第 3 章　常用 Shell（Bash）命令

在前一章中，我们学习了 Shell 变量和 Shell 环境相关的一些知识。想必你对 Linux Shell 已经有了一个初步的认识。

在这一章，我们将学习一些常用的 Shell 命令，应用好这些命令，将使你在 Linux 下执行一些任务具有事半功倍的效果。

3.1　查看文件和目录

3.1.1　ls 命令实例：列出文件名和目录

ls 命令是 Linux 中最常用的命令之一。我相信 ls 命令可能是你进入 Linux 命令行提示符时第一个输入的命令。虽然你还没有意识到，你可能每天都在频繁地使用 ls 命令。

在命令行提示符下，直接输入 ls 命令，不带任何选项，将列出当前目录下所有文件和目录，但不会显示详细的信息，比如，文件类型、大小、修改日期和时间、权限等等：

```
$ ls
acpid         conman.old  libvirt     pm            secure.1     wtmp
anaconda.log     cron     mail        ppp           secure.2    xferlog
anaconda.syslog  cron.1   maillog     prelink       secure.3    xferlog.1
audit            cron.2   maillog.1   rpmpkgs       secure.4    xferlog.2
boot.log         cron.3   maillog.2   rpmpkgs.1     spooler     xferlog.3
boot.log.1       cron.4   maillog.3   rpmpkgs.2     spooler.1   xferlog.4
boot.log.2       cups     maillog.4   rpmpkgs.3     spooler.2   Xorg.0.log
boot.log.3       dmesg    messages    rpmpkgs.4   spooler.3   Xorg.0.log.old
boot.log.4       faillog  messages.1  sa              spooler.4     xrdp
brcm-iscsi.log   gdm      messages.2  samba           tallylog     yum.log
btmp             httpd    messages.3  scrollkeeper.log  tomcat5    yum.log.1
conman           lastlog  messages.4  secure            virusReport
```

使用-1 选项，将每行显示一条记录：

```
$ ls -1
acpid
anaconda.log
anaconda.syslog
audit
boot.log
boot.log.1
boot.log.2
boot.log.3
boot.log.4
```

使用-l 选项，将以长列表格式显示文件和目录，包括文件类型、大小、修改日期和时

间、权限等信息：

```
$ ls -l
total 50716
-rw-r----- 1 root root      21630 Aug 19 00:01 acpid
-rw------- 1 root root     381429 Oct  7  2012 anaconda.log
-rw------- 1 root root      24902 Oct  7  2012 anaconda.syslog
drwxr-x--- 2 root root       4096 Aug 18 22:26 audit
-rw------- 1 root root          0 Aug 18 04:02 boot.log
-rw------- 1 root root          0 Aug 11 04:02 boot.log.1
-rw------- 1 root root          0 Aug  4 04:02 boot.log.2
-rw------- 1 root root          0 Jul 28 04:02 boot.log.3
-rw------- 1 root root          0 Jul 21 04:02 boot.log.4
```

从上面的输出中可以看到每行有 7 个字段，那么每个字段的含义是什么呢？下面就来介绍一下。

- 第一个字符——文件类型。在上面的例子中，第一个字符有的是连字符“-”，有的是字母“d”。使用 ls –l 命令的输出中，第一个字符可能有如下几种，其代表的含义也如下所示。

-——普通文件。

d——目录。

s——套接字文件。

l——链接文件。

- 字段 1：文件权限。接下来的 9 个字符指示文件的权限。每 3 个字符分别涉及所有者、用户组和其他用户的读、写、执行权限。例如，rw-r-----表示所有者有读和写的权限，用户组有读的权限，而其他用户没有任何权限。
- 字段 2：链接数。1 表示只有一个链接到此文件。
- 字段 3： 所有者。在上例中，文件的所有者均为 root。
- 字段 4：用户组。在上例中，文件的用户组也均为 root。
- 字段 5：文件大小。默认输出的大小单位是字节。
- 字段 6：文件最近一次被修改的日期时间。例如上例中，文件 acpid 的修改时间是 Aug 19 00:01。
- 字段 7：文件名。

使用-lh 选项可以将文件大小显示为符合人类阅读习惯的格式：

```
$ ls -lh
total 50M
-rw-r----- 1 root root   22K Aug 19 00:01 acpid
-rw------- 1 root root  373K Oct  7  2012 anaconda.log
-rw------- 1 root root   25K Oct  7  2012 anaconda.syslog
drwxr-x--- 2 root root  4.0K Aug 18 22:26 audit
-rw------- 1 root root     0 Aug 18 04:02 boot.log
-rw------- 1 root root     0 Aug 11 04:02 boot.log.1
-rw------- 1 root root     0 Aug  4 04:02 boot.log.2
-rw------- 1 root root     0 Jul 28 04:02 boot.log.3
-rw------- 1 root root     0 Jul 21 04:02 boot.log.4
-rw------- 1 root root  1.1M Aug 18 04:02 cron.1
-rw------- 1 root root  952K Aug 11 04:02 cron.2
-rw------- 1 root root  856K Aug  4 04:02 cron.3
-rw------- 1 root root  1.1M Jul 28 04:02 cron.4
```

使用-F 选项，将使用不同的特殊字符归类不同的文件类型：

```
$ ls -F /lib
bdevid/                                libgmodule-2.0.so@
cpp@                                   libgmodule-2.0.so.0@
dbus-1/                                libgmodule-2.0.so.0.1200.3*
firmware/                              libgobject-2.0.a
```

如上例中所示：

- /——表示目录。
- 无特殊字符——表示普通文件。
- @——表示链接文件。
- *——表示可执行文件。

注意：你也可以使用 ls --color=auto 命令，将不同的文件类型显示为不同的颜色，或将--color 与-F 选项联合使用。

联合使用-ld 选项，可以以长列表格式列出某个目录的信息：

```
$ ls -ld /var/log
drwxr-xr-x 16 root root 4096 Aug 21 04:02 /var/log
```

使用-R 选项，将递归地列出子目录的内容：

```
$ ls -R /etc/sysconfig/
/etc/sysconfig/:
apmd         crond        irqbalance    network-scripts   smartmontools
apm-scripts  desktop      kernel           nfs            snmpd.options
atd          dund         keyboard         ntpd           snmptrapd.options
auditd       firstboot    krb524           pand           syslog
authconfig   grub         kudzu            pm-action      sysstat
autofs       hidd         libvirtd         prelink        sysstat.ioconf
bluetooth  httpd  libvirt-guests  raid-check  system-config-securitylevel
cbq          hwconf       lincase         rawdevices     system-config-users
cfenvd       i18n         lm_sensors       readonly-root    tomcat5
cfexecd      init            mkinitrd      rhn            udev-stw
cfservd      ip6tables-config  modules     samba           vncservers
clock        ipmi             nasd         samba.rpmnew    wpa_supplicant
conman       iptables         netconsole   saslauthd        xinetd
console      iptables-config   network     selinux          yppasswdd
cpuspeed     irda              networking   sendmail

/etc/sysconfig/apm-scripts:
apmscript

/etc/sysconfig/cbq:
avpkt  cbq-0000.example

/etc/sysconfig/console:

/etc/sysconfig/mkinitrd:
multipath

/etc/sysconfig/modules:
other.modules  udev-stw.modules

/etc/sysconfig/networking:
```

```
devices  profiles

/etc/sysconfig/networking/devices:

/etc/sysconfig/networking/profiles:
default

/etc/sysconfig/networking/profiles/default:

/etc/sysconfig/network-scripts:
ifcfg-eth0  ifdown-ipv6  ifdown-tunnel  ifup-ipv6  ifup-routes  network-functions
ifcfg-lo  ifdown-isdn  ifup    ifup-ipx  ifup-sit  network-functions-ipv6
ifdown       ifdown-post    ifup-aliases   ifup-isdn   ifup-sl
ifdown-bnep   ifdown-ppp     ifup-bnep     ifup-plip   ifup-tunnel
ifdown-eth    ifdown-routes  ifup-eth      ifup-plusb  ifup-wireless
ifdown-ippp   ifdown-sit     ifup-ippp     ifup-post   init.ipv6-global
ifdown-ipsec  ifdown-sl      ifup-ipsec    ifup-ppp    net.hotplug

/etc/sysconfig/rhn:
sources
```

联合使用-ltr 选项，将以长列表格式按文件或目录的修改时间倒序地列出文件和目录：

```
$ ls -ltr
total 51880
drwxr-xr-x 2 root root       4096 Jul 12  2007 conman.old
drwxr-xr-x 2 root root       4096 Jul 12  2007 conman
drwx------ 2 root root       4096 Jan 22  2009 ppp
drwx------ 2 root root       4096 Aug 31  2010 httpd
drwxr-xr-x 2 root root       4096 Oct  7  2012 mail
drwxr-xr-x 2 root root       4096 Oct  7  2012 pm
-rw------- 1 root root      24902 Oct  7  2012 anaconda.syslog
-rw------- 1 root root     381429 Oct  7  2012 anaconda.log
-rw------- 1 root root          0 Oct  7  2012 tallylog
drwx------ 5 root root       4096 Oct  7  2012 libvirt
```

联合使用-ls 选项，将以长列表格式按文件大小顺序列出文件和目录：

```
$ ls -ls
total 51884
-rw-r--r-- 1 root root   115383508 Aug 21 10:04 lastlog
-rw------- 1 root root     9560095 Jul 28 04:02 maillog.4
-rw------- 1 root root     9557915 Aug 18 04:02 maillog.1
-rw------- 1 root root     8423447 Aug 11 04:02 maillog.2
-rw------- 1 root root     7380013 Aug  4 04:02 maillog.3
-rw------- 1 root root     4464058 Aug 21 10:28 maillog
-rw------- 1 root root     1413012 Aug 11 04:00 messages.2
-rw------- 1 root root     1403975 Aug 18 04:00 messages.1
-rw------- 1 root root     1402187 Aug  4 03:58 messages.3
-rw------- 1 root root     1388015 Jul 28 04:00 messages.4
```

使用-a 选项，将列出包括隐藏文件或目录在内的所有文件和目录，包括“.”（当前目录）和“..”（父目录）：

```
$ ls -a
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc
```

使用-A 选项，将列出包括隐藏文件或目录（不包含“.”和“..”）在内的所有文件和目录：

```
$ ls -A
.bash_history  .bash_logout  .bash_profile  .bashrc
```

使用-i 选项，将显示文件或目录的 inode 编号，有时在系统维护操作时，你可能想知道文件的 inode 编号：

```
$ ls -i /etc/
1279950 a2ps.cfg                                    1278405 man.config
1280419 a2ps-site.cfg                                  1343677 maven
1279260 acpi                                           1311078 mc
1278826 adjtime                                   1278112 mgetty+sendfax
```

注意：在 find 命令中，你可以使用 inode 编号移除文件名中含有特殊字符的文件。

使用-n 选项，其输出的内容类似于-l 选项，指示显示 uid 和 gid，替代显示所有者和用户组：

```
$ ls -n
total 51964
-rw-r----- 1 0  0       21630 Aug 19 00:01 acpid
-rw------- 1 0  0      381429 Oct  7  2012 anaconda.log
-rw------- 1 0  0      24902 Oct  7  2012 anaconda.syslog
drwxr-x--- 2 0  0       4096 Aug 18 22:26 audit
-rw------- 1 0  0        0 Aug 18 04:02 boot.log
-rw------- 1 0  0        0 Aug 11 04:02 boot.log.1
-rw------- 1 0  0        0 Aug  4 04:02 boot.log.2
-rw------- 1 0  0        0 Jul 28 04:02 boot.log.3
-rw------- 1 0  0        0 Jul 21 04:02 boot.log.4
```

3.1.2　cat 命令实例：连接显示文件内容

cat 命令也是 Linux 系统中最常使用的命令之一。cat 命令让我们可以查看文件的内容、连接文件、创建一个或多个文件和重定向输出到终端或文件。

cat 命令的语法如下所示：

```
$ cat [OPTION] [FILE]…
```

使用 cat 命令查看文件/etc/group 的内容：

```
$ cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
```

显示多个文件的内容：

```
$ cat /etc/redhat-release /etc/issue
Scientific Linux SL release 5.5 (Boron)
Scientific Linux SL release 5.5 (Boron)
Kernel \r on an \m
```

使用-n 选项，可以显示文件内容的行号：

```
$ cat -n /etc/fstab
    1  LABEL=/               /                  ext3    defaults        1 1
    2  LABEL=/local          /local             ext3    defaults        1 2
    3  tmpfs                 /dev/shm           tmpfs   defaults        0 0
```

```
    4  devpts           /dev/pts              devpts  gid=5,mode=620  0 0
    5  sysfs            /sys                 sysfs    defaults        0 0
    6  proc             /proc                proc     defaults        0 0
    7  LABEL=SWAP-hda2     swap                 swap    defaults        0 0
```

-b 选项和-n 选项类似，但只标识非空白行的行号。

使用-e 选项，将在每一行的结尾显示“$”字符。这个选项在需要将多行内容转换成一行时是很有用的。

```
$ cat -e /etc/fstab
LABEL=/               /                    ext3     defaults          1 1$
LABEL=/local           /local              ext3     defaults          1 2$
tmpfs                /dev/shm             tmpfs    defaults          0 0$
devpts               /dev/pts             devpts   gid=5,mode=620    0 0$
sysfs                /sys                 sysfs    defaults          0 0$
proc                 /proc                proc     defaults          0 0$
LABEL=SWAP-hda2        swap                 swap    defaults          0 0$
```

当你只输入 cat 命令，而没有任何参数时，它只是接收标准输入的内容并在标准输出中显示（关于标准输入输出的内容，将在 11.1 节中介绍）。所以在你输入一行内容并回车后，会在接下来的一行显示相同的内容。你也可以重定向（重定向的内容将在第 11 章做详细介绍）标准输出到一个新文件，类似如下所示：

```
$ cat >test
```

这时会等待用户的输入，输入需要的内容后，按 Ctrl+D 组合键退出。输入的内容将会写入到文件 test 中。

```
$ cat >test
hello everyone!

$ cat test
hello everyone!
```

使用 cat 命令并利用重定向，还可以连接多个文件的内容到一个新文件，如下所示：

```
$ cat test
hello everyone!

$ cat test1
hello world!

$ cat test test1 > test2

$ cat test2
hello everyone!
hello world!
```

注意：Linux 下还有一个 tac 命令，从命令的名字你可能已经想到，tac 命令与 cat 命令相反，它将以倒序的形式（先显示文件的最后一行）显示文件的内容。

3.1.3　less、more 命令实例：分屏显示文件

more 命令在你使用小的 xterm 窗口时，或是想不使用文本编辑器而只是简单地阅读一

个文件时是很有用的。more 命令是一个用于一次翻阅一整屏文件的过滤器。

使用 more 命令查看一个文件：

```
$ more /etc/inittab
```

它会自动地清空屏幕并显示文件的开始部分：

```
#
# inittab      This file describes how the INIT process should set up
#              the system in a certain run-level.
#
# Author:      Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#              Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
```

如果你按空格键，more 会将文件向下移动一个你当前终端窗口的高度，来为你显示下一页的内容。

使用-num（num 是一个整数）选项，可以指定一次显示的行数：

```
$ more -5 /etc/inittab
#
# inittab      This file describes how the INIT process should set up
#              the system in a certain run-level.
#
# Author:      Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
--More--(11%)
```

你也可以通过管道流将 cat 命令显示的内容输出到 more 命令。比如，有时你想输出一个文件的全部内容，但要慢慢地查看它：

```
$ cat README | more
```

与 more 命令相比，我个人更喜欢使用 less 命令查看文件。less 命令与 more 命令类似，但 less 命令向前和向后翻页都支持，而且 less 命令不需要在查看前加载整个文件，即 less 命令查看文件更快速。当你尝试使用 Vim 编辑器和 less 打开同一个大的 log 文件，你会发现速度是不同的。

使用 less 命令查看一个文件：

```
$ less /etc/fstab
LABEL=/                 /                       ext3    defaults          1 1
LABEL=/local            /local                  ext3    defaults          1 2
tmpfs                   /dev/shm                tmpfs   defaults          0 0
devpts                  /dev/pts                devpts  gid=5,mode=620    0 0
sysfs                   /sys                    sysfs   defaults          0 0
proc                    /proc                   proc    defaults          0 0
LABEL=SWAP-hda2         swap                    swap    defaults          0 0
/etc/fstab (END)
```

当你用 less 命令打开了一个文件后，你可以使用搜索功能，搜索指定的关键字。默认情况下，所有匹配的关键字将会自动地高亮显示。

向前搜索：

- /——在 less 命令打开的文件中，输入字符“/”后跟要搜索的关键字，然后输入回车，显示内容的第一行将自动跳转到关键字第一次出现的位置，并高亮显示所有搜索到的关键字。
- n——输入字母“n”，显示内容的第一行将向前跳转到下一个匹配。
- N——输入字母“N”，显示内容的第一行将向回跳转到前一个匹配。

向后搜索：

- ?——与字符“/”的功能相反，在问号“?”后输入要搜索的关键字，然后输入回车，将向回搜索关键字。
- n——向回搜索下一个匹配。
- N——向前搜索下一个匹配。

在使用 less 命令浏览较大的文件时，可以使用如下屏幕导航命令：

- Ctrl+F——向前翻一个窗口的内容。
- Ctrl+B——向回翻一个窗口的内容。
- Ctrl+D——向前翻半个窗口的内容。
- Ctrl+U——向回翻半个窗口的内容。
- G——跳转到文件的末尾。
- G——跳转到文件的开头。
- q or ZZ——退出 less。

你可以使用 less 命令打开多个文件：

```
$ less file1
```

当你查看文件 file1 时，使用“:e”可以打开第二个文件 file2：

```
$ less file1
#some lines
```

这时你输入“:e”，内容将变为：

```
#some lines
Examine:
```

然后你输入文件名 file2：

```
#some lines
Examine: file2
```

然后输入回车：

```
#this is file2
#some lines
file2 (file 2 of 2) (END)
```

当你使用 less 命令打开了两个以上文件时，可以使用如下的关键字切换文件。

- :n：跳转到下一个文件。
- :p：跳转到前一个文件。

less 命令允许你在文件的特定位置做一个标记，当需要时，可以使用这个标记再次返回到标记的位置。

- m：后跟任意小写字母，使用这个字母标记当前位置。
- '（单引号）：后跟任意小写字母，返回到这个小写字母标记的位置。

使用 less 查看一个文件，并输入“m”，将在窗口底部看到类似如下内容：

```
#some lines
mark:
```

此时输入一个小写字母将在当前位置做一个标记，当想再次回到此标记位置时，输入单引号字符“'”，将在窗口底部看到类似如下内容：

```
#some lines
goto mark:
```

此时输入我们刚才用做标记的字母，窗口的内容将跳转到我们之前标记的位置。

一旦你已经使用 less 命令打开了一个文件，在此之后添加到此文件的内容将不会自动显示出来。然而，你可以在 less 中输入大写字母“F”显示新写入的内容。输入大写字母“F”，在窗口的底部会显示类似如下内容：

```
#some lines
Waiting for data... (interrupt to abort)
```

3.1.4　head 命令实例：显示文件头部

head 命令用于打印指定输入的开头部分内容。默认情况下，打印每个指定输入的前 10 行内容。

使用-n 选项可以指定打印文件的前 N 行，如下所示：

```
$ head -n 5 /etc/inittab
#
# inittab      This file describes how the INIT process should set up
#             the system in a certain run-level.
#
# Author:      Miquel van Smoorenburg, miquels@drinkel.nl.mugnet.org
```

你也可以不使用-n 选项，而只是简单地在连字符“-”后跟一个正整数来指定要打印的行数，类似如下所示：

```
$ head -5 /etc/inittab
#
# inittab      This file describes how the INIT process should set up
#             the system in a certain run-level.
#
# Author:      Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
```

使用-n 选项时，如果在正整数参数 N 前加一个连字符“-”，将打印输出除最后 N 行以外的所有行。类似如下所示：

```
$ cat -n /proc/meminfo
    1  MemTotal:     3850316 kB
    2  MemFree:      2874464 kB
    3  Buffers:       180964 kB
```

```
    4  Cached:        674652 kB
    5  SwapCached:         0 kB
    6  Active:        233200 kB
    7  Inactive:      652108 kB
    8  HighTotal:    2973316 kB
    9  HighFree:     2256976 kB
   10  LowTotal:      877000 kB
   11  LowFree:       617488 kB
   12  SwapTotal:    4192956 kB
   13  SwapFree:     4192956 kB
   14  Dirty:            292 kB
   15  Writeback:          0 kB
   16  AnonPages:      29684 kB
   17  Mapped:         19136 kB
   18  Slab:           75016 kB
   19  PageTables:      3132 kB
   20  NFS_Unstable:       0 kB
   21  Bounce:             0 kB
   22  CommitLimit:   6118112 kB
   23  Committed_AS:   339852 kB
   24  VmallocTotal:   116728 kB
   25  VmallocUsed:    11856 kB
   26  VmallocChunk:   100612 kB
   27  HugePages_Total:     0
   28  HugePages_Free:      0
   29  HugePages_Rsvd:      0
   30  Hugepagesize:     2048 kB
$ head -n -20 /proc/meminfo
MemTotal:     3850316 kB
MemFree:      2874340 kB
Buffers:       180968 kB
Cached:        674652 kB
SwapCached:         0 kB
Active:        233204 kB
Inactive:      652108 kB
HighTotal:    2973316 kB
HighFree:     2256852 kB
LowTotal:      877000 kB
```

你也可以使用-c 选项打印文件的前 N 个字节的数据，类似如下所示：

```
$ head -c 10 /etc/inittab
#
# initta$
```

注意：-c 选项和选项-n 类似，在正整数 N 前加上连字符“-”，将打印除最后 N 字节以外的全部字节。

3.1.5　tail 命令实例：显示文件尾部

tail 命令与 head 命令相反，它打印指定输入的结尾部分的内容。默认情况下，它打印指定输入的最后 10 行内容。

使用-n 选项可以指定打印文件的最后 N 行，如下所示：

```
$ tail -n 10 /etc/inittab
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
```

```
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon
```

使用-f 选项可以即时打印文件中新写入的行。此命令类似如下所示：

```
$ tail -f /var/log/messages
```

注意：使用这个选项对于监控日志文件是非常有用的。

--pid 选项和-f 选项同时使用，可以在特定的进程结束时终结 tail 命令，命令类似如下所示：

```
$ tail -f /tmp/debug.log --pid= 24184
```

有时，你有意想使用 tail 命令打开一个稍后才会创建或即使不可用的文件。这时，你可以使用--retry 选项持续尝试打开这个文件。此命令类似如下所示：

```
$ tail -f  /tmp/debug.log --retry
tail: warning: --retry is useful only when following by name
tail: cannot open '/tmp/debug.log' for reading: No such file or directory
```

如果不加--retry 选项，输出将类似如下所示：

```
$ tail -f  /tmp/debug.log
tail: cannot open '/tmp/debug.log' for reading: No such file or directory
tail: no files remaining
$
```

3.1.6　file 命令实例：查看文件类型

file 命令用于接收一个文件作为参数并执行某些测试，以确定正确的文件类型。

下面这个例子是使用 file 命令确定一个文件类型的基本方法：

```
$ file /etc/inittab
/etc/inittab: ASCII English text

$ file /etc/init.d/network
/etc/init.d/network: Bourne-Again shell script text executable

$ file /usr/bin/file
/usr/bin/file: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.6.9, dynamically linked (uses shared libs), for GNU/Linux
2.6.9, stripped

$ file /etc
/etc: directory
```

使用-i 选项，可以 MIME 类型的格式显示文件类型的信息：

```
$ file -i /etc/inittab
/etc/inittab: text/plain; charset=us-ascii
```

```
$ file -i /etc/init.d/network
/etc/init.d/network: application/x-shellscript

$ file -i /usr/bin/file
/usr/bin/file: application/x-executable, for GNU/Linux 2.6.9, dynamically
linked (uses shared libs), for GNU/Linux 2.6.9, stripped

$ file -i /etc
/etc: application/x-not-regular-file
```

使用-N 选项，输出的队列可以以在文件名之后无空白填充的形式显示，其格式对比如下：

```
$ file -N *
a2ps.cfg: ASCII English text
a2ps-site.cfg: ASCII English text
acpi: directory
adjtime: ASCII text
aliases: ASCII English text
aliases.db: writable, regular file, no read permission
aliases.ORIG: ASCII English text
alsa: directory
alternatives: directory
anacrontab: ASCII text
ant.conf: ASCII text
ant.d: directory
anthy-conf: ASCII text
asound.state: ASCII text, with very long lines
at.deny: writable, regular file, no read permission
audisp: directory
audit: directory
auto.agl: ASCII text
autofs_ldap_auth.conf: writable, regular file, no read permission

$ file *
a2ps.cfg:                                          ASCII English text
a2ps-site.cfg:                                      ASCII English text
acpi:                                              directory
adjtime:                                           ASCII text
aliases:                                           ASCII English text
aliases.db:                                        writable,regular file,
no read permission
aliases.ORIG:                                       ASCII English text
alsa:                                              directory
alternatives:                                       directory
anacrontab:                                        ASCII text
ant.conf:                                          ASCII text
ant.d:                                             directory
anthy-conf:                                        ASCII text
asound.state:                                       ASCII text, with very
long lines
at.deny:                                           writable, regular file,
no read permission
audisp:                                            directory
audit:                                             directory
auto.agl:                                          ASCII text
autofs_ldap_auth.conf:                                  writable, regular
file, no read permission
```

3.1.7　wc 命令实例：查看文件统计信息

wc 命令用于查看文件的行数、单词数和字符数等信息。其语法类似如下所示：

```
$ wc filename
  X  Y Z /etc/inittab
```

X：表示行数。
Y：表示单词数。
Z：表示字节数。
filename：表示文件名。

```
$ wc /etc/inittab
  53  229 1666 /etc/inittab
```

上述输出表示/etc/inittab 文件中，有 53 行、229 个单词和 1666 个字节。
使用-l 选项，可以只统计文件的行数信息，如下所示：

```
$ wc -l /etc/inittab
53 /etc/inittab
```

使用-w 选项，可以只统计文件的单词数信息，如下所示：

```
$ wc -w /etc/inittab
229 /etc/inittab
```

使用-c 选项，可以只统计文件的字节数信息，如下所示：

```
$ wc -c /etc/inittab
1666 /etc/inittab
```

使用-L 选项，可以统计文件中最长的行的长度，如下所示：

```
$ wc -L /etc/inittab
77 /etc/inittab
```

3.1.8　find 命令实例：查找文件或目录

find 命令是 Linux 系统中最重要也是最常用的命令之一。find 命令用于根据你指定的参数搜索和定位文件和目录的列表。find 命令可以在多种情况下使用，比如你可以通过权限、用户、用户组、文件类型、日期、大小和其他可能的条件来查找文件。

简单地使用 find 命令查找指定目录下的某个文件的方法如下所示：

```
# find /etc -name inittab
/etc/inittab
```

注意：此处的命令行提示符是“#”号，表示当前用户账号是 root。

在当前目录下，查找名称为 inittab 的文件：

```
# find . -name inittab
./inittab
```

找出当前目录下，文件名不区分大小写是 example 的所有文件：

```
$ find . -iname example
./example
./Example
```

找出当前目录下，目录名是 tmp 的目录：

```
$ find . -type d -name tmp
./tmp
```

找出当前目录下所有 php 文件：

```
$ find . -type f -name "*.sh"
./login.php
./index.php
```

找出当前目录下，文件权限是 777 的所有文件：

```
$ find . -type f -perm 0777
```

找出当前目录下，文件权限不是 777 的所有文件：

```
$ find . -type f ! -perm 777
```

找出/etc/目录下所有只读文件：

```
# find . -type f ! -perm /a+w
```

找出你账号的主目录下的所有可执行文件：

```
$ find ~ -type f -perm /a+x
```

找出/tmp 目录下的.log 文件并将其删除：

```
$ find /tmp/ -type f -name "*.log" -exec rm -f {} \;
```

找出当前目录下的所有空文件：

```
$ find . -type f -empty
```

找出当前目录下的所有空目录：

```
$ find . -type d -empty
```

找出/tmp 目录下的所有隐藏文件：

```
$ find /tmp/ -type f -name ".*"
```

找出/tmp 目录下，所有者是 root 的文件和目录：

```
$ find /tmp/ -user root
```

找出/tmp 目录下，用户组是 developer 的文件和目录：

```
$ find /tmp/ -group developer
```

找出你账号的主目录下，3 天前修改的文件：

```
$ find ~ -type f -mtime 3
```

找出你账号的主目录下，30 天以前修改的所有文件：

```
$ find ~ -type f -mtime +30
```

找出你账号的主目录下，3 天以内修改的所有文件：

```
$ find ~ -type f -mtime -3
```

找出你账号的主目录下，30 天以前 60 天以内修改的所有文件：

```
$ find ~ -type f -mtime +30 -mtime -60
```

找出/etc 目录下，一小时以内变更过的文件：

```
# find /etc -type f -cmin -60
```

找出/etc 目录下，一小时以内访问过的文件：

```
# find /etc -type f -amin -60
```

找出你账号的主目录下，大小是 50MB 的所有文件：

```
$ find ~ -type f -size 50MB
```

找出你账号的主目录下，大于 50MB 小于 100MB 的所有文件：

```
$ find ~ -type f -size +50MB -size -100MB
```

找出你账号的主目录下，大于 100MB 的文件并将其删除：

```
$ find ~ -type f -size +100MB -exec rm -rf {} \;
```

3.2　操作文件和目录

3.2.1　touch 命令实例：创建文件

在 Linux 中，每个文件都关联一个时间戳，并且每个文件都会存储最近一次访问的时间、最近一次修改的时间和最近一次变更的时间等信息。所以，无论何时我们创建一个新文件，访问或修改一个已存在的文件，文件的时间戳都会自动地更新。

touch 命令就可用于创建、变更和修改文件的时间戳。它是 Linux 操作系统的标准程序。touch 命令有如下选项。

- -a：只改变访问时间。
- -c：不创建任何文件。
- -m：只改变修改时间。
- -r：使用指定文件的时间替代当前时间。
- -t：使用 [[CC]YY]MMDDhhmm[.ss] 替代当前时间。

使用 touch 命令创建一个名称是 effyl 的新的空文件（0 字节）：

```
$ touch effyl
```

使用 touch 命令，你同样可以创建多个文件。例如，如下命令将创建名称分别为 sheffyl、myeffyl 和 lueffyl 的三个文件：

```
$ touch sheffyl myeffyl lueffyl
```

使用-a 选项，可以改变或更新文件的最新访问时间。如下命令更新文件的访问时间，如果文件 effyl 不存在，它将创建一个以 effyl 命名的新的空文件：

```
$ touch -a effyl
```

使用-c 选项，可以避免创建一个新文件，并用当前时间更新文件的时间戳：

```
$ touch -c effyl
```

使用-m 选项，可以只改变文件的修改时间，而访问时间不变：

```
$ touch -m effyl
```

你可以同时使用-c 和-t 选项，来明确设置文件的时间，命令格式如下所示：

```
$ touch -c -t YYMMDDHHMM filename
```

例如，我们将文件 effyl 的修改时间和访问时间设置为 12 年 12 月 13 日 10 点 30 分：

```
$ touch -c -t 1212131030 effyl
```

如果想使用文件 myeffyl 的时间戳更新文件 effyl 的时间戳，那么可以使用-r 选项：

```
$ touch -r myeffyl effyl
```

3.2.2　mkdir 命令实例：创建目录

mkdir 命令用于创建一个新目录。最基本的 mkdir 命令的使用方法如下所示：

```
$ mkdir <dirname>
```

上述命令将用给定的目录名在当前目录下创建一个目录。

你并不一定要在一个目录下来创建这个目录的新目录。你可以使用相对或绝对路径来创建它们。我们假设你在你账号的主目录，并且刚创建了一个目录 backup（/home/yantaol/backup）。现在你想在 backup 中创建一个名为 old 的目录，那么你既可以使用相对路径，类似如下所示：

```
$ mkdir backup/old
```

也可以使用绝对路径，类似如下所示：

```
$ mkdir /home/yantaol/backup/old
```

使用-p 选项，mkdir 命令会自动创建所有还不存在的父目录，比如我想在我账号的主目录的 backup 目录下创建一个 old 目录，但是 backup 目录还不存在，就可以使用如下命令来创建 old 目录：

```
$ mkdir -p backup/old
```

或

```
$ mkdir -p /home/yantaol/backup/old
```

上述命令会自动创建目录 backup。如果在上述情况中，不使用-p 选项，将会出现类似如下错误：

```
$ mkdir backup/old
mkdir: cannot create directory 'backup/old': No such file or directory
```

注意：当你需要递归地创建目录时，使用-p 选项将是一个很简便的方法。

使用-p 选项，同样可以在要创建的目录已经存在的情况下，阻止错误的发生。比如，你要创建目录 backup，但这个目录已经存在，系统将会显示类似如下错误信息：

```
$ mkdir backup
mkdir: cannot create directory 'backup': File exists
```

但如果你使用-p 选项，将阻止错误信息的输出，类似如下所示：

```
$ mkdir -p backup
$
```

使用-m 选项，你可以设置你将要创建的目录的权限。例如，你想创建一个任何人都有读写访问权限的目录：

```
$ mkdir -p -m 777 backup/old
```

或

```
$ mkdir -p -m a=rwx backup/old
```

3.2.3　cp 命令实例：复制文件或目录

在 Linux 下，我们可以使用 cp 命令复制文件和目录。cp 命令用于将文件从一个地方复制到另一个地方。原来的文件保持不变，新文件可能保持原名或用一个不同的名字。

使用 cp 命令复制文件和目录的语法有以下几种：

```
$ cp [OPTION] SOURCE DEST               #复制源文件到目标文件
$ cp [OPTION] SOURCE… DIRECTORY         #复制一个或多个源文件到一个目录
$ cp [OPTION] -t DIRECTORY SOURCE… #同上
```

例如，在当前目录下，创建一个文件 file.txt 的副本，取名为 newfile.txt。如下所示：

```
$ cp file.txt newfile.txt
```

复制当前目录下的文件 file.txt 到/tmp 目录下：

```
$ cp file.txt /tmp/
```

复制当前目录下的所有文件到/tmp 目录：

```
$ cp * /tmp
```

使用-p 选项，可以使复制一个文件到新文件时，保留源文件的所有者、用户组、权限、修改和访问时间，以及一些扩展属性等信息：

```
$ cp -p filename /path/to/new/location/myfile
```

使用-R 或-r 选项，可以递归地复制一个目录，即将一个目录及其下的所有文件和子目录都复制到另一个目录：

```
$ cp -R * /home/yantaol/backup
```

还有比较常用的归档模式复制：

```
$ cp -a * /home/yantaol/backup
```

- -a：存档模式。相当于-dpR。
- -d：保留软链接。
- -p：保留权限、所有权和时间戳等信息。
- -R：递归地复制目录。

3.2.4　ln 命令实例：链接文件或目录

ln 命令用于创建软链接或硬链接。我们在 3.1.1 小节介绍 ls 命令时，讲述了 ls-l 命令列出的每一条目的第一个字符指示文件类型，当一个字符是“l”（小写的 L）时，即表示它是一个软链接。

- 软链接又称符号链接，是一类特殊的文件，这个文件包含了另一个文件或目录的路径名（绝对路径或相对路径）。在对符号文件进程读或写操作时，系统会自动把该操作转换为对源文件或目录的操作，但输出链接文件时，系统仅仅删除链接文件，而不删除源文件或目录本身。软链接可以链接不同文件系统的文件。
- 硬链接可以理解为一个文件的一个或多个文件名。它引用的是文件在文件系统中的物理索引（也称为 inode）。当你移动或删除原始文件时，硬链接不会被破坏，因为它所引用的是文件的物理数据而不是文件在文件结构中的位置。硬链接的文件不需要用户有访问原始文件的权限，也不会显示原始文件的位置，这样有助于文件的安全。如果你删除的文件有相应的硬链接，那么这个文件依然会被保留，直到所有对它的引用都被删除，即硬链接数为 0。硬链接只能链接同一文件系统中的文件。

使用-s 选项，可以创建一个软链接：

```
$ ln -s /full/path/of/original/file /full/path/of/symbolic/link/file
```

ln 命令不使用任何选项，默认将创建一个硬链接：

```
$ ln -s /full/path/of/original/file /full/path/of/hard/link/file
```

在目录/home/yantaol/lib 下创建一个软链接 library.so，链接到/home/yantaol/src/library.so：

```
$ ln -s /home/yantaol/src/library.so /home/yantaol/lib

$ ls -l /home/yantaol/lib/ library.so
lrwxrwxrwx 1 yantaol yantaol 29 Sep  4 19:27 /home/yantaol/lib/library.so
-> /home/yantaobl/src/library.so
```

创建目录的软链接和创建文件的软链接类似：

```
$ mkdir /home/yantaol/src

$ cd /tmp
```

```
$ ln -s /home/yantaol/src source

$ ls -l source
lrwxrwxrwx 1 yantaol yantaol 18 Sep  4 19:38 source -> /home/yantaol/src
```

在当前目录下，创建文件 src_original.txt 的硬链接，名称为 dst_link.txt。两个文件的 inode 编码应该相同：

```
$ ln src_original.txt dst_link.txt

$ ls -i src_original.txt
1638423 src_original.txt

$ ls -i dst_link.txt
1638423 dst_link.txt
```

注意：Linux 不允许给目录创建硬链接。

当你创建一个软链接时，如果已经存在一个与此软链接同名的文件，那么你可以使用 --backup 选项，让 ln 命令在创建这个新链接之前，先备份已经存在的同名文件：

```
$ ln -s source.txt dst.txt
ln: creating symbolic link 'dst.txt' to 'source.txt': File exists

$ ln --backup -s source.txt dst.txt

$ ls -l dst.txt*
lrwxrwxrwx 1 yantaobl cc_rdr_tdd_03 10 Sep  4 19:48 dst.txt -> source.txt
-rw-r--r-- 1 yantaobl cc_rdr_tdd_03  0 Sep  4 19:48 dst.txt~
```

注意：如果你不想备份而是覆盖已经存在的文件，则使用-f 选项。

3.2.5　mv 命令实例：重命名文件或目录

mv 命令用于将文件和目录从一个位置移到另外一个位置。除了移动文件，mv 命令还可用于修改文件或目录的名字。

mv 命令的基本语法如下所示：

```
$ mv SOURCE... DIRECTORY
```

比如，将当前目录下的文件 source.txt 移到目录/tmp 下：

```
$ mv source.txt /tmp
```

或将目录 dir1、dir2 移到目录 dir_dist 下：

```
$ mv dir1 dir2 dir_dist
```

使用 mv 命令将当前目录下的文件 old.txt 更名为 new.txt：

```
$ mv old.txt new.txt
```

或者将目录 oldDir 更名为 newDir：

```
$ mv oldDir newDir
```

默认情况下，如果目标文件或目录已存在，mv 命令并不会提示你任何信息，而是直接将其重写覆盖。为了避免这个问题，可以使用-i 选项，让 mv 命令在重写覆盖目标文件或目录之前给出提示信息。这样你可以通过输入字符“y”或“n”来接受或拒绝此操作。

比如我准备将文件 old.txt 更名为 new.txt，但是文件 new.txt 已经存在，在使用-i 选项时将看到类似如下结果：

```
$ mv -i old.txt new.txt
mv: overwrite 'new.txt'?
```

使用 mv 命令，也可以同时移动多个文件或目录。比如，我想将当前目录下的所有文件移到目录/tmp 下：

```
$ mv * /tmp/
```

然而，如果想只从源目录中移动那些在目标目录中不存在的文件到目标目录，可以使用-u 选项。下面的实例，就是只将目录 dir1 中的文件 file2 和 file3 移到目录 dir2，因为文件 file1 已经存在于目录 dir2 中了：

```
$ ls dir1/
file1  file2  file3

$ ls dir2/
file1

$ mv -u dir1/* dir2/

$ ls dir1/
file1

$ ls dir2/
file1  file2  file3
```

3.2.6　rm 命令实例：删除文件或目录

rm 命令用于删除指定的文件和目录。其语法如下所示：

```
$ rm [OPTIONS]… FILE…
```

在下面这个实例中，我们使用 rm 命令删除文件 file1.txt、file2.txt 和 file3.txt，假设这 3 个文件都在当前目录下：

```
$ rm file1.txt file2.txt file3.txt
```

注意：接下来的这些实例，在执行后可能会导致你的系统崩溃或数据丢失，所以请谨慎运行这些实例，并清楚你在做什么。

星号“*”可以表示所有文件名，所以使用星号“*”可以删除所有文件。比如，删除当前目录下的所有文件：

```
$ rm *
```

删除你当前账号主目录下的 temp 目录中的所有文件：

```
$ rm ~/temp/*
```

使用-i 选项，可以让 rm 命令在删除每一个文件和目录前提示用户确认：

```
$ rm -i *
```

删除当前目录下所有以“.doc”为后缀的文件：

```
$ rm *.doc
```

删除当前目录下所有文件名中包含“movie”字符串的文件：

```
$ rm *movie*
```

删除当前目录下所有以小写字母“a”开头的文件：

```
$ rm a*
```

问号“?”用于匹配一个字符，比如，“???”可以表示任何只含有 3 个字符的文件名。下面的例子，就是用于删除当前目录下整个文件名（包括扩展名）只有 3 个字符的所有文件：

```
$ rm ???
```

删除当前目录下文件扩展名有两个字符的所有文件：

```
$ rm *.??
```

方括号“[]”可以用于匹配括号内的任意一个字符。例如，如下语句将使用 rm 命令删除当前目录下文件名中含有字母 a，或字母 b，或字母 c 的所有文件：

```
$ rm * [abc]*
```

删除当前目录下文件名中包含数字 0～9 的所有文件。即，文件名中至少有一个数字的文件：

```
$ rm * [0-9]*
```

删除当前目录下文件扩展名是字母 c 或 h 的所有文件：

```
$ rm*.[ch]
```

删除/tmp 目录下的所有文件及其子目录：

```
$ rm -rf /tmp/*
```

- -f：删除前不提示用户确认，并忽略不存在的文件。
- -r（或 -R）：递归地删除目录及其下的内容。

3.3　管理文件或目录权限

3.3.1　ls -l：显示文件和目录权限

在 3.1.1 小节中，我们学习了 ls -l 命令的输出，已经知道了它的输出结果中第一列的

第一个字符表示文件的类型（目录、文件或链接），从第 2～10 这 9 个字符即指示文件的 3 种用户类型的权限。

```
-rwxr-xr-x  1 yantaol yantaol      759 Jun  4 09:53 test.sh
```

如上例所示，每个文件或目录都有 3 个用户权限组。

- 所有者权限（第一组的 3 个字符 rwxr-xr-x）：只应用于文件或目录的所有者，它们将不影响其他用户的行为。
- 用户组权限（第二组的 3 个字符 rwxr-xr-x）：只应用于已经指定给文件或目录的组，它们将同样不影响其他用户的行为。
- 其他用户权限（第三组的 3 个字符-rwxr-xr-x）：只应用于系统中的所有其他用户。

每个文件或目录还有三个基本的权限类型：

- r：读权限。指用户的读文件内容或列出目录内容的权限。
- w：写权限。指用户的写或修改文件或目录的权限。
- x：执行权限。指用户执行文件或进入目录的权限。

我们来看一个例子：

```
$ ls -dl /etc
drwxr-xr-x 121 root root 20480 Sep 10 08:35 /etc
```

目录/etc 的权限是 rwxr-xr-x。该目录的所有者是 root，用户组也同样是 root。所有者权限是 rwx，这些权限允许用户 root 对该目录的读、写和执行访问。

用户组权限是 r-x。这里没有给写权限，所以用户组 root 的成员只能查看和列出目录中的内容。他们不能在该目录中创建文件或子目录，同样不能删除任何文件或对目录内容做改动。

其他用户权限同样是 r-x。其他用户也只能查看和列出目录中的内容。

我们假设如果你看到一个目录的权限是 drw-r--r--，那么该目录的所有者可以列出和修改该目录的内容，但是不能进入到这个目录，因为没有执行权限。需要进入目录并列出目录中的内容必须有读和执行（r-x）的权限。用户组的成员和其他用户也会有同样的问题，他们可以列出目录的内容，但是不能进入目录，因为没有执行（x）的权限。

我们再来看几个权限的例子。

- -r--r--r--：这个表示所有者、用户组成员和其他用户对文件都只有读权限。
- -rw-rw-rw：这个表示所有者、用户组成员和其他用户对文件有读和写的权限。
- -rwxrwxrwx：这个表示所有者、用户组成员和其他用户有所有权限，他们全部可以读、写和执行此文件。

3.3.2　chmod 命令实例：修改权限

chmod 命令用于修改文件或目录的权限。chmod 命令根据相应的模式修改每个给定文件的权限。这里的模式有两种：一种是符号表达式模式，另一种是八进制位模式。

使用符号表达式模式的格式如下所示：

```
$ chmod [OPTION]…[ugoa][[+-=][rwxug]][,...] FILE…
```

字母“ugoa”的组合控制哪些用户对文件的访问权限将被改变。

- u：指文件或目录的所有者。
- g：指文件或目录的用户组的成员。
- o：指不在文件或目录的用户组中的其他用户。
- a：指所有用户，即（ugo）。

注意：如果使用 chmod 命令的符号表达式模式时，不给出“ugoa”的组合，则得到的结果和使用“a”相同。

操作符“+-=”表示权限的授予或撤销。

- +：选定的权限将被添加。
- -：选定的权限将被移除。
- =：文件只拥有选定的权限。

下面我们来看几个使用符号表达式模式的 chmod 命令实例：

移除用户组成员的写权限：

```
$ ls -l example.sh
-rwxrwxr-- 1 yantaol yantaol 0 Sep 10 18:40 example.sh

$ chmod g-w example.sh

$ ls -l example.sh
-rwxr-xr-- 1 yantaol yantaol 0 Sep 10 18:40 example.sh
```

赋予其他用户执行文件的权限：

```
$ chmod o+x example.sh

$ ls -l example.sh
-rwxr-xr-x 1 yantaol yantaol 0 Sep 10 18:40 example.sh
```

只给文件的所有者写权限：

```
$ chmod u=w example.sh

$ ls -l example.sh
--w-r-xr-x 1 yantaol yantaol 0 Sep 10 18:40 example.sh
```

用文件的用户组权限替换文件的所有者权限：

```
$ chmod u=g example.sh

$ ls -l example.sh
-r-xr-xr-x 1 yantaol yantaol 0 Sep 10 18:40 example.sh
```

赋予所有人对文件读、写和执行的权限：

```
$ chmod ugo+rwx example.sh
```

chmod 命令的数字模式是用数字来表示读（4）、写（2）和执行（1）的权限，而每个用户权限组的值就是表示读、写和执行权限的这 3 个数字（4、2、1）组合的相加得到的八进制数（0～7）。各数字所表示的权限如下所示：

4：r（读权限）。

2：w（写权限）。
1：x（执行权限）。
表示 rwx 权限就是 4+2+1=7。
表示 rw-权限就是 4+2+0=6。
表示 r--权限就是 4+0+0=4。
表示 r-x 权限就是 4+0+1=5。
赋予所有人对文件的读、写和执行权限：

```
$ chmod 777 example.sh

$ ls -l example.sh
-rwxrwxrwx 1 yantaol yantaol 0 Sep 10 18:40 example.sh
```

赋予文件的所有者和用户组成员读写权限，其他用户只读权限：

```
$ chmod 664 example.sh

$ ls -l example.sh
-rw-rw-r-- 1 yantaol yantaol 0 Sep 10 18:40 example.sh
```

使用-R 选项，chmod 命令可以递归地修改目录的权限。比如递归地将当前目录及其下的所有文件和子目录的权限修改为 775：

```
$ chmod -R 775
```

然而，如果你只想修改当前目录下的所有子目录的权限，而不修改文件的权限，你可以将 chmod 命令与 find 命令结合使用：

```
$ find . -type d -exec chmod -R 775 {} \;
```

3.3.3　chown、chgrp 命令实例：修改文件所有者和用户组

chown 命令用于修改文件或目录的所有者和用户组信息。其语法如下所示：

```
$ chown [OPTION]… [OWNER][:[GROUP]] FILE
```

比如，我们将文件 example.sh 的所有者修改为 root：

```
# ls -l example.sh
-rw-rw-r-- 1 yantaol yantaol 0 Sep 10 18:40 example.sh

# chown root example.sh

# ls -l example.sh
-rw-rw-r-- 1 root yantaol 0 Sep 10 18:40 example.sh
```

注意：当前的命令行提示符是“#”号，表示当前用户账号是 root。

将文件 example.sh 的用户组也修改为 root：

```
# chown :root example.sh

# ls -l example.sh
-rw-rw-r-- 1 root root 0 Sep 10 18:40 example.sh
```

同时修改文件 example.sh 的所有者和用户组：

```
# chown yantaol:yantaol example.sh

# ls -l example.sh
-rw-rw-r-- 1 yantaol yantaol 0 Sep 10 18:40 example.sh
```

修改软链接文件的所有者和用户组信息，我们看一下会有什么样的结果：

```
# ls -l tmpfile_symlnk
lrwxrwxrwx 1 root root 7 Sep 11 11:33 tmpfile_symlnk -> tmpfile

# ls -l tmpfile
-rw-r--r-- 1 root    root        0 Sep 11 11:33 tmpfile

# chown yantaol:yantaol tmpfile_symlnk

# ls -l tmpfile_symlnk
lrwxrwxrwx 1 root root 7 Sep 11 11:33 tmpfile_symlnk -> tmpfile

# ls -l tmpfile
-rw-r--r-- 1 yantaol yantaol 0 Sep 11 11:33 tmpfile
```

注意：默认情况下，当你使用 chown 命令修改软链接文件时，它实际修改的是软链接所指向的文件。

使用-h 选项，可以强制地修改软链接的所有者和用户组信息，而不是修改软链接所指向的文件的所有者和用户组信息：

```
# ls -l tmpfile_symlnk
lrwxrwxrwx 1 root root 7 Sep 11 11:33 tmpfile_symlnk -> tmpfile

# ls -l tmpfile
-rw-r--r-- 1 yantaol yantaol 0 Sep 11 11:33 tmpfile

# chown -h yantaol:yantaol tmpfile_symlnk

# ls -l tmpfile_symlnk
lrwxrwxrwx 1 yantaol yantaol 7 Sep 11 11:33 tmpfile_symlnk -> tmpfile

# ls -l tmpfile
-rw-r--r-- 1 yantaol yantaol 0 Sep 11 11:33 tmpfile
```

使用--from 选项，可以仅在文件或目录的当前的所有者或用户组匹配所指定的用户或组时，才修改此文件或目录的所有者或用户组：

```
# ls -l tmpfile
-rw-r--r-- 1 yantaol yantaol 0 Sep 11 11:33 tmpfile

# chown --from=guest root:root tmpfile

# ls -l tmpfile
-rw-r--r-- 1 yantaol yantaol 0 Sep 11 11:33 tmpfile

# chown --from= yantaol root:root tmpfile

# ls -l tmpfile
-rw-r--r-- 1 root root 0 Sep 11 11:33 tmpfile

# chown --from=:root yantaol:yantaol tmpfile
```

```
# ls -l tmpfile
-rw-r--r-- 1 yantaol yantaol 0 Sep 11 11:33 tmpfile
```

使用-R 选项，chown 命令可以递归地修改目录下的文件及其子目录的所有者和用户组信息。比如，修改/root 目录下的所有文件和子目录的所有者和用户组信息：

```
# chown -R root:root /root
```

如果使用 chown 命令递归地修改指向某个目录的软链接的所有者和用户组，让我们来看一下会发生什么情况：

```
# ls -l linux_symlnk
lrwxrwxrwx 1 root root 5 Sep 11 14:40 linux_symlnk -> linux

# ls -dl linux
drwxr-xr-x 5 root root 4096 Sep 11 14:40 linux

# ls -l linux_symlnk/
total 12
drwxr-xr-x 2 root root 4096 Sep 11 14:40 lib
drwxr-xr-x 2 root root 4096 Sep 11 14:40 os
drwxr-xr-x 2 root root 4096 Sep 11 14:40 src

# chown -R yantaol:yantaol linux_symlnk

# ls -l linux_symlnk
lrwxrwxrwx 1 yantaol yantaol 5 Sep 11 14:40 linux_symlnk -> linux

# ls -dl linux
drwxr-xr-x 5 root root 4096 Sep 11 14:40 linux

# ls -l linux_symlnk/
total 12
drwxr-xr-x 2 root root 4096 Sep 11 14:40 lib
drwxr-xr-x 2 root root 4096 Sep 11 14:40 os
drwxr-xr-x 2 root root 4096 Sep 11 14:40 src
```

我们看到软链接的所有者和用户组已被修改，但是软链接所指向的目录并没有被修改。这是因为默认情况下，chown 命令不能横越软链接。如果想递归地修改软链接所指向的目录的所有者或用户组，需要使用-H 选项：

```
# chown -R -H yantaol:yantaol linux_symlnk
```

chgrp 命令与 chown 命令类似，但 chgrp 命令只用于修改文件或目录的用户组（不能修改所有者）。其语法如下所示：

```
$ chgrp [OPTION]… GROUP FILE…
```

3.3.4　设置 setuid 和 setgid 权限位实例：设置用户和组权限位

setuid（设置用户标识）是允许用户以文件所有者的权限执行一个程序的权限位。

setgid（设置组标识）是允许用户以用户组成员的权限执行一个程序的权限位。

任意用户可以以所有者的权限，执行一个设置了 setuid 的脚本。同样，任意用户可以以用户组成员的权限，执行一个设置了 setgid 的脚本。

注意：在类 Unix 系统中，理解 setuid 和 setgid 权限位的原理是重要的，特别是知道它们为什么被使用，更重要的是避免不当地使用它们。当你设置 setuid 或 setgid 权限位时，一定要非常谨慎，这些权限位可能导致安全风险。例如，用户通过执行一个设置了 setuid 权限位并且所有者是 root 的程序，可以获得超级用户的权限。

与其他权限位类似，可以使用 chmod 命令设置 setuid 和 setgid 权限位。

查看一个文件是否有 setuid 和 setgid，可以使用 ls -l 命令或 stat 命令。如果有字母“s”在所有者权限组表示设置了 setuid，有字母“s”在用户组权限组则表示设置了 setgid。例如，passwd 命令的权限：

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 23324 Aug 11  2010 /usr/bin/passwd

$ stat /usr/bin/passwd
  File: '/usr/bin/passwd'
  Size: 23324         Blocks: 48         IO Block: 4096   regular file
Device: 2101h/8449d    Inode: 532483      Links: 1
Access: (4755/-rwsr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2013-09-09 04:02:12.000000000 +0800
Modify: 2010-08-11 19:52:23.000000000 +0800
Change: 2012-10-07 18:39:40.000000000 +0800
```

使用 chmod 命令的符号表达式模式设置 setuid：

```
# chmod u+s example.sh

# ls -l example.sh
-rwsr-xr-x 1 root root 0 Sep 10 18:40 example.sh
```

注意：当前的命令行提示符是“#”号，表示当前用户账号是 root。

使用 chmod 命令的符号表达式模式设置 setgid：

```
# chmod g+s example.sh

# ls -l example.sh
-rwsr-sr-x 1 root root 0 Sep 10 18:40 example.sh
```

使用 chmod 命令的数字模式设置 setuid 的方法是，在 3 个权限位的前面加数字 4：

```
# chmod 4755 example.sh

# ls -l example.sh
-rwsr-xr-x 1 root root 0 Sep 10 18:40 example.sh
```

使用 chmod 命令的数字模式设置 setgid 的方法是，在 3 个权限位的前面加数字 2：

```
# chmod 2755 example.sh

# ls -l example.sh
-rwxr-sr-x 1 root root 0 Sep 10 18:40 example.sh
```

使用 chmod 命令移除 setuid 和 setgid 权限位的方法是，在 3 个权限位的前面加数字 0：

```
# chmod 0755 example.sh

# ls -l example.sh
-rwxr-xr-x 1 root root 0 Sep 10 18:40 example.sh
```

3.4　文 本 处 理

3.4.1　sort 命令实例：文本排序

sort 命令用于将文本文件的行排序。默认情况下，sort 命令是按照字符串的字母顺序排序。

现在有一个包含如下内容的文件：

```
$ cat example.txt
abc
def
ghi
jkl
mno
def
```

不使用任何选项，sort 命令简单地将文件内容按字母顺序排序：

```
$ sort example.txt
abc
def
def
ghi
jkl
mno
```

使用-u 选项，sort 命令可以移除所有重复的行：

```
$ sort -u example.txt
abc
def
ghi
jkl
mno
```

上例中，重复的记录“def”被移除。

现在有一个文件，里面的内容都是数字：

```
$ cat example.txt
20
10
35
100
69
83
```

直接简单地使用 sort 命令，将得到如下结果：

```
$ sort example.txt
10
100
20
35
69
83
```

在上例中，100 被直接排到了 10 的下面，而不是按照数字的大小排在末尾。这与 sort 命令的默认排序机制有关。

使用-n 选项，可以使 sort 命令将数字按数值的大小排序：

```
$ sort -n example.txt
10
20
35
69
83
100
```

使用-r 选项，可以使 sort 命令以倒序方式排序：

```
$ sort -n -r example.txt
100
83
69
35
20
10
```

sort 命令可以同时将多个文件的内容排序：

```
$ sort file1 file2
```

如果一个文件有多个列（下例中，每列以逗号“，”分隔），如下所示：

```
$ cat example.txt
abc,20
def,10
ghi,35
jkl,100
mno,69
def,83
```

默认情况下，sort 命令按文件第一列的字符串字母顺序将文件内容排序：

```
$ sort example.txt
abc,20
def,10
def,83
ghi,35
jkl,100
mno,69
```

指定 sort 命令按照第二列的字符串顺序将文件内容排序：

```
$ sort -t ',' -k2,2 example.txt
def,10
jkl,100
abc,20
ghi,35
mno,69
def,83
```

上例中，-t 选项用于指定列的分隔符，上例中的列分隔符是逗号“，”；-k 选项用于指定进行排序的列，这里我们指定的是第二列。

指定 sort 命令按照第二列的数值顺序将文件内容排序：

```
$ sort -t ',' -k2n,2 example.txt
def,10
abc,20
ghi,35
mno,69
def,83
jkl,100
```

指定 sort 命令按照第二列的数值顺序的倒序将文件内容排序：

```
$ sort -t ',' -k2nr,2 example2.txt
jkl,100
def,83
mno,69
ghi,35
abc,20
def,10
```

3.4.2　uniq 命令实例：文本去重

uniq 命令用于移除或发现文件中重复的条目。
现在有一个文件，其内容如下所示：

```
$ cat example.txt
aaa
aaa
bbb
bbb
bbb
ccc
```

使用 uniq 命令，不带有任何选项时，它将移除文件中重复的行并显示单一行：

```
$ uniq example.txt
aaa
bbb
ccc
```

使用-c 选项，可以统计重复行出现的次数：

```
$ uniq -c example.txt
2 aaa
3 bbb
1 ccc
```

使用-d 选项，只显示文件中有重复的行并只显示一次：

```
$ uniq -d example.txt
aaa
bbb
```

使用-D 选项，与-d 选项类似，但它显示文件中所有重复的行：

```
$ uniq -D example.txt
aaa
aaa
bbb
bbb
bbb
```

使用-u 选项，只显示文件中不重复的行：

```
$ uniq -u example.txt
ccc
```

现在我们将示例文件修改成如下内容：

```
$ cat example.txt
aaa bbb
aaa ccc
bbb aaa
bbb ccc
ccc ccc
```

使用-w 选项，可以限制 uniq 命令只比较每行的前 N 个字符。例如，下面的实例中，限制 uniq 命令只比较每行的前 3 个字符是否重复：

```
$ uniq -w 3 example.txt
aaa bbb
bbb aaa
ccc ccc
```

使用-s 选项，可以避免 uniq 命令比较每行的前 N 个字符，即跳过每行的前 N 个字符，只比较后面的字符。例如，下面的实例中，避免 uniq 命令比较每行的前 3 个字符，只比较后面的字符是否重复：

```
$ uniq -s 3 example.txt
aaa bbb
aaa ccc
bbb aaa
bbb ccc
```

使用-f 选项，可以避免 uniq 命令比较前 N 列，即跳过前 N 列（这里列以空格分隔），只比较后面的字符。例如，下面的实例中，避免 uniq 命令比较第一列的内容，只比较后面的字符是否重复：

```
$ uniq -f 1 example.txt
aaa bbb
aaa ccc
bbb aaa
bbb ccc
```

3.4.3　tr 命令实例：替换或删除字符

tr 命令用于转换字符、删除字符和压缩重复的字符。它从标准输入读取数据并将结果输出到标准输出。

tr 命令的语法如下所示：

```
$ tr [OPTION]… SET1 [SET2]
```

我们首先了解一下 tr 命令的转换字符的功能。如果参数 SET1 和 SET2 同时指定，并且没有指定-d 选项，那么 tr 命令将把 SET1 中指定的每个字符替换为 SET2 中相同位置的字符。

下面这个实例用于将字符串中所有的小写字母转换为大写字母：

```
$ echo linuxShell | tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
LINUXSHELL
```

下面的实例同样可以将字符串中的所有小写字母转换为大写字母：

```
$ echo linuxShell | tr [:lower:] [:upper:]
LINUXSHELL
```

还有一种方法可以得到和上例同样的结果：

```
$ echo linuxShell | tr a-z A-Z
LINUXSHELL
```

使用 tr 命令转换一个文件中的内容，并将转换的结果输出到另一个文件：

```
$ cat inputfile
{linuxShell}

$ tr '{}' '()' < inputfile > outputfile

$ cat outputfile
(linuxShell)
```

上例中，使用 tr 命令将文件 inputfile 中的大括号转换为了小括号，并将转换的结果输出到了文件 outputfile 中。

将字符串中的空格转换为制表符：

```
$ echo "This is for testing" | tr [:space:] '\t'
This    is      for     testing
```

如果上例中有两个以上空格同时出现，那么 tr 命令将把每个空格都转换为制表符，类似如下所示：

```
$ echo "This  is   for testing" | tr [:space:] '\t'
This          is                  for     testing
```

这时，我们可以使用-s 选项压缩这些重复的空格：

```
$ echo "This  is   for testing" | tr -s [:space:] '\t'
This    is      for     testing
```

下面我们来学习使用 tr 命令删除指定的字符。使用-d 选项，tr 命令可以删除指定的字符：

```
$ echo "The Linux Shell" | tr -d a-z
T L S
```

上例中，使用 tr 命令删除了字符串中的所有小写字母。

使用-d 选项，删除字符串中的所有数字：

```
$ echo "my username is yantaol123" | tr -d [:digit:]
my username is yantaol
```

将-c 和-d 选项结合使用，删除字符串中除数字以外的所有字符：

```
$ echo "my username is yantaol123" | tr -cd [:digit:]
123
```

3.4.4　grep 命令实例：查找字符串

grep 命令用于搜索文本或指定的文件中与指定的字符串或模式相匹配的行。默认情况下，grep 命令只显示匹配的行。

grep 命令的语法如下所示：

```
$ grep [OPTION]… PATTERN [FILE]…
$ grep [OPTION]… [ -e PATTERN | -f FILE] [FILE]…
```

下面的实例中，我们使用 grep 命令查找文件/etc/passwd 中账号 yantaol 的信息：

```
$ grep yantaol /etc/passwd
```

会得到类似如下的输出：

```
yantaol:x:12107:25:SDE-LiuYanTao:/home/yantaol:/bin/bash
```

使用-i 选项，可以强制 grep 命令忽略搜索关键字的大小写：

```
$ grep -i YantaoL /etc/passwd
yantaol:x:12107:25:SDE-LiuYanTao:/home/yantaol:/bin/bash
```

使用-r 选项，可以递归搜索指定目录下的所有文件：

```
$ grep -r yantaol /etc/
```

或者

```
$ grep -R yantaol /etc/
```

将-r 选项与-l 选项结合使用，grep 命令可以只打印输出包含匹配指定模式的行的文件的名字：

```
$ grep -rl yantaol /etc/
/etc/passwd
/etc/shadow
```

默认情况下，当搜索字符串 yantaol 时，grep 命令也会匹配 yantaol123、yantaolb 和 liuyantaol 等字符串。使用-w 选项，就可以强制 grep 命令只匹配包含指定单词的行。比如，查找文件/etc/passwd 中只包含指定单词 yantaol 的行，类似如下所示：

```
$ grep -w yantaol /etc/passwd
```

使用-c 选项，grep 命令可以报告文件或文本中模式被匹配的次数：

```
$ grep -c yantaol /etc/passwd
1
```

使用-n 选项，grep 命令可以显示每一个匹配的行的行号：

```
$ grep -n yantaol /etc/passwd
36:yantaol:x:12107:25:SDE-LiuYanTao:/home/yantaol:/bin/bash
```

使用-v 选项， grep 命令可以输出除匹配指定模式的行以外的其他所有行：

```
$ grep -v yantaol /etc/passwd
```

上例中，所有不包含字符串 yantaol 的行都将被打印输出。

使用--color 选项，grep 命令在输出中将匹配的字符串以彩色的形式标出：

```
# grep --color yantaol /etc/passwd
yantaol:x:12107:25:SDE-LiuYanTao:/home/yantaol:/bin/bash
```

grep 命令通常与 Shell 管道一起结合使用，如以下实例所示：

```
$ cat /etc/passwd | grep -i YantaoL
```

关于管道的相关知识我们将在第 12 章中做详细讲解。

3.4.5　diff 命令实例：比较两个文件

diff 命令用于比较两个文件，并找出它们之间的不同。diff 命令的语法如下所示：

```
$ diff [OPTION]… from-file to-file
```

我们以如下两个文件为例，其文件内容如下：

```
$ cat -n nsswitch.conf
1  passwd:     files nis
2  shadow:     files nis
3  group:      files nis
4
5
6  # Example - obey only what nisplus tells us...
7  #services:  nisplus [NOTFOUND=return] files
8  #networks:  nisplus [NOTFOUND=return] files
9  #protocols: nisplus [NOTFOUND=return] files
10  #rpc:      nisplus [NOTFOUND=return] files
11  #ethers:   nisplus [NOTFOUND=return] files

$ cat -n nsswitch.conf.org
1  passwd:     files
2  shadow:     files
3  group:      files
4
5  #hosts:     db files nisplus nis dns
6  hosts:      files dns
7
8  # Example - obey only what nisplus tells us...
9  #services:   nisplus [NOTFOUND=return] files
10  #networks:   nisplus [NOTFOUND=return] files
11  #protocols:  nisplus [NOTFOUND=return] files
12  #rpc:       nisplus [NOTFOUND=return] files
13  #ethers:    nisplus [NOTFOUND=return] files
```

简单地使用 diff 命令比较上述两个文件的方法如下所示：

```
$ diff nsswitch.conf nsswitch.conf.org
1,3c1,3
< passwd:     files nis
< shadow:     files nis
< group:      files nis
---
```

```
> passwd:    files
> shadow:    files
> group:      files
4a5,6
> #hosts:    db files nisplus nis dns
> hosts:     files dns
7,11c9,13
< #services:  nisplus [NOTFOUND=return] files
< #networks:  nisplus [NOTFOUND=return] files
< #protocols: nisplus [NOTFOUND=return] files
< #rpc:      nisplus [NOTFOUND=return] files
< #ethers:    nisplus [NOTFOUND=return] files
---
> #services:   nisplus [NOTFOUND=return] files
> #networks:   nisplus [NOTFOUND=return] files
> #protocols:  nisplus [NOTFOUND=return] files
> #rpc:       nisplus [NOTFOUND=return] files
> #ethers:     nisplus [NOTFOUND=return] files
```

上例中，“1,3c1,3”表示第一个文件的 1～3 行与第二个文件的 1～3 行在内容上有差别。“<”表示第一个文件的行，“>”表示第二个文件的行。“4a5,6”表示第二个文件与第一个文件相比，在第 4 行后多了 5、6 两行内容。“7,11c9,13”表示第一个文件的 7～11 行与第二个文件的 9～13 行在内容上有差别。

你可能已经注意到，上例中的第一个文件的 7～11 行与第二个文件的 9～13 行相比，只是在“:”和“nisplus”之间多了一个空格。如果想在比较两个文件时忽略这些空格，该如何处理呢？

使用-w 选项，diff 命令就将在比较两个文件时忽略这些空格：

```
$ diff -w nsswitch.conf nsswitch.conf.org
1,3c1,3
< passwd:    files nis
< shadow:    files nis
< group:     files nis
---
> passwd:    files
> shadow:    files
> group:     files
4a5,6
> #hosts:    db files nisplus nis dns
> hosts:     files dns
```

使用-y 选项，diff 命令可以并排的格式输出两个文件的比较结果：

```
$ diff -yw nsswitch.conf nsswitch.conf.org
passwd:    files nis                          | passwd:    files
shadow:    files nis                          | shadow:    files
group:     files nis                          | group:     files

                                             > #hosts:    db files nisplus nis dns
                                              > hosts:     files dns

# Example - obey only what nisplus tells us...
                                             # Example - obey only what nisplus tells us...
#services:  nisplus [NOTFOUND=return] files
                                             #services:   nisplus [NOTFOUND=return] files
```

```
#networks:  nisplus [NOTFOUND=return] files
                            #networks:   nisplus [NOTFOUND=return] files
#protocols: nisplus [NOTFOUND=return] files
                            #protocols:  nisplus [NOTFOUND=return] files
#rpc:       nisplus [NOTFOUND=return] files
                            #rpc:       nisplus [NOTFOUND=return] files
#ethers:    nisplus [NOTFOUND=return] files
                            #ethers:    nisplus [NOTFOUND=return] files
```

上例中，“|”表示内容有差异的行，“>”表示第二个文件中多出的行。如果是“<”，则表示第一个文件中多出的行。

如果使用-y 选项时，每行显示的内容不完整，可以与-W 选项结合使用，指定并列输出格式的列宽，使每行的内容可以完整地显示：

```
$ diff -yw -W 160 nsswitch.conf nsswitch.conf.org
```

使用-c 选项，diff 命令会以上下对比的格式输出两个文件的比较结果：

```
$ diff -cw nsswitch.conf nsswitch.conf.org
*** nsswitch.conf      2013-09-24 10:09:19.000000000 +0800
--- nsswitch.conf.org   2013-09-24 10:09:58.000000000 +0800
***************
*** 1,7 ****
! passwd:    files nis
! shadow:    files nis
! group:     files nis


  # Example - obey only what nisplus tells us...
  #services:  nisplus [NOTFOUND=return] files
--- 1,9 ----
! passwd:    files
! shadow:    files
! group:     files

+ #hosts:    db files nisplus nis dns
+ hosts:     files dns

  # Example - obey only what nisplus tells us...
  #services:   nisplus [NOTFOUND=return] files
```

上例中，“!”表示两个文件中内容有差别的行，“+”表示第二个文件比第一个文件多出的行。

3.5　其他常用命令

3.5.1　hostname 命令实例：查看主机名

hostname 命令用于查看系统的主机名，或是修改系统的主机名。

直接简单地使用 hostname 命令，不指定任何参数，将显示你的系统的当前主机名：

```
# hostname
yantaol-laptop
```

注意：当前的命令行提示符是“#”号，表示当前用户账号是 root。

使用 hostname 命令修改你的系统的主机名：

```
# hostname yantaol-system
yantaol-system

# hostname
yantaol-system
```

注意：上述命令只是临时地修改系统的主机名。当系统重启后，这个新修改的主机名将不会被使用。

使用-F 选项，hostname 命令可以从指定的文件中读取主机名。注释行（以符号“#”开头的行）将被忽略。如下所示：

```
# cat /root/hostname.txt
yantaol-laptop

# hostname -F /root/hostname.txt

# hostname
yantaol-laptop
```

3.5.2　w、who 命令实例：列出系统登录的用户

w 命令用于显示登录的用户及他们当前运行的进程。

w 命令输入的内容格式如下：

```
# w
 11:35:16 up 1 day, 17:32,  1 user,  load average: 0.05, 0.01, 0.00
USER     TTY        FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    yantaol-laptop   11:35    0.00s   0.00s   0.00s  w
```

上例中，w 命令输出的第一行内容与 uptime 命令默认输出的内容相同（uptime 命令我们将在下一小节中介绍）。第三行分别显示的是：登录账号的用户名、tty 名称、从哪台主机登录、登录时间、空闲时间、tty 上的所有进程所使用的 CPU 时间、当前进程所使用的 CPU 时间以及当前运行的进程。

who 命令有与 w 命令类似的用途，但它的功能比 w 命令更强大一些。who 命令的语法如下所示：

```
$ who [OPTION]… [ FILE | ARG1 ARG2 ]
```

直接使用 who 命令，不知道任何参数，将显示当前登录的所有用户的信息：

```
$ who
yantaol pts/0        2013-09-25 13:45 (yantaol-laptop)
root     pts/1        2013-09-25 13:52 (yantaol-laptop)
```

使用-b 选项，who 命令可以显示系统的启动时间：

```
$ who -b
        system boot  2013-09-23 18:03
```

使用-l 选项，who 命令会显示出系统登录进程：

```
$ who -l
LOGIN    tty1        2013-09-23 18:04           3685 id=1
LOGIN    tty2        2013-09-23 18:04           3686 id=2
LOGIN    tty3        2013-09-23 18:04           3687 id=3
LOGIN    tty4        2013-09-23 18:04           3689 id=4
LOGIN    tty6        2013-09-23 18:04           3700 id=6
```

使用-m 选项，who 命令将只显示与当前标准输入关联的用户信息：

```
$ who -m
yantaol pts/0       2013-09-25 13:45 (yantaol-laptop)
```

使用-r 选项，who 命令将显示系统的运行级别：

```
$ who -r
        run-level 5  2013-09-23 18:03                last=S
```

使用-q 选项， who 命令将只显示所有登录用户的用户名和登录的用户数：

```
$ who -q
yantaol root
# users=2
```

3.5.3　uptime 命令实例：查看系统运行时间

uptime 命令用于打印系统的运行时间等信息。

uptime 命令的使用很简单，只需简单地在命令行提示符下输入 uptime 命令，将显示类似如下信息：

```
$ uptime
 14:19:27 up 1 day, 20:16,  2 users,  load average: 0.86, 0.88, 0.97
```

上例的输出中，“14:19:27”表示当前时间，“up 1 day, 20:16”表示系统已经连续运行 1 天 20 小时 16 分，“load average: 0.86, 0.88, 0.97”中显示的分别是系统的过去 1 分钟、5 分钟和 15 分钟的平均负载。

3.5.4　uname 命令实例：查看系统信息

uname 命令用于打印内核名称和版本、主机名等系统信息。命令的语法如下所示：

```
$ uname [OPTION]…
```

直接简单地输入 uname 命令，不指定任何选项，uname 命令将只打印内核的名称。类似如下所示：

```
$ uname
Linux
```

使用-n 选项，uname 命令将只打印系统的主机名，其输出与 hostname 命令相同：

```
$ uname -n
yantaol-laptop
```

使用-r 选项，uname 命令将打印内核版本信息：

```
$ uname -r
2.6.18-238.9.1.el5PAE
```

使用-m 选项，uname 命令将打印系统的硬件名称：

```
$ uname -m
i686
```

使用-p 选项，uname 命令将打印系统的处理器类型的信息：

```
$ uname -p
i686
```

使用-i 选项，uname 命令将打印系统的硬件平台信息：

```
$ uname -i
i386
```

使用-a 选项，uname 命令将打印上述所有示例中的信息，其打印结果与 uname -snrvmpio 命令相同：

```
$ uname -a
Linux yantaol-laptop 2.6.18-238.9.1.el5PAE #1 SMP Tue Apr 12 19:28:32 EDT
2011 i686 i686 i386 GNU/Linux
```

3.5.5　date 命令实例：显示和设置系统日期和时间

date 命令用于以多种格式显示日期和时间，或设置系统的日期和时间。

date 命令的语法如下所示：

```
$ date [OPTION]... [+FORMAT]
$ date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

直接简单地输入 date 命令，不指定任何选项，date 命令将以默认格式直接显示系统的当前日期时间：

```
$ date
Wed Sep 26 09:02:49 CST 2013
```

使用-d 或--date 选项，你可以指定日期和时间的字符串值，date 命令会将此输入的字符串转换为相应的日期时间格式：

```
$ date --date="10/1/2013"
Tue Oct  1 00:00:00 CST 2013

$ date --date="1 Oct 2013"
Tue Oct  1 00:00:00 CST 2013

$ date --date="Oct 1 2013"
Tue Oct  1 00:00:00 CST 2013

$ date --date="Oct 1 2013 12:13:14"
Tue Oct  1 12:13:14 CST 2013

$ date --date="next mon"
Mon Sep 30 00:00:00 CST 2013
```

```
$ date --date="next week"
Thu Oct  3 09:47:06 CST 2013

$date --date="last week"
Thu Sep 19 09:47:50 CST 2013

$ date --date="1 week ago"
Thu Sep 19 09:47:58 CST 2013

$ date --date="yesterday"
Wed Sep 25 09:53:35 CST 2013

$ date --date="last day"
Wed Sep 25 09:53:44 CST 2013
```

使用-f 或--file 选项，date 命令可以从文件中读取多个日期时间字符串，并将其转换为相应的日期时间格式打印输出：

```
$ cat  datefile
Oct 17 1986
Dec 13 1989

$ date --file=datefile
Fri Oct 17 00:00:00 CST 1986
Wed Dec 13 00:00:00 CST 1989
```

使用-s 或--set 选项，date 命令可以设定系统的日期和时间。比如当前系统时间如下所示：

```
$ date
Thu Sep 26 09:57:35 CST 2013
```

现在修改系统时间到 10:10:00：

```
$ date -s "Thu Sep 26 10:10:00 CST 2013"
Thu Sep 26 10:10:00 CST 2013

$ date
Thu Sep 26 10:10:01 CST 2013
```

使用-u 或--utc 或--universal 选项，date 命令将打印输出世界标准时间：

```
$ date
Thu Sep 26 10:18:27 CST 2013

$ date -u
Thu Sep 26 02:18:29 UTC 2013
```

使用-r 选项，date 命令可以打印输出指定文件的最近修改时间：

```
$ stat datefile
  File: 'datefile'
  Size: 24              Blocks: 8          IO Block: 4096   regular file
Device: 2101h/8449d     Inode: 1638410     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2013-09-26 09:44:06.000000000 +0800
Modify: 2013-09-26 09:43:41.000000000 +0800
```

```
Change: 2013-09-26 09:43:41.000000000 +0800

$ date -r datefile
Thu Sep 26 09:43:41 CST 2013
```

你还可以使用格式化选项来自定义 date 命令打印输出的时间和日期的格式，其语法类似如下所示：

```
$ date +%<format-option>
```

关于格式化选项的详细信息，请参见 date 命令的参考手册（$ man date）。

3.5.6　id 命令实例：显示用户属性

id 命令用于打印输出用户的 uid、gid、用户名和组名等用户身份信息。id 命令的语法如下所示：

```
$ id [OPTION]... [USERNAME]
```

直接输入 id 命令，将打印输出当前用户的 uid、用户名、gid、组名，以及用户属于的所有群组信息：

```
$ id
uid=12107(yantaol) gid=25(yantaol) groups=24(xxxx),25(yantaol)
```

使用-u 选项，id 命令将只打印输出用户的 uid：

```
$ id -u
12107

$ id -u root
0
```

将-u 选项和-n 选项结合使用，可以打印输出账号的用户名，而不是 uid：

```
$ id -un
yantaol
```

使用-g 选项，id 命令将只打印输出账号当前起作用的 gid：

```
$ id -g
25
```

同样，-g 与-n 选项结合使用，可以打印输出账号当前起作用的用户组名，而不是 gid：

```
$ id -gn
yantaol
```

使用-G 选项，id 命令将打印输出账号所属于的所有群组的 id：

```
$ id -G root
0 1 2 3 4 6 10 8 7 5 9 12
```

同样，-G 与-n 选项结合使用，可以打印输出账号所属于的所有群组的名称：

```
$ id -Gn root
root bin daemon sys adm disk wheel mem lp tty kmem mail
```

3.6　小　　结

下面我们总结一下本章所学的主要知识。

ls 命令可以列出文件和目录的信息，包括文件类型、所有者、大小、修改日期和时间、权限等等。

cat 命令可以查看文件的内容、连接文件、创建一个或多个文件和重定向输出到终端或文件。

more 命令在你使用小的 xterm 窗口时，或是想不使用文本编辑器而只是简单地阅读一个文件时是很有用的。它是一个用于一次翻阅一整屏文件的过滤器。

less 命令与 more 命令类似，但 less 命令向前和向后翻页都支持，而且 less 命令不需要在查看前加载整个文件，即 less 命令查看文件更快速。当你尝试使用 Vim 编辑器和 less 打开同一个大的 log 文件，你会发现速度是不同的。

head 命令用于打印指定输入的开头部分内容。默认情况下，打印每个指定输入的前 10 行内容。

tail 命令与 head 命令相反，它打印指定输入的结尾部分的内容。默认情况下，它打印指定输入的最后 10 行内容。

file 命令用于接收一个文件作为参数，并执行某些测试以确定正确的文件类型。

wc 命令用于查看文件的行数、单词数和字符数等信息。

find 命令用于根据你指定的参数搜索和定位文件和目录的列表。find 命令可以在多种情况下使用，比如你可以通过权限、用户、用户组、文件类型、日期、大小和其他可能的条件来查找文件。

touch 命令用于创建、变更和修改文件的时间戳。它是 Linux 操作系统的标准程序。

mkdir 命令用于创建一个新目录。

cp 命令用于将文件从一个地方复制到另一个地方。原来的文件保持不变，新文件可能保持原名或用一个不同的名字。

ln 命令用于创建软链接或硬链接。

mv 命令用于将文件和目录从一个位置移到另外一个位置。除了移动文件，mv 命令还可用于修改文件或目录的名字。

rm 命令用于删除指定的文件和目录。

chmod 命令用于修改文件或目录的权限。

chown 命令用于修改文件或目录的所有者和用户组信息。

chgrp 命令与 chown 命令类似，但 chgrp 命令只用于修改文件或目录的用户组（不能修改所有者）。

setuid（设置用户标识）是允许用户以文件所有者的权限执行一个程序的权限位。setgid（设置组标识）是允许用户以用户组成员的权限执行一个程序的权限位。

sort 命令用于将文本文件的行排序。默认情况下，sort 命令是按照字符串的字母顺序排序。

uniq 命令用于移除或发现文件中重复的条目。

tr 命令用于转换字符、删除字符和压缩重复的字符。它从标准输入读取数据并将结果输出到标准输出。

grep 命令用于搜索文本或指定的文件中与指定的字符串或模式相匹配的行。

diff 命令用于比较两个文件，并找出它们之间的不同。

hostname 命令用于查看系统的主机名，或是修改系统的主机名。

w 命令用于显示登录的用户及他们当前运行的进程。who 命令有与 w 命令类似的用途，但它的功能比 w 命令更强大一些。

uptime 命令用于打印系统的运行时间等信息。

uname 命令用于打印内核名称和版本、主机名等系统信息。

date 命令用于以多种格式显示日期和时间，或设置系统的日期和时间。

id 命令用于打印输出用户的 uid、gid、用户名和组名等用户身份信息。