Linux

第一章 系统概述

1 目录结构

- /bin：
  bin是Binary的缩写, 这个目录存放着最经常使用的命令。
- /boot：
  这里存放的是启动Linux时使用的一些核心文件，包括一些连接文件以及镜像文件。
- /dev ：
  dev是Device(设备)的缩写, 该目录下存放的是Linux的外部设备，在Linux中访问设备的方式和访问文件的方式是相同的。
- /etc：
  这个目录用来存放所有的系统管理所需要的配置文件和子目录。
- /home：
  用户的主目录，在Linux中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。
- /lib：
  这个目录里存放着系统最基本的动态连接共享库，其作用类似于Windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。
- /lost+found：
  这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些文件。
- /media：
  linux系统会自动识别一些设备，例如U盘、光驱等等，当识别后，linux会把识别的设备挂载到这个目录下。
- /mnt：
  系统提供该目录是为了让用户临时挂载别的文件系统的，我们可以将光驱挂载在/mnt/上，然后进入该目录就可以查看光驱里的内容了。
- /opt：
  这是给主机额外安装软件所摆放的目录。比如你安装一个ORACLE数据库则就可以放到这个目录下。默认是空的。
- /proc：
  这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。这个目录的内容不在硬盘上而是在内存里，我们也可以直接修改里面的某些文件.
- /root：
  该目录为系统管理员，也称作超级权限者的用户主目录。
- /sbin：
  s就是Super User的意思，这里存放的是系统管理员使用的系统管理程序。
- /srv：
  该目录存放一些服务启动之后需要提取的数据。

- /sys：
  这是linux2.6内核的一个很大的变化。该目录下安装了2.6内核中新出现的一个文件系统 sysfs 。sysfs文件系统集成了下面3种文件系统的信息：针对进程信息的proc文件系统、针对设备的devfs文件系统以及针对伪终端的devpts文件系统。该文件系统是内核设备树的一个直观反映。当一个内核对象被创建的时候，对应的文件和目录也在内核对象子系统中被创建。
- /tmp：
  这个目录是用来存放一些临时文件的。
- /usr：
  这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似于windows下的program files目录。
  - /usr/bin/ 存放系统命令，普通用户和超级用户都可以执行。这些命令和系统启动无关，在单用户模式下不能执行
  - /usr/sbin/ 存放根文件系统不必要的系统管理命令，如多数服务程序，只有 root 可以使用。
  - /usr/lib/ 应用程序调用的函数库保存位置
  - /usr/XllR6/ 图形界面系统保存位置
  - /usr/local/ 手工安装的软件保存位置。我们一般建议源码包软件安装在这个位置
  - /usr/share/ 应用程序的资源文件保存位置，如帮助文档、说明文档和字体目录
  - /usr/src/ 源码包保存位置。我们手工下载的源码包和内核源码包都可以保存到这里。不过笔者更习惯把手工下载的源码包保存到 /usr/local/src/ 目录中，把内核源码保存到 /usr/src/linux/ 目录中
  - /usr/include C/C++ 等编程语言头文件的放置目录
- /var：
  这个目录中存放着在不断扩充着的东西，我们习惯将那些经常被修改的目录放在这个目录下。包括各种日志文件。
  - /var/lib/ 程序运行中需要调用或改变的数据保存位置。如 MySQL 的数据库保存在 /var/lib/mysql/ 目录中
  - /var/log/ 登陆文件放置的目录，其中所包含比较重要的文件如 /var/log/messages, /var/log/wtmp 等。
  - /var/run/ 一些服务和程序运行后，它们的 PID（进程 ID）保存位置
  - /var/spool/ 里面主要都是一些临时存放，随时会被用户所调用的数据，例如 /var/spool/mail/ 存放新收到的邮件，/var/spool/cron/ 存放系统定时任务。
  - /var/www/ RPM 包安装的 Apache 的网页主目录
  - /var/nis和/var/yp NIS 服务机制所使用的目录，nis 主要记录所有网络中每一个 client 的连接信息；yp 是 linux 的 nis 服务的日志文件存放的目录
  - /var/tmp 一些应用程序在安装或执行时，需要在重启后使用的某些文件，此目录能将该类文件暂时存放起来，完成后再行删除
- /run：
  是一个临时文件系统，存储系统启动以来的信息。当系统重启时，这个目录下的文件应该被删掉或清除。如果你的系统上有 /var/run 目录，应该让它指向 run。在 Linux 系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。

2 帮助命令

在linux终端，面对命令不知道怎么用，或不记得命令的拼写及参数时，我们需要求助于系统的帮助文档； linux系统内置的帮助文档很详细，通常能解决我们的问题，我们需要掌握如何正确的去使用它们；

在只记得部分命令关键字的场合，我们可通过man -k来搜索；
需要知道某个命令的简要说明，可以使用whatis；而更详细的介绍，则可用info命令；
查看命令在哪个位置，我们需要使用which；
而对于命令的具体参数及使用方法，我们需要用到强大的man；
下面介绍这些命令；

whatis

查看命令的简要说明

```
$whatis command
```

正则匹配:

```
$whatis -w "loca*"
```

更加详细的说明文档:

```
$info command
```

man

查询命令command的说明文档:

```
$man command
eg: man date
```

在man的帮助手册中，将帮助文档分为了9个类别，对于有的关键字可能存在多个类别中， 我们就需要指定特定的类别来查看； （一般我们查询bash命令，归类在1类中）；

man页面所属的分类标识(常用的是分类1和分类3)

1. 用户可以操作的命令或者是可执行文件
2. 系统核心可调用的函数与工具等
3. 一些常用的函数与数据库
4. 设备文件的说明
5. 设置文件或者某些文件的格式
6. 游戏
7. 惯例与协议等。例如Linux标准文件系统、网络协议、ASCⅡ，码等说明内容
8. 系统管理员可用的管理条令
9. 与内核有关的文件

当whatis下的命令有多个分类时。使用数字表示某个分类下的帮助手册。

```
$man 3 printf
```

```
$man -k keyword
```

查询关键字 根据命令中部分关键字来查询命令，适用于只记住部分命令的场合；

查找GNOME的config配置工具命令，对于某个单词搜索，可直接使用/word来使用:

```
$man -k GNOME config| grep 1
```

which & whereis

查看程序的binary文件所在路径:

```
$which command
```

查看程序的搜索路径:

```
$whereis command
```

当系统中安装了同一软件的多个版本时，不确定使用的是哪个版本时，这个命令就能派上用场；

第二章 文件及目录管理

> - ls: 列出目录
> - cd：切换目录
> - pwd：显示目前的目录
> - mkdir：创建一个新的目录
> - rmdir：删除一个空的目录
> - cp: 复制文件或目录
> - rm: 移除文件或目录
> - mv: 移动文件与目录，或修改文件与目录的名称
> - find locate:文件的查询和检索:
> - cat head tail more: 查看文件内容：
> - 管道和重定向: ; | && > >>
> - 打包 tar -cvf
> - 解包 tar -xvf
> - 压缩 gzip bzip zip 7z
> - 解压缩 gunzip,bunzip,unzip
> - 写出命令以及主要的参数说明，具体内容查看参考手册。

1 文件目录操作

ls：列出目录

简介：

在Linux系统当中， ls 命令可能是最常被运行的。

语法：

```
[root@www ~]# ls [-aAdfFhilnrRSt] 目录名称
[root@www ~]# ls [--color={never,auto,always}] 目录名称
[root@www ~]# ls [--full-time] 目录名称
```

选项：

- -a ： 全部的文件，连同隐藏档(开头为 . 的文件) 一起列出来(常用)
- -d ： 仅列出目录本身，而不是列出目录内的文件数据(常用)
- -l ： 长数据串列出，包含文件的属性与权限等等数据；(常用)

实例：

```
# 将家目录下的所有文件列出来(含属性与隐藏档)
[root@www ~]# ls -al ~
```

cd：切换目录

简介：

cd是Change Directory的缩写，这是用来变换工作目录的命令。

语法：

```
cd [相对路径或绝对路径]
```

实例：

```
#使用 mkdir 命令创建 runoob 目录
[root@www ~]# mkdir runoob

#使用绝对路径切换到 runoob 目录
[root@www ~]# cd /root/runoob/

#使用相对路径切换到 runoob 目录
[root@www ~]# cd ./runoob/

# 表示回到自己的家目录，亦即是 /root 这个目录
[root@www runoob]# cd ~

# 表示去到目前的上一级目录，亦即是 /root 的上一级目录的意思；
[root@www ~]# cd ..
接下来大家多操作几次应该就可以很好的理解 cd 命令的。
```

pwd：显示目前所在的目录

简介：

pwd 是 Print Working Directory 的缩写，也就是显示目前所在目录的命令。

```
[root@www ~]# pwd [-P]
```

选项：

- -P ： 显示出确实的路径，而非使用连结 (link) 路径。

实例：

```
[root@www ~]# pwd
/root    <== 显示出目录啦～
实例显示出实际的工作目录，而非连结档本身的目录名而已。

[root@www ~]# cd /var/mail    <==注意，/var/mail是一个连结档
[root@www mail]# pwd
/var/mail             <==列出目前的工作目录
[root@www mail]# pwd -P
/var/spool/mail    <==怎么回事？有没有加 -P 差很多～
[root@www mail]# ls -ld /var/mail
lrwxrwxrwx 1 root root 10 Sep  4 17:54 /var/mail -> spool/mail
# 看到这里应该知道为啥了吧？因为 /var/mail 是连结档，连结到 /var/spool/mail
# 所以，加上 pwd -P 的选项后，会不以连结档的数据显示，而是显示正确的完整路径啊！
```

mkdir (创建新目录)

简介：

如果想要创建新的目录的话，那么就使用mkdir (make directory)吧。

语法：

```
mkdir [-mp] 目录名称
```

选项：

- -m： 配置文件的权限喔！直接配置，不需要看默认权限 (umask) 的脸色～
- -p： 帮助你直接将所需要的目录(包含上一级目录)递归创建起来！

```
# 请到/tmp底下尝试创建数个新目录看看：
[root@www ~]# cd /tmp
[root@www tmp]# mkdir test     <==创建一名为 test 的新目录
[root@www tmp]# mkdir test1/test2/test3/test4
mkdir: cannot create directory `test1/test2/test3/test4':
No such file or directory           <== 没办法直接创建此目录啊！
[root@www tmp]# mkdir -p test1/test2/test3/test4
加了这个 -p 的选项，可以自行帮你创建多层目录！

# 实例：创建权限为 rwx--x--x 的目录。
[root@www tmp]# mkdir -m 711 test2
[root@www tmp]# ls -l
drwxr-xr-x  3 root  root 4096 Jul 18 12:50 test
drwxr-xr-x  3 root  root 4096 Jul 18 12:53 test1
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
上面的权限部分，如果没有加上 -m 来强制配置属性，系统会使用默认属性。

如果我们使用 -m ，如上例我们给予 -m 711 来给予新的目录 drwx--x--x 的权限。
```

rmdir (删除空的目录)

语法：

```
rmdir [-p] 目录名称
```

选项：

- -p： 连同上一级『空的』目录也一起删除

实例：

```
删除 runoob 目录
[root@www tmp]# rmdir runoob/
将 mkdir 实例中创建的目录(/tmp 底下)删除掉！

[root@www tmp]# ls -l    <==看看有多少目录存在？
drwxr-xr-x  3 root  root 4096 Jul 18 12:50 test
drwxr-xr-x  3 root  root 4096 Jul 18 12:53 test1
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
[root@www tmp]# rmdir test    <==可直接删除掉，没问题
[root@www tmp]# rmdir test1  <==因为尚有内容，所以无法删除！
rmdir: `test1': Directory not empty
[root@www tmp]# rmdir -p test1/test2/test3/test4
[root@www tmp]# ls -l          <==您看看，底下的输出中test与test1不见了！
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
利用 -p 这个选项，立刻就可以将 test1/test2/test3/test4 一次删除。

不过要注意的是，这个 rmdir 仅能删除空的目录，你可以使用 rm 命令来删除非空目录。
```

cp (复制文件或目录)

简介：

cp 即拷贝文件和目录。

语法:

```
[root@www ~]# cp [-adfilprsu] 来源档(source) 目标档(destination)
[root@www ~]# cp [options] source1 source2 source3 .... directory
```

选项：

- -a：相当於 -pdr 的意思，至於 pdr 请参考下列说明；(常用)
- -d：若来源档为连结档的属性(link file)，则复制连结档属性而非文件本身；
- -f：为强制(force)的意思，若目标文件已经存在且无法开启，则移除后再尝试一次；
- -i：若目标档(destination)已经存在时，在覆盖时会先询问动作的进行(常用)
- -l：进行硬式连结(hard link)的连结档创建，而非复制文件本身；
- -p：连同文件的属性一起复制过去，而非使用默认属性(备份常用)；
- -r：递归持续复制，用於目录的复制行为；(常用)
- -s：复制成为符号连结档 (symbolic link)，亦即『捷径』文件；
- -u：若 destination 比 source 旧才升级 destination！

实例：

```
用 root 身份，将 root 目录下的 .bashrc 复制到 /tmp 下，并命名为 bashrc

[root@www ~]# cp ~/.bashrc /tmp/bashrc
[root@www ~]# cp -i ~/.bashrc /tmp/bashrc
cp: overwrite `/tmp/bashrc'? n  <==n不覆盖，y为覆盖
```

rm (移除文件或目录)

语法：

```
 rm [-fir] 文件或目录
```

选项与参数：

- -f：就是 force 的意思，忽略不存在的文件，不会出现警告信息；
- -i：互动模式，在删除前会询问使用者是否动作
- -r：递归删除啊！最常用在目录的删除了！这是非常危险的选项！！！

实例：

```
将刚刚在 cp 的实例中创建的 bashrc 删除掉！

[root@www tmp]# rm -i bashrc
rm: remove regular file `bashrc'? y
如果加上 -i 的选项就会主动询问喔，避免你删除到错误的档名！
```

mv (移动文件与目录，或修改名称)

语法：

```
[root@www ~]# mv [-fiu] source destination
[root@www ~]# mv [options] source1 source2 source3 .... directory
```

选项：

- -f：force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖；
- -i：若目标文件 (destination) 已经存在时，就会询问是否覆盖！
- -u：若目标文件已经存在，且 source 比较新，才会升级 (update)

实例：

```
复制一文件，创建一目录，将文件移动到目录中
[root@www ~]# cd /tmp
[root@www tmp]# cp ~/.bashrc bashrc
[root@www tmp]# mkdir mvtest
[root@www tmp]# mv bashrc mvtest
将某个文件移动到某个目录去，就是这样做！
将刚刚的目录名称更名为 mvtest2
[root@www tmp]# mv mvtest mvtest2
```

2 文件查找

find

简介：

搜寻文件或目录:

实例：

```
$find ./ -name "core*" | xargs file
```

查找目标文件夹中是否有obj文件:

```
$find ./ -name '*.o'
```

递归当前目录及子目录删除所有.o文件:

```
$find ./ -name "*.o" -exec rm {} \;
```

查看当前目录下文件个数:

```
$find ./ | wc -l
```

locate

简介：

find是实时查找，如果需要更快的查询，可试试locate；locate会为文件系统建立索引数据库，如果有文件更新，需要定期执行更新命令来更新索引库:

语法：

```
$locate string
$updatedb
```

3 文件内容

cat

简介：

由第一行开始显示文件内容

语法：

```
cat [-AbEnTv]
```

选项与参数：

- -A： 相当於 -vET 的整合选项，可列出一些特殊字符而不是空白而已；
- -b： 列出行号，仅针对非空白行做行号显示，空白行不标行号！
- -E： 将结尾的断行字节 $ 显示出来；
- -n： 列印出行号，连同空白行也会有行号，与 -b 的选项不同；
- -T： 将 [tab] 按键以 ^I 显示出来；
- -v： 列出一些看不出来的特殊字符

实例：

```
检看 /etc/issue 这个文件的内容：

[root@www ~]# cat /etc/issue
CentOS release 6.4 (Final)
Kernel \r on an \m
```

tac

简介：

tac与cat命令刚好相反，文件内容从最后一行开始显示，可以看出 tac 是 cat 的倒着写！如：

语法：

```
[root@www ~]# tac /etc/issue
```

nl

简介：

显示行号

语法：

```
nl [-bnw] 文件
```

选项与参数：

- -b： 指定行号指定的方式，主要有两种：
- -b a： 表示不论是否为空行，也同样列出行号(类似 cat -n)；
- -b t： 如果有空行，空的那一行不要列出行号(默认值)；
- -n： 列出行号表示的方法，主要有三种：
- -n ln： 行号在荧幕的最左方显示；
- -n rn： 行号在自己栏位的最右方显示，且不加 0；
- -n rz： 行号在自己栏位的最右方显示，且加 0；
- -w： 行号栏位的占用的位数。

实例：

```
实例一：用 nl 列出 /etc/issue 的内容

[root@www ~]# nl /etc/issue
       1  CentOS release 6.4 (Final)
       2  Kernel \r on an \m
```

more

简介：

一页一页翻动

语法：

```
[root@www ~]# more /etc/man_db.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.6d
....(中间省略)....
--More--(28%)  <== 重点在这一行喔！你的光标也会在这里等待你的命令
```

选项：

- 空白键 (space)：代表向下翻一页；
- Enter　　：代表向下翻『一行』；
- /字串　　：代表在这个显示的内容当中，向下搜寻『字串』这个关键字；
- :f　　：立刻显示出档名以及目前显示的行数；
- q　　：代表立刻离开 more，不再显示该文件内容。
- b 或 [ctrl]-b：代表往回翻页，不过这动作只对文件有用，对管线无用。

less

简介：

一页一页翻动，以下实例输出/etc/man.config文件的内容：

语法：

```
[root@www ~]# less /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.6d
....(中间省略)....
:   <== 这里可以等待你输入命令！
```

选项：

- 空白键　：向下翻动一页；
- [pagedown]：向下翻动一页；
- [pageup]：向上翻动一页；
- /字串　：向下搜寻『字串』的功能；
- ?字串　：向上搜寻『字串』的功能；
- n　　：重复前一个搜寻 (与 / 或 ? 有关！)
- N　　：反向的重复前一个搜寻 (与 / 或 ? 有关！)
- q　　：离开 less 这个程序；

head

简介：

取出文件前面几行

语法：

```
head [-n number] 文件
```

选项与参数：

- -n： 后面接数字，代表显示几行的意思

实例：

```
[root@www ~]# head /etc/man.config
默认的情况中，显示前面 10 行！若要显示前 20 行，就得要这样：

[root@www ~]# head -n 20 /etc/man.config
```

tail

简介：

取出文件后面几行

语法：

```
tail [-n number] 文件
```

选项与参数：

- -n： 后面接数字，代表显示几行的意思
- -f： 表示持续侦测后面所接的档名，要等到按下[ctrl]-c才会结束tail的侦测

```
[root@www ~]# tail /etc/man.config
# 默认的情况中，显示最后的十行！若要显示最后的 20 行，就得要这样：
[root@www ~]# tail -n 20 /etc/man.config
```

$diff file1 file2

```
动态显示文本最新信息:
```

使用egrep查询文件内容:

```
egrep '03.1\/CO\/AE' TSF_STAT_111130.log.012
egrep 'A_LMCA777:C' TSF_STAT_111130.log.035 > co.out2
```

4 链接（快捷方式）

Linux 链接分两种，一种被称为硬链接（Hard Link），另一种被称为符号链接（Symbolic Link）。默认情况下，ln 命令产生硬链接。

硬连接

硬连接指通过索引节点来进行连接。在 Linux 的文件系统中，保存在磁盘分区中的文件不管是什么类型都给它分配一个编号，称为索引节点号(Inode Index)。在 Linux 中，多个文件名指向同一索引节点是存在的。比如：A 是 B 的硬链接（A 和 B 都是文件名），则 A 的目录项中的 inode 节点号与 B 的目录项中的 inode 节点号相同，即一个 inode 节点对应两个不同的文件名，两个文件名指向同一个文件，A 和 B 对文件系统来说是完全平等的。删除其中任何一个都不会影响另外一个的访问。

硬连接的作用是允许一个文件拥有多个有效路径名，这样用户就可以建立硬连接到重要文件，以防止“误删”的功能。其原因如上所述，因为对应该目录的索引节点有一个以上的连接。只删除一个连接并不影响索引节点本身和其它的连接，只有当最后一个连接被删除后，文件的数据块及目录的连接才会被释放。也就是说，文件真正删除的条件是与之相关的所有硬连接文件均被删除。

软连接

另外一种连接称之为符号连接（Symbolic Link），也叫软连接。软链接文件有类似于 Windows 的快捷方式。它实际上是一个特殊的文件。在符号连接中，文件实际上是一个文本文件，其中包含的有另一文件的位置信息。比如：A 是 B 的软链接（A 和 B 都是文件名），A 的目录项中的 inode 节点号与 B 的目录项中的 inode 节点号不相同，A 和 B 指向的是两个不同的 inode，继而指向两块不同的数据块。但是 A 的数据块中存放的只是 B 的路径名（可以根据这个找到 B 的目录项）。A 和 B 之间是“主从”关系，如果 B 被删除了，A 仍然存在（因为两个是不同的文件），但指向的是一个无效的链接。

实验

```
[oracle@Linux]$ touch f1              #创建一个测试文件f1
[oracle@Linux]$ ln f1 f2              #创建f1的一个硬连接文件f2
[oracle@Linux]$ ln -s f1 f3          #创建f1的一个符号连接文件f3
[oracle@Linux]$ ls -li                 # -i参数显示文件的inode节点信息
total 0
9797648 -rw-r--r--  2 oracle oinstall 0 Apr 21 08:11 f1
9797648 -rw-r--r--  2 oracle oinstall 0 Apr 21 08:11 f2
9797649 lrwxrwxrwx  1 oracle oinstall 2 Apr 21 08:11 f3 -> f1
```

从上面的结果中可以看出，硬连接文件 f2 与原文件 f1 的 inode 节点相同，均为 9797648，然而符号连接文件的 inode 节点不同。

```
[oracle@Linux]$ echo "I am f1 file" >>f1
[oracle@Linux]$ cat f1
I am f1 file
[oracle@Linux]$ cat f2
I am f1 file
[oracle@Linux]$ cat f3
I am f1 file
[oracle@Linux]$ rm -f f1
[oracle@Linux]$ cat f2
I am f1 file
[oracle@Linux]$ cat f3
cat: f3: No such file or directory
```

通过上面的测试可以看出：当删除原始文件 f1 后，硬连接 f2 不受影响，但是符号连接 f1 文件无效

总结

依此您可以做一些相关的测试，可以得到以下全部结论：

- 删除符号连接f3,对f1,f2无影响；
- 删除硬连接f2，对f1,f3也无影响；

- 删除原文件f1，对硬连接f2没有影响，导致符号连接f3失效；
- 同时删除原文件f1,硬连接f2，整个文件会真正的被删除。

5 管道和重定向

```
批处理命令连接执行，使用 |
串联：使用分号 ；
前面成功，则执行后面一条，否则，不执行:&&
前面失败，则后一条执行：||
ls /proc && echo  suss! || echo failed.
能够提示命名是否执行成功or失败；
```

与上述相同效果的是:

```
if ls /proc; then echo suss; else echo fail; fi
```

重定向:

```
ls  proc/*.c > list 2> &l 将标准输出和标准错误重定向到同一文件；
```

等价的是:

```
ls  proc/*.c &> list
```

清空文件:

```
:> a.txt
```

重定向到文件的末尾

```
echo aa >> a.txt
```

6 打包压缩

打包/ 压缩

在linux中打包和压缩和分两步来实现的；

1. 打包是将多个文件归并到一个文件:

```
tar -cvf etc.tar /etc <==仅打包，不压缩!
-c :打包选项
-v :显示打包进度
-f :使用档案文件
注：有的系统中指定参数时不需要在前面加上-，直接使用tar xvf
```

用tar实现文件夹同步，排除部分文件不同步:

```
tar --exclude '*.svn' -cvf - /path/to/source | ( cd /path/to/target; tar -xf -)
```

2. 压缩,生成 demo.txt.gz

```
$gzip demo.txt
```

解包/解压缩

解包

```
tar -xvf demo.tar
-x 解包选项
```

解压后缀为 .tar.gz的文件

1. 解压缩

```
$gunzip     demo.tar.gz
```

2. 解包:

```
$tar -xvf  demo.tar
```

bz2解压解包:

```
tar jxvf demo.tar.bz2
```

如果tar 不支持j，则同样需要分两步来解包解压缩，使用bzip2来解压，再使用tar解包:

```
bzip2 -d  demo.tar.bz2
tar -xvf  demo.tar
-d decompose,解压缩
```

tar解压参数说明:

```
-z 解压gz文件
-j 解压bz2文件
-J 解压xz文件
```

7 属性修改

chgrp：更改文件属组

语法:

```
chgrp [-R] 属组名 文件名
```

参数选项:

-R：递归更改文件属组，就是在更改某个目录文件的属组时，如果加上-R的参数，那么该目录下的所有文件的属组都会更改。

chown：更改文件属主，也可以同时更改文件属组

语法:

```
chown [ - R] 属主名 文件名
chown [-R] 属主名：属组名 文件名
```

chmod

Linux文件属性有两种设置方法，一种是数字，一种是符号。

Linux文件的基本权限就有九个

- 三种身份：owner/group/others
- 三种权限：read/write/execute

各权限的分数对照表如下：

- r:4、 w:2、 x:1

```
chmod [-R] xyz 文件或目录
chmod   u/g/o/a  +(加入)/-(除去)/=(设定)   r/w/x  文件或目录
```

第三章 用户管理

> useradd passwd userdel usermod chmod chown .bashrc .bash_profile

1 用户管理

添加用户，为用户创建相应的帐号和用户目录/home/username；

```
$useradd -m username
```

用户添加之后，设置密码，密码以交互方式创建:

```
$passwd username
```

删除用户

```
$userdel -r username
```

不带选项使用 userdel，只会删除用户。用户的家目录将仍会在/home目录下。要完全的删除用户信息，使用-r选项；

帐号切换 登录帐号为userA用户状态下，切换到userB用户帐号工作:

```
$su userB
```

进入交互模型，输入密码授权进入；

2 用户组管理

将用户加入到组
默认情况下，添加用户操作也会相应的增加一个同名的组，用户属于同名组； 查看当前用户所属的组:

```
$groups
```

一个用户可以属于多个组，将用户加入到组:

```
$usermod -G groupNmame username
```

变更用户所属的根组(将用户加入到新的组，并从原有的组中除去）：

```
$usermod -g groupName username
```

查看系统所有组，系统的所有用户及所有组信息分别记录在两个文件中：/etc/passwd , /etc/group 默认情况下这两个文件对所有用户可读。查看所有用户及权限:

```
$more /etc/passwd
```

查看所有的用户组及权限:

```
$more /etc/group
```

3 环境变量

bashrc & profile

bashrc与profile都用于保存用户的环境信息，bashrc用于交互式non-loginshell，而profile用于交互式login shell。

```
/etc/profile，/etc/bashrc 是系统全局环境变量设定
~/.profile，~/.bashrc用户目录下的私有环境变量设定
```

login过程

当登入系统获得一个shell进程时，其读取环境设置脚本分为三步:

- 首先读入的是全局环境变量设置文件/etc/profile，然后根据其内容读取额外的文档，如/etc/profile.d和/etc/inputrc
- 读取当前登录用户Home目录下的文件~/.bash_profile，其次读取~/.bash_login，最后读取~/.profile，这三个文档设定基本上是一样的，读取有优先关系
- 读取~/.bashrc。

~/.profile与~/.bashrc的区别

这两者都具有个性化定制功能

- ~/.profile可以设定本用户专有的路径，环境变量，等，它只能登入的时候执行一次
- ~/.bashrc也是某用户专有设定文档，可以设定路径，命令别名，每次shell script的执行都会使用它一次

例如，我们可以在这些环境变量中设置自己经常进入的文件路径，以及命令的快捷方式：

```
.bashrc
alias m='more'
alias cp='cp -i'
alias mv='mv -i'
alias ll='ls -l'
alias lsl='ls -lrt'
alias lm='ls -al|more'

log=/opt/applog/common_dir
unit=/opt/app/unittest/common

.bash_profile
. /opt/app/tuxapp/openav/config/setenv.prod.sh.linux
export PS1='$PWD#'
```

通过上述设置，我们进入log目录就只需要输入cd $log即可；

第四章 磁盘管理

> - df：列出文件系统的整体磁盘使用量
> - du：检查磁盘空间使用量
> - fdisk：用于磁盘分区

1 df：磁盘使用量

df命令参数功能:

检查文件系统的磁盘空间占用情况。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。

语法：

```
df [-ahikHTm] [目录或文件名]
```

选项与参数：

- -a ： 列出所有的文件系统，包括系统特有的 /proc 等文件系统；
- -k ： 以 KBytes 的容量显示各文件系统；
- -m ： 以 MBytes 的容量显示各文件系统；
- -h ： 以人们较易阅读的 GBytes, MBytes, KBytes 等格式自行显示；
- -H ： 以 M=1000K 取代 M=1024K 的进位方式；
- -T ： 显示文件系统类型, 连同该 partition 的 filesystem 名称 (例如 ext3) 也列出；
- -i ： 不用硬盘容量，而以 inode 的数量来显示

2 du：使用空间

Linux du命令也是查看使用空间的，但是与df命令不同的是Linux du命令是对文件和目录磁盘使用的空间的查看，还是和df命令有一些区别的，这里介绍Linux du命令。

语法：

```
du [-ahskm] 文件或目录名称
```

选项与参数：

- -a ： 列出所有的文件与目录容量，因为默认仅统计目录底下的文件量而已。
- -h ： 以人们较易读的容量格式 (G/M) 显示；
- -s ： 列出总量而已，而不列出每个各别的目录占用容量；
- -S ： 不包括子目录下的总计，与 -s 有点差别。
- -k ： 以 KBytes 列出容量显示；
- -m ： 以 MBytes 列出容量显示；

3 fdisk：磁盘分区

fdisk 是 Linux 的磁盘分区表操作工具。

语法：

```
fdisk [-l] 装置名称
```

选项与参数：

- -l： 输出后面接的装置所有的分区内容。若仅有 fdisk -l 时， 则系统将会把整个系统内能够搜寻到的装置的分区均列出来。

第五章 进程管理

> ps top lsof kill pmap

查询正在运行的进程信息

```
$ps -ef
```

eg:查询归属于用户colin115的进程

```
$ps -ef | grep colin115
$ps -lu colin115
```

查询进程ID（适合只记得部分进程字段）

```
$pgrep 查找进程
```

eg:查询进程名中含有re的进程

```
[/home/weber#]pgrep -l re
```

以完整的格式显示所有的进程

```
$ps -ajx
```

显示进程信息，并实时更新

```
$top
```

查看端口占用的进程状态：

```
lsof -i:3306
```

查看用户username的进程所打开的文件

```
$lsof -u username
```

查询init进程当前打开的文件

```
$lsof -c init
```

查询指定的进程ID(23295)打开的文件：

```
$lsof -p 23295
``
查询指定目录下被进程开启的文件（使用+D 递归目录）：
```

$lsof +d mydir1/

```
## 2 终止进程
杀死指定PID的进程 (PID为Process ID)
```

$kill PID

```
    杀死相关进程
```

kill -9 3434

```
    杀死job工作（job为job number）
```

$kill %job

```
    ## 3 进程监控
    查看系统中使用CPU、使用内存最多的进程；
```

$top
(->)P

```
    输入top命令后，进入到交互界面；接着输入字符命令后显示相应的进程状态：

    对于进程，平时我们最常想知道的就是哪些进程占用CPU最多，占用内存最多。以下两个命令就可以满足要求：
```

P：根据CPU使用百分比大小进行排序。
M：根据驻留内存大小进行排序。
i：使top不显示任何闲置或者僵死进程。

```
    这里介绍最使用的几个选项,对于更详细的使用，详见 top linux下的任务管理器 ；

    ## 4 分析线程栈
    使用命令pmap，来输出进程内存的状况，可以用来分析线程堆栈；
```

$pmap PID

```
    5.5. 综合运用
    将用户colin115下的所有进程名以av_开头的进程终止：
```

ps -u colin115 | awk '/av_/ {print "kill -9 " $1}' | sh

```
    将用户colin115下所有进程名中包含HOST的进程终止：
```

第六章 安装软件

1 RPM

格式：rpm [选项] RPM包文件

常用选项：

-i：安装一个新的rpm软件包

-h：以“#”号显示安装的进度

-v：显示安装过程中的详细信息

--force：强制安装所指定的rpm软件包

--nodeps：安装软件时，忽略依赖关系

```
    卸载指定的.rpm软件包
    格式：rpm -e 软件名
    --nodeps：卸载软件时，忽略依赖关系
```

```
查询已安装的.rpm软件包
格式：rpm -q[子选项]［软件名］
常用子选项命令：
-qa：查看系统中已安装的所有RPM软件包列表
-qi：查看指定软件的详细信息
-ql：查询指定软件包所安装的目录、文件列表
-qc：仅显示指定软件包安装的配置文件
-qd：仅显示指定软件包安装的文档文件
```

2 YUM

格式：yum install 软件名 [-y]

-y：如果使用-y，那么在安装软件时命令行就不会出现"Is this ok[y/N]"这条提醒语句了，更不需要在命令行输入y或N了，直接安装软件。