        Ss   0:00 ora_cjq0_med
8852 ?        Ss   0:01 ora_pmon_med
8854 ?        Ss   0:00 ora_psp0_med
8911 ?        Ss   0:02 oraclemed (LOCAL=NO)
```

查看当前用户的进程

以下命令显示当前用户的所有进程

```
$ ps U $USER
PID TTY      STAT TIME COMMAND
10329 ?        S    0:00 sshd: ramesh@pts/1,pts/2
10330 pts/1    Ss   0:00 -bash
10354 pts/2 Ss+ 0:00 -bash
10530 pts/1 R+  0:00 ps U ramesh
```

技巧 92：df 命令

df 命令（磁盘空余）显示已挂载的文件系统上总的磁盘空间和剩余磁盘空间

```
语法: df [选项] [文件系统名称]
```

我系统上有多少 GB 的空余磁盘空间?

如下使用 df －h。选项 h 以人类可读的形式显示数值(例如：K 代表 Kb,M 代表 Mb 和 G 代表 Gb)。例子输出结果如下，/文件系统有 17GB 的磁盘空间可用，而/home/user 文件系统有 70GB 可用。

```
# df -h
Filesystem  Size Used Avail Use% Mounted on
/dev/sda1   64G  44G   17G 73%    /
/dev/sdb1  137G  67G   70G 49%    /home/user
```

我系统的文件系统是什么类型的？

选项 T 将显示文件系统类型的信息。在这个例子中/和/hone/user 文件系统是 ext2 类型的。选项 a 将显示所有的文件系统，包括系统使用的 0 大小的特殊文件系统

```
# df -Tha
Filesystem Type Size Used Avail Use% Mounted on
/dev/sda1     ext2 64G 44G 17G 73% /
/dev/sdb1     ext2 137G 67G 70G 49% /home/user
none         proc     0   0      0 -   /proc
none        sysfs    0    0     0 -   /sys
none      devpts     0    0     0 -   /dev/pts
none       tmpfs 2.0G      0 2.0G 0% /dev/shm
```

技巧 93：kill 命令

kill 命令通常用来终止一个正在运行的进程。典型地，这个命令通常用来杀死挂起及未响应的进程

```
语法: kill [选项] [进程 ID 或者命令名称]
```

怎样杀死一个挂起的进程?

首先，用 ps 命令获取想要终止的进程 id。只要知道进程 id，作为参数传递给 kill 命令即可。下面的例子说明了如何终止一个挂起的 apache httpd 进程。请注意，一般来讲你应该用“apachectl stop”来停止 apache。

```
    # ps aux | grep httpd
    USER      PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
    apache 31186       0.0    1.6 23736 17556 ? S     Jul26
0:40 /usr/local/apache2/bin/httpd
    apache 31187       0.0    1.3 20640 14444 ? S     Jul26
0:37 /usr/local/apache2/bin/httpd
    # kill 31186 31187
```

请注意，上面的命令试图通过发送一个叫 SIGTERM 的信号来终止进程。如果进程没有终止，你可以通过使用选项 9 传递一个叫 SIGKILL 的信号强制终止进程，如下所示。只有你进程所有者或特权用户才能杀死进程。

```
# kill -9 31186 31187
```

将如下两个函数添加到.bashrc 者.bash_profile,可以使 kill 轻易的终止多个进程

```
function psgrep ()
{
 ps aux | grep "$1" | grep -v ‘grep’
}
function psterm ()
{
 [ ${#} -eq 0 ] && echo "usage: $FUNCNAME STRING" && return 0
 local pid
 pid=$(ps ax | grep "$1" | grep -v grep | awk ‘{ print $1 }’)
 echo -e "terminating ‘$1’ / process(es):\n$pid"
 kill -SIGTERM $pid
}
```

为标识和终止所有 httpd 进程，按如下方法操作：

```
# psgrep http
USER       PID %CPU %MEM    VSZ    RSS TTY  STAT START TIME
COMMAND
apache   31186       0.0        1.6 23736 17556 ?        S
Jul26       0:40  /usr/local/apache2/bin/httpd
apache   31187       0.0        1.3 20640 14444 ?        S
Jul26       0:37  /usr/local/apache2/bin/httpd
```

```
# psterm httpd
terminating ‘httpd’ / process(es):
31186
31187
```

技巧 94：du 命令

du 命令（磁盘用量）可以打印出一个指定目录及其子目录的空间使用情况。

我的主目录及其子目录占用了多少空间？

下面的例子中，选项 s 代表只有摘要，例如它只显示/home/jsmith 的总共大小而不是/home/jsmith 所有子目录的单独大小。选项 h 以人易读的方式显示信息，例如 K 代表 KB,M 代表 MB 和 G 代表 GB。～指明用户主目录。这个命令和 “du -sh /home/jsmith”一样。

```
# du -sh ~
320M    /home/jsmith
```

为列出/home/jsmith 的子目录的空间使用信息，不带 s 选项执行上面命令

技巧 95：lsof 命令

lsof 会列出系统中所有打开的文件。打开的文件包括网络连接，设备和目录。lsof 命令的输出有以下几列：

- COMMAND 进程名称.
- PID 进程ID
- USER 用户名
- FD 文件描述符
- TYPE 文件的节点类型
- DEVICE 设备号
- SIZE 文件大小
- NODE 节点号
- NAME 文件名的绝对路径

查看系统中所有打开的文件

不带任何参数执行 lsof 命令如下。

```
# lsof | more
    COMMAND PID          USER
```

```
FD      TYPE    DEVICE    SIZE    NODE
NAME
     init       1     root cwd     DIR     8,1     4096      2/
     init       1     root rtd    DIR     8,1     4096      2/
     init       1     root txt    REG      8,1 32684 983101 /sbin/init
     init       1     root mem      REG      8,1 106397 166798
/lib/ld-
2.3.4.so
     init       1     root mem      REG      8,1 1454802 166799
/lib/tls/libc-2.3.4.so
     init       1     root mem      REG      8,1 53736 163964
/lib/libsepol.so.1
     init       1     root mem      REG      8,1 56328 166811
/lib/libselinux.so.1
     init       1     root 10u FIFO       0,13            972
/dev/initctl
     migration 2         root cwd     DIR     8,1    4096      2/
     skipped...
```

lsof 命令的输出可能返回许多记录，这些信息除了让你对系统打开的文件有粗略印象外，没有多少意义。

```
# lsof | wc -l
3093
```

查看指定用户打开的文件

使用 lsof -u 选项显示指定用户所有打开的文件

```
# lsof -u ramesh
vi      7190 ramesh  txt    REG        8,1   474608
475196 /bin/vi
sshd    7163 ramesh    3u IPv6    15088263
TCP dev-db:ssh->abc-12-12-12-12.socal.res.rr.com:2631
(ESTABLISHED)
```

系统管理员可以用这个命令获取一些关于用户在系统上正在执行的任务信息。

列出打开指定文件的用户

如果你想查看使用某指定文件的所有用户，按照如下的方法使用 lsof。在这个例子中，显示的是现在正在使用 vi 的所有用户。

```
# lsof /bin/vi
```

```
COMMAND PID   USER    FD TYPE DEVICE   SIZE   NODE NAME
vi      7258  root   txt  REG    8,1 474608 475196 /bin/vi
vi      7300  ramesh txt  REG    8,1 474608 475196 /bin/vi
```

技巧 96：sar 命令

Sar 命令在 sysstat 软件包中 。请确定 sysstat 已经安装。如果你没有在系统上安装 Sar, 可以从 Sysstat project 得到。

Sar 是一个出色的监视工具，它可以显示几乎所有的系统资源（包括 CPU，内存，IO，页式调度，网络，中断等等）的性能数据。

Sar 收集，报告（显示）并保存性能数据，让我们分别看看这三个方面。

Sadc — 系统活动数据收集器

/usr/lib/sadc（系统活动数据收集器）以一个指定的时间间隔收集系统数据。它使用位于 /va/log/sa/sa[dd]（dd 指当前日期）文件来存储数据。

Sa1 shell 脚本

/usr/lib/sa1 顺序调用/usr/lib/sadcs。如下 sa1 被 crontab 调用。根据你的需要决定是 5 分钟，还是 15 分钟运行一次。我更趋向于在 crontab 里把它定为 5 分钟运行一次，如下所示。

```
*/5 * * * * root /usr/lib/sa/sa1 1 1
```

Sa2 shell 脚本

/usr/lib/sa2 是一个向/var/log/sa/sa[dd]（dd 指当前日期）里写日常报告的脚本。每天午夜 crontab 调用 sa2 一次。

```
# 59 23 * * * root /usr/lib/sa/sa2 -A
```

[注：/etc/cron.d/sysstat 文件和包括一些 sa1 和 sa2 默认值的 sysstat 包有关，你可以相应地做出修改。]

用 Sar 命令显示 CPU 统计信息

```
# sar -u

Linux 2.6.9-42.ELsmp (dev-db)      01/01/2009
12:00:01 AM CPU %user %nice    %system   %iowait %idle
12:05:01 AM all 3.70      0.00    0.85      0.00 95.45
12:10:01 AM all 4.59      0.00    1.19      0.06 94.16
12:15:01 AM all 3.90      0.00    0.95      0.04 95.11
12:20:01 AM all 4.06      0.00    1.00      0.01 94.93
12:25:01 AM all 3.89      0.00    0.87      0.00 95.23
12:30:01 AM all 3.89      0.00    0.87      0.00 95.23
```

```
Skipped..
Average: all     4.56     0.00    1.00      0.15 94.29
```

```
[注：如果你要把性能数据分成单个 CPU 的，执行下面的命令。]

# sar -u -P ALL
```

用 Sar 命令显示磁盘 IO 统计信息

```
# sar -d

Linux 2.6.9-42.ELsmp (dev-db)       01/01/2009
12:00:01 AM    DEV            tps    rd_sec/s wr_sec/s
12:05:01 AM    dev2-0          1.65      1.28    45.43
12:10:01 AM    dev8-1         4.08      8.11    21.81
Skipped..
Average:       dev2-0          4.66    120.77    69.45
Average:       dev8-1         1.89      3.17     8.02
```

用 Sar 命令显示网络统计信息

```
# sar -n DEV | more

Linux 2.6.9-42.ELsmp (dev-db)      01/01/2009
12:00:01 AM     IFACE   rxpck/s txpck/s   rxbyt/s  txbyt/s
rxcmp/s   txcmp/
s rxmcst/s
12:05:01 AM        lo      0.17    0.16     25.31    23.33
0.00      0.0
0      0.00
12:10:01 AM      eth0     52.92   53.64  10169.74 12178.57
0.00      0.0
0      0.00
# sar -n SOCK |more

Linux 2.6.9-42.ELsmp (dev-db)      01/01/2009
12:00:01 AM    totsck    tcpsck  udpsck    rawsck  ip-frag
12:05:01 AM        50        13       3         0        0
12:10:01 AM        50        13       4         0        0
12:15:01 AM        53        13       5         0        0
```

技巧 97：vmstat 命令

出于典型的的性能监控的需要，只要 vmstat 命令即可满足需求。它可以显示内存，交换空间，IO，系统和 CPU 性能信息。

下面的命令执行 vmstat 每秒 100 次.

```
# vmstat 1 100

procs -----------memory---------- ---swap-- -----io---- --system--
----cpu----
 r b swpd free buff cache si so            bi    bo in      cs us sy
id wa
 0 0     0 282120 134108 5797012       0   0     0     2     0     0 0 0 100
0
 0 0     0 282120 134108 5797012       0   0     0     0 1007 359 0 0 100
0
 0 0 0 282120 134108 5797012 0 0 0 0 1117 577 0 0 100 0
 0 0 0 282120 134108 5797012 0 0 0 0 1007 366 0 0 100 0
```

Vmstat 的 procs 段

- r 字段：可运行进程总数
- b 字段：阻塞的进程总数

Memory段

- Swpd字段：使用的交换空间
- Free字段：可用的空闲RAM
- Buff字段：用于缓冲区的RAM
- Cache字段：用于文件系统cache的RAM

Swap段

- Si字段：每秒从磁盘交换的内存数量
- So字段：每秒交换到磁盘的内存数量

IO段

- Bi字段：从磁盘读取的块
- Bo字段：写入到磁盘的块

System段

- In字段：每秒的中断数
- Cs字段：每秒的上下文切换数

CPU 段

- Us字段：运行用户代码花费的时间（非内核代码）
- Sy字段：运行内核代码花费的时间
- Id字段：空闲时间
- Wa字段：等待IO花费的时间

技巧 98：netstat 命令

netstat 命令显示网络相关的信息，例如网络连接，路由表，网络接口信息。以下是几个使用 netstat 命令的例子。

用 netstat 显示活动的网络连接和域套接字

```
# netstat -an

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign
Address             State
tcp        0      0 0.0.0.0:5666                0.0.0.0:*
LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*
LISTEN
tcp        0      0 0.0.0.0:4086                0.0.0.0:*
LISTEN
skipped..
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix 2       [ ACC ]     STREAM     LISTENING     7894
/tmp/.font-unix/fs7100
unix 2       [ ACC ]     STREAM     LISTENING     9662
/tmp/.gdm_socket
unix 2       [ ACC ]     STREAM     LISTENING     10897
@/tmp/fam-root-
```

显示带进程 ID 和程序名的活动连接

这对标识哪个程序初始化一个指定的网络连接很有用。

```
# netstat -tap

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign
Address             State       PID/Program name
tcp        0      0 *:nrpe                      *:*
LISTEN      16277/xinetd
tcp        0      0 localhost.localdomain:smtp *:*
LISTEN      7263/sendmail: acce
```

```
tcp       34      0 localhost.localdomain:54221
localhost.localdomain:4089 CLOSE_WAIT 29881/httpd
tcp        0   3216 dev-db:ssh                cpe-76-
94-215-154.soca:4682 ESTABLISHED 11717/sshd: ramesh
```

显示路由表

```
# netstat -tap

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign
Address             State       PID/Program name
tcp        0      0 *:nrpe                      *:*
LISTEN      16277/xinetd
tcp        0      0 localhost.localdomain:smtp *:*
LISTEN      7263/sendmail: acce
tcp       34      0 localhost.localdomain:54221
localhost.localdomain:4089 CLOSE_WAIT 29881/httpd
tcp        0   3216 dev-db:ssh                cpe-76-
94-215-154.soca:4682 ESTABLISHED 11717/sshd: ramesh
```

显示 RAW 网络统计信息

```
# netstat --statistics --raw
Ip:
    11080343 total packets received
    0 forwarded
    1 with unknown protocol
    0 incoming packets discarded
    11037744 incoming packets delivered
    11199763 requests sent out
Icmp:
    577135 ICMP messages received
    64 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 537
        timeout in transit: 65
        source quenches: 2
        echo requests: 576476
        echo replies: 12
        timestamp request: 3
        address mask request: 3
    581558 ICMP messages sent
```

```
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 5079
        echo replies: 576476
        timestamp replies: 3
```

其他的 Netstat 命令

- # netstat --tcp --numeric 列出进出该机器的TCP连接 .
- # netstat --tcp --listening --programs 显示服务器正在监听的TCP端口和正在监听指定端口的程序
- #netstat -rnC 显示路由缓存

技巧 99：sysctl 命令

Liunx 内核参数可以通过 sysctl 命令在运行时改变。

sysctl 有助于运行时配置 Linux 内核参数

```
# sysctl -a

dev.cdrom.autoclose = 1
fs.quota.writes = 0
kernel.ctrl-alt-del = 0
kernel.domainname = (none)
kernel.exec-shield = 1
net.core.somaxconn = 128
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_wmem = 4096        16384
net.ipv6.route.mtu_expires = 600
sunrpc.udp_slot_table_entries = 16
vm.block_dump = 0
```

在/etc/sysctl.conf 永久修改内核参数

在/etc/sysctl.conf 修该完内核参数以后，执行 sysctl -p提交改动。重启之后改变仍然保持。

```
# vi /etc/sysctl.conf
# sysctl -p
```

临时修改内核参数

为临时修改内核参数，执行下面的命令。请注意，重启后这些改变不会存在。

```
# sysctl -w {variable-name=value}
```

技巧 100：nice 命令

内核根据进程的 nice 值决定进程需要多少处理器时间。nice 值的取值范围是是：-20 到 20。一个具有-20 的 nice 值的进程有很高的优先级。一个 nice 值为 20 的进程的优先级则很低。

用 ps axl 显示所有正在运行的进程的 nice 值，如下所示。

```
# ps axl
F   UID   PID PPID PRI NI     VSZ RSS WCHAN STAT TTY
TIME COMMAND
4     0     1     0 16    0 2172 552 -        S    ?
0:17 init [5]
1     0     3     1 34 19       0    0 ksofti SN   ?
3:18 [ksoftirqd/0]
1     0    10     1   5 -10     0    0 worker S<   ?
0:01 [events/0]
4     0 5145      1 25 10 32124 18592 -       SNs ?
0:08 /usr/bin/python /usr/bin/rhn-applet-gui --sm-client-id
default4
4     0 5147 5142 16      0 3528 604 -        S    ?
0:00 /sbin/pam_timestamp_check -d root
1   503 17552 4180 16     0 14208 3920 -      S    ?
0:01 /home/www/apache2/bin/httpd -f
/home/www/apache2/conf/httpd.conf -k start
```

如何给一个 shell 脚本分配一个低的优先级（更高的 nice 值）？

在下面的例子里，当我在后台启动 nice-test.sh 脚本，nice 值为 0。

```
$ ./nice-test.sh &
[3] 13009
$ ps axl | grep nice-test
0   509 13009 12863 17    0 4652 972 wait S
pts/1 0:00 /bin/bash ./nice-test.sh
```

［注：第六列数值为 0 的是 nice 值］

现在，让我们以不同的 nice 值来执行相同的脚本，如下所示

```
$ nice -10 ./nice-test.sh &
[1] 13016
```

```
$ ps axl | grep nice-test
0   509 13016 12863 30 10 4236 968 wait  SN
pts/1      0:00 /bin/bash ./nice-test.sh
```
［注：第六列数值为 10 的是该 shell 脚本的 nice 值］

如何给一个 shell 脚本分配一个高的优先级（更低的 nice 值）？

下面的例子里，我们分配给 shell 脚本 nice-test.sh 一个－10 的 nice 值。

```
$ nice --10 ./nice-test.sh &
[1] 13021
$ nice: cannot set priority: Permission denied
```

注意：只有 root 用户可以设置一个负的 nice 值。用 root 用户登录再次尝试。注意在下面的 nice 命令里的 10 前面有一个双破折号。

```
# nice --10 ./nice-test.sh &
[1] 13060
# ps axl | grep nice-test
4     0 13060 13024 10 -10 5388 964 wait S<
pts/1      0:00 /bin/bash ./nice-test.sh
```
［注：第六列数值为-10 的是该 shell 脚本的 nice 值］

技巧 101：renice 命令

renice 可以给正在运行的进程设置调度优先级。

怎样降低一个运行进程的优先级（增加 nice 值）？

下面的例子里，一个存在的 shell 脚本运行时的 nice 值为 10。（ps 输出的第 6 列）

```
$ ps axl | grep nice-test
0   509 13245 13216 30 10 5244 968 wait  SN
pts/1      0:00 /bin/bash ./nice-test.sh
```

为增加 nice 值（因此降低优先级），按如下方法执行 renice 命令。

```
$ renice 16 -p 13245
13245: old priority 10, new priority 16
$ ps axl | grep nice-test
```

```
0   509 13245 13216 36 16 5244 968 wait  SN
pts/1      0:00 /bin/bash ./nice-test.sh
[注：：现在，nice-test.sh (PID 13245)的第 6 列显示新的 nice 值是 16]
```

怎样增加运行进程的优先级（减少 nice 值）？

下面的例子里，一个已有的 shell 脚本运行时 nice 值为 10。（ps 输出第 6 列）

```
$ ps axl | grep nice-test
0   509 13254 13216 30 10 4412 968 wait  SN
pts/1      0:00 /bin/bash ./nice-test.sh
```

为提升其优先级，赋予其一个较低的nice值。然而，只有root可以提升运行进程的优先级，否则你会得到以下错误信息。

```
$ renice 5 -p 13254
renice: 13254: setpriority: Permission denied
Login as root to increase the priority of a running
process
$ su -
# renice 5 -p 13254
13254: old priority 10, new priority 5
# ps axl | grep nice-test
0   509 13254 13216 25    5 4412 968 wait     SN
pts/1      0:00 /bin/bash ./nice-test.sh
[注：第 6 列现在显示了一个较低的 nice 值 5（提升的优先级）]
```

第十三章 一些额外的技巧

额外技巧 1：让 cd 命令对参数大小写不敏感

只显示小写

下面的这个例子中 cd 命令会列出所有以小写 m 开头的的目录

```
# cd m<tab><tab>

may   myticket
```

[注：你需要在 m 之后按两下 tab 键]

只显示大写

下面的例子中 cd 命令会列出所有以大写 M 开头的目录

```
# cd M<tab><tab>

March   Music
```

大写小写同时显示

如果你需要同时显示大写和小写的目录名（即便是你给的参数只是小写的），执行下面的 bind 命令

```
# bind "set completion-ignore-case on"
```

现在，如果你在 cd 命令的参数中输入小写的 m 并按下两次 tab，就会自动将大写小写的 m 开头的目录名同时显示出来。

```
# cd m<tab><tab>
```

```
March    may    Music    myticket
```

额外技巧 2：一次动作为多次 SSH 连接指定密码

用 SSH 的 ControlMaster 特性，你可以在多次建立 ssh 连接时只在第一次连接时输入密码。这就可以让你通过一次网络连接共享多个 SSH 会话。

将下面几行加入~/.ssh/config 文件。

```
$ vi ~/.ssh/config

Host *
ControlMaster auto
ControlPath ~/.ssh/master-%r@%h:%p
```

Host * ——所有的主机

ControlPath ~/.ssh/master-%r@%h:%p ——建立控制文件的路径，确定这个文件不能被别人使用。

- %r - remote login name
- %h - host name (remote)
- %p - port

第一次当你执行 SSH 连接一台远程主机时，你需要提供密码来建立主连接。

之后在要与同一台机器建立 ssh，scp，或者 sftp 这些会话，你无需再提供密码。当主连接还存在时，这种方法始终可行。在随后的 ssh 连接中，将会使用第一次连接时创建的已经存在的套接字。

要使这个功能对于特定的一些主机有效，将如下几行加入到~/.ssh/config 文件中。

```
Host 192.168.1.?
Host *.com
```

更改控制文件路径：

```
ControlPath ~/.mysecretfiles/master-%r@%h:%p
```

你还可以用 ControlMaster 特性来控制以下行为：

- ControlMaster auto - 这样设置将会自动对所后续连接使用主连接的套接字

o ControlMaster autoask - 这样设置会在下一次连接时进行确认。如果回答“yes”，下次连接将会使用主连接的套接字。

额外技巧 3：rar 命令用法示例

如何用 rar 压缩一个单独文件呢？

```
用法: rar a {.rar file-name} {file-name}

# rar a syslog.rar /var/log/syslog

  RAR 3.90 beta 2   Copyright (c) 1993-2009
    Alexander Roshal   3 Jun 2009
  Shareware version        Type RAR -? for help
  Evaluation copy. Please register.

    Creating archive syslog.rar
  Adding    /var/log/syslog   OK
  Done
```

如何压缩多个文件呢？

```
用法: rar a {.rar file-name} {file-names}

# rar a var-log-cups.rar /var/log/cups/*

  RAR 3.90 beta 2   Copyright (c) 1993-2009
    Alexander Roshal   3 Jun 2009
  Shareware version        Type RAR -? for help
  Evaluation copy. Please register.

  Creating archive var-log-cups.rar
  Adding    /var/log/cups/access_log OK
  Adding    /var/log/cups/access_log.1.gz OK
  Adding    /var/log/cups/access_log.2.gz OK
  Adding    /var/log/cups/error_log OK
  Adding    /var/log/cups/error_log.1.gz OK
  Adding    /var/log/cups/error_log.2.gz OK
  Done
```

你可以通过下面的办法压缩多个文件

```
# rar a -r var-log-cups.rar /var/log/cups/
```

如何用 unrar 解压扩展名为 rar 的压缩文件呢？

```
用法: unrar e {.rar file-name}

# unrar e var-log-cups.rar

  UNRAR 3.90 beta 2 freeware Copyright (c) 1993-20
    Alexander Roshal

  Extracting from var-log-cups.rar
  Extracting  error_log OK
  Extracting  access_log.1.gz OK
  Extracting  error_log.2.gz OK
  Extracting  error_log.1.gz OK
  Extracting  access_log OK
  Extracting  access_log.2.gz OK
  All OK
```

如何不解压而列出一个 rar 压缩文件的内容呢？

```
用法: unrar l {.rar file-name}

# unrar l var-log-cups.rar

  UNRAR 3.90 beta 2 freeware Copyright (c) 1993-2009
    Alexander Roshal

  Archive var-log-cups.rar

   Name            Size   Packed Ratio  Date   Time
Attr     CRC   Meth Ver
  -------------------------------------------------
------------------------------
   access_log.1.gz     106     133 125% 14
20:37 -rw-r----- A329619E m3b 2.9
   access_log.2.gz     190     243 127% 30
09:44 -rw-r----- CAE8E2DB m3b 2.9

.....
```

额外技巧 4：用 Comm 命令比较两个文件

Comm 命令用于逐行比较两个已排序文件

```
用法：comm [OPTION]... FILE1 FILE2
```

这条命令将分别显示 FILE1 所独有的行、FILE2 中独有的行，再显示 FILE1 和 FILE2 共有的行，具体效果如下命令所示。

在下面的例子中，

- 第一列显示的是第一个文件中独有的行（示例文件 name_list.txt）
- 第二列显示的是第二个文件中独有的行（示例文件 name_list _new.txt）
- 第三列显示的是两个文件中共有的行

```
$ cat name_list.txt

Bram Moolenaar
Ken Thompson
Linus Torvalds

$ cat name_list_new.txt

Bram Moolenaar
Dennis Ritchie
Richard Stallman

$ comm name_list.txt name_list_new.txt
      Bram Moolenaar
    Dennis Ritchie
  Ken Thompson
  Linus Torvalds
    Richard Stallman
```

额外技巧 5：Compact-Disk (CD)操作

如何烧录一张 CD 呢？

```
用法：cdrecord -V -eject dev={device-file-path} {iso-file}
```

```
$ cdrecord -V -eject dev=/dev/cdrom data-backup.iso
```

如何降低 CD 的写入速度？

```
用法: --speed=1

# cdrecord -V -eject dev=$CDPath $ISOFileName --speed=1
```

如何清空一个可擦写的 CD？

```
用法: cdrecord blank=fast dev={device-file-path}

# cdrecord blank=fast dev=/dev/cdrw

  Device type    : Removable CD-ROM
  Version        : 5
Response Format: 2
Capabilities   :
Vendor_info    : 'Optiarc '
Identification : 'DVD RW AD-7240S '
Revision       : '1.02'
Device seems to be: Generic mmc2 DVD-R/DVD-RW.
Using generic SCSI-3/mmc   CD-R/CD-RW driver
Driver flags   : MMC-3 SWABAUDIO BURNFREE
Supported modes: TAO PACKET SAO SAO/R96R RAW/R1
Speed set to 1764 KB/s
Starting to write CD/DVD at speed  10.0 in real
Last chance to quit, starting real write 0 sec.
 Operation starts.
```

额外技巧 6：DVD 操作

如何烧录一张 DVD？

```
用法: growisofs -dvd-compat -Z {device-path}={iso-file}

# growisofs -dvd-compat -Z /dev/dvdrw=data.iso
```

```
  Executing 'builtin_dd if=data.iso of=/dev/dvdrw
obs=32k seek=0'
  /dev/dvdrw: "Current Write Speed" is 2.0x1352KBps.
  builtin_dd: 192*2KB out @ average infx1352KBps
  /dev/dvdrw: flushing cache
  /dev/dvdrw: writing lead-out
```

如何清空一张可擦写的 DVD？

```
用法: dvd+rw-format -force {device-file-path}

# dvd+rw-format -force /dev/dvdrw
  * BD/DVD±RW/-RAM format utility by
<appro@fy.chalmers.se>, version 7.0.
  * 4.7GB DVD-RW media in Restricted Overwrite mode
detected.
  * formatting 86.2%
```

如何写一个多区段的 DVD？

首先，创建一个 ISO 镜像之后就可以按照上面提到的方法正常烧录了。

没有必要去创建第二个镜像，你可以将增添的文件加入到已有的 iso 文件中。

```
用法:
  -M to merge a new session
  Final argument indicates the files to be appended.

# growisofs -dvd-compat -M /dev/dvdrw -R -J t1/

  Executing 'mkisofs -C 16,560 -M /dev/fd/3 -R -J
t1/ | builtin_dd of=/dev/dvdrw obs=32k seek=35'
  I: -input-charset not specified, using utf-8
(detected in locale settings)
  Rock Ridge signatures found
  Total translation table size: 0
  Total rockridge attributes bytes: 404
  Total directory bytes: 0
  Path table size(bytes): 10
  Max brk space used 0
  742 extents written (1 MB)
  /dev/dvdrw: "Current Write Speed" is 2.0x1352KBps.
  builtin_dd: 192*2KB out @ average infx1352KBps
```

```
  /dev/dvdrw: flushing cache
  /dev/dvdrw: copying volume descriptor(s)
  /dev/dvdrw: flushing cache
  /dev/dvdrw: reloading tray
```

额外技巧 7：从 CD 或者 DVD 创建 ISO 文件

首先，获取 CD 或者 DVD 的容量大小，块大小这些信息，可以使用 isoinfo 命令获取：

```
用法: isoinfo -d -i DEVICE-Of-CD-DVD

# isoinfo -d -i /dev/cdrom

  CD-ROM is in ISO 9660 format
  System id: WIN32
  Volume id: RQ0010
  Volume set id:
  Publisher id:
  Data preparer id:
  Application id: ULTRAISO V9.3 CD & DVD CREATOR,
(C)2008 EZB SYSTEMS, INC.
  Copyright File id:
  Abstract File id:
  Bibliographic File id:
  Volume set size is: 1
  Volume set sequence number is: 1
  Logical block size is: 2048
  Volume size is: 1825
  Joliet with UCS level 3 found
  NO Rock Ridge present
```

接着，按如下方法使用 dd 命令将 CD 或者 DVD 的镜像复制为 ISO 文件：

```
# dd if=/dev/cdrom bs=2048 count=1825 of=mydata.iso

  1825+0 records in
  1825+0 records out
  3737600 bytes (3.7 MB) copied, 4.43782 seconds,
842 kB/s
```

额外技巧 8：OD 命令用法示例

od 命令可以将文件按照八进制，十进制或者其他的格式显示出来。

将 OD 命令用在一个文本文件上

下面的样本文本文件会被用在下面的示例中

```
$ cat sample-file.txt

abc   de
f     h
```

选项 -b 和 -c 的典型用法如下：

- -b 与选项 -t oC 功能相同，显示为八进制模式
- -c 与选项 -t c 功能相同，显示为 ASCII 字符或者转义符

```
# od -c special-chars.txt

0000000   a   b   c     \t   d   e  \n   f
\t   h
0000020  \n
0000021

# od -bc special-chars.txt

0000000 141 142 143 040 011 144 145 012 146 040 040 040
040 040 011 150
          a   b   c     \t   d   e  \n   f
\t   h
0000020 012
         \n
0000021
```

将 OD 命令用在一个二进制文件上

读取前 16 字节，然后显示成等价 ASC II 字符或者转义字符：

```
# od -N16 -c /usr/bin/h2xs

0000000   #   !   /   u   s   r   /   b   i   n   /   p
e   r   l  \n
0000020
```

读取前 16 字节并且显示等价的字符：

```
# od -N16 -a /usr/bin/h2xs

0000000   #   !   /   u   s   r   /   b   i   n   /   p
e   r   l  nl
0000020
```

读取前 16 字节并且以八进制格式显示：

```
# od -N16 -bc /usr/bin/h2xs

0000000 043 041 057 165 163 162 057 142 151 156 057 160
145 162 154 012
        #   !   /   u   s   r   /   b   i   n   /   p
e   r   l  \n
0000020
```

额外技巧 9：Gpg 命令用法示例

这个例子中我们会讲解如何使用 gpg 命令制作私人或者公共密钥来对文件进行加密和解密。

第一步：创建一个新的 GPG 密钥对

下面示例中显示的粗体文字是用户的输入。

```
# gpg --gen-key

  gpg --gen-key
  gpg (GnuPG) 1.4.9; Copyright (C) 2008 Free
Software Foundation, Inc.
```

```
  This is free software: you are free to change and
redistribute it.
  There is NO WARRANTY, to the extent permitted by
law.

  Please select what kind of key you want:
     (1) DSA and Elgamal (default)
     (2) DSA (sign only)
     (5) RSA (sign only)

  Your selection?
  DSA keypair will have 1024 bits.
  ELG-E keys may be between 1024 and 4096 bits long.

  What keysize do you want? (2048)
  Requested keysize is 2048 bits
  Please specify how long the key should be valid.
          0 = key does not expire
       <n>  = key expires in n days
       <n>w = key expires in n weeks
       <n>m = key expires in n months
       <n>y = key expires in n years
  Key is valid for? (0)
  Key does not expire at all
  Is this correct? (y/N) y

  You need a user ID to identify your key; the
software constructs the user ID
  from the Real Name, Comment and Email Address in
this form:
      "Heinrich Heine (Der Dichter)
<heinrichh@duesseldorf.de>"

Real name: Ramesh Natarajan
Email address: ramesh.thegeekstuff@gmail.com
Comment: testing demo key

You selected this USER-ID:
    "Ramesh Natarajan (testing demo key)
<ramesh.thegeekstuff@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

```
  You need a Passphrase to protect your secret key.

Enter passphrase:
Repeat passphrase:

  We need to generate a lot of random bytes. It is a
good idea to perform some other action (type on the
keyboard, move the mouse, utilize the disks) during the
prime generation; this gives the random number
generator a better chance to gain enough entropy.

  .++++++++++++++++++++++++.++++++++++++++++++++++++

  gpg: key 90130E51 marked as ultimately trusted
  public and secret key created and signed.


  gpg: checking the trustdb
  gpg: 3 marginal(s) needed, 1 complete(s) needed,
PGP trust model
  gpg: depth: 0  valid:   1  signed:   0  trust: 0-,
0q, 0n, 0m, 0f, 1u
  pub   1024D/90130E51 2010-01-02
        Key fingerprint = B8BD 46EF 41E7 44B9 F934
7C47 3215 5713 9013 0E51
  uid  Ramesh Natarajan (testing demo key)
<ramesh.thegeekstuff@gmail.com>
  sub   2048g/35C5BCDB 2010-01-02
```

第二步：导出你的公钥

```
用法: gpg --export {user-name}

# gpg --export ramesh > ramesh-pub.gpg

# file ramesh-pub.gpg
ramesh-pub.gpg: GPG key public ring

# gpg --armor --export ramesh > ramesh-pub-asc.gpg
```

第三步：导入其他的公钥

用 –import 选项导入其他的公共密钥

```
用法： gpg --import FileName
```

第四步：发送加密的信息

在这个例子中，让我们看看 John 如何给 Bob 发送一个加密的信息。

John 用 Bob 的公钥将输入文件加密。以下示例即创建了这个二进制文件。

```
$ gpg --recipient bob --encrypt filename
```

由于某些原因，如果 John 没能将加密的二进制文件发给 Bob，他可以创建一个加密的 ASCII 文件，如下所示：

```
$ gpg --recipient bob --armor --encrypt filename
```

第五步：读取加密的消息

在这个例子中，让我们看看 Bob 如何读取来自 John 的加密文件。

用你的私钥解密信息

```
用法： gpg --decrypt file

$ gpg --decrypt test-file.asc

  You need a passphrase to unlock the secret key for
  user: "ramesh (testing demo key)
<ramesh.thegeekstuff@gmail.com>"
  2048-bit ELG-E key, ID 35C5BCDB, created 2010-01-
02 (main key ID 90130E51)

  Enter passphrase:
```

[注：输入密码之后，解密的文件就会被打印到标准输出。]

用下面的命令可以将解密的信息重定向到一个文本文件中。

```
# gpg --decrypt test-file.asc > file.txt
```

额外的 GPG 命令：

你可以按下面的方法列出所有的 GPG 密钥。

```
# gpg --list-keys

/home/ramesh/.gnupg/pubring.gpg
---------------------------------
pub   1024D/90130E51 2010-01-02
uid   ramesh (testing demo key)
<ramesh.thegeekstuff@gmail.com>
sub   2048g/35C5BCDB 2010-01-02

# gpg --list-secret-keys

/home/ramesh/.gnupg/secring.gpg
---------------------------------
sec   1024D/90130E51 2010-01-02
uid   ramesh (testing demo key)
<ramesh.thegeekstuff@gmail.com>
ssb   2048g/35C5BCDB 2010-01-02
```

额外的技巧 10：Tee 命令示例

Tee 命令用来存储和查看（同时进行）别的命令的输出

在下面的例子中，Tee 命令同时将命令结果写入到标准输出和文件中。

示例一：将输出写入标准输出中和文件中

下面的的命令仅仅在屏幕中显示（标准输出）。

```
$ ls
```

下面的命令将输出仅仅写入到文件中而没有在屏幕上显示。

```
$ ls > file
```

下面的命令（在 tee 命令的帮助下）将结果显示在到屏幕（标准输出）的同时将其也写入到文件 file 中

```
$ ls | tee file
```

示例二：将输出写给两个命令

你也可以用 tee 命令将一个命令的输出到一个文件中并且重定向同样的输出作为另一个命令的输入。

下面的命令将会给 crontab 的输出形成一个备份，并且将 crontab 的输出作为 sed 命令的输入。替换之后，它就会成为一个新的 cron 作业。

```
$ crontab -l | tee crontab-backup.txt | sed
's/old/new/' | crontab -
```

tee 命令的其他用法

默认的 tee 命令会覆盖已有文件，你可以用 -a 参数指挥 tee 命令在文件末尾追加内容。

```
$ ls | tee -a file
```

按照下面的方法你还可以将输出写入到多个文件中

```
$ ls | tee file1 file2 file3
```

12 本精彩的 Linux 书籍

对于想更深入了解 Linux 的学习者，我推荐你们阅读下面这些书。下面提到的这 12 本有关 Linux 的书籍，不能囊括所有 Linux 的内容，也不是绝对权威的。但是这 12 本书是我这几年最喜欢读的书，而且如果你以前没有读过，那么我坚信你读了之后会提升你驾驭 Linux 的能力。

1. Sed and Awk，作者Dale Dougherty and Arnold Robbins。这本书改变了我在Linux命令行下的工作方式。这本书可能是你学习Sed和Awk时唯一需要的参考资料。一旦你掌握了Sed和Awk的基础，你就会着迷于能够快速并且高效的完成一系列复杂工作。我平常用于sed 和awk 的参考的是由相同作者所著的《Sed and Awk Pocket Reference》。
2. Learning The Vi and Vim Editor，作者Arnold Robbins。 我对于命令行下的操作情有独钟。因此，自然地我非常喜欢Vi 和 Vim 进行文本编辑。回首数年之前，当我需要在Linux下写大量的C程序的代码时，我总会参考Vi文本编辑器的快速参考手册。即使你已经用了很长时间的Vi和vim编辑器，如果你没有读过这本书的话，那么请你读一下这本书。你会惊讶于Vim编辑器的强大功能。
3. Bash Cookbook ，作者Carl Albing, JP Vossen and Cameron Newham。无论你是系统管理员还是数据库管理员，或者是个开发人员，你都会或多或少的写些shell脚本。聪明的系统管理员都知道掌握了shell脚本编程之后，可以让shell脚本作一些微小繁杂的工作，从而使你的服务器处于自动运行状态。要达到这种境界，你就需要掌握本书中所举出的例子。市面上这方面的书籍相当的多，但这本书以丰富细致的实例而完全超越它们。
4. SSH,The Secure Shell ，作者Daniel J. Barrett, Richard E. Silverman and Robert G. Byrnes。这无疑是关于SSH方面最好的书了。这本书介绍了SSH的理论和实践的各个方面。作为最终却用户使用SSH是相当方便的。但是作为一个管理员配置SSH相对来讲比较复杂，同时要对SSH有更深的理解。对于每个系统管理员，这本书都是必读书籍。这本书中的例子直接给出了该如何根据需求(SHH1,SHH2和OpenSSH)来量身定制SSH的不同口味。
5. Essential System Administrator，作者Æleen Frisch。对于想成为UNIX系统管理员的人来说这是一本相当棒的书。这本书囊括了所有的系统管理工作。很好得整合了AIX,FreeBSD, HP-UX，Linux,Solaris 和Tru64等多种UNIX版本。所有当你在处理不同版本的UNIX时，它是个不错的参考。以前当我同时管理多种UNIX系统时候就用了这本书的便携版----Essential System Administration Pocket Reference。
6. Linux Server Hacks，卷一，作者Rob Flickenger。这本书中收藏了100相当不错的技巧。搭建一个Linux实验平台并且一一测试这些技巧。这些技巧被分成了不同的章节—服务器基础、版本控制、备份、网络、监控、SSH、脚本和数据服务器。当你完全读懂Linux Server Hacks，就可以掌握了这些技巧；卷二，作者Willian von Hagen 和 Brian Jones。这本书也包含了100个技巧，集中在认证,监控,安全,性能和连通性。
7. DNS and BIND，作者Cricket Liu 和 Paul Albitz。几年前我通过阅读在线文档第一次配置了DNS. 本着了解DNS 和 BIND 是如何工作的目的，我买了这本书。之后它的版本又更新了两次，每次我都购买了新版本。如果你是个认真的系统管理员，这本书应该在你的书库里。
8. Understanding the Linux Kernel，作者Daniel Bovet 和 Marco Cesati。如果你是一个Linux环境下的开发人员或者系统管理员，那么这本书是必读的。这本书以一种结构化且符合逻辑

的方式阐述了Linux 2.6内核的工作原理。这本书介绍了内核的内存管理，进程调度， I/O架构以及块设备管理等内容。这本书是为那些想深入了解Linux的Geek们而量身打造的。

9. Linux Cookbook，**作者Carla Schroder**。这本书分别以用户和管理员的角度阐释Linux的各种特性。其中两个章节介绍了如何在基于RPM的系统以及Debian下安装和管理软件。如果你使用的是RedHat，由Daniel J. Barrett所写，包括了Linux命令的所有示例用法的《the Linux Pocker Guide》对你来说将会是个不错的选择。
10. Linux Firewalls，**作者Michael Rash**。如果想要建立一个安全的Linux系统，那么这本书是必读的。关于防火墙有很多相关书籍。但是这本书详述了如何用防火墙，psad, fwsnort配置一个入侵检测系统。如果你想要一本有关防火墙的详细参考，那么由Gregory N. Purdy 所著的《Linux Iptables Pocket Reference》将是你的最佳选择。
11. Linux Administration Handbook，**作者Evi Nemeth， Garth Snydet 和 Trent R. Hein**。早年，在我做系统程序员的时候，我经常参考这本书。这是一本相当非常详细的书，分成了三章Basic Administration， Networking 和 Bunch O’Stuff， 共有将近1000页，30节。
12. Beginning Ubuntu Linux，**作者Keir Thomas 和 Jaime Sicam。**对于那些想从Windows转向Linux并在自己的老机器上安装Ubuntu的人，这本书就是你所需要的。我坚信这本书可以将Linux的信息介绍给那些不用Linux的人。如果你想要你的密友或朋友学习Linux系统的话，那么装上Ubuntu并用把这本书作为礼物送给他，那么他一定会非常感激你的。

扩展阅读

下面是一些源自The Geek Stuff博客的扩展阅读材料。Best Of The Blog部分有更多的文章。

- Unix LS Command: 15 Practical Examples
- Turbocharge PuTTY with 12 Powerful Add-Ons
- wget Tutorial: 15 Awesome Examples to Download Files from Internet
- Ping Tutorial: 15 Effective Ping Command Examples
- Nagios – 监控程序
  - Nagios Jumpstart Guide
  - Monitor Window Server
  - Monitor Linux Server
  - Monitor Network Sever
  - Monitor VPN Device
- 在不输入密码的情况下执行SSH和SCP:
  - From openSSH to openSSH
  - From openSSH to SSH2
  - From SSH2 to SSH2
- Hello World Examples (Learn a programming language
- UNIX Sed Tips and Tricks
- Ubuntu Tips and Tricks
- MySQL Tutorials
- PostgreSQL Tutorials
- vi/vim Tips and Tricks
  - Vim Macro Tutorial: How To Record and Play
  - 怎样像Perl IDE和C/C++ IDE一样使用Vim
  - Automatic Word Completion in Vim
  - 3 Steps to Add Custom Header to a File Using Vim
- The Ultimate Guide for Creating Strong Passwords
- 6 Steps to Secure Your Home Wireless Network
- Firefox Add-On: Hire 7 Personal Bodyguards to Browse Internet Securely
- Tripwire Tutorial: Linux Host Based Intrusion Detection System
- Midnight Commander (mc) Guide: Powerful Text based File Manager for Unix

您的反馈与支持

我希望 Linux 101 Hacks 这本电子书对您能有所帮助。谢谢您的阅读。我对读者对 TGS 长期的支持表示衷心的感谢。如果没有你们给予的极大支持，我很难找到写出这本书的动力。

获取新的 Linux 方面的文章

如果你想得在现有基础上的 LINUX 方面的技巧和指导，请订阅 The Geek Stuff blog http://www.thegeekstuff.com/subscribe/ 。订阅后后，我们会直接把 TGS 上的最新文章发送到您的邮箱或者 RSS 阅览器中。

您的反馈

这本书也有在线版，网址是—http://linux.101hacks.com/toc您可以在这个网上将你的反馈写出来，我们也可以在线讨论这本书中提到的技巧。

你还可以将你的反馈，疑问以及建议通过这个地址发给我
http://www.thegeekstuff.com/contact/

您的支持

如果你喜欢《Linux 101 Hacks》这本电子书，而且想对我的工作表示支持（或者想更高效地使用 Vim 编辑器），那么请购买 《Vim 101 Hacks》这本书吧。