Linux 操作系统及应用
第三章 — 系统 Shell 和常用命令

李亦农　唐晓晟
hoplee@bupt.edu.cn　txs@bupt.edu.cn

Beijing University of Posts and Telecommunications (BUPT)
School of Information and Communication Engineering

内容简介 I

1. 登录及退出
2. 设置用户密码
3. 使用 Shell
4. Shell 命令行
5. 建立 Shell 命令脚本程序
6. UNIX 文件系统

内容简介 II

内容简介 III

内容简介 IV

登录及退出

- 由于 UNIX 是一个多用户的操作系统，所以必须让系统能区分不同的用户。用户使用 ID 和密码（口令）进行身份的登记。系统根据用户的身份进行资源的分配以及用户数据的保护。
- 终端登录
- 远程登录

登录过程 I

- 首先确定你的机器已经和某台 UNIX 相连（通过以太网卡或是终端接口），或者你直接可以使用某台 UNIX 机器的控制台（console）
- 然后向 UNIX 的系统管理员申请一个帐号，你可以得到一个 user id（或 account name）和相应的 password
- 启动你的终端或某个终端仿真程序，指定 UNIX 主机的地址
- 如果连接正常的话在你的终端上将看到如下信息（以 Ubuntu 14.10 为例）：

```
1 Ubuntu 14.10 Apple :0
2 login:
```

- 现在输入你的用户名并回车；接着系统将提示你输入密码：

```
1 Password:
```

登录过程 II

- 如果用户名和密码都正确的话就可以进入系统了。此时将出现如下的内容：

```
1 Last login: Tue Sep 26 17:06:14 from :0
2 [Apple]$
```

- 如果用户名或者密码错误，系统将提示你登录错误，但是并不会指出是用户名错误还是密码错误，即使根本就没有这个帐户也一样：

```
1 login incorrect
2 login:
```

- 登录正确后你最后会看到一个 shell 命令提示符，一般是一个 $ 符号。

退出

- 成功登录之后，你就可以在 shell 提示符下向系统提交你的任务。
- 当你使用完系统之后就应该退出，以便将占用的资源释放。退出系统的命令为：`logout`或`exit`或直接键入`^d`。
- 关闭系统的命令为：`shutdown -h now`或`halt`或`init 0`。
- 重新启动计算机的命令为：`reboot`或`init 6`。
- 关闭和重启系统的命令必须以超级用户身份运行。

密码的保护

- 密码的保密关系着用户信息的安全，因此我们应该特别注意密码的保护；
- 密码的保护主要要注意密码的选择和养成定时更换密码的习惯；
- 密码的选择应该遵循以下原则：
  - 不能是任何有意义的单词及其反序、组合
  - 不能是自己或家人、朋友的生日、电话号码及其反序、组合
  - 必须同时包含大写字母、小写字母、数字和符号
  - 密码长度至少为六位
  - 输入时尽可能双手平均使用
  - 密码应该定时更换，一般应该三到四周更换一次，新密码与旧密码之间至少应该有四位以上的不同符号。
  - 密码不应记录在笔记、便签、书籍等上，而应该牢牢记住。

密码的修改

- 密码的修改使用passwd命令。过程如下所示：

```
1 [Apple]$ passwd
2 Changing password for student
3 (current) UNIX password:
4 New UNIX password:
5 Retype new UNIX password:
```

- 为了保证密码修改的合法性，系统要求用户先输入原来的密码。
- 为了防止用户输入错误，系统要求用户输入两次新密码。
- 如果你输入的原来的密码正确，并且两次输入的新密码完全一致，那么修改密码成功，系统显示：

```
passwd: all authentication tokens updated successfully
```

- 系统还要求新密码至少要有三位与旧密码不同，并且其长度至少为六位。

什么是 Shell

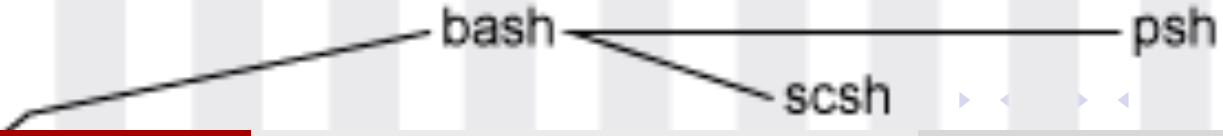
- shell 提供了一个到 Linux 操作系统的界面以方便运行程序。事实上，shell 也只不过是另外一个 Linux 操作系统程序而已。
- shell 是一个命令解释器，它可以用来启动、挂起、停止甚至编写程序。shell 是 Linux 操作系统的一个整体组成部分，也是 Linux 操作系统和 UNIX 设计的一部分。
- 如果把 Linux 操作系统的内核想象成一个球体的中心，那么 shell 就是包围内核的外层。从 shell 或其他程序向 Linux 操作系统传递命令的时候，内核就会做出相应的反应。
- 用户 shell 配置的位置：`/etc/passwd`文件中每条记录的最后一个字段。

学习 Shell 应注意学习哪方面的内容？

- 这种 shell 的内建命令都有哪些？
- 怎样进行任务控制？
- 这种 shell 是否支持命令行编辑？
- 这种 shell 是否支持命令行历史记录？
- 什么是它的重要的开机启动文件或者配置文件？
- 各个 shell 的重要环境变量有哪些？
- 可以使用什么样的命令行提示符？
- 它支持什么样的编程框架？

系统中有几种 Shell

- 常见非图形化的 shell：
  - ash 袖珍的sh兼容的 shell
  - bash Bourne Again Shell（与ksh和sh兼容）
  - csh 对tcsh的一个符号链接
  - ksh pdksh，公共域 Korn（与ksh兼容）shell
  - sh 对bash的一个符号链接
  - tcsh 与csh兼容的 shell
  - zsh Z-shell，一个与csh、ksh、和sh兼容的 shell
- 基于 curse 的 shell：clex, mc（Midnight Commander）, pdmenu 等。
- 图形化的 shell：Nautilus, Konqueror, Dolphin, Thunar, ROX-Filer, PCMan File Manager, emelFM2, GNOME Commander, Tux Commander, FlyBird, Nao, muCommander, Krusader, BSCommander, Xfe, gentoo 等等。



ash

- 由 Kenneth Almquist 编写的ash是 Linux 操作系统上尺寸最小的 shell 之一。这个 shell 有二十多个不同的内建命令和十多个不同的命令行参数。
- ash支持大多数常见的 shell 命令，如cd命令以及大多数普通的命令行操作符。
- 它是一个很流行的 shell，通常在以下几种情况下使用：以单用户状态（在 LILO 启动提示符下输入linux single）启动到 Linux 操作系统、用于安全恢复状态或者用于 Linux 操作系统的软盘版本。它自身的尺寸很小，通常只有bash的十分之一大小，这使得 ash成为小文件系统理想的选择。
- dash是 NetBSD 版ash的后代，Herbert Xu 于 1997 年将其移植到 Linux 上，并于 2002 年正式改名为dash。

bash

- bash就是由 Brian Fox 和 Chet Ramey 编写的 Bourne Again Shell，它是 Linux 操作系统上最流行的 shell 之一。它有 59 个内建的命令和十多个命令行参数。
- bash有许多特色。可以使用方向键查阅以前输入的命令（历史记录功能）、可以对某个命令行进行编辑、而且在忘记了某个程序的名字的时候，甚至可以请求这个 shell 使用命令行补充功能对你进行帮助。只要敲入一个命令的一部分然后再按下 Tab 键就可以了。
- bash还有内建的帮助功能，能够列出所有的内建命令和关于每个命令的帮助信息。
- bash有几个非常重要的文件叫资源文件、运行配置文件和 shell 启动文件。第一次登录进入 Linux 操作系统的时候，这些文件被用来定义或者共享预定义的设置值和定义值，比如所使用的终端类型、缺省的文本编辑器程序和打印机以及可执行文件存放的位置等等。

bash II

- 初始化设置文件/etc/profile用来设置全局（对所有用户都起作用的）参数，比如环境变量或者在第一次登录进入的时候给你发送一条消息（比如一条欢迎标语）。也可以使用用户主目录中的.bashrc登录文件按照个人喜好控制bash启动运行的方式或者对不同的击键（如退格键）的响应；在用户主目录中还有一个.profile文件，它被用来通知 shell 在登录进入 Linux 操作系统之后应该使用哪一个资源文件。

pdksh

- pdksh，也就是公共域 Korn shell，最初是由 Eric Gisin 于 1995 年编写的，它有 42 个内建的命令和 20 个命令行参数。这个 shell 存放在子目录/bin下，在子目录/usr/bin中还有一个符号链接。目前由 Michael Rendell 维护。
- pdksh在 Linux 操作系统中叫做ksh。它和bash一样，如果在用户主目录中找不到一个名为.profile的文件的话，就会去读取初始化设置文件/etc/profile。不幸的是，这个 shell 并不支持与bash使用的相同的命令行提示符。但这个 shell 支持任务控制，所以可以从命令行上挂起、后台执行、唤醒或终止程序。
- 这个 shell 和商业化 UNIX 中的版本几乎是完全兼容的。关于这个 shell 的有关文档存放在ksh的使用手册页和目录/usr/doc/pdksh中。
- ksh能在自身进程内进行浮点数运算。

tcsh

- 由 William Joy（以及其他 47 名参加人员）编写的tcsh有 52 个内建的命令和 18 个命令行参数。这个 shell 仿真csh，但是增加了许多功能，包括一个带拼写检查功能的命令行编辑器程序。
- 这个 shell 不仅能够与bash提示符兼容，而且还可以提供比bash更多的提示符定制参数。如果在用户主目录中找不到.tcshrc或.cshrc文件的话，tcsh就会使用目录/etc中的csh.cshrc文件。与bash类似，可以查看曾经输入过的命令并编辑命令行。
- 使用builtins命令（tcsh没有像bash那样的帮助信息功能）可以得到一份tcsh中允许使用的命令的清单。

zsh

- 由 Paul Falstad 于 1990 年开始编写的zsh是 Linux 操作系统所使用的最大的 shell 之一，它有 84 个内建的命令和超过 50 个命令行参数，并能仿真sh和ksh的命令。
- 与bash和tcsh类似，zsh也可以让你查看曾经输入过的命令并对命令进行补充、编辑或者拼写检查。它还可以让你使用任务控制来管理运行中的程序。这个 shell 有高级的命令行参数用来搜索和匹配文件格式。
- 这个 shell 的系统级初始化文件存放在子目录/etc中，包括：/etc/zlogin, /etc/zlogout, /etc/zshenv和/etc/zshrc。
- 如果zsh在用户主目录中找不到以句号.开头的与上面列出的文件相对应的文件如.zlogin等的话，就会分析这些保存在/etc目录中的文件。与其他的 shell—如bash或者tcsh相比，这个 shell 有着更多的命令行参数。你会发现这个 shell 有类似于所有其他 shell 的特色。

zsh II

- zsh能在自身进程内进行浮点数运算。
- 关于这个 shell 有许多的资料：它有多达 17 章的手册页，另外还有一个装满帮助文件、示范以及其他最新的有用信息的/usr/doc/zsh目录。

对 shell 进行定制设置

- 使用 shell 的时候，是在一个包含了“环境变量”的环境中运行 shell 的。环境变量是那些在用户主目录和目录/etc中的各种资源文本文件中预先定义使用的数值。对bash来说，缺省的全局性系统环境变量是在文件/etc/profile中定义的，而一些定制的设置可以在用户主目录中的文件.bashrc中找到。
- 有许多不同的环境变量。使用printenv命令或者set命令可以看到一个当前正在使用中的环境变量的清单。
- 对于我们来说，目前最重要的就是$PATH变量。可以修改.bash-profile或者/etc/profile文件来改变这个变量的缺省值。
- 此外，需要了解的还有：$PS1变量和alias命令和/etc/shells文件。

在后台运行程序 I

- 大多数的 shell 还都提供了一种以“后台”进程的方式启动并执行程序的方法。在后台启动某个程序的意思是这个命令继续在内存中执行，而与此同时，shell 命令行的控制权已经返回到了控制台。这是完成工作的一个便利的方法。
- 对巨型文件进行排序或者对子目录和其他类型的文件系统进行搜索等等操作都是可以放到后台去执行的好例子。
- 在后台运行一个进程：

```
1 command &
2 nohup command &
```

- 显示运行进程：

```
1 ps -ax
```

在后台运行程序 II

- 杀掉一个进程：

```
1 kill pid
2 killall pname
```

作业控制

- bash拥有很强的作业管理能力。
- 可以简单地将命令行上一行输入当作是一个作业。
- 作业控制相关的命令和快捷键有：

&	在后台执行作业
Ctrl-Z	挂起前台作业
Ctrl-C	向前台作业发送SIGINT信号
Ctrl+\	向前台作业发送SIGQUIT信号
jobs	列出当前存在的所有作业
fg n	将n号作业放到前台来执行
bg n	将n号作业放到后台去执行
kill	向指定的进程/作业发送指定的信号

使用重定向和管道 I

- 重定向操作符：改变命令的标准输入和/或标准输出。

<	标准输入重定向
>	标准输出覆盖重定向
>>	标准输出追加重定向

使用重定向和管道 II

- 管道：将前一个命令的标准输出直接联结到后一个命令的标准输入上去。
- 例子：

```
1 ps ax | grep ftpd | wc - l
2 find /home -name *.doc | xargs | \
3   fgrep administration | less
```

建立 Shell 命令脚本程序

- 对于经常需要做的冗长的重复性工作，我们可以建立一个文件，并将其所有命令写入到该文件中，并将该文件设置为可以执行的（脚本文件）
- 过程：编辑文件、存储、添加执行权限、运行
- 脚本编程是 UNIX 环境下很复杂的一个课题。

UNIX 文件系统 I

- UNIX 文件系统简介
  - UNIX 文件系统由一组普通文件、目录文件、设备文件和符号链接组成。所有的成员形成一个树状结构，为用户提供了一种组织、检索和管理信息的便捷、高效的方法。
  - 普通文件是记录在存储介质上的 ASCII 或二进制流。它可以是普通的正文文档、程序的源代码、数据文件、图象文件、声音文件或可执行文件。
  - 目录文件是包含若干个文件和子目录的文件。
  - 设备文件代表连接在系统上的物理设备，分为字符设备和块设备两大类。如终端或磁盘设备。
  - 符号链接是指向另一个文件的文件。
- 用户主目录（**Home Directory**）
  - 用户主目录又称为用户的登录目录或起始目录或用户家目录
  - 用户主目录由系统管理员在创建帐号时建立，每个合法的用户在文件系统中都有一个唯一的起始目录。一般都位于/home目录下

UNIX 文件系统 II

- 用户的主目录属于用户，用户可以在此目录下进行各种文件操作
- 很多系统使用 “~” 代表用户主目录
- 用户每次登录后自动位于其主目录下
- 使用`echo $HOME`可以查看自己的主目录

- 当前目录
  - 当前目录就是用户当前所处的工作目录
  - 用户登录进入系统后的当前目录即为用户的主目录
  - 使用命令`pwd`(print working directory) 可以显示当前工作目录
  - 可以在任何时候使用`pwd`命令来判断你所在文件系统中的位置。

UNIX 文件系统 III

- 路径（path）
  - UNIX 系统中每个文件都有一个唯一的路径名。路径名表明了文件的位置并提供到达他的向导
  - 路径名分为两种：绝对路径和相对路径
  - 绝对路径指出从根目录到此文件的路径
  - 相对路径指明从当前目录到此文件的路径，相对路径名一下列方式开始：
    - 目录或文件名
    - “.”，代表当前目录
    - “..”，代表当前目录的父目录
  - 例；假设用户目前处于目录/home/shop，则document/summary代表最后一行的左边的文件；../marry代表另一个用户的主目录；/home/shop/bin和bin都代表用户shop的bin目录；如果用户改变当前工作目录到根目录/，那么bin则代表/bin目录，而不再是用户shop自己的bin目录了。

UNIX 文件系统 IV

- 目录和文件的命名规则
  - 除/和空字符（\0）外，所有字符都合法
  - 某些具有特别意义的字符最好不用：
    ?,@,#,$,&,(,),\,|,;,',",`,<,>,[,] 等
  - 如果在文件名中使用了空格或制表符，那么在引用文件时必须用引号将其括起来
  - 避免使用+、- 和，作为文件名的第一个字符
  - UNIX 是大小写敏感的

常见目录解释 I

`/bin` binary(二进制) 的缩写，这个目录是对 UNIX 系统习惯的沿袭，存放着使用者最经常使用的命令，例如：`cp`, `ls`, `cat`等

`/boot` 启动 Linux 时使用的一些核心文件

`/dev` device(设备) 的缩写，这个目录下是所有 Linux 的外部设备，其功能类似 DOS 下的`.sys`和 Win 下的`.vxd`。在 Linux 中设备和文件是用同样的方法访问的。例如: `/dev/hda`代表第一个物理 IDE 硬盘

`/etc` 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，对于用户 young 而言，其主目录就是`/home/young`也可以用young 表示

`/lib` library(库) 的缩写，存放着系统最基本的动态链接共享库，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

常见目录解释 II

`/lost+found` 平时是空的，当系统不正常关机后，这里就成了一些无家可归的文件的避难所，有点类似于 DOS 下的`.chk`文件

`/mnt` mount(挂载) 的缩写，系统提供这个目录是为用户挂载别的文件系统时提供统一的挂载点

`/proc` process(进程) 的缩写，虚拟的目录，它是系统内存的映射，可以通过直接访问这个目录来获取系统信息。也即，这个目录的内容不在硬盘上而是在内存里

`/root` 系统管理员、也叫作超级权限者的主目录

`/sbin` s 表示 Super user，存放的是一些系统管理程序

`/tmp` 存放一些临时文件的地方，每次系统重启后会自动清空

`/usr` user 的缩写，最庞大的目录，我们用到的很多应用程序和文件几乎都存放在这个目录下（类似于 Windows 下的“`Programs and Files`”文件夹）

常见目录解释 III

/usr/X11R6	存放 X-Windows 相关内容
/usr/bin	一些应用程序
/usr/sbin	系统应用程序
/usr/doc	各种文档都在这里
/usr/include	Linux 下开发和编译应用程序需要的头文件
/usr/lib	存放一些常用的动态链接共享库和静态档案库
/usr/local	用户安装的东西一般放在这里，里面是一整套类似/usr下的目录设置
/usr/man	手册页存放位置
/usr/src	应用程序的源代码（例如 Linux 内核的）

/var variable(易变的) 的缩写，存放着那些不断变化的东西，比方说：系统日志（各种应用的日志）、用户邮件、打印作业等

UNIX 文件系统的操作

1. 显示文件列表
2. 浏览和加工文件
3. 文件的拷贝、移动和删除
4. 目录操作
5. 设置文件的操作权限
6. 文件的链接

ls

语法: `ls [-options] [filelist]`

说明: `ls`（列出子目录内容清单）命令会是你最常使用的命令之一。通过使用它最简单的格式，`ls`命令可以列出当前子目录下几乎所有的文件。虽然这个命令本身只有两个字母，但是它的命令行参数可能比其他任何程序都多！

如果不指定`filelist`参数，则列出当前目录中的所有文件；`filelist`参数既可以是绝对路径也可以是相对路径

不带任何选项的`ls`命令只列出文件名

ls ll

选项：

-l	长文件名列表，包括文件属性与权限等信息
-d	仅列出目录文件
-a	显示所有文件（包括隐含文件）
-A	显示所有文件（包括隐含文件，以. 开头的文件，但是不包括. 和.. 两个目录）
-m	把文件用逗号分隔显示在一行上
-x	按水平对齐的方式列出文件
-F	特殊显示目录和可执行文件
-R	显示子目录
-t	按时间排序
-r	反序排列
-h	以人类易读的方式（如 GB，KB 等）列出
-i	列出 inode 号码

ls III

-n	列出文件的 uid 和 gid，而非用户与组的名称
-S	基于文件大小排序列出
--full-time	以完整时间模式输出
-c	输出文件的状态修改时间
--time=atime	输出文件的访问时间
--time=ctime	输出文件的修改时间

ls IV

范例：

```
1 [Apple]$ ls -l /home/shop/document
2 drwxr-xr-x 2 shop staff 96  Oct 2 07:15 backup
3 -rwxr-xr-x 3 shop staff 348 Oct 2 08:23 summary
```

- 每行的第一个字符表明文件的类型：

-	普通文件
d	目录
l	符号链接文件
b	块设备文件
c	字符设备文件
p	管道文件
s	套接字文件

ls V

- 后面的九个字符表明文件的操作权限，三位一组，依次代表文件属主、同组用户和其他用户的读、写、执行权限。“-”表示禁止
- 文件大小以字节为单位，时间为文件的最后修改时间，可以使用-u选项显示文件的最后访问时间
- 在用户的主目录下有一隐含文件 .bashrc（用ls -a可查到）在其中加入 alias ls="ls --color" 后运行source .bashrc就可以让ls命令将不同类型的文件显示为不同的颜色。

ls VI

- 颜色含义为（颜色的定义可修改/etc/DIR_COLORS配置文件。）：

绿色	可执行文件
蓝色	目录文件
红色	压缩文件
青色	符号链接文件
灰色	一般文件（没有定义的文件）

显示和加工文件

- 显示文件的命令：cat, more, less, head, tail, od等
- 文件处理命令：diff, grep, sort, wc等
- 文件信息相关命令：file, size等
- 压缩、解压缩、归档相关命令：gzip, bzip2, tar等

cat

语法： cat [-options] [files]

说明： cat命令用于串接文件并将结果打印到标准输出上去。
省略files参数或files参数设为-时cat命令将从标准输入读取数据。
cat还可以用来把文件内容送到屏幕上去显示、通过输出重定向把文件内容送到其他的文件中去（合并文件）。
tac命令类似于cat，不同的是，该命令反向显示文件的内容（以行为单位）。

选项： 如下表所示：

-b	给非空行标上行号
-E	在每行末尾显示 $ 字符
-n	给所有行都标上行号
-s	将连续的多个空行压缩为一个
-T	将 TAB 键显示为^I
-v	显示非打印字符

cat II

范例： 以下为几个例子：

1. 如果想使用cat命令查看一个短文件，可以输入如下的命令：

```
[Apple]$ cat t1.txt
```

2. 同时查看几个文件是cat命令的一种用法（cat *.txt），还可以使用cat命令和重定向操作符（“>”）来合并文件。如果想把文件t1.txt和文件t2.txt合并到另外一个叫做t3.txt的文件中去，可以使用下面的方法：

```
[Apple]$ cat t* > t3.txt
```

相当于

cat III

```
1 [Apple]$ cat t1.txt t2.txt > t3.txt
```

3. 如果只是想把t1.txt和t2.txt文件合并，但是并不想再生成另外一个更大的文件的时候时，可以考虑把t1.txt的内容加到文件t2.txt中去，或者把t2.txt的内容加入t1.txt中去。然后，使用cat命令和重定向符 “>>”，敲入下面的内容：

```
1 [Apple]$ cat t1.txt >> t2.txt
```

4. 建立一个短的文本文件。因为cat命令可以读取标准输入，所以可以使用cat命令建立一个文件并通过键盘直接向这个文件中输入内容。我们以myfile.txt为示例文件说明如下。先输入以下命令：

cat IV

```
1 [Apple]$ cat > myfile.txt
```

输入一些文本内容，按下Ctrl+D（EOF）组合键关闭这个文件。看看是否完成了操作：

```
1 [Apple]$ ls -l myfile.txt
2 [Apple]$ cat myfile.txt
```

其他： hexedit可以十六进制的形式查看/编辑文本/二进制文件。

more

语法： more [-options] [files]

说明： more命令是 Linux 操作系统命令中我们称之为“页命令”的家族中的一员。页命令使你在浏览文件的时候可以一次阅读一屏或者一行。这在阅读大量使用手册页的时候特别有帮助，因为man命令是使用了一个页命令来显示每一页的。more命令是一个传统意义上的页命令，因为它提供了早期页命令的基本特色，可以在命令行上像下面这样使用more命令，如下所示：

```
1 [Apple]$ more longfile.txt
```

more II

需要帮助时，按下“h”键，将看到一个帮助画面。如果使用了惊叹号（!），还可以从more命令中去执行其他的命令。阅读一个文本文件是相当容易的，因为可以敲空格键阅读后一页，也可以敲b键阅读前一页。

more命令也有一些命令行参数。在各种常见的参数之外，还可以自行设置屏幕提示（more命令会显示正在阅读的文件的读取百分比）、设置屏幕画面大小（前后翻阅文本时显示的行数）、使用多个文件名或通配符及打开或关闭滚屏功能。

less

语法: less [-options] [files]

说明: 类似于more命令，但是具有反向翻页浏览的功能。
less不需要读取整个文件就可以开始显示，因此在操作大文件时速度较快。
虽然可能会觉得在阅读文件方面more命令的表现相当不错，可是less页命令可能会更让你喜欢。
less命令和more命令一样都是页命令。但是它的编写者Mark Nudelman，改进了more命令中的一些特色，并又添加了许多其他的特色：

- 可以使用光标键在文本文件中前后滚屏。
- 可以用行号或百分比作为书签来浏览文件。
- 可以实现在多个文件中进行复杂的检索、格式匹配、高亮显示等操作。
- 键盘操作与字处理程序如emacs兼容。

less II

- 阅读到文件结束或者标准输入结束的时候less命令不会退出。
- 屏幕底部的信息提示更容易控制使用，而且提供了更多的信息。
- 带有许多的附件，包括一个独立的键定义程序lesskey，这样就可以定义使用哪些按键来控制less命令。

当安装了 Linux 操作系统之后，less页命令将是许多程序（比如man命令等）使用的缺省的页命令。如果想阅读压缩文件（那些带有.gz后缀的文件，可以使用保存在子目录/usr/bin下的zless命令。

head

语法: `head [-options] [files]`

说明: 将每个指定文件的前 10 行显示在标准输出上。不指定files参数或files参数为-时将从标准输入中读取文本。

选项:

-cN	输出前N个字节对应的文本
-nN	输出前N行文本
-q	不输出相应的文件名
-v	输出相应的文件名
--help	显示简短帮助信息
--version	显示版本信息

tail

语法: tail [-options] [files]

说明: 将每个指定文件的最后 10 行显示在标准输出上。不指定files参数或files参数为-时将从标准输入中读取文本。

选项:

-cN	输出最后N个字节对应的文本
-nN	输出最后N行文本
-f	如果文件在查看时发生了改变：被改名、文件尾部追加了新的文本，则tail将会自动跟踪这些变化
-q	不输出相应的文件名
-v	输出相应的文件名
-sN	每隔N秒钟重新打开文件并产生输出

od

语法: od [-t TYPE] [files]

说明: 按照用户指定的形式输出文件的内容。

选项: -t选项后的TYPE参数的取值如下表所示:

a	默认方式显示
c	使用 ASCII 字符来输出
d[size]	使用 decimal 来输出数据，每个整数占用size bytes
f[size]	使用 float 来输出数据，每个数占用size bytes
o[size]	使用 octal 来输出数据，每个整数size bytes
x[size]	使用 hexadecimal 来输出数据，每个整数占用size bytes

diff

语法： `diff [-options] file1 [file2]`

说明： 命令`diff`用于识别文件之间的差别，它将`file2`作为参考点，返回将`file1`改变成和`file2`一样的`ed`命令。

wc

语法： wc [-options] [files]

说明： 命令wc用于统计指定文件中的字符、词和行的数量，可同时处理多个文件。

选项：

-l	仅统计行数
-w	仅统计单词数
-c	仅统计字节数
-m	仅统计字符数

file

语法： `file [-options] files`

说明： 命令`file`用于判断给定文件的类型、采用的字符编码集等信息。

size

语法: size [-options] [files]

说明: 命令size用于输出给定目标文件或归档文件的代码段、数据段等各个部分的大小以及总计大小，以字节为单位。若未给出files，则缺省处理当前目录下的a.out文件。

touch

语法： touch [-options] [files]

说明： 修改指定文件的时间标记，缺省时将指定文件的 LAT 和 LMT 均改为当前时间。若指定文件不存在，则创建空文件（若未指定-c选项）。

选项：

-a	只修改 LAT
-c	不创建文件
-h	处理符号链接文件本身
-m	只修改 LMT
-r FILE	使用FILE的时间戳而不是当前时间
-t STAMP	使用指定的时间而不是当前时间

gzip

语法: gzip [options] [filenames]

说明: gzip命令使用 LZ77 算法压缩/解压缩指定的文件。每个指定的待压缩文件将被替换为相应的压缩后的文件，gzip命令会自动在原文件名后面加上 gz 的后缀。解压缩时会自动去掉后缀。gzip命令自动忽略符号链接文件。
gzip命令还可用于解压缩那些由deflate, compress, lzh和pack命令压缩后的文件。

其他: 相关命令还有gunzip, zcat.

gzip II

选项：

-c	将压缩/解压缩的结果送到标准输出上
-d	解压缩
-v	回显
-l	列出被压缩文件的详细信息
-f	强制覆盖现存的文件
-r	递归
-t	检查待解压缩文件的完整性
-1 to -9	fast to best

bzip2

语法： bzip2 [options] [filenames]

说明： bzip2命令使用 Burrows-Wheeler 算法和 Huffman 编码压缩/解压缩指定的文件。其压缩效果普遍优于基于 LZ77/LZ78 算法的传统压缩工具，甚至能接近基于 PPM 算法的统计压缩工具。
每个指定的待压缩文件将被替换为相应的压缩后的文件，bzip2命令会自动在原文件名后面加上 bz2 的后缀。解压缩时会自动去掉后缀。

其他： 相关命令还有bunzip2, bzcat, bzip2recover.

bzip2 II

选项：

-c	将压缩/解压缩的结果送到标准输出上
-d	解压缩
-z	压缩
-t	检查待解压缩文件的完整性
-f	强制覆盖现存的文件
-1 to -9	fast to best

tar

语法： tar <operation> [-options]

说明： tar命令用于将若干文件保存到一个单独的归档文件中去，或是从指定的归档文件中提取文件。这个归档文件通常被称为**tarfile**。最早的tar命令是被设计用于磁带机这种外部存储设备的（**tar** = tape archive）。

操作：

-A	串接
-c	创建
-d	比较
-r	追加
-t	列表
-u	更新
-x	提取
--delete	删除

tar II

选项：

-f	指定归档/提取的文件
-j	使用bzip2
-v	回显
-z	使用gzip

tar III

示例：

```
1  tar -xvf foo.tar
2        # verbosely extract foo.tar
3
4  tar -xzf foo.tar.gz
5        # extract gzipped foo.tar.gz
6
7  tar -cjf foo.tar.bz2 bar/
8        # create bzipped tar archive of the
9        # directory bar called foo.tar.bz2
10
11 tar -xzf foo.tar.gz blah.txt
12       # extract blah.txt from foo.tar.gz
```

cp

语法: `cp [options] src_file_list dst_file`

说明: 源文件可以是一个文件列表，此时目的文件应该是一个目录，如果目的文件是一个目录，那么将源文件（列表）拷贝到目的目录。

在目录备份操作中，经常会使用到`-a`(相当于`-pdr`) 选项，`-p`意为复制文件属性，`-d`是当文件为连接文件，复制连接文件属性而非文件本身，`-r`为递归拷贝整个目录

`-l`的意思是以硬连接的方式复制文件，`-s`为以符号连接的方式复制文件

`-u`若目标文件比源文件旧时，才更新目标文件

`-i`交互式方式拷贝，若目标文件已存在，则事先询问

mv

语法： `mv [options] src_file dst_file`

说明： mv可以对一个文件/目录重命名，或者把文件/目录从一个目录移到另外一个目录。
选项有-f和-i，分别对应强制和交互。

rm

语法： `rm [-options] [files]`

说明： `rm`用于删除不需要的文件
使用`-r`选项可以递归地删除一个目录，`-f`选项强制删除（无论是否写保护），`-i`选项表示交互（要求用户确认）。

cd

语法： `cd [-L|-P] [dir]`

说明： 改变当前目录至给定的dir位置。不指定dir参数时，将使用 shell 中的系统环境变量HOME的值。当dir指定为-时表示返回最后所处的目录。

选项：

-P	使用物理目录结构而不是缺省的按照符号链接来进行
-L	强制使用符号链接

cd II

范例：

```
1 [Apple]$ cd /usr/bin
2 [Apple]$ cd ..
3 [Apple]$ cd ../..
```

你总能够用下面的命令回到自己的主目录：

```
1 [Apple]$ cd ~
```

或者

```
1 [Apple]$ cd
```

mkdir

语法： `mkdir [-options] directories`

说明： 创建目录。

选项：

-m	在创建时指定权限，而不是采用777-umask
-v	回显

rmdir

语法: `rmdir [-options] directories`

说明: 删除空目录。如果给定的目录不为空，则删除失败。

选项:

-p	删除完指定的目录后，继续删除给定的路径中的每一个目录：‘rmdir -p a/b/c’ 等效于 ‘rmdir a/b/c a/b a’
-v	回显

文件权限详解 I

- 使用`ls -l`列出当前目录，输出结果可能如下所示：

```
-rw-r--r-- 2 root root 12306 Sep 4 18:45 install.log
```

- 依次解释如下：
  - 第一个字符为文件类型（普通文件、目录、符号链接、管道、块设备、字符设备）
  - 三位一组的访问权限，共计 9 个字符，分别表示用户、同组用户以及其他用户的读、写、执行权限
  - 链接数（有多少个不同的目录项指向同一个 i-node）
  - 文件所有者用户名以及文件所有者所属的首要组组名
  - 文件大小，默认单位为字节
  - 创建日期时间或最后修改日期时间
  - 文件名
- 文件访问权限
  - `r`，可以读取此文件的实际内容，如读取文本文件的文字内容

文件权限详解 II

- w，可以编辑、新增或者是修改该文件的内容（但是不包括删除该文件本身）
- x，该文件具有被系统执行的权限

- 目录访问权限
  - r，表示具有读取目录内容的权限，例如，该权限位设置后，即可用 ls 命令列出该目录下的文件名
  - w，表示具有更改该目录内容的权限，也即：创建新文件或目录；删除已经存在的文件或子目录（不管该文件或子目录的自身权限）；将已存在的文件或子目录重新命名；移动该目录内的文件、子目录位置
  - x，表示用户能否进入该目录，使之成为当前工作目录
- 当创建新文件时，其默认权限为0666-umask；创建目录时，其默认权限为0777-umask
- 文件隐藏属性可通过chattr修改或者lsattr查看
  - chattr [+-=][ASacdistu] 文件或目录
  - +-= 增加、删除或者设置某个特殊参数

文件权限详解 III

- A，atime 保持不变
- S，文件同步写入磁盘
- a，该文件只能追加数据，不能删除数据也不能修改数据
- c，设置自动压缩存储
- d，不会被dump(8) 程序备份
- i，该文件不能被删除、改名、设置链接、修改数据
- s，文件删除时，被完全从磁盘空间删除
- u，文件删除时，实际数据内容还在磁盘，未来可恢复

- 一些特殊权限，SUID，SGID，SBIT
  1. SUID
     - 示例：/usr/bin/passwd的权限
     - SUID 只对二进制程序有效（脚本无效），对目录无效
     - 执行者对于该程序需要有可执行权限
     - 本权限只在该程序执行过程中生效
     - 执行者在执行该程序过程中，具有该程序所有者的权限
  2. SGID

文件权限详解 IV

- 示例：/usr/bin/locate，其数据文件为/var/lib/mlocate/mlocate.db
- SGID 只对二进制程序有效，执行者在执行过程中，会获得该程序所有者所属的用户组的权限

3. SBIT
   - 示例：/tmp目录，Sticky Bit，只对目录有效
   - 对于用户在此目录下创建的文件或目录，只有自己或 root 才有权利删除
4. 特殊权限 bit 位说明，SUID 为 4，SGID 为 2，SBIT 为 1，例如：`chown 07777 somefile`，此外，SUID 可以通过 u+s 设置，SGID 可以通过 g+s 设置，SBIT 可以通过 o+t 设置

新建文件的权限

- 当建立一个文件时，系统根据一个掩码（umask）自动授予各个用户一定的权限。
- 这个掩码在/etc/bashrc中被指定为 002 或 022，每当用户创建一个新文件时，其权限将被设定为666-umask，当用户创建一个新的目录时，其权限设定为777-umask。
- 可以通过umask命令返回和修改这个缺省值。可以通过chmod命令直接修改现有文件的访问权限。
- 文件的属性还包括所有者、所有者所属的组等。对这两个信息的修改可以使用chown和chgrp命令。

chmod

语法: chmod [-options] mode files

说明: 对现有文件的操作权限进行修改

选项:

-R	递归
-f	抑制绝大多数的错误信息
--reference=RFILE	将文件的权限设为与参考文件RFILE一样

chmod II

mode为设定的权限，有两种设定方式：符号方式和八进制数值方式

1. 符号方式的mode格式为：user operator access
   user表示用户的分类：

u	文件的属主
g	同组的成员
o	其他用户
a	所有用户

chmod III

operator表示设置运算符：

+	添加
−	清除
=	赋予

chmod IV

access表示权限，可以用字母或八进制数两种形式：

r	允许读
w	允许写
x	允许运行
s	设置用户/组 ID
t	设置粘着位

chmod V

2. 当使用八进制数方式时，每位八进制数代表一组用户的读、写、执行权限，其值应该是下表中一个或数个值的和：

4000	The set-user-ID-on-execution bit.
2000	The set-group-ID-on-execution bit.
1000	The sticky bit.
0400	Allow read by owner.
0200	Allow write by owner.
0100	For files, allow execution by owner. For directories, allow the owner to search in the directory.
0040	Allow read by group members.
0020	Allow write by group members.

chmod VI

0010	For files, allow execution by group members. For directories, allow group members to search in the directory.
0004	Allow read by others.
0002	Allow write by others.
0001	For files, allow execution by others. For directories allow others to search in the directory.

chmod VII

范例： 如果用户希望将当前目录下的文件table的权限改为 -rw-r-xr--，那么可以使用命令

```
1 chmod u-x,o-x table
2 chmod uo-x table
3 chmod u=rw,o=r table
4 chmod 654 table
```

中的一个。

chown

语法： `chown [-options] owner[:[group]] files`

说明： 改变文件的所有者和组。`-R`选项表示递归。

chgrp

语法： `chgrp [-options] group files`

说明： 改变文件的组。`-R`选项表示递归。

文件的链接

- 在 UNIX 系统中，多个文件名可以指向存储介质中的同一个数据区，类似于 Windows 中的快捷方式。这种方式称为文件的链接。
- 文件的链接分为符号链接和硬链接两种。
  1. 符号链接的文件中存储的是原文件的绝对/相对路径。
  2. 硬链接的文件则确实指向了原文件的数据区。

ln

语法: `ln [-options] src [dst]`

说明: `ln`命令用于在文件之间创建链接。

选项:

`-b`	为每个目标文件创建一个备份
`-d`	为目录创建硬链接
`-i`	交互
`-s`	创建符号链接

man

语法: man [options] [section] names

说明: man命令格式化并输出指定关键字的联机手册页。
如果指定了section，则man只在指定的小节中搜索关键字。

选项:

-a	显示搜索到的所有和指定的关键字有关的手册页，而非缺省时的第一个匹配
-f	等效于whatis命令
-k	等效于apropos命令
-w	不输出手册页的内容，而是输出手册页对应文件的路径名
-P pager	使用指定的分页显示程序替代缺省的less

man II

关于 Linux 操作系统首先要知道的事情之一是可以非常容易获得帮助。象大多数 UNIX 运行版本一样，Linux 操作系统发行版本也为几乎每个程序、工具、命令或系统调用编制了使用手册页。可以得到几乎所有命令的有关信息，包括man命令本身。举例来说，输入下列命令就可以阅读man命令的使用手册页：

```
1 [Apple]$ man man
```

使用手册页就像一部 UNIX 的命令汇总。每页使用手册页文件的名称以一个个位数字作为文件后缀，存放在/usr/man的一个子目录下。许多 Linux 操作系统命令的使用手册页都在最初的安装过程中或者在单独安装某个命令程序的时候拷贝到硬盘。比如，man命令的使用手册页文件被命名为：/usr/man/man1/man.1.gz

man III

- Linux 操作系统的使用手册页的组成部分

/usr/man/man1	命令 —在 shell 中执行的命令
/usr/man/man2	系统调用 —关于核心函数的文档
/usr/man/man3	库调用 —libc 函数的使用手册页
/usr/man/man4	特殊文件 —关于/dev目录中的文件的信息
/usr/man/man5	文件格式 —/etc/passwd和其他文件的详细格式
/usr/man/man6	游戏
/usr/man/man7	宏命令包 —对 Linux 文件系统、使用手册页等的说明
/usr/man/man8	系统管理 —根操作员操作的使用手册页
/usr/man/man9	核心例程 —关于 Linux 操作系统内核源例程或者内核模块技术指标的文档

man IV

- 手册页的组织格式

小节名称	说明
Name	命令的名称及简单说明
Synopsis	如何使用这个命令及命令行参数
Description	对这个程序命令及其参数的解释
Files	这个命令用到的文件清单和它们存放的位置
See Also	有相互联系的使用手册页的清单
Diagnostics	特殊输出情况的说明
Bugs	编程漏洞
Author	程序的主要编写者和其他维护人员

man V

- man命令的配置
  man命令对使用手册页进行检索的时候，其依据主要是根据在子目录 /etc中man.conf文件内容中详细说明的原则。这些原则规定了查找使用手册页的缺省子目录。查找这些使用手册页的缺省位置有：MANPATH=/usr/man:/usr/local/man: /usr/X11R6/man

pwd

语法： pwd [-LP]

说明： 打印当前工作目录的绝对路径。

选项： 如下表所示：

-L	在输出中允许使用符号链接
-P	在输出中不允许使用符号链接

pwd命令告诉你自己所在的位置，显示当前工作子目录。例如，执行命令：

```
1 [Apple]$ cd /usr/bin
2 [Apple]$ pwd
```

会看到：/usr/bin

find

- 使用find命令在子目录中搜索匹配的文件，详见第六章。

whereis

语法: whereis [-options] [files]

说明: 定位指定命令的源文件、二进制文件和手册页文件的位置。

选项:

-b	只搜索二进制文件
-m	只搜索手册页文件
-s	只搜索源文件
-u	搜索不寻常的位置
-B	指定二进制文件的搜索路径
-M	指定手册页文件的搜索路径
-S	指定源文件的搜索路径
-f	用于表示-BMS的结束和files参数的开始

whereis II

whereis命令可以迅速地找到文件，而且它还可以提供这个文件的二进制可执行文件、源代码文件和使用手册页存放的位置。例如，下面的命令给出 find命令是放在子目录/usr/bin中的；而它的使用手册页是放在子目录/usr/man/man1 中的：

```
1 [Apple]$ whereis -b find
2 find: /usr/bin/find /usr/man/man1/find.1.gz
```

范例： 搜索当前目录下那些没有文档的文件：

```
1 [Apple]$ whereis -m -u *
```

which

语法： `which [-a] files`

说明： 返回给定可执行文件（命令）的路径名。

选项： `-a`选项显示相应命令的全部路径名。

whatis

语法: whatis keywordlist

说明: whatis在一系列包括一些简短描述信息的数据库文件中搜索指定的关键字并将搜索结果打印到标准输出上去。搜索模式为精确匹配。

whatis命令的数据库文件是由/usr/sbin/makewhatis命令创建的。

whatis命令能够从某个程序的使用手册页中抽出一行简单的介绍性文字，帮助你迅速了解这个程序的具体功能。例如，如果希望了解who命令有什么作用，可以输入下面的内容：

```
1 [Apple]$ whatis who
```

显示器清屏，然后whatis命令显示下面这样的一行文字：

```
who(1) - show who is logged on
```

df

语法： df [-options] [files]

说明： 显示磁盘空间的使用情况。不带任何参数的df命令将依次显示各个文件系统的名称、空间总额、已用和剩余的空间大小、已用空间的百分比以及此文件系统的挂装点。

选项： 如下表所示：

-a	包括那些大小为 0 块的文件系统
-h	以更容易阅读的形式输出
-k,-m	以 1024 字节或 1048576 字节为单位
-l	只输出本地文件系统的信息
-T	显示每个文件系统的类型范例

范例：

df ll

```
1 [Apple]$ df
2 Filesystem  1k-blocks      Used Available Use% Mounted on
3 /dev/hda2     5036316   2328576   2451908  49% /
4 /dev/hda1       62193      8850     50132  16% /boot
5 /dev/hda5     2869900     50864   2673248   2% /home
6 none            63316         0     63316   0% /dev/shm
```

du

语法： `du [-options] [files]`

说明： 计算每个指定文件占用的磁盘空间。对于目录会递归地处理。

选项： `-s`不回显每个文件的详细信息，`-h`以便于阅读（human readable）的形式给出结果。

范例： 命令`du -sh lecture`用于统计当前目录下的`lecture`子目录所占用的磁盘空间。

top

语法: top [options]

语法: top -bcHisS -d sec -n ite -p pid[,pid ...]

说明: top命令以表格的形式给出当前正在运行的系统的一个动态的实时报告。

选项:

-b	批处理模式。top将不接受输入，迭代运行直到迭代次数到达-n选项指定的数值ite
-c	命令行模式
-H	显示线程信息
-i	显示 idle 和 zombie 状态的任务
-s	安全模式
-S	此选项打开时，显示的任务时间为此进程极其已死的子进程所用的总 CPU 时间

top II

内部命令： top拥有一些列的内部命令，最基本的是帮助（'h' 或 '?'）和退出（'q'）。当然 Ctrl-c 也可以终止top的运行。

- top的输出形式如下所示：

```
[Apple]$ top
top - 15:06:43 up 67 days,  1:25, 23 users,  load average: 5.20, 5.20, 5.00
Tasks: 303 total,  11 running, 289 sleeping,   2 stopped,   1 zombie
Cpu(s): 48.2%us,  2.7%sy,  0.0%ni, 49.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3359752k total,  2209228k used,  1150524k free,   541496k buffers
Swap:  3148732k total,      108k used,  3148624k free,   728504k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
25442 yhzhao    25   0  1484  368  316 R  100  0.0 344375:29 a.out
20030 hop       15   0  133m  21m  12m S    1  0.6   1:51.73 gnome-terminal
12607 hop       15   0 16016  11m 3380 S    0  0.3   3:36.39 Xvnc
    1 root      15   0  2032  640  552 S    0  0.0   0:00.40 init
    2 root      RT   0     0    0    0 S    0  0.0   0:00.57 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/0
    4 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/0
    5 root      RT   0     0    0    0 S    0  0.0   0:00.44 migration/1
    6 root      39  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/1
    7 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/1
    8 root      10  -5     0    0    0 S    0  0.0   0:00.42 events/0
    9 root      10  -5     0    0    0 S    0  0.0   0:00.06 events/1
```

top III

```
21    10 root      17  -5     0    0     0 S    0  0.0   0:00.00 khelper
22    11 root      10  -5     0    0     0 S    0  0.0   0:00.00 kthread
23    15 root      10  -5     0    0     0 S    0  0.0   0:01.17 kblockd/0
24    16 root      20  -5     0    0     0 S    0  0.0   0:00.47 kblockd/1
25    17 root      14  -5     0    0     0 S    0  0.0   0:00.00 kacpid
26   123 root      14  -5     0    0     0 S    0  0.0   0:00.00 cqueue/0
27   124 root      14  -5     0    0     0 S    0  0.0   0:00.00 cqueue/1
```

- 整个终端屏幕被分为总结信息区、消息提示行和一个由 task 构成的表格。
- 第 1 行：命令名、系统运行时间、当前登录用户数、平均负载（1/5/15 分钟内的负载数，即运行队列中的平均进程数量，大于 5 的话就表示性能有严重问题）。
- 第 2 行：进程总数及各类进程数量；
- 第 3 行：CPU 占用情况，分别为用户态、系统态、改变过优先级的进程、空闲、等待 I/O、硬中断、软中断占用 CPU 的百分比；
- 第 4 行：内存使用情况；

top IV

- 第 5 行：交换分区使用情况；
- 实际的可用内存大约等于第 4 行的 free + 第 4 行的 buffers + 第 5 行的 cached;
- task 表头：

PID	任务的 pid
PPID	任务的 ppid
RUSER	任务所有者的真实 user name
UID	任务所有者的有效 user id
USER	任务所有者的有效 user name
GROUP	任务所有者的有效 group name
TTY	任务的控制终端
PR	任务的优先级
NI	任务的 nice 值，负值表示高优先级，正值表示低优先级

top V

P	最后使用的 CPU (SMP)
%CPU	上次更新到现在的 CPU 时间占用百分比
TIME	任务使用的 CPU 时间总计，单位秒
TIME+	任务使用的 CPU 时间总计，单位 1/100 秒
%MEM	任务所使用的物理内存的百分比
VIRT	任务所使用的虚拟内存总量，VIRT=SWAP+RES
SWAP	任务所使用的交换分区数量 (kB)
RES	任务所使用的非交换内存数量，RES=CODE+DATA
SHR	任务所使用的共享内存数量
S	任务状态，D= 不可中断的睡眠状态，R= 运行，S= 睡眠，T= 跟踪/停止，Z= 僵尸
COMMAND	任务的启动命令

top VI

- 'b', 'x', 'y' 切换高亮形式
- '>', '<' 改变排序字段,'R' 切换排序模式（升序/降序）
- 'z' 切换彩色和黑白
- 类似的工具还有：

iotop	simple top-like I/O monitor
iftop	display bandwidth usage on an interface by host
powertop	a power consumption and power management diagnosis tool
htop	interactive process viewer
atop	an interactive monitor to view the load on a Linux system

free

语法： free [options]

说明： 显示系统当前的内存和交换分区使用情况。

选项：

-b|k|m	显示的单位分别为字节、千字节和兆字节
-l	显示详细的低、高内存统计信息
-o	使用旧式格式
-t	额外显示一行总计信息
-s n	每n秒更新一次
-c m	共更新m次

date

语法： `date [options] [+format]`
`date [-u|--utc|--universal] [MMDDhhmm[[CC]YY] [.ss]]`

说明： 此命令用于显示和修改系统时间（仅超级用户）。

范例：

```
1 [Apple]$ date +%b%d日%A%H时%M分
2 10月15日星期二18时18分
```

选项：

-I,-R	选择日期时间显示格式的标准
-s	设置日期时间
+FORMAT	设置输出格式（见下表）

date II

若不以加号作为开头，则表示要设定时间，此时时间格式应为 MMDDhhmm[[CC]YY][.ss]，其中MM为月份，DD为日，hh为小时，mm为分钟，CC为年份前两位数字，YY为年份后两位数字，ss为秒数。

输出格式控制符：

- 时间方面：

%%	输出%
%n	下一行
%t	跳格
%H	小时（00..23）
%I	小时（01..12）
%k	小时（0..23）
%l	小时（1..12）

date III

%M	分钟（00..59）
%p	显示本地 AM 或 PM
%r	直接显示时间（12 小时制，格式为 hh:mm:ss [A|P]M）
%s	从 1970 年 1 月 1 日 00:00:00 UTC 到目前为止的秒数
%S	秒（00..60）
%T	直接显示时间（24 小时制）
%X	相当于%H:%M:%S
%Z	显示时区

date IV

- 日期方面:

%a	星期几（Sun..Sat）
%A	星期几（Sunday..Saturday）
%b	月份（Jan..Dec）
%B	月份（January..December）
%c	直接显示日期与时间
%d	日（01..31）
%D	直接显示日期（mm/dd/yy）
%h	同%b
%j	一年中的第几天（001..366）
%m	月份（01..12）
%U	一年中的第几周（00..53）（以 Sunday 为一周的第一天的情形）

date V

%w	一周中的第几天（0..6）
%W	一年中的第几周（00..53）（以 Monday 为一周的第一天的情形）
%x	直接显示日期（mm/dd/yy）
%y	年份的最后两位数字（00.99）
%Y	完整年份（0000..9999）

cal

语法: cal [-options]

语法: ncal [-options]

说明: 显示当月日历，可指定月份。ncal可显示儒略日及本年度的复活节。

其他: 类似的命令还有

- calendar 显示“历史上的今天”
- calcurse 提供日历和日程安排服务
- gcal GNU 版本的cal，拥有非常丰富和强大的功能
- remind 提供复杂的提醒服务
- wyrd remind的前端，提供复杂的日历和提醒功能
- taskwarrior 是一个基于命令行的 TODO 列表管理工具

who

语法： who [-options] [FILE | ARG1 ARG2]

说明： 根据文件/var/log/utmp中的数据显示当前登录的用户信息。参数FILE用于指定信息的来源，例如/var/log/wtmp；ARG1和ARG2通常是 'am i' 或 'mom likes'

选项： 常用选项有-H（显示标题），-q（只显示用户名和相应用户名登录的个数）

last

语法: last [-options] [names] [ttys]

说明: 显示自文件/var/log/wtmp创建以来用户登录信息的历史记录。参数[names] [ttys]用于指定特定的用户及登录的终端。

其他: 命令lastb显示失败的登录尝试的相关信息。

选项: 常用选项:

-f FILE	使用指定的文件FILE作为数据来源而不是/var/log/wtmp
-n NUM	只显示NUM行输出
-t YYYYMMDDHHMMSS	显示在指定的时间的登录信息
-R	不显示主机名字段
-a	将主机名字段放在最右边显示

last II

-d	将远程登录记录中的远程 IP 地址翻译为主机名
-F	输出完整的登录和退出日期时间
-w	显示完整的用户名和主机域名
-x	显示系统关机和运行级别改变记录

nohup

语法： `nohup command [arg]...`

说明： 使运行的命令忽略SIGHUP信号。

nice

语法： nice [-options] [command [arg]...]

说明： 以指定的调度优先级运行命令。如果没有给出参数COMMAND则输出当前的调度优先级。
缺省的优先级为 10，取值范围从 −20 到 19.
普通用户只能将优先级的数值加大，只有超级用户才能减小优先级的值。优先级的值越低表示对 CPU 的占用率越大。

选项： -n表示将优先级增加n。

passwd

语法： `passwd [-l] [-u [-f]] [-d] [-n mindays] [-x maxdays] [-w warndays] [-i inactivedays] [-S] [username]`

说明： 修改登录密码。只有超级用户才能修改其它用户的密码。

选项： `-l`和`-u`选项表示锁定/解除锁定用户，only for root；`-d`选项使指定的用户登录时不需要密码（但是密码依然保存在用户数据库中）；`-S`选项显示某个用户的密码的情况。其它选项用于指定密码的有效期等相关信息。

echo

语法： `echo [-options] [strings]`

说明： 显示字符串。

选项： `-n`表示不显示字符串尾部的新行符，`-e`将解析后续的反斜线转义序列。

uname

语法： uname [-options]

说明： 输出系统信息。

选项： 如下表所示：

-a	all information
-s	kernel name
-n	node name
-r	kernel release
-v	kernel version
-m	machine hardware name
-p	processor type
-i	hardware platform
-o	operating system name

dpkg I

语法： dpkg [options] action

说明： dpkg命令用于安装、构建、删除和管理 Debian 软件包。

- deb 是 Debian Linux 的安装格式，跟 Red Hat 的 rpm 非常相似。
- dpkg是 Debian Package 的简写，是为 Debian 专门开发的套件管理系统，方便软件的安装、更新及移除。
- 所有源自 Debian 的 Linux 发行版都使用dpkg，例如 Ubuntu、Knoppix 等。
- 以下是一些dpkg的普通用法：
  1. dpkg -i pkg.deb
     安装一个 Debian 软件包，通常是手动下载的文件。
  2. dpkg -c pkg.deb
     列出pkg.deb的内容。

dpkg II

3. `dpkg -I pkg.deb`
   从`pkg.deb`中提取软件包信息。
4. `dpkg -r pkg-name`
   移除一个已安装的软件包。
5. `dpkg -P pkg-name`
   完全清除一个已安装的软件包。和`remove`不同的是，`remove`只是删掉数据和可执行文件，`purge`另外还删除所有的配制文件。
6. `dpkg -L pkg-name`
   列出`pkg-name`安装的所有文件清单。同时可用`dpkg -c pkg.deb`来检查一个`pkg.deb`文件的内容。
7. `dpkg -s pkg-name`
   显示已安装软件包的信息。同时可用`apt-cache`显示 Debian 存档中的软件包信息，以及`dpkg -I pkg.deb`来显示从一个`pkg.deb`文件中提取的软件包信息。

dpkg III

8. `dpkg-reconfigure pkg-name`
   重新配制一个已经安装的软件包，如果它使用的是debconf (debconf为软件包安装提供了一个统一的配制界面)。

apt-get

语法： apt-get [options] [subcmd] [pkg-name]

说明： 软件包管理。

- 如果一个软件依赖关系过于复杂，使用dpkg来安装它，并不是一个明智的选择，这个时候就需要用到apt软件包管理系统。apt可以自动地检查依赖关系，通过您预设的方式来获得相关软件包，并自动安装配置它。事实上，在多数情况下，我们推荐您使用apt软件包管理系统。
- apt系统需要一个软件信息数据库和至少一个存放着大量 deb 包的软件仓库，我们称之为“源”。源可以是网络服务器、安装 CD 或者本地软件仓库。您需要修改/etc/apt/sources.list文件，使apt系统能够连接到源。
- Ubuntu 的“系统设置”中的“软件和更新”可以根据主机的网络环境，自动选择速度最快的源。

apt-get II

- apt系统主要包括“apt-get”和“apt-cache”等命令。它们通常都是复合命令，包含若干个子命令。
  - apt-get install pkg-name，安装pkg-name
    -d 仅下载
    -f 强制安装
  - apt-get remove pkg-name，卸载pkg-name
  - apt-get update，更新软件信息数据库
  - apt-get upgrade，进行系统升级
  - apt-cache search pkg-name，搜索软件包
- aptitude是apt命令的一个字符环境下的前端，利用 curse 技术为用户提供了一个好用得多的用户界面。
- 类似的软件包管理工具还有synaptic（新立得）以及各个桌面系统自带的软件更新管理工具。

rpm

语法： rpm [options] [pkg]

说明： RPM (Redhat Package Manager) 软件包管理。

- 安装 RPM 包时需要选项-i，即可以用这样的命令来安装：
  ```
  rpm -i pkg.rpm
  ```
- 但是总是希望能看到一些信息，这时就可以用这样的命令：
  ```
  rpm -ivh pkg.rpm
  ```
  这时就会显示出软件包的安装进度以及安装中的信息等。
- 如果想要安装的软件包系统中已经存在而只需要升级一下的话可以这样来做：
  ```
  rpm -U pkg.rpm
  ```
- 卸载一个已安装的软件包的命令是这样的：
  ```
  rpm -e pkg-name
  ```

rpm II

- 要想列出系统已经安装的 RPM 包的清单可以输入下面的命令：

  ```
  rpm -qa
  ```

  但是此时列出的清单会是很长的，而我们所需要的只是其中的一个，这时我们可以用下列的命令来达到我们的目的：

  ```
  rpm -qa | grep -i 'name'
  ```

- 在软件包的安装过程中我们有可能用到的参数如下：

--force	强行安装
-v	用符号“#”来显示安装进度
--percent	用百分比来显示安装进度
-nodeps	忽视已丢失的依赖性文件强行进行安装
-test	这个参数并不进行实际的安装，而只是检查软件包能否成功安装
-v	让rpm报告每一步的情况，即回显

yum

- **YUM**（全称为 Yellow dog Updater, Modified）是一个在 Fedora 和 RedHat 以及 SUSE 中的文本模式前端软件包管理器。起初是由 Yellow Dog 这一发行版的开发者 Terra Soft 研发，用 Python 写成，那时还叫做**YUP** (Yellow Dog Updater)，后经杜克大学的 Linux@Duke 开发团队进行改进，遂有此名。
- YUM 基於 RPM 包管理，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软体包，无须繁琐地一次次下载、安装。YUM 提供了查找、安装、删除某一个、一组甚至全部软件包的命令，而且命令简洁而又好记。
- YUM 的关键之处是要有可靠的 repository。顾名思义，这是软件的仓库，它可以是 http 或 ftp 站点，也可以是本地软件池，但必须包含 RPM 的 header，header 包括了 RPM 包的各种信息，包括描述、功能、提供的文件、依赖性等。正是收集了这些 header 并加以分析，才能自动化地完成余下的任务。

yum II

- YUM 有以下特点：
  - 可以同时配置多个资源库 (Repository)
  - 简洁的配置文件 (/etc/yum.conf)
  - 自动解决增加或删除 RPM 包时遇到的倚赖性问题
  - 使用方便
  - 保持与 RPM 数据库的一致性
- YUM 的一切配置信息都储存在一个叫yum.conf的配置文件中，通常位于/etc目录下，这是整个yum系统的重中之重：

yum III

```
[Apple]$ more /etc/yum.conf
[main]
  cachedir=/var/cache/yum
  keepcache=0
  debuglevel=2
  logfile=/var/log/yum.log
  exactarch=1
  obsoletes=1
  gpgcheck=1
  plugins=1
  metadata_expire=1800
# PUT YOUR REPOS HERE OR IN separate files
# named file.repo in /etc/yum.repos.d
```

yum IV

- 下面简单的对这一文件作简要的说明：

  `cachedir` YUM 缓存的目录，YUM 在此存储下载的 RPM 包和数据库，一般是`/var/cache/yum`

  `debuglevel` 除错级别，0—10, 默认是 2

  `logfile` YUM 的日志文件，默认是`/var/log/yum.log`

  `exactarch` 有两个选项 1 和 0, 代表是否只升级和你已安装软件包的 CPU 平台一致的包。如果设为 1，则 YUM 不会用 686 的包来升级已安装的 i386 的 RPM 包。

  `gpgchkeck` 有 1 和 0 两个选择，分别代表是否是否进行 gpg 校验

- YUM 的命令一般是如下形式：

  `yum [options] [cmd] [pkg ...]`

  其中常用选项包括`-h`（帮助），`-y`（当安装过程提示选择全部为“yes”），`-q`（不显示安装的过程）等等。`[cmd]`为所要进行的操作，`[pkg ...]`是操作的对象。

yum V

- 常用的命令：
  - 自动搜索最快镜像插件：`yum install yum-fastestmirror`
  - 安装 YUM 图形窗口插件：`yum install yumex`
  - 查看可以批量安装的列表：`yum grouplist`
  - 安装

    `yum install` 全部安装

    `yum install pkg-name` 安装指定的安装包`pkg-name`

    `yum groupinsall grp-name` 安装程序组`grp-name`
  - 更新和升级

    `yum update` 全部更新

    `yum update pkg-name` 更新指定程序包`pkg-name`

    `yum check-update` 检查可更新的程序

    `yum upgrade pkg-name` 升级指定程序包`pkg-name`

    `yum groupupdate grp-name` 升级程序组`grp-name`

yum VI

- 查找和显示
  yum info pkg-name 显示安装包信息pkg-name
  yum list 显示所有已经安装和可以安装的程序包
  yum list pkg-name 显示指定程序包安装情况pkg-name
  yum groupinfo grp-name 显示程序组grp-name信息
  yum search string 根据关键字string查找安装包
- 删除程序
  yum remove|erase pkg-name 删除程序包pkg-name
  yum groupremove grp-name 删除程序组grp-name
  yum deplist pkg-name 查看程序pkg-name依赖情况
- 清除缓存
  yum clean pkgs 清除缓存目录下的软件包
  yum clean headers 清除缓存目录下的headers
  yum clean oldheaders 清除缓存目录下旧的headers
  yum clean, yum clean all (等效于yum clean pkgs; yum clean oldheaders) 清除缓存目录下的软件包及旧的headers

源码包 I

- 对于绝大多数软件，我们建议使用 apt 系统来安装它。在少数情况下，例如某软件没有以 deb 包的格式发布，或者需要定制适合自己的软件，则可以通过编译源代码的方式安装它。
- 首先需要下载软件的源码包，并且将它解包为源代码文件。源码包通常有两种格式，一种是.tar.gz或.tgz，另一种就是.tar.bz2，可分别用以下命令解开：

```
tar -xzvf pkg.tar.gz
tar -xvjf pkg.tar.bz2
```

- 进入解包后的源码目录，仔细阅读README文件和INSTALL文件
- 源码目录中通常有一个configure脚本，用来配置即将开始的编译过程。执行它

```
./configure [--prefix=/usr/loca/xxx ...]
```

它会自动检测软件的编译环境和依赖关系，并且生成Makefile文件。

源码包 II

- 可以使用带参数的命令./configure --help，或者阅读INSTALL文件，查看该脚本允许的参数。例如使用--prefix=/usr/local/xxx参数，将软件的安装目录设定为/usr/local/xxx/。（如果您一定要将软件安装在一个目录下，我们建议您安装在这里）
- 现在执行make命令，系统会根据Makefile文件中的设定，通过make工具调用编译器和所需资源文件，将源代码文件编译成目标文件。

  ```
  make
  ```

- 执行make install命令，make工具会自动将连接目标文件，将最终生成的文件拷贝到Makefile文件设定的路径中，并且完成更改文件的属性，删除残留文件等动作。

  ```
  sudo make install
  ```

  注意：这一步需要超级用户权限，否则安装会失败或者安装的软件只有当前用户能够使用。

源码包 III

- 现在，编译安装过程已经完成，为了更方便地使用它，通常应该给程序的可执行文件作一个符号链接。如果PATH环境变量设置合理的话可以省略这一步。

```
sudo ln -sf /usr/local/xxx/exe /usr/local/bin/exe
```

- 如果在上述的./configure, make和sudo make install三个步骤中任何一步出了问题，不要惊慌！请仔细阅读并保存错误信息，然后自行解决或请教周围的高手或上网 Google。
- 最常见的错误是缺少某个必要的库，只需找到并安装好缺少的库即可解决。

mail

语法: `mail [[-s subject] [-c cc-addr] [-b bcc-addr] to-addr...]`

说明: 收发邮件。不带任何参数时为读取邮件

talk

语法： `talk username [ttyname]`

说明： 与其它用户交谈。若某个用户名有多个登录，可通过`who`命令查看其登录信息并通过`ttyname`来进行区分。

write

语法： `write username [ttyname]`

说明： 从发送方的标准输入读取文本并将其直接发送到目标用户的标准输出上。

wall

语法： `wall username [ttyname]`

说明： 向当前所有在线用户发送消息。

网络相关 I

ip	显示/处理路由、设备、路由策略及隧道等相关信息
ifconfig	显示/设置网络接口配置
dig	DNS 查询工具
netstat	输出网络连接、路由表、网口统计信息、伪装连接和多播成员信息等
nicstat	输出网络流量统计信息
ifstat	输出网口统计信息
tcpdump	卸出指定网络接口上的通信数据
nethogs	一个网络top类工具，以进程为依据统计网络流量
slrum	实时网络负载监测工具
ping	向指定的主机发送 ICMP 协议的ECHO_REQUEST数据报并测量其响应时间

网络相关 II

traceroute	在 IP 层跟踪并输出数据包的路由过程
curl	下载指定的 URL 到本地
wget	非交互式 URL 下载工具，通常用于镜像整个网站到本地
wireshark	功能强大的、图形模式的网络分析仪。甚至扩展到了蓝牙和 USB 接口上
tshark	网络协议分析器，其功能大致为wireshark的子集
mtr	网络诊断工具，整合了traceroute和ping
lynx	文本模式 WWW 浏览器
w3m	文本模式 WWW 浏览器，支持 HTML 页面中的 table 和 frame 特性，同时还可作为 pager 使用
lftp	文本模式 FTP 客户端，支持多种文件传输协议，甚至包括 BitTorrent
gftp	FTP 图形客户端

网络相关 III

`filezilla` 目前最流行的 FTP 图形客户端

`ssh` 安全的远程登录客户端

Just for Fun

- `cowsay`, `cowthink`
- `figlet`
- `fortune`, `fortune-zh`
- `sl`
- `linuxlogo`
- `hack`, `nethack`, `slashem`
- `pacman4console`
- `snake`, `nsnake`, `snake4`
- `celestia`, `stellarium`
- ...

附录一：元字符概述 I

- Shell 对许多字符的处理不按其字面的意思，这样的特殊字符称为**元字符（meta character）**。
- UNIX 中的元字符有：`*!<>(){}[];|?/\$"'` `#&` 以及空格、制表符和新行符。
- 元字符的意义

*	匹配包括空字符的任何字符串，但不包括以点开头的文件名
?	匹配任何单个字符，除了带点文件名的第一个点
[xyz···]	匹配括号内的任意单个字符
[a-z]	匹配此范围内的任何字符
[!xyz]	匹配任何不在括号内的字符
[!a-z]	匹配任何不在此范围内的字符

附录一：元字符概述 II

>	标准输出重定向
>>	标准输出重定向（追加）
<	标准输入重定向
<<	Here document
|	管道。cmd1 | cmd2：将cmd1的输出作为cmd2的输入
;	cmd1 ; cmd2表示先执行cmd1然后再执行cmd2（顺序执行）
&	cmd1 & cmd2表示在后台执行cmd1，而cmd2不必等待cmd1执行完便可执行（异步执行）
`	`cmdlist`表示执行cmdlist列表，并且用最后命令的输出作为对此表达式的引用的输出（命令替换）
()	(cmdlist)表示在子 shell 中运行cmdlist
{}	{cmdlist}在当前 shell 中运行cmdlist
\c	取字符c的字面意思（转义）
\	出现在行尾时表示续行（取消新行符的特殊含义）

附录一：元字符概述 III

'	'str'取字符串（单引号转义）
"	"str"：在$、' 和\ 之外取字符串str
#	shell 注释符

附录二：bash 命令行提示符 I

- 命令行提示符中的颜色是通过 ANSI 转义序列表示的。所有的颜色序列应该用\[\033[和m\] 括起来。

前景色转义序列			
0;30	Black	1;30	Dark Gray
0;31	Red	1;31	Light Red
0;32	Green	1;32	Light Green
0;33	Brown	1;33	Yellow
0;34	Blue	1;34	Light Blue
0;35	Purple	1;35	Light Purple
0;36	Cyan	1;36	Light Cyan
0;37	Light Gray	1;37	White

附录二：bash 命令行提示符 II

背景色转义序列	
40	Black
41	Red
42	Green
43	Brown
44	Blue
45	Purple
46	Cyan
47	Light Gray

附录二：bash 命令行提示符 III

其它颜色转义序列		
0	No-colour	清除
1	Hilight	高亮
4	Underline	下划线
5	Blink	闪烁
7	Inverse	反显
8	Concealed	消隐

附录二：bash 命令行提示符 IV

	光标处理转义序列
nA	光标上移 n 行
nB	光标下移 n 行
nC	光标右移 n 行
nD	光标左移 n 行
y;xH	设置光标位置
2J	清屏
K	清除从光标到行尾的内容
s	保存光标位置
u	恢复光标位置
?25l	隐藏光标
?25h	显示光标

附录二：bash 命令行提示符 V

- 通过下表中的转义序列，用户可以自己定制个性化的命令行参数。

命令行提示符转义序列	
\a	响铃（ACSII 字符 007）
\d	以 “Weekday Month Date” 的格式显示日期
\e	ESC（ACSII 字符 033）
\h	主机名
\H	主机名加域名
\j	shell 目前管理的任务数
\l	shell 终端设备名的 basename
\n	新行符
\r	回车符
\s	当前 shell 的 basename
\t	当前时间的 24 小时格式：HH:MM:SS

附录二：bash 命令行提示符 VI

\T	当前时间的 12 小时格式：HH:MM:SS
\@	当前时间的 12 小时格式：HH:MM:SS am/pm
\u	当前用户名
\v	当前 shell 的版本号
\V	当前 shell 的发布号 version + patchlevel
\w	当前工作目录
\W	当前工作目录的 basename
\!	当前命令的历史编号
\#	当前命令的命令编号
\$	超级用户显示 '#'，普通用户显示 '$'
\nnn	八进制数 0nnn 表示的 ASCII 字符
\\	一个反斜线
\[	界定一个不可打印字符串的开始
\]	界定一个不可打印字符串的结束

附录二：bash 命令行提示符 VII

- 缺省命令行提示符：`[\u@\h \W]\$`
- 一个复杂的命令行提示符示例：

```
1  # Prompt String
2  #
3  case $TERM in
4      xterm*|rxvt*)
5          TITLEBAR='\[\033]0;\u@\h:\w\007\]'
6          ;;
7      *)
8          TITLEBAR=""
9          ;;
10 esac;
11 PS1="${TITLEBAR}\T\[\033[0m\]\[\033[0;33m\]::\
12       \[\033[0m\]\[\033[1;30m\]\[\033[0m\]\
13       \[\033[0;37m\]\u\[\033[0m\]\[\033[0;33m\]@\
14       \[\033[0m\]\[\033[0;37m\]\h\[\033[0m\]\
```

附录二：bash 命令行提示符 VIII

```
15      \[\033[0;33m\]::\[\033[0m\]\[\033[1;33m\]\w\
16      \[\033[0m\]\[\033[0;33m\] \[\033[0m\]\
17      \[\033[0;32m\]\[\033[0m\]\[\033[1;37m\]\
18      \[\033[0m\]\[\033[1;37m\](\[\033[0m\]\
19      \[\033[0;33m\]\$(ls -l | grep \"^-\" | \
20      wc -l | tr -d \" \")\[\033[0m\]\
21      \[\033[1;37m\] \[\033[0m\]\[\033[1;30m\]\
22      \[\033[0m\]\[\033[0;37m\]files\[\033[0m\]\
23      \[\033[1;37m\], \[\033[0m\]\[\033[0;33m\]\
24      \$(ls --si -s | head -1 |\
25      awk '{print \$2}') \[\033[0m\]\[\033[0;37m\]\
26      total\[\033[0m\]\[\033[0;37m\])\n\[\033[0m\]\
27      \[\033[1;37m\]\[\033[0;0m\]"
```

附录三：bash 命令行操作 I

- 历史命令记录的操作。bash的命令行历史记录被保存在~/.bash_history文件中，记录的条数由环境变量HISTSIZE决定。

命令行历史记录操作	
!n	第 n 个历史记录
!-n	倒数第 n 个历史记录
!#	目前已经键入的整个命令行内容
!foo	最近一个以 'foo' 开始的历史记录
!!	上一个命令，等效于!-1
!?foo?	最近一个含有字符串 'foo' 的历史记录，若 'foo' 后有一换行符则尾部的? 可以省略
^foo^bar	重复上一个历史记录，并将其中的 'foo' 替换为 'bar'
!!:0	上一个历史记录的命令名

附录三：bash 命令行操作 II

!!:^	上一个历史记录的第一个参数
!$	上一个历史记录的最后一个参数
!!:*	上一个历史记录的所有参数
!!:x-y	上一个历史记录的第 x 到第 y 个参数
C-s	在历史记录中向后搜索
C-r	在历史记录中向前搜索（重点推荐）

附录三：bash 命令行操作 III

- 当键入一个很长的命令行时出现了输入错误或者想要修改之前的某个历史记录时，就会用到命令行编辑。下表中 C—Ctrl, M—Meta(Alt).

命令行编辑操作	
C-a	移到行首
C-e	移到行尾
C-b	左移一个字符
C-f	右移一个字符
M-b	左移一个单词
M-f	右移一个单词
C-h	删除光标左边的字符
C-d	删除光标所在位置的字符
C-_	撤销编辑

附录三：bash 命令行操作 IV

M-d	剪切到单词尾
C-w	剪切到左边第一个空白
M-BS	剪切到左边单词的第一个字符（含）
C-k	剪切到行尾
C-u	剪切到行首
C-y	粘贴
M-y	循环 kill-ring
M-r	恢复本行的所有修改
C-]	在行内向后搜索
M-C-]	在行内向前搜索
C-t	交换当前光标及其左边的两个字符
M-t	交换当前光标及其左边的两个单词
M-u	将当前单词转换为大写
M-l	将当前单词转换为小写
M-c	将当前单词转换为首字母大写

附录三：bash 命令行操作 V

M-.	上一次命令的最后一个参数打出来
M-*	列出你可以输入的命令

附录三：bash 命令行操作 VI

命令行自动补全操作	
M-/	补全文件名
M-~	补全用户名
M-@	补全主机名
M-$	补全变量名
M-!	补全命令名
M-^	补全历史记录

The End

The End of Chapter III.