Unix/Linux 操作系统使用基础

李会民

hmli@ustc.edu.cn, lihm@qibebt.ac.cn

中国科学院青岛生物能源与过程研究所 超级计算中心

2009 年 12 月

内容

Linux 操作系统简介

- Linux 以它的高效性和灵活性著称。具有多任务、多用户的能力
- Linux 实际应该只是内核，绝大多数基于 Linux 内核的操作系统使用了大量的 GNU 软件，包括了一个 Shell 程序、工具、程序库、编译器及工具，还有许多其他程序，例如 Emacs。正因为如此，GNU(GNU's not Unix) 计划的开创者 Richard Matthew Stallma-RMS 博士提议将 Linux 操作系统改名为 GNU/Linux，但多数人仍然习惯性地使用 Linux
- Linux 之所以受到广大计算机爱好者的喜爱，其主要原因有两个：
  - 它属于自由软件，用户不用支付任何费用就可以它的源代码，并且可以根据自己的需要对它进行必要的修改
  - 它具有 Unix 的全部功能，任何使用 Unix 操作系统或想要学习 Unix 操作系统的人都可以从 Linux 中获益

Linux 操作系统起源

Linux 是一个诞生于网络、成长于网络且成熟于网络的操作系统

- 1991 年，芬兰大学生 Linus Torvalds 萌发了开发一个自由的 UNIX 操作系统的想法，当年 Linux 诞生，为了不让这个羽毛未丰的操作系统夭折，Linus 将自己的作品 Linux 通过 Internet 发布。从此一大批知名的、不知名的 hack、编程人员加入到开发过程中来，Linux 逐渐成长起来
- Linux 一开始是要求所有的源码必须公开，并且任何人均不得从 Linux 交易中获利。然而这种纯粹的自由软件的理想对于 Linux 的普及和发展是不利的，于是 Linux 开始转向 GPL(GNU General Public License)，成为 GNU 阵营中的主要一员
- Linux 凭借优秀的设计，不凡的性能，加上 IBM、INTEL、CA、ORACLE 等国际知名企业的大力支持，市场份额逐步扩大，逐渐成为主流操作系统之一
- Linux 只是内核，即操作系统中允许用户软件与硬件通信的部分

Linux 产商借网络爱好者升级的内核，通过优化、增加功能出售各个版本的 Linux 操作系统

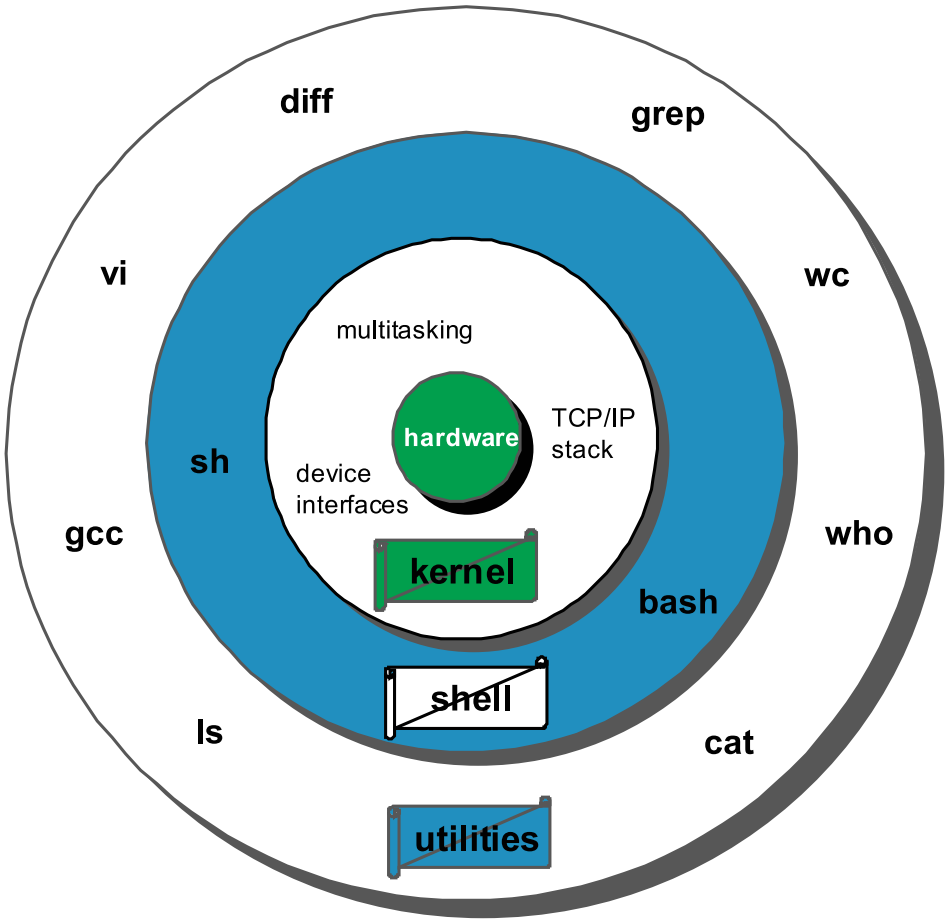
Linux 特点 I

- 开放性：指系统遵循世界标准规范，特别是遵循开放系统互连（OSI）国际标准
- 多用户：是指系统资源可以被不同用户使用，每个用户对自己的资源（例如：文件、设备）有特定的权限，互不影响
- 多任务：它是指计算机同时执行多个程序，而且各个程序的运行互相独立
- 良好的用户界面：Linux 向用户提供两种界面：用户界面和系统调用。Linux 还为用户提供了图形用户界面。它利用鼠标、菜单、窗口、滚动条等设施，给用户呈现一个直观、易操作、交互性强的友好的图形化界面
- 设备独立性：是指操作系统把所有外部设备统一当作成文件来看待，只要安装它们的驱动程序，任何用户都可以象使用文件一样，操纵、使用这些设备，而不必知道它们的具体存在形式。Linux 是具有设备独立性的操作系统，它的内核具有高度适应能力

Linux 特点 II

- 提供了丰富的网络功能：完善的内置网络是 Linux 一大特点
- 可靠的安全系统：Linux 采取了许多安全技术措施，包括对读、写控制、带保护的子系统、审计跟踪、核心授权等，这为网络多用户环境中的用户提供了必要的安全保障
- 良好的可移植性：是指将操作系统从一个平台转移到另一个平台使它仍然能按其自身的方式运行的能力。Linux 是一种可移植的操作系统，能够在从微型计算机到大型计算机的任何环境中和任何平台上运行。诺基亚的 Maemo、谷歌的 Android 和 Chrome OS，以及英特尔的 Moblin

Linux 的主要组件



Kernel 系统启动时将内核装入内存管理系统各种资源

Shell 用户界面，提供用户与内核交互处理接口，是命令解释器，提供强大编程环境 Bash、Csh、Tcsh、Ksh、Zsh···

Utility 提供各种管理工具，应用程序

Linux 操作系统内核版本

版本号码命名约定：
主版本号.稳定（偶）|开发版本（奇）.发布号-patch 号
如：

- 2.5.19-6 为开发版本
- 2.6.20 为 稳定版本

当前最新稳定版本为 2009 年 12 月 14 日发布的 2.6.32.1
Linux-kernel 官方站点：http://www.kernel.org/

Unix/Linux 操作系统发行版本

一些主要发行版：

- Linux：Arch, CentOS, Debian, Fedora, Gentoo, Mandriva, Red Hat Enterprise Linux Server (RHEL), Slackware, SUSE Linux Enterprise Desktop (SLED), SUSE Linux Enterprise Server (SLES), OpenSuSE, Ubuntu, ...
- Unix：
  - 学院派 BSD：FreeBSD, OpenBSD, NetBSD, ...
  - 商业 Unix：IBM AIX, HP UX, Sun Solaris, OpenSolaris[1], Mac OS X[2], SGI IRIX, ...

Linux、BSD 发布版：http://distrowatch.com/

[1]Sun 公司按照 CDDL 授权开源

[2]以 FreeBSD 源代码和 Mach 微内核为基础

系统的运行

- 登录及退出系统
- 修改密码
- Linux 的命令结构

登录及退出系统

- 本地登录：文本或 X 界面
- 远程登录：ssh 或 telnet
  - ssh：
    - Linux 系统：$ ssh UserName@HostName[3]
    - Windows 系统：利用 putty 等支持 ssh 协议的客户端
  - telnet：
    $ telnet HostName
- 退出系统：exit 或 <ctrl−d>

[3]HostName 指的是要登录系统的域名或 IP 地址

创建或修改密码：passwd 与 yppasswd

- 一般系统：在系统提示符下输入 passwd
- 采用 NIS[4] 的集群系统等：需利用 yppasswd 修改密码，普通单机应该使用 passwd 修改

[4]网络信息服务，可用于在各节点系统中同步用户信息、系统配置等

命令的格式

命令一般具有如下格式，具体各个命令有所不同，需要查看命令的帮助（man command）

command options arguments

命令中 [] 之内的表示是可选选项，| 表示是或，几种中选择一种

$ 为普通用户的默认命令行提示符，超级用户 root 的为 #，例如：

$ mail −f newmail

一些常见错误：

正确	错误
- 与选项分离	
$ mail −f newmail	$ mail − f newmail
选项与参数的顺序	
$ mail −f newmail	$ mail newmail −f
多选项	
$ who −m −u	$ who −m−u
$ who −mu	$ who −m u

键入命令

以下如非特殊说明，将以 Bash(> 3.0 版本) 做为默认 Shell 说明，其余 Shell 也许略有不同

- 在 Shell 提示符下输入命令，然后按回车键
- Shell 区分大小写
- 如找不到输入的命令，会显示 Command not Found
- 如命令太长，要在此行行尾键入 \ 和按下回车键，在下一行的 > 后接着输入

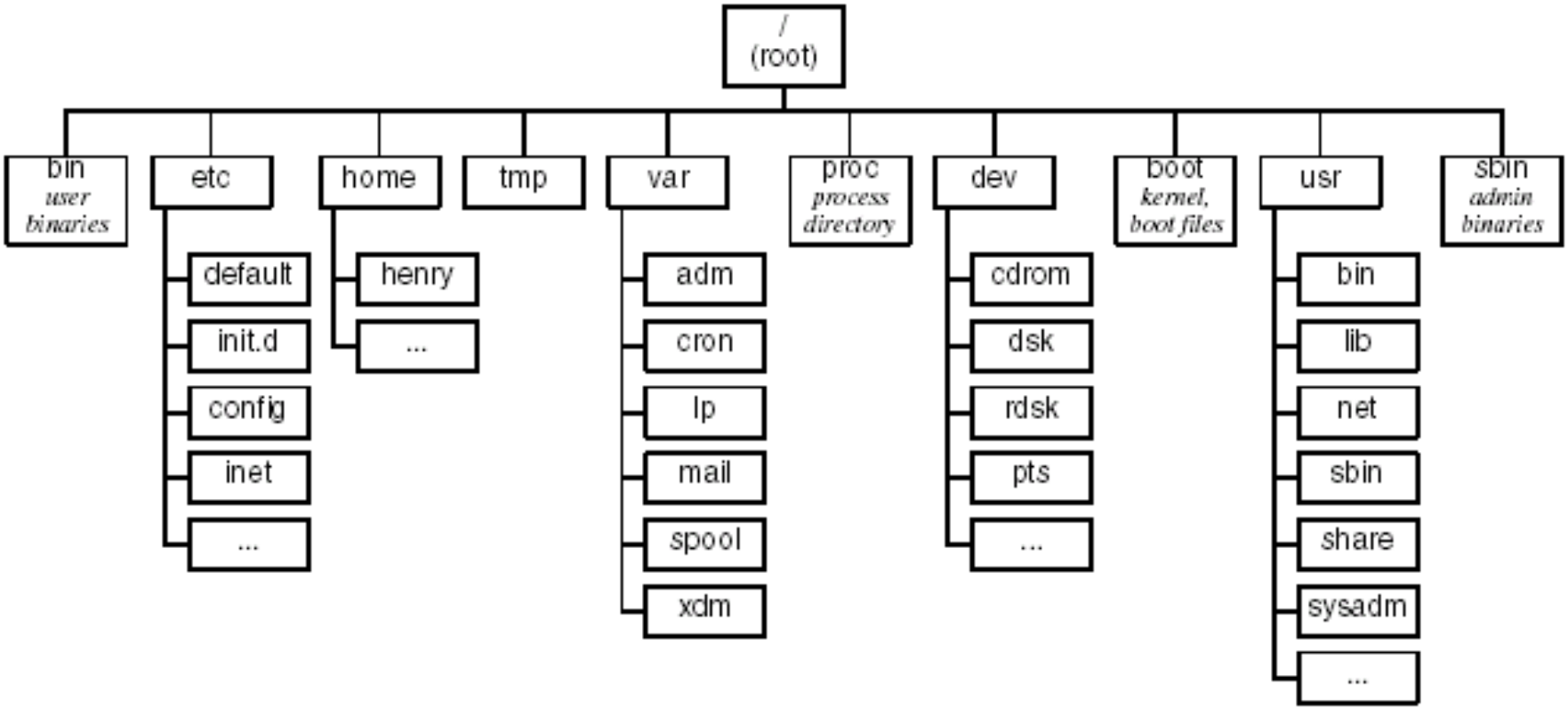
键盘的快捷方式

- <ctrl−c>：停止命令
- <ctrl−d>：结束传输或文件输入
- <ctrl−s>：临时停止输出
- <ctrl−q>：恢复输出
- <ctrl−u>：擦除光标以前的
- <ctrl−k>：擦除光标以后的
- <backspace>：纠正错误
- <ctrl−r>：在以前的命令中搜索

文件和目录

- 描述 Linux 文件系统的结构
- 描述不同的文件格式
- 文件的绝对路径和相对路径
- 创建、删除和列出路径
- 复制、显示、打印、移动、删除和连接文件

文件系统的层次结构


/
(root)
bin
user
binaries
etc
default
init.d
config
inet
...
home
henry
...
tmp
var
adm
cron
lp
mail
spool
xdm
proc
process
directory
dev
cdrom
dsk
rdsk
pts
...
boot
kernel,
boot files
usr
bin
lib
net
sbin
share
sysadm
...
sbin
admin
binaries

Linux 中的标准系统目录

目录	用途
/	Linux 系统根目录，包含所有目录
/bin	Binary 的缩写，存放用户的可执行程序，如 ls、cp，也包含其它的 Shell 如：bash 等
/boot	包含 vmlinuz、initrd.img 等启动文件，随便改动可能无法正常开机
/dev	接口设备文件目录，如你的硬盘：sda
/etc	passwd 等系统设置与管理的文件
/etc/x11	X Windows System 的设置目录
/home	一般用户的主目录
/lib	包含执行 /bin 和 /sbin 目录的二进制文件时所需的共享函数库 library
/lib32	在某些 64 位系统中包含 32 位的库文件
/lib64	在某些 32 位系统中包含 64 位的库文件
/mnt	各项装置的文件系统加载点，如：/mnt/cdrom 是光驱的加载点
/opt	提供空间，较大的且固定的应用程序存储文件之用
/proc	ps 命令查询的信息与这里的相同，都是系统内核与程序执行的信息
/root	管理员的主目录
/sbin	系统启动时所需的二进制程序
/tmp	Temporary，存放暂存盘的目录，此目录下的文件系统重启后将被自动清空
/usr	存放用户使用系统命令和应用程序等信息
/usr/bin	存放用户可执行程序，如 grep、mkdir 等
/usr/doc	存放各式程序文件的目录
/usr/include	保存提供 C 语言加载的 header 文件
/usr/include/X11	保存提供 X Windows 程序加载的 header 文件
/usr/info	GNU 程序文件目录
/usr/lib(/lib64)	函数库
/usr/lib(/lib64)/X11	保存提供 X Windows 程序的函数库
/usr/local	提供自行安装的应用程序位置
/usr/man	存放在线说明文件目录
/usr/sbin	存放经常使用的程序，如 nfsstat
/usr/src	保存系统的源码文件
/usr/X11R6/bin	存放 X Windows System 的执行程序
/var	Variable,具有变动性质的相关程序目录，如 log

文件系统的概念

- 文件系统：磁盘上有特定格式的一片区域
- 文件：存储数据的一个命名的对象
- 目录：包含许多文件项目的一类特殊文件
- 子目录：被包含在另一个目录中的目录
- 父目录：包含子目录的目录称为父目录
- 文件名：用来标识文件的字符串，保存在一个目录文件项中
- 路径名：由“/”字符结合在一起的一个或多个文件名的集合，它指定一个文件在文件系统中的位置

文件结构

无论文件是一个程序、文档、数据库、或是目录，操作系统都会赋予它下面的结构：

- 索引节点（Inode）：在文件系统结构中，包含有关相应文件信息（文件权限、文件所有者、文件大小等）的一个记录
- 数据：文件的实际内容

文件名称

- 包含大写字母、小写字母、数字、$、@、_、-
- 一般不包含以下字符：*、?、>、<、/、;、$、@、\、’、”
- 最好不以 + 或 - 开头
- 最长文件名 255 字符
- 可以有扩展名以便于识别和管理，用扩展名作为文件名的一部分，中间用 . 隔开
- 区分大小写

文件的主要类型

主要有以下文件类型，利用 ls –l 输出对应文件名的行第一个字符可判断

- 普通文件：
  - 文本文件：ASCII 码形式存储，以 - 开头，如：
    –rw–r––r–– 1 root root 39599 Mar 8 12:15 x.sh
  - 二进制文件：以二进制形式存储在计算机中，不可直接读，要通过相应的软件读取，以 - 开头，如：
    –rw–r––r–– 1 root root 69599 Mar 8 12:25 x
- 目录文件：以 d 开头，如：
  drwxr–xr–x 2 root root 4096 Aug 2 2006 bin
- 设备文件：
  - 块设备文件：以 b 开头，如：
    brw–rw–––– 1 root disk 3, 1 Jan 30 2003 /dev/hda1
  - 字符设备文件：以 c 开头，如：
    crw––––––– 1 root root 4, 1 Jul 31 13:49 /dev/tty1
- 链接文件：存放文件系统中通向文件的路径，以 l 开头，如：
  lrwxrwxrwx 1 root root 11 Sep 30 2008 c –> /mnt/c

利用 file 命令可查看文件类型

普通文件

普通文件也称常规文件，含各种长度的字符串，如：信件、报告和脚本

- 文本文件：由 ASCII 字符构成
- 数据文件：由来自应用程序的数字型和文本型数据构成，如：电子表格、数据库等
- 可执行的二进制程序：由机器命令和数据构成

目录文件

- 由成对的“Inode/文件名”构成的列表，利用目录文件可以构成文件系统的分层树形结构
- Inode 是检索 Inode 表的下标，Inode 存放所有文件的状态信息
- 文件名是给一个文件分配的文本形式的字符串，用来标识文件

路径的名称

主要分为两种类型：

- 绝对路径，以 / 或 ~ 开头，如 /home/hmli/linux
- 相对路径，不以 / 或 ~ 开头，如当前目录为 /home/hmli，那么 linux 既是相对路径

目录操作命令

- ls：显示目录中的内容
- pwd：显示当前和工作目录
- cd：改变用户工作目录
- mkdir：建立用户目录
- rmdir：删除目录

列出目录内容：ls

ls 命令列出一个子目录中的全部文件和目录名，一般格式为：

ls [OPTION]... [FILE]...

它有多个命令行参数，参数可组合使用，下面列出它最常用的几个[5]：

- -a：显示所有文件或目录，包括以“.”为名称开头字符的隐藏文件、现行目录“.”与上层目录“..”
- -l：使用详细格式列表。将权限标示、硬件接数目、拥有者与群组名称、文件或目录大小及更改时间一并列出
- -R：递归处理，将指定目录下的所有文件及子目录一并处理
- -t：按照时间排序
- -S：按照大小排序
- -r：逆向排序
- –color[=WHEN]：对不同类型的文件显示不同的颜色，WHEN 默认为 always，可为 never 和 auto

[5]其它命令类似

列出目录内容：ls 举例

- 使用长列表方式列出某个子目录中的全部文件：
  $ ls -la

```
total 16
drwxr-xr-x    4 root     root         4096 Jan  1 11:28 .
drwxr-x---   11 root     root         4096 Jan  1 11:27 ..
drwxr-xr-x    2 root     root         4096 Jan  1 11:27 dir1
drwxr-xr-x    2 root     root         4096 Jan  1 11:28 dir2
```

- 列出子目录中以字母 v 打头的全部非隐藏文件：
  $ ls /boot/v*

```
-rw-r--r-- 1 root root 2078016 Nov 17 09:28 vmlinuz-2.6.31-hmli
-rw-r--r-- 1 root root 2138528 Dec  8 11:11 vmlinuz-2.6.32-hmli
```

显示当前工作目录：pwd

pwd 没有参数，唯一的作用就是显示当前工作目录的绝对路径的名称

```
$ pwd
```

```
/home/hmli
```

环境变量 PWD 存储当前目录名，OLDPWD 存储执行 cd 命令切换到当前目录之前的目录名，利用 echo 可以查看，如：

```
$ echo $PWD
$ echo $OLDPWD
```

改变用户工作目录：cd

cd 命令可以让用户在不同的目录间切换，一般格式为 cd [−L|−P] [dir]，几个常用方式：

- cd dirname：进入名字为 dirname 的目录
- cd ~username：进入用户 username 的主目录
- cd ~/dirname：进入当前用户的主目录下的 dirname 子目录
- cd −：返回进入这个目录之前所在的目录
- cd ..：进入上一级目录
- cd ../..：进入上两级目录
- cd：不跟任何参数时返回用户根目录

建立用户目录：mkdir

mkdir 可建立目录，一般格式为：
mkdir [OPTION]... DIRECTORY...
主要选项为：

- -p：若所要建立目录的上层目录目前尚未建立，则会一并建立上层目录，如 $ mkdir −p 1/2
- -m：建立目录时，同时设置目录的权限。权限的设置法与 chmod 命令相同，如 $ mkdir −m 700 3

删除目录：rmdir

rmdir 命令可删除空目录，若所给予的目录非空目录，则出错

- rmdir dirname：删除 dirname 目录
- -p：删除指定目录之后，若该目录的上层目录已变成空目录，则将其一并删除，如 $ rmdir −p 1/2
- rmdir 更常用的替代命令为后面所说的 rm 命令

文件操作命令

这里不仅仅指的是普通文件，也包括目录等文件

- cp：复制文件或目录
- mv：移动文件和文件换名
- rm：删除文件或目录
- ln：在文件间建立连接
- find：查找特定的文件
- locate：查找特定的文件
- which：查看命令的路径
- touch：改变文件的时间参数

复制文件或目录：cp

cp 命令用来复制文件，在缺省的情况下，工作时不做任何显示，只有在出现错误情况的时候才显示状态信息，一般格式为：

cp [OPTION]... SOURCE DEST

主要选项：

- -a：复制时尽可能保持文件的结构和属性
- -f：不提示直接覆盖存在的目标文件
- -i：无论是否覆盖现存文件都作提示，建议打开此选项，避免误操作覆盖掉源文件
- -p：保持原始文件的所有者、组、许可和时间表属性
- -r：递归地复制目录

移动/重命名文件：mv

mv 命令用来把文件从一个位置移动到另外一个位置，一般格式为：
mv [OPTION]... SOURCE DEST
主要选项：

- -i：无论是否覆盖现存文件都作提示，建议打开此选项，避免误操作覆盖掉源文件
- -f：不提示直接覆盖存在的目标文件

删除文件或目录：rm

rm 命令用来删除文件[6]，一般格式为：
rm [OPTION]... FILE... 主要选项：

- -f：不提示直接删除指定的目标文件
- -i：指定交互模式，在执行删除前提示确认
- -r：删除文件列表中的目录

删除文件名以 - 开头文件，比如 '-foo' 可以使用以下方法之一：

- rm −− −foo
- rm ./−foo

注：在使用 cp、mv 等许多命令时遇到此类文件名可类似处理

[6]删除了就没有了，基本不存在恢复的可能，切记

在文件间建立连接：ln

ln 命令用来建立硬连接和符号连接

- 硬连接是一个文件的额外的名字，相当于一个同步更新的副本，删除源文件，硬连接的内容还存在
- 符号连接相当于快捷方式，当源文件被删除后，符号连接仍然存在，但链接的内容已经不存在

一般格式为：

ln [options] source [dest]

主要选项：

- -d：用于建立目录的硬连接
- -f：覆盖已存在的目的文件
- -i：提示是否删除已存在的目的文件。
- -s：建立符号连接以替代硬连接

查找文件：find

find 命令可以根据各种检索条件查找文件，一般格式为：

find [−H] [−L] [−P] [−D debugopts] [−Olevel] [path...] [expression]

- path…：准备寻找文件所在的路径以及它的子路径，也可以是多个路径
- expression：包含要搜索文件的条件，可以包含文件名、拥有者、修改时间、权限等

find 主要选项

- -atime n：至少 n*24 小时内没有访问过的文件
- -ctime n：至少 n*24 小时内没有修改过的文件
- -amin n：n 分钟之前访问过的文件
- -cmin n：n 分钟之前修改过的文件
- -empty：文件为空
- -name name：指定要寻找的文件或目录的名称，区分大小写
- -iname name：指定要寻找的文件或目录的名称，不区分大小写
- -type x：以文件的类型作为寻找的条件，若 x 为：
  - d：表示寻找目录
  - f：表示寻找普通文件
  - c：表示寻找字符特殊设备
  - b：表示寻找特殊块设备
  - p：表示寻找命名管道
  - l：表示寻找符号连接
  - s：表示寻找套接字

find 举例

- 查找 /home 子目录中至少 7 天没有被访问过的文件：
  $ find /home −atime +7 −print
- 找出 /usr/src 子目录中名字为 core 的文件并删除它们：
  $ find /usr/src −name core −exec rm {} \;
- 找出 /home 中以 .jpg 结尾并且长度超过 100KB 的文件：
  $ find /home −name ”∗.jpg” −size 100k
- 找出当前目录下权限不是 755 的目录并将其权限设置为 755：
  $ find . −type d ! −perm 755 −exec chmod 755 {} \;

查找文件所在目录：locate

从系统保存文件位置信息的数据库中搜索文件：locate
如查看名字中含有 libmkl_intel 的库文件：$ locate libmkl_intel

```
warning: locate: warning: database /var/lib/slocate/s
locate.db' is more than 8 days old
/opt/intel/Compiler/11.0/083/mkl/lib/32/libmkl_intel.a
/opt/intel/Compiler/11.0/083/mkl/lib/32/libmkl_intel.so
/opt/intel/Compiler/11.1/059/mkl/lib/32/libmkl_intel.a
/opt/intel/Compiler/11.1/059/mkl/lib/32/libmkl_intel.so
```

注意：如果系统的数据库不够新，那么查找的结果将未必正确，比如上面提示数据库的信息至少已经 8 天没更新，root 可以运行 updatedb 更新数据库，普通用户无权更新

显示命令的路径：which

which 查看命令的所在路径，如查看使用的是哪个 mpif90：

$ which mpif90

```
/opt/mpich2-1.2/bin/mpif90
```

which 实际上是在环境变量 PATH 中指定的路径中显示第一个搜索到的命令的路径，利用 echo $PATH 可以看到以 : 分隔的命令路径，如：

```
/usr/local/bin:/usr/bin:/bin:/opt/mpich2-1.2/bin:/opt/mpich2/bin
```

改变文件的时间参数 touch

touch 改变文件访问和修改时间或用指定时间建立新文件，一般格式为：

touch [OPTION]... FILE...

主要选项：

- 不带参数：将文件的时间改为当前时间
- -a：只更改访问时间
- -c：若目标文件不存在，不建立空的目标文件
- -t [[CC]YY]MMDDhhmm[.ss]：以指定时间修改文件的时间，如：
  $ touch −t 01201500 file1

显示文件内容的命令

- cat：显示和合并文件
- paste：横向合并文件，将多个文件对应行合并
- more：分屏显示文件
- less：分屏显示文件
- head：显示文件的前几行
- tail：显示文件的最后几行

显示和合并文件：cat

cat 可以结合多个文件，并将它们的内容输出到标准输出设备或者定向到某个文件，一般格式为：

cat [OPTION]... [FILE]...

主要选项：

- -b：列出文件内容时，在所有非空白列之开头标上编号，从 1 开始累加
- -E：在每一行的最后标上“$”符号
- -n：列出文件内容时，在每一列之开头标上编号，从 1 开始累加

cat 举例

- 让 cat 命令从标准输入设备（如键盘）读取数据，转而输出至标准输出设备（如显示器）：
  $ cat 不加任何参数之后回车，键入文字，按下回车键，系统将回应一模一样的文字
- 利用特殊字符 > 将名称为 file1 与 file2 的文件合并成一个文件 file3：
  $ cat file1 file2 > file3
  - 若文件 file3 已经存在，则其内容会被覆盖过去
  - 欲避免上述状况发生，可用 >> 代替 >，新的内容就会附加在原有内容之后，而不会覆盖它

横向合并文件：paste

paste 横向合并文件，将多个文件对应行合并，一般格式为：
paste [OPTION]... [FILE]...
一般选项：

- -d, –delimiters=LIST：用 LIST 代替 TABs
- -s, –serial：串行而非并行合并文件

例如：

file1 的内容如下：

```
1
2
3
```

file2 的内容如下：

```
a
b
c
```

执行 paste file1 file2 将显示：

```
1	a
2	b
3	c
```

分屏显示文件：more

more 可将文件内容显示于屏幕上，每次只显示一页。可以往下翻页，但无法回退翻页，一般格式为：

more [−dlfpcsu] [−num] [+/pattern] [+linenum] [file ...]

主要选项：

- +/<字符串>：在文件中查找选项中指定的字符串，然后显示字符串所在该页的内容
- +<行数>：从指定的行数开始显示
- -n：每次只显示 n 行
- -c：不滚屏，在显示下一屏之前先清屏

主要操作：

- 空格：翻页
- /pattern：查找匹配 pattern 的字符
- v：调用编辑器进行编辑
- <ctrl-L>；刷新屏幕
- q：退出

more 举例

- 在文件 file1 中查找“123”字符串，然后从该页开始显示内容：
  $ more +/123 file1
- 显示文件 file1 的内容，每 10 行显示一次，且在显示之前先清屏：
  $ more −c −10 file1

分屏显示文件：less

less 类似 more，也可以用来浏览超过一页的文件。less 命令除了可按空格键向下显示文件外，还可利用上下键来卷动文件。当要结束浏览时，只要下按 q 键即可。显示中按 v，也可调用编辑器直接进行编辑。建议一般使用功能更强大方便的 less，man less 仔细看看用法以及功能

显示文件的前几行：head

head 在显示指定文件的开头若干行，默认值是 10 行，一般格式为：
head [OPTION]... [FILE]...

- -c N：显示前 N 个字节
- -n N：显示前 N 行
- -N：显示前 N 行，如：
  $ head −20 file

显示文件的最后几行：tail

tail 在显示指定文件的末尾若干行，默认值是 10 行，一般格式为：

tail [OPTION]... [FILE]...

主要选项：

- -c N：显示后 N 个字节
- -n N：显示后 N 行
- -N：显示后 N 行
- -f：连续监测需要显示文件的最后，如果文件更新，将自动显示出更新，常用于监测输出日志，比如：

  $ tail −f myjob.log

比较文件内容命令：comm 与 diff

- comm：比较两个已排过序的文件
- diff：比较文件的差异

显示两个文件共同部分：comm

comm 用来对两个已排过序的文件进行逐行比较显示其共同部分，一般格式为：

comm [OPTION]... FILE1 FILE2

OPTION 可以为：

- -1：不显示只在第一个文件里出现的行
- -2：不显示只在第二个文件里出现的行
- -3：不显示在第一、二个文件里都出现的行

comm 举例

file1 的内容如下：

```
main (
{
   printf("Hello!\n");
}
```

file2 的内容如下：

```
main (
{
printf("Good!\n");
}
```

用 comm 命令对这两个文件进行比较只显示它们共有的行：

comm −12 file1 file2

```
        main (
        {
         }
```

显示多个文件不同部分：diff

diff 可以比较多个文本文件，并显示它们的不同，一般格式为：
diff [OPTION]... FILES
主要选项：

- -c：输出格式是带上下文的三行格式
- -C n：输出格式是带上下文的 n 行格式
- -r：两个文件都是目录时，递归比较找到的各子目录
- 输出的一般形式如下：
  - n1 a n3,n4
  - n1,n2 d n3
  - n1,n2 c n3,n4
- a-附加，d-删除，c-修改
- a、d、c 前面的表示第一个文件的行号，后面的表示第二个文件的行号

上面的 n1 a n3,n4 表示两个文件若要变成相同，那么第一个文件的 n1 行将附加第二个文件的行号为 n3 到 n4 行的内容

diff 举例

file1 的内容如下：

```
1main ()
2{
3 printf("Hello!\n");
4}
5
```

file2 的内容如下：

```
1 main ()
2 {
3  int n,m;
4   n=10;
5 printf("%d\n",m=n*10);
6}
```

输入命令：

$ diff file1 file2

将显示：

```
 3,5c 3,6
<3 printf("Hello!\n") ;
<4 }
<5
---
>3  int n,m;
>4  n=10;
>5 printf("%d\n",m=n*10);
>6}
```

文件权限操作

- chmod：改变文件或目录的许可权限
- chown：改变文件的所有权
- chgrp：改变用户分组

文件的保护和权限

文件权限：

rwx	rwx	rwx
user	group	others

- 普通文件：
  - r：可以查看文件内容
  - w：可以修改文件内容
  - x：可以执行文件
- 目录：
  - r：可以查看目录下的文件
  - w：可以在目录下创建和删除文件
  - x：可以进入目录或访问目录下的文件

权限

权限格式

	user	group	others
符号	rwx	rw-	r–
二进制	111	110	100
八进制	7(4+2+1)	6(4+2+0)	4(4+0+0)

缺省的文件权限：

- 文件：–rw–r––r–– 644
- 目录：drwxr–xr–x 755

改变文件权限：chmod

chmod 用于改变文件或目录的权限，一般格式为：
chmod [OPTION]... MODE[,MODE]... FILE...
MODE 模式：

- u：文件所属帐户
- g：文件所属组
- o：其余帐户
- a：所有帐户
- +：添加权限
- -：去除权限
- =：使得指定文件只具有这些权限

主要选项：

- -v：详细显示权限改变的信息
- -c：类似 -v，仅权限改变时显示
- -R：对所有某目录下的文件进行递归处理

chmod 举例

- 使文件 file 在各个级别拥有所有权限：
  $ chmod 777 file
- 允许所有人读 file，但只有拥有者能改变它：
  $ chmod 644 file
- 给所有人增加写权：
  $ chmod a+w file
- 对组级和其他用户除去写权和读权：
  $ chmod o−wr,g−wr file
- 建立其他用户的只读权：
  $ chmod o=r file

设置缺省权限：umask

利用 umask 可以设置缺省的文件权限，一般格式为：

umask [−p] [−S] [mode]

mode 与 chmod 的类似，应该设其 mode 为 ls 看到的权限的异或，简单说就是 777-ls 看到的

如避免其他用户查看自己文件，可用 umask 077，将设置缺省权限为：

- 文件：–rw———— 600
- 目录：drwx———— 700

如果想要每次登录都起作用，需要加入 Shell 的启动初始文件，比如 bash 的 ~/.bashrc

改变文件的所有权：chown

chown 可把一个文件的所有权修改为其他用户的[7]，一般格式为：
chown [OPTION]... [OWNER][:[GROUP]] FILE...
主要选项：

- -v：详细显示所有权的变化
- -c：类似 -v，仅所有权改变时显示
- -R：递归改变目录及其内容的所有权

比如，将所有文件的所有者修改 hmli（hmli 所在的组为 nic）：
$ chown −R hmli:nic dirname

[7]只有 root 用户能够进行这样的操作

改变用户分组：chgrp

chgrp 命令可以改变一个文件的用户分组设置情况[8]，一般格式为：
chgrp [OPTION]... GROUP FILE...
主要选项：

- -v：详细说明文件所属的用户组的变化
- -c：类似 -v，仅所有权改变时显示
- -r：改变本目录及其所有子目录中的文件所属的用户组

[8] 只有 root 才可以执行

进程

- 定义进程
- 进程监视
- 调用后台进程
- 中断进程
- nohup
- 控制进程
- 定义系统进程

进程概念

- 一个进程就是一个运行的程序，是动态的
- Linux 为每一个进程分配一个进程标识号（PID）指定和跟踪进程
- 每一个进程有一个父进程（PPID）和多个子进程

进程和程序的关系

- 进程是程序的执行过程
- 程序是一个静态的命令集，进程是动态的
- 进程之间是并发执行的，而程序本身没有并发行
- 进程是分配资源的单位，在运行过程中使用系统资源

父进程和子进程

进程（PID）	父进程（PPID）
201	1
206	201
207	206

查看进程：ps

ps 查看当前系统中运行的进程的信息，一般格式为：ps [options]
主要选项：

- -a：显示系统中与 tty 相关的所有进程的信息
- -f：显示所有进程的信息
- -r：只显示正在运行的进程
- -o format：以指定格式显示
- -u：显示面向用户的格式
- -x：显示所有终端上的进程信息

ps 举例

- 以标准方式显示所有进程：
  - $ ps −e
  - $ ps −ef
  - $ ps −eF
  - $ ps −ely
- 以 BSD 方式显示所有进程
  - $ ps ax
  - $ ps axu
- 打印进程树：
  - $ ps −ejH
  - $ ps axjf
- 显示进程信息：
  - $ ps −eLf
  - $ ps axms

ps 举例

- 显示进程安全信息：
  - $ ps -eo euser,ruser,suser,fuser,f,comm,label
  - $ ps axZ
  - $ ps -eM
- 显示所有 root 用户运行的进程：
  - $ ps -U root -u root u
- 按照用户设定格式显示进程：
  - $ ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
  - $ ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
  - $ ps -eopid,tt,user,fname,tmout,f,wchan
- 显示 syslogd 进程的信息：
  - $ ps -C syslogd -o pid=
- 显示进程号（PID）为 42 的进程信息：
  - $ ps -p 42 -o comm=

查看进程树：pstree

pstree 可以显示进程树，一般格式为：pstree [OPTIONS]

主要选项：

- -a：显示命令执行时的参数
- -h：高亮显示当前进程与其父进程
- -n：按照进程号进行排序
- -p：显示进程号进行排序

输出实例：

```
init-+-acpi_fakekeyd
     |-atd
     |-avahi-daemon---avahi-daemon
     |-console-kit-dae---63*[{console-kit-dae}]
     |-cron
     |-fail2ban-server---4*[{fail2ban-server}]
     |-gdm---gdm-+-Xorg
     |           '-x-window-manage-+-2*[FvwmAuto]
     |                             |-FvwmIconMan
     |                             |-FvwmPager
     |                             |-FvwmTaskBar
     |                             |-firefox-bin-+-mplayer---mplayer
     |                             |             '-9*[{firefox-bin}]
     |                             '-ssh-agent
```

终止进程：kill、killall、pkill 等命令

kill 、 killall 、 pkill 等可以用于终止进程

- 前台进程：
  - ctrl－c
  - kill
  - killall
  - pkill
- 后台进程：
  - kill
  - killall
  - pkill

终止进程

利用下述命令显示出当前进程：

```
$ ps -f
```

```
UID        PID        PPID       ···TTY  ···COMMAND
hmli       206         201       ···1    ···bash
hmli       209         206       ···1    ···vim x.txt
```

利用下述命令之一可以终止进程：

- 利用进程号杀掉进程： `kill 206`
- 利用发送信号 9 和进程号杀掉进程： `kill -9 206`
- 杀掉所有 vim： `killall vim`
- 杀掉所有 vim： `pkill vim`

后台运行，运行耗时长进程：nohup 与 &

默认情况下退出终端，当前运行的进程将自动终止，可以利用 nohup 保持程序的运行，一般格式为：nohup COMMAND [ARG]...

- $ nohup ls −R l > out &
- $ nohup ls −R l &

一般在运行命令最后添加 & 将程序放到后台运行，比如：

$ ./myprog >myprog.log &

暂停、前台与后台运行：<ctrl−z>、fg 与 bg

如果程序没有用 & 指定后台运行，可以在启动此进程的 Shell 利用 <ctrl−z> 暂停运行，暂停之后运行 bg 将其变为后台运行，如果再想变成前台运行，可以执行 fg。

查看进程运行情况：top

top 可监视系统当前运行的各进程 CPU 内存等利用情况等，格式为：

top −hv | −bcHisS −d delay −n iterations −p pid [, pid ...]

主要选项：

- -b：以批处理模式运行，可用于连续监测并将输出存储到文件中
- -d delay：两此连续刷新之间的间隔
- -n：最大刷新次数
- -u|U username：仅查看 username 用户的进程
- -p PID,PID：仅查看特定进程号的进程

非批处理模式进入 top 后的主要操作：

- 1：数字 1 显示隐藏多 CPU 利用情况
- h：显示帮助信息，查看可执行的操作
- c：显示进程路径参数等
- d|s：设置刷新间隔时间
- F|O：设置排序方式
- n|#：设置最大显示进程数
- M：按照内存/SWAP 使用排序显示
- q：退出
- z：开启关闭颜色显示

注意：IBM AIX 系统为 topas，HP UX 有 top 与 glance

时间和日历：date 和 cal

- 在屏幕上显示或设置系统的时间 date：
  - 显示当前时间：$ date
  - 按照某种格式显示：$ date +%Y−%m−%d\ %H:%M:%S
  - 一个计算俺儿子年龄的脚本：

```
#!/bin/bash
#Program: son−age
# Author: HM Li <hmli@ustc.edu.cn>
#   Date: 09−10−05
#Comment: 计算儿子年龄，单位天
BIRTH=$(date −d "2009−08−25 13:56:00" +%s)
NOW=$(date −d "$(date +%Y−%m−%d %H:%M:%S)" +%s)
AGE=$(($(($NOW − $BIRTH))/3600/24))
echo −e 儿子满"\e[35;1m$AGE\e[0m"天了
```

- 显示日历 cal：
  - 显示当前月份日历：$ cal
  - 显示 2012 年全年月份日历：$ cal 2012

计算程序运行时间：time

使用 time 命令可以了解单个程序和它的同步子程序的性能特征。它报告实际时间，也就是从程序开始到结束所使用的时间。它也报告由该程序使用的 CPU 时间量。CPU 时间分为 user 和 sys。user 值是由程序自身和它所调用的任何库子例程所使用的时间。sys 值是由程序调用（直接或间接）的系统调用所使用的时间。

$user + sys$ 的和是执行程序的总的直接 CPU 花费。这不包括内核部件的 CPU 花费，这些部件可以说是代表程序而运行的，但实际并不在它的线程上运行。例如，当程序启动时窃取页面帧来代替从自由列表中取得的页面帧的花费并不作为该程序 CPU 耗费的一部分来报告。

- 在一个单处理器上，实际时间和总 CPU 时间之间的差值，即：
  $real - (user + sys)$
- 在一个多核系统中，可以有如下近似：
  $real * number\_of\_processors - (user + sys)$

比如：

```
$ time ./myprog
```

清屏：clear

clear 清除屏幕上的信息。清除后，提示符移到屏幕的左上角

重新初始化终端：reset

如果发现终端出现稀奇古怪的字符等，可以利用 reset 重新初始化终端

显示变量内容：echo

利用 echo 可以显示变量内容或字符串到标准输出（即屏幕）上，一般格式为：echo [SHORT−OPTION]... [STRING]...

- -n：表示输出字符串后，光标不换行
- -e：对 \ 进行转义，比如 \ 表示颜色等
- -E：取消对 \ 进行转义，默认选项

几个例子：

- $ echo $LD_LIBRARY_PATH
- $ echo ”This is a command.”
- $ echo This is a command.
- $ echo −n ”Enter data−>”
- $ echo −e ”\e[35;1mPATH\e[0m”=$PATH

查找文件内容：grep

grep 可用于查找文件中的内容，一般格式为：

grep [OPTIONS] [−e PATTERN | −f FILE] [FILE...]

- 如果在搜索模式中包含空格，应用单引号把模式字符串括起来
- 在文件列表中可以使用通配符

主要选项：

- –color[=WHEN]：WHEN 可为 never、always 或 auto，颜色可利用环境变量 GREP_COLOR 设置，默认为红色
- -c：不显示匹配行，只显示匹配总数
- -e PATTERN：按正则表达式方式匹配
- -i：忽略大小写查找
- -n：同时显示匹配行号
- -v：显示不匹配的行
- -l：只显示有匹配的内容的文件名

几个实例：

- 查找所有进程中含有 vim 的进程：$ ps axu | grep vim
- 不区分大小写查包含 subroutine 的 .f90 文件：
  $ grep −i subroutine ∗.f90

Linux 与 Windows 文本格式转化：dos2unix 与 u

对于文本文件来说，Windows(DOS) 利用回车换行（\r\n）作为和新行的开始，Unix/Linux 用换行（\n）作为新行的开始，因此当文本文件在这两个系统使用时也许会因此出现问题，可以利用 dos2unix|fromdos 和 unix2dos|todos（实际这四个命令是 tofrodos 包的一部分）来分别进行转化：

```
$ file filename
filename: ISO-8859 text
```

- 转化成 Windows 格式：

```
$ unix2dos filename
$ file filename
filename: ISO-8859 text, with CRLF line terminators
```

- 转化成 Linux 格式：

```
$ dos2unix filename
$ file filename
filename: ISO-8859 text
```

打印文件字数和行数：wc

wc 可统计文件字数行数等，一般格式为：wc [OPTION] filename
主要选项：

- -c：只显示字节数
- -l：只显示行数
- -w：只显示字数

查看 /etc/passwd 的行数：

```
$ wc –l /etc/passwd
```

生成序列：seq

seq 可以生成数字序列，并且支持浮点数，可用于循环处理某些操作，一般格式为：

- seq [OPTION]... LAST
- seq [OPTION]... FIRST LAST
- seq [OPTION]... FIRST INCREMENT LAST

主要选项：

- -f, –format=FORMAT：指定格式
- -s, –separator=STRING：以 STRING 分割，默认为回车 \\n
- -w：等宽显示

对比以下命令：

- $ seq 6 12
- $ seq −w 6 12
- $ seq 6 0.5 12
- $ seq −s , 6 12

生成文件名 test01.dat-test10.dat 的十个文件：

```
$ for i in $(seq −w 1 10);do touch test−$i.dat;done
```

测试网络通断：ping

ping 用来检测一个系统是否已连接上并在运行，如：
$ ping 202.38.64.1

```
PING 202.38.64.1 (202.38.64.1) from 202.38.64.1 : 56(84) bytes of data.
64 bytes from 202.38.64.1: icmp_seq=1 ttl=255 time=0.092 ms
64 bytes from 202.38.64.1: icmp_seq=2 ttl=255 time=0.022 ms
64 bytes from 202.38.64.1: icmp_seq=3 ttl=255 time=0.020 ms
64 bytes from 202.38.64.1: icmp_seq=4 ttl=255 time=0.019 ms

--- 202.38.64.1 ping statistics ---
4 packets transmitted, 4 received, 0% loss, time 2997ms
rtt min/avg/max/mdev = 0.019/0.038/0.092/0.031 ms
```

上面显示网络是通的，如果一直没反应，那么说明网络有可能无法到达对方（也许对方禁止 ping）

追踪网络路由：traceroute

traceroute 可追踪网络路由，并可判断网络在哪里出了问题等，如：
$ traceroute www.g.cn

```
traceroute: Warning: www.g.cn has multiple addresses; using 203.208.37.104
traceroute to www.g.cn (203.208.37.104), 30 hops max, 46 byte packets
 1  local-gw (202.38.64.126)  0.241 ms  0.180 ms  0.188 ms
 2  wlt (202.38.64.59)  0.165 ms  0.145 ms  0.135 ms
 3  210.45.224.251 (210.45.224.251)  1.644 ms  0.768 ms  0.757 ms
 4  202.112.53.225 (202.112.53.225)  0.488 ms  0.697 ms  0.644 ms
 5  * * *
```

从上面可以看出网络通路，并得出到 202.112.53.225 是通的，之后也许断掉了

获取用户信息：finger

finger 获得网络中其他用户的信息。可以查看一个用户最后登陆的时间、他所使用的 Shell 类型、主目录的路径等，一般格式为：

finger [−lmsp] [user ...] [user@host ...]

例如：

$ finger hmli

```
Login: hmli                             Name: Li HuiMin
Directory: /home/nic/hmli               Shell: /bin/bash
Office: hmli@ustc.edu.cn
On since Wed Dec 16 19:38 (CST) on pts/10 from 202.38.64.91
On since Wed Dec 16 19:39 (CST) on pts/11 from 124.16.151.106
   3 seconds idle
No mail.
No Plan.
```

查看命令手册：man

man 可以格式化并显示某命令的联机帮助手册，输出的手册页主要包括以下几个部分：

- NAME：命令的名称和用法
- SYNOPSIS：显示命令的语法格式，列出其所有可用的选用的选项及参数。一般来说这里 [ ] 内的表示可选，不在 [ ] 内的为必选，| 表示或，如：who [OPTION] $\cdots$ [FILE — ARG1 ARG2]
- DESCRIPTION：描述命令的详细用法及每个选项的功能
- OPTION：对命令的每一个选项进行详细的说明

主要选项：

- [section]：显示某 [section] 内的该命令帮助，[section] 是 man 的分类，一般为：1、命令；1P、Posix 命令；2、系统调用；3、库函数；3P、Posix 程序；4、特殊文件；5、文件格式与约定；6、游戏；7、约定与杂项；8、系统管理命令；9、内核进程
- -a：显示所有 [section] 内的帮助
- -L：以某种语言显示帮助，比如 $ man −L zh_CN.GBK man

环境变量 MANPATH 设置 man 的搜索路径

阅读 info 帮助：info/pinfo

info 和 pinfo 用于阅读 info 文档，此种文档相比 man 文档更友好，pinfo 比 info 操作更方便，支持颜色加亮等，一般格式为：

- info [OPTION]... [MENU-ITEM...]
- pinfo [options] [infopage]

将光标移到链接处（pinfo 支持彩色链接，info 看不出来），按回车进入子章节，按小键盘的左右箭头可以来回跳转

显示命令简要说明：–help|-h 等

许多命令支持运行时添加如下选项查看简要帮助：

- -h
- -help
- –help
- -H
- -?

查看目录大小命令：du

du 总结每个文件的磁盘用量，目录则取总用量，一般格式为：
du [OPTION]... [FILE]..
主要选项：

- -h：以人性化方式显示大小
- -s：仅显示总大小

du −h /home/hmli/data/

```
590M    /home/hmli/data/surface/old
926M    /home/hmli/data/surface
34M     /home/hmli/data/asem
2.8M    /home/hmli/data/line/old
105M    /home/hmli/data/line
1.1G    /home/hmli/data/
```

查看分区大小：df

df 显示系统某个文件所在的分区系统的信息，默认是显示所有文件系统，一般格式为：

df [OPTION]... [FILE]...

主要选项：

- -B Block-Size：以指定块大小显示
- -h：人性化显示，比如直接显示 1K、234M、2G 等
- -H：用 1000 而不是 1024 进制计算
- -T：显示文件系统类型
- -l：仅显示本地文件系统，不显示 NFS 等网络文件系统

例如：

$ df -h

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1              19G   17G  790M  96% /
tmpfs                 442M     0  442M   0% /lib/init/rw
udev                   10M  284K  9.8M   3% /dev
tmpfs                 442M     0  442M   0% /dev/shm
/dev/sda2              92M   31M   57M  35% /boot
/dev/sda4              73G   60G  8.8G  88% /home
```

远程会话工具：Screen

- 你是不是经常需要 SSH 或 telent 远程登录到 Linux 服务器
- 你是不是经常为一些长时间运行的任务而头疼，比如系统备份、ftp 传输等等
- 通常情况下我们都是为每一个这样的任务开一个远程终端窗口，因为它们执行的时间太长了。必须等待执行完毕，在此期间可不能关掉窗口或者断开连接，否则这个任务就会被杀掉，一切半途而废了

不想有以上烦恼，那就使用 GNU Screen 吧，Screen 可以看作是窗口管理器的命令行界面版本，它提供了统一的管理多个会话的界面和相应的功能

screen 功能

- 会话恢复
  只要 Screen 本身没有终止，在其内部运行的会话都可以恢复。这一点对于远程登录的用户特别有用——即使网络连接中断，用户也不会失去对已经打开的命令行会话的控制。只要再次登录到主机上执行 screen −r 就可以恢复会话的运行。同样在暂时离开的时候，也可以执行分离命令 detach，在保证里面的程序正常运行的情况下让 Screen 挂起（切换到后台）。这一点和图形界面下的 VNC 很相似。
- 多窗口
  在 Screen 环境下，所有的会话都独立的运行，并拥有各自的编号、输入、输出和窗口缓存。用户可以通过快捷键在不同的窗口下切换，并可以自由的重定向各个窗口的输入和输出。Screen 实现了基本的文本操作，如复制粘贴等；还提供了类似滚动条的功能，可以查看窗口状况的历史记录。窗口还可以被分割和命名，还可以监视后台窗口的活动。
- 会话共享
  Screen 可以让一个或多个用户从不同终端多次登录一个会话，并共享会话的所有特性（比如可以看到完全相同的输出）。它同时提供了窗口访问权限的机制，可以对窗口进行密码保护。

Screen 启动方式

- 不加任何参数的 screen：启动新的 Screen
- screen −r：登录到上次退出的 Screen 会话，如有多个会有提示登录到哪个会话，如下：

```
There are screens on:
here are several suitable screens on:
    10219.pts-7.HM_Li   (12/15/09 14:34:14) (Detach
    10181.pts-7.HM_Li   (12/15/09 14:34:10) (Detach
There is no screen to be resumed.
```

之后可类似用 $ screen −r 10181.pts−2.HM_Li 登录某个会话

screen 进入后的操作

- <ctrl-a d>：退出 Screen，Screen 内的操作等继续保留，下次可以用 screen −r 重新进入此会话继续查看和运行
- <ctrl-A>：命名会话
- <ctrl-w>：显示会话名称
- <ctrl-a c>：创建一个新的 Shell
- <ctrl-a ctrl-a>：在 Shell 间切换
- <ctrl-a n>：切换到下一个 Shell
- <ctrl-a p>：切换到上一个 Shell
- <ctrl-a 0···9>：切换各个 Shell
- <ctrl-a d>：退出 Screen 会话

如果当前 Screen 是运行 Shell，那么直接 <ctrl-d> 或 exit，将退出 Shell 并退出此 screen 会话，下次无法恢复此会话中的运行

网络上传下载：ftp

ftp 可用于客户端和服务器之间上传下载数据，一般用法为：

ftp [−pinegvd] [host [port]]

- 连接服务器：
  - ftp Host
  - ftp User@Host
  - ftp User:Passwd@Host
- 进入后基本命令：
  - 查看文件列表：ls
  - 单个/多个文件下载：get/mput
  - 单个/多个文件上载：put/mget
  - 建立删除文件夹：mkdir/rmdir
  - 删除文件：delete
  - 二进制传输：bin
  - ASCII 传输：ascii
  - 连接新服务器：open
  - 退出：quit 或 bye
  - 打开关闭交互模式：prompt，mput 和 mget 命令事情前应关闭交互模式，否则每次都询问是否上载或下载
  - 查看帮助：help，help CMD

强大的网络上传下载：lftp

lftp 类似 ftp，但功能要强大的多，一般格式为：

- lftp [−d] [−e cmd] [−p port] [−u user[,pass]] [site]
- lftp −f script_file
- lftp −c commands

与 ftp 相比之外的主要特殊命令：

- 目录镜像下载：mirror
- 目录镜像上载：mirror -R
- 查看文件：cat、zcat
- 运行本地命令：!CMD，比如 !ls
- 下载已知地址文件： Slftp ftp://url

网络下载上传：wget/curl/axel

wget、curl 支持网络上传下载，可以批量下载及镜像网站，并可以模拟网页点击，进行网络登录后的操作，比如设置网络通等，非常强大

- 文件续传：wget −c http://url
- 下载 http://vcd.gro.clinux.org/doc/usr_01.html 到 http://vcd.gro.clinux.org/doc/usr_15.html：
  $ curl −O http://vcd.gro.clinux.org/doc/usr_[01−15].html

axel 支持并发多线程同时从多个或单个服务器下载，不要太狠，小心文明使用，如以两个线程下载：

$ axel −n 2 http://scc.ustc.edu.cn/zh_CN/article/46/4b29ea78/4b29ea92.pdf

Shell 基础

- 通配符
- 输入输出重定向
- 管道
- 命令组

Shell 简介

- Shell 的作用是解释执行用户的命令，用户输入一条命令，Shell 就解释执行一条，这种方式称为交互式（Interactive）
- Shell 还有一种执行命令的方式称为批处理（Batch），用户事先写一个 Shell 脚本（Script），其中有很多条命令，让 Shell 一次把这些命令执行完，而不必一条一条地敲命令
- Shell 脚本和编程语言很相似，也有变量和流程控制语句，但 Shell 脚本是解释执行的，不需要编译，Shell 程序从脚本中一行一行读取并执行这些命令，相当于一个用户把脚本中的命令一行一行敲到 Shell 提示符下执行

- 作为操作系统的交互式命令解释程序，它在用户和操作系统之间提供了一个面向行的可交互接口
- 作为一种命令级的程序设计语言，具有变量设置、结构控制、子程序调用、参数传递、中断处理等

常见 Shell 有：Csh、Tcsh、Ksh、Bash 等，以下主要基于 Bash 说明

Shell 编程部分主要参考：Linux C 编程一站式学习之 31. Shell 脚本（宋劲杉，北京亚嵌教育研究中心）

Bash 配置文件

Bash 运行时会读取初始配置文件，退出时也会执行相关文件中的命令，主要包括以下文件：

- /etc/profile：系统范围初始文件，当用户登录进时执行，一般用于系统默认的设置
- /etc/bash.bashrc：系统范围初始文件，每个新终端都执行
- /etc/bash.bash.logout：系统范围清理文件，每次登录退出时执行
- ~/.bash_profile：用户个人的初始化文件，登录的 Shell 会执行
- ~/.bashrc：个人初始化文件，每个新终端都执行
- ~/.bash_logout：个人登录退出时的清理文件，每个登录 Shell 退出时会执行

文件名中的字符代用字

- 单字符代用字：？
- 多字符代用字：*
- 包含代用字：[ ] ! [-]

标准输入输出文件

- 标准输入（0）
- 标准输出（1）
- 标准错误输出（2）

输入输出重定向：>

- 输入重定向：command < filename
- 输出重定向：command > filename
- 错误重定向：command 2> filename
- 正常和错误重定向同一个文件：
  - command &>filename
  - command >&filename
  - command >filename 2>&1

管道：|

管道 | 可以使一个命令的标准输出成为另一个命令的标准输入，利用管道可以把一堆命令组合起来，合作完成任务，一般格式为：

cmd1 | cmd2

比如，统计多少个进程名含有 sh 的进程数：

```
$ ps axu | grep sh | wc -l
```

如果能利用好 | 将命令组合起来可以完成非常复杂的操作，解放双手，一定要记住勤用管道功能

自动补全命令行

自动补全命令行就是在输入命令时不必把命令输全，按 <Tab> 时 Shell 能判断出所要输入的命令，如果有多个匹配将会列出，比如：

- $ ls /boot/v<Tab>：系统将自动补全 /boot 下以 v 开头的文件或目录
- $ pass<Tab>：系统会补全 passwd 命令

别名：alias 与 unalias

alias 可以定义某些命令的别名以简化操作等，比如
alias psl='ps aux | grep'，运行 psl sh 就相当于运行 ps aux | grep sh
定义的命令仅仅在当前 Shell 中有效，为了每次登录都有效，应该放入下面所说的 ~/.bashrc 之类的文件中
取消别名可以用 unalias，比如 unalias psl

~/.bashrc 实例

```
# ~/.bashrc: executed by bash(1) for non-login shells.
umask 022
alias ls='ls $LS_OPTIONS'
alias ll='ls $LS_OPTIONS -l'
alias la='ls $LS_OPTIONS -lA'
alias l='ll'
alias cp='cp -i'
alias mv='mv -i'
alias rm='rm -i'
MPIROOT=/opt/mpich2
export PATH=$PATH:$MPIROOT/bin
export MANPATH=$MANPATH:$MPIROOT/share/man
. /opt/intel/Compiler/11.1/059/bin/ifortvars.sh ia32
```

其中 # 后面的做为注释

Shell 变量

变量是可赋值的名字，其值可以是字符串、数字等

- 用户变量：由用户创建和赋值的变量
- 环境变量：由 Shell 维护，用于配置系统工作环境的一组变量，可以由用户改变
- 特殊变量：由 Shell 设置的，不能改变。例如参数个数，进程号退出状态。

Shell 脚本中的变量除非用 export 输出做为环境变量，否则都是本地变量，仅在本脚本中有效，脚本结束，变量随之消亡

环境变量

系统具有许多环境变量，利用 env 命令可以打印出所有已经设置的环境变量及其值，以下是一些系统常用的环境变量（其它系统未必和我的路径一致，只能参考）：

- man 文档路径：MANPATH=/opt/intel/Compiler/11.1/059/man/en_US:/opt/mpich/share/man
- Shell 路径：SHELL=/bin/bash
- 终端 TERM 名：TERM=xterm
- 库路径：
  LIBRARY_PATH=/opt/intel/Compiler/11.1/059/lib/ia32:/opt/intel/Compiler/11.1/059/mkl/lib/32
- Fortran 头文件路径：FPATH=/opt/intel/Compiler/11.1/059/mkl/include
- 用户：USER=hmli
- 链接库路径：
  LD_LIBRARY_PATH=/opt/intel/Compiler/11.1/059/lib/ia32:/opt/intel/Compiler/11.1/059/mkl/lib/3
- C 头文件路径：CPATH=/opt/intel/Compiler/11.1/059/mkl/include
- 用户名：USERNAME=hmli
- 命令搜索路径：
  PATH=/opt/intel/Compiler/11.1/059/bin/ia32:/usr/local/bin:/usr/bin:/bin:/opt/mpich/bin
- 当前路径：PWD=/home/hmli/tex/linux
- 语言环境：LANG=zh_CN.GBK
- 用户主目录：HOME=/home/hmli
- 登录名：LOGNAME=hmli
- 头文件目录：INCLUDE=/opt/intel/Compiler/11.1/059/mkl/include

对于 Bash 用户可以在 ~/.bashrc 中设置利用 export 命令来设置，如将自己根目录下的 bin 目录加入命令搜索路径，以直接运行此目录中的命令可以在 ~/.bashrc 中添加：export PATH=$HOME/bin:$PATH

环境变量影响用户的使用，一定要小心按照变量要求的格式设置

用户变量

变量名可以是由字母开头的任意字母、数字组成的序列，变量值以字符形式存储

- 申报和管理用户变量：
  VAR=string
- 取消变量的定义：
  unset VAR
- 连接变量：
  VAR=$VAR1"xxx"$VAR2
- 引用变量的值：
  变量前添加 $，如 $VAR
- 显示变量的值：
  echo $VAR

如果变量值符合数字格式，则可以直接用于计算

命令代换：` 或 $()

反引号 ` 或 $() 将其内命令执行结果赋值给变量，比如将系统时间赋值给变量 DATE 可用下列格式之一：

- DATE=`date`
- DATE=$(date)

算数代换：$(())

$(()) 用于数学计算，比如执行加法：

```
VAR=2
SUM=$((VAR+3))
echo $SUM
```

转义字符：\

和 C 语言类似，\ 在 Shell 中被用作转义字符，用于去除紧跟其后的单个字符的特殊意义（回车除外），换句话说，紧跟其后的字符取字面值，如对比：

- $ echo $SHELL
- $ echo \$SHELL

单双引号：’与”

和 C 语言不一样，Shell 脚本中的单引号 ' 和双引号 ” 一样都是字符串的界定符，而不是字符的界定符

- 单引号用于保持引号内所有字符的字面值，即使引号内的 \ 和回车也不例外，但是字符串中不能出现单引号。如果引号没有配对就输入回车，Shell 会给出续行提示符，要求用户把引号配上对
- 双引号用于保持引号内所有字符的字面值（回车也不例外），但以下情况除外：
  - \$ 加变量名可以取变量的值
  - 反引号 ‘ 仍表示命令替换
  - \$ 表示 $ 的字面值
  - \‘ 表示 ‘ 的字面值
  - \” 表示 ” 的字面值
  - \\ 表示 \ 的字面值
  - 除以上情况之外，在其它字符前面的 \ 无特殊含义，只表示字面值

举例对比：

- echo '$SHELL'
- echo ”$SHELL”

条件测试：test 与 [

命令 test 或 [可以测试一个条件是否成立，如果测试结果为真，则该命令的 Exit Status 为 0，如果测试结果为假，则命令的 Exit Status 为 1（注意与 C 语言的逻辑表示正好相反）
虽然看起来很奇怪，但左方括号 [确实是一个命令的名字，传给命令的各参数之间应用空格隔开，如 $VAR、-gt、3、] 是 [命令的四个参数，它们之间必须用空格隔开。test 或 [的参数形式是相同的，只不过 test 命令不需要] 参数。以 [命令为例，常见的测试命令如下表所示：

[-d DIR]	如果 DIR 存在并且是一个目录则为真
[-f FILE]	如果 FILE 存在且是一个普通文件则为真
[-z STRING]	如果 STRING 的长度为零则为真
[-n STRING]	如果 STRING 的长度非零则为真
[STRING1 = STRING2]	如果两个字符串相同则为真
[STRING1 != STRING2]	如果字符串不相同则为真
[ARG1 OP ARG2]	ARG1 和 ARG2 应该是整数或者取值为整数的变量，OP 是 -eq（等于）-ne（不等于）-lt（小于）-le（小于等于）-gt（大于）-ge（大于等于）之中的一个

与、或、非的测试命令

和 C 语言类似，测试条件之间还可以做与、或、非逻辑运算：

[! EXPR]	EXPR 可以是上表中的任意一种测试条件，! 表示逻辑反
[EXPR1 -a EXPR2]	EXPR1 和 EXPR2 可以是上表中的任意一种测试条件，-a 表示逻辑与
[EXPR1 -o EXPR2]	EXPR1 和 EXPR2 可以是上表中的任意一种测试条件，-o 表示逻辑或

if/then/elif/else/fi 语句

和 C 语言类似，在 Shell 中用 if、then、 elif 、 else、 fi 这几条命令实现分支控制。这种流程控制语句本质上也是由若干条 Shell 命令组成的

```
#! /bin/sh
echo "Is it  morning? Please answer yes or no."
read YES_OR_NO
if  [ "$YES_OR_NO" = "yes" ]; then
      echo "Good morning!"
 elif  [ "$YES_OR_NO" = "no" ]; then
      echo "Good afternoon!"
else
      echo "Sorry, $YES_OR_NO not recognized. Enter yes or no."
      exit 1
fi
exit 0
```

case/esac 选择语句

case 命令可类比 C 语言的 switch/case 语句，esac 表示 case 语句块的结束。C 语言的 case 只能匹配整型或字符型常量表达式，而 Shell 脚本的 case 可以匹配字符串和 Wildcard，每个匹配分支可以有若干条命令，末尾必须以 ;; 结束，执行时找到第一个匹配的分支并执行相应的命令，然后直接跳到 esac 之后，不需要像 C 语言一样用 break 跳出。

```
#! /bin/sh
echo "Is it morning? Please answer yes or no."
read YES_OR_NO
case "$YES_OR_NO" in
yes|y|Yes|YES)
    echo "Good Morning!";;
[nN]*)
    echo "Good Afternoon!";;
*)
    echo "Sorry, $YES_OR_NO not recognized. Enter yes or no."
esac
```

for/do/done 循环

Shell 脚本的 for 循环结构和 C 语言很不一样，它类似于某些编程语言的 foreach 循环。例如：

```
#! /bin/sh
for FRUIT in apple banana pear; do
    echo "I like $FRUIT"
done
```

以下脚本巡检 node1 到 node100 并打印系统名

```
#! /bin/sh
for i in $(seq 1 100)
do
    ssh node$i hostname
done
```

while/do/done 循环

while 的用法和 C 语言类似。如一个监视进程内存利用情况的脚本：

```
#!/bin/bash
#Program: monitor.sh
# Author: HM Li <hmli@ustc.edu.cn>
#   Date: 09-12-14
#Comment: Monitor CPU and Memory
while [ 0 ]
do
    date
    echo ----------------------------------------
    echo "RUSER   PID %CPU COMMAND %MEM SZ"
    ps -o "%u %p %C %c" -o "%mem,size" -U mgbouc,muxd,heyong | sort -nr -k 5 | head -n 10
    sleep 10
done
```

以上脚本是个死循环，只能不用时利用 kill 等杀掉

位置参数和特殊变量

有很多特殊变量是被 Shell 自动赋值的：

$0	相当于 C 语言 main 函数的 argv[0]
$1、$2...	这些称为位置参数（Positional Parameter）， 相当于 C 语言 main 函数的 argv[1]、argv[2]...
$#	相当于 C 语言 main 函数的 argc - 1，注意 # 后面不表示注释
$@	表示参数列表 "1" "2" ...，例如可用在 for 循环中的 in 后面
$*	表示参数列表 "1"c"2"c...，c 为 Shell 分割符 IFS 的第一个字符，如未设置则为空格
$?	上一条命令的 Exit Status
$$	当前 Shell 的进程号

位置参数移动 shift

位置参数可以用 shift 命令左移。比如 shift 3 表示原来的 $4 现在变成 $1，原来的 $5 现在变成 $2 等，原来的 $1、$2、$3 丢弃，$0 不移动。不带参数的 shift 命令相当于 shift 1。例如：

```
#! /bin/sh

echo "The program $0 is now running"
echo "The first parameter is $1"
echo "The second parameter is $2"
echo "The parameter list is $@"
shift
echo "The first parameter is $1"
echo "The second parameter is $2"
echo "The parameter list is $@"
```

函数

和 C 语言类似，Shell 中也有函数的概念，但是函数定义中没有返回值也没有参数列表。例如：

```
#! /bin/sh
foo(){ echo "Function foo is called";}
echo "--=start=--"
foo
echo "--=end=--"
```

注意函数体的左花括号 { 和后面的命令之间必须有空格或换行，如果将最后一条命令和右花括号 } 写在同一行，命令末尾必须有 ;。
在定义 foo() 函数时并不执行函数体中的命令，就像定义变量一样，只是给 foo 这个名字一个定义，到后面调用 foo 函数的时候（注意 Shell 中的函数调用不写括号）才执行函数体中的命令。Shell 脚本中的函数必须先定义后调用，一般把函数定义都写在脚本的前面，把函数调用和其它命令写在脚本的最后（类似 C 语言中的 main 函数，这才是整个脚本实际开始执行命令的地方）。
Shell 函数没有参数列表并不表示不能传参数，事实上，函数就像是迷你脚本，调用函数时可以传任意个参数，在函数内同样是用 $0、$1、$2 等变量来提取参数，函数中的位置参数相当于函数的局部变量，改变这些变量并不会影响函数外面的 $0、$1、$2 等变量。函数中可以用 return 命令返回，如 return 后面跟一个数字则表示函数的 Exit Status。

函数举例

下面这个脚本可以一次创建多个目录，各目录名通过命令行参数传入，脚本逐个测试各目录是否存在，如果目录不存在，首先打印信息然后试着创建该目录。

```
#! /bin/sh
is_directory ()
{
  DIR_NAME=$1
  if [ ! -d $DIR_NAME ]; then
    return 1
  else
    return 0
  fi
}

for DIR in "$@"; do
  if  is_directory  "$DIR"; then :
  else
    echo "$DIR doesn't exist. Creating it now..."
    mkdir $DIR > /dev/null 2>&1
    if [ $? -ne 0 ]; then
      echo "Cannot create directory $DIR"
      exit 1
    fi
  fi
done
```

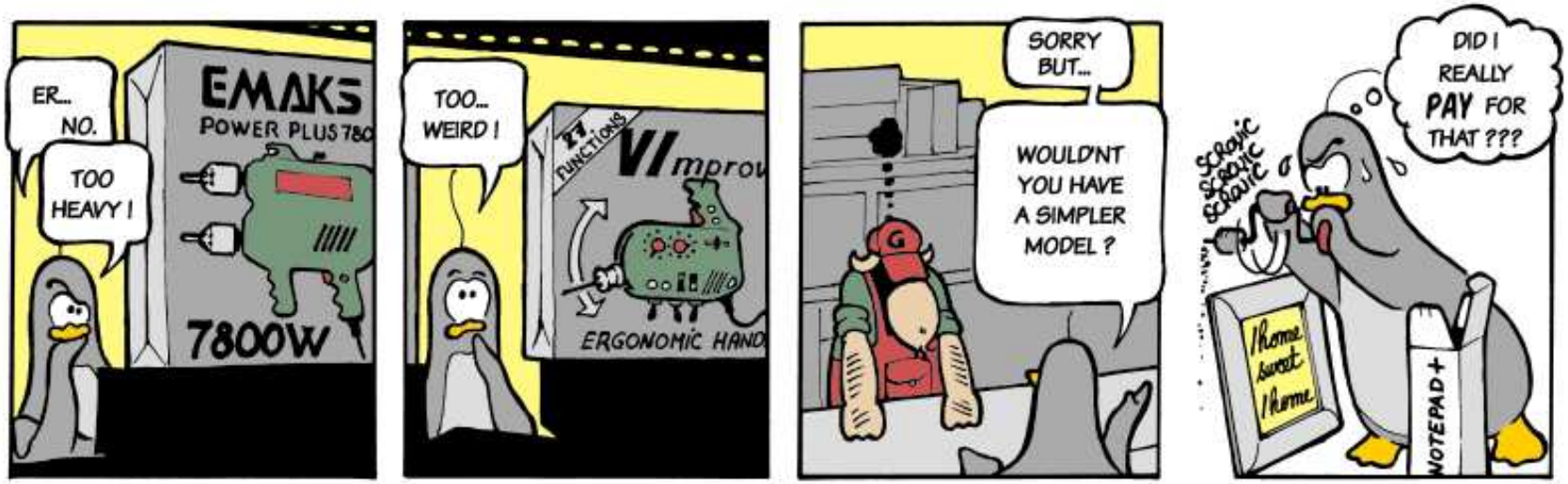
高效编辑：两大编辑器 VIM 与 Emacs

- 有人说：世界上的程序员分三种，一种使用 Emacs，一种使用 VIM，剩余的是其它（完全无视其他编辑器的存在……）
- VIM 号称编辑器的神，Emacs 号称神的编辑器，用户群都非常广
- 由于区分了模式，导致 VI 的命令非常简洁，而无模式编辑器比如 Emacs，所有的命令都需要加上控制键 Ctrl 或 Alt，所以有个笑话说 Emacser 们最希望计算机备一个脚踏板，这样就可以用脚踩 Ctrl 和 Alt 键了（编辑器圣战——在 VI 和 Emacs 之间有很多口水战，自然也引出非常多的幽默）
- VI 继承了 ed 的理念，VI 追求的是快捷——启动程序迅速，编辑文本高效，功能专注
- Emacs 追求的是功能的丰富强大以及集成带来的方便，在 Emacs 里头可以发邮件，上新闻组，听 mp3，浏览网页，玩游戏，几乎可以 login->Emacs->logout 了:-)
- VI 和 Emacs 都是程序员的编辑器，相比而言，Emacs 更是提供了一种程序员的生活氛围

建议看看 VIM 作者 Bram Moolenaar 写的 Seven Habits of Effective Text Editing，可从以下地址下载：

http://124.16.151.186/docs/doc-main.php?dir=linux

有趣图片


ER...
NO.
TOO HEAVY !
EMAKS
POWER PLUS 780
7800W
TOO... WEIRD !
77 FUNCTIONS
VImprov
ERGONOMIC HAND
SORRY BUT...
WOULD'NT YOU HAVE A SIMPLER MODEL ?
DID I REALLY PAY FOR THAT ???
Home sweet Home
NOTEPAD+

VI/VIM 编辑器简介

- Bram Moolenaar 80 年代末从一个开源的 VI 复制 Stevie 开始，开发了 VIM 的 1.0 版本。最初的目标只是完全复制 VI 的功能，那个时候的 VIM 是Vi IMitation（模拟）的简称。1992 年 1.22 版本的 VIM 被移植到了 UNIX 和 MS-DOS 上。从那个时候开始，VIM 的全名就变成 Vi IMproved（改良）了。
- VIM 加入了不计其数的新功能：多视窗编辑模式（分割视窗），同一萤幕显示的编辑文件数可以不止一个；图形界面（GUI）；highlight（语法高亮）；代码折叠、插件、多国语言支持、垂直分割视窗；拼字检查、上下文相关补全，标签页编辑；脚本的浮点数支持。
- 目前，VIM 是按照 GPL 协议发布的开源软件。它的协议中包含一些慈善条款，建议用户向荷兰 ICCF 捐款，用于帮助乌干达的艾滋病患者。VIM 启动时会显示 Help poor children in Uganda! 的字样，在中文版本中则是帮助乌干达的可怜儿童！

VI/VIM 模式

VI/VIM 大体可以分为两种模式：

- 命令模式：从键盘上输入的任何字符都被作为编辑命令来解释，按 <ESC> 进入此模式。
- 输入模式：从键盘上输入的所有字符都被插入到正在编辑的缓冲区中，被当作正文，按 a、i、c、o 等进入此模式。

实际 VIM 具有 6 种基本模式和 5 种派生模式。以下介绍基于 VIM，VI 仅支持部分以下功能

VIM 基本模式

- **普通模式**
  在普通模式中，用户可以执行一般的编辑器命令，比如移动光标，删除文本等等。这也是 VIM 启动后的默认模式。这正好和许多新用户期待的操作方式相反（大多数编辑器默认模式为插入模式）。
  VIM 强大的编辑能力中很大部分是来自于其普通模式命令。普通模式命令往往需要一个操作符结尾。例如普通模式命令 ”dd” 删除当前行，但是第一个 ”d” 的后面可以跟另外的移动命令来代替第二个 ”d”，比如用移动到下一行的 ”j” 键就可以删除当前行和下一行。另外还可以指定命令重复次数，”2dd”（重复 ”dd” 两次），和 ”dj” 的效果是一样的。用户学习了各种各样的文本间移动/跳转的命令和其他的普通模式的编辑命令，并且能够灵活组合使用的话，能够比那些没有模式的编辑器更加高效的进行文本编辑。
  在普通模式中，有很多方法可进入插入模式。比较普通的方式是按 ”a”（append/追加）或者 ”i”（insert/插入）键。
- **插入模式**
  在这个模式中，大多数按键都会向文本缓冲中插入文本。大多数新用户希望文本编辑器编辑过程中一直保持这个模式。
  在插入模式中，可以按 <ESC> 键回到普通模式。
- **可视模式**
  这个模式与普通模式比较相似。但是移动命令会扩大高亮的文本区域。高亮区域可以是字符、行或者是一块文本。当执行一个非移动命令时，命令会被执行到这块高亮的区域上。VIM 的”文本对象”也能和移动命令一样用在这个模式中。
- **选择模式**
  这个模式和无模式编辑器的行为比较相似（Windows标准文本控件的方式）。这个模式中，可以用鼠标或者光标键高亮选择文本，不过输入任何字符的话，VIM 会用这个字符替换选择的高亮文本块，并且自动进入插入模式。
- **命令行模式**
  在命令行模式中可以输入会被解释成并执行的文本。例如执行命令（”:” 键），搜索（”/” 和 ”?” 键）或者过滤命令（”!” 键）。在命令执行之后，VIM 返回到命令行模式之前的模式，通常是普通模式。
- **Ex 模式**
  这和命令行模式比较相似，在使用 ”:visual” 命令离开 Ex 模式前，可以一次执行多条命令。

VIM 派生模式

- 操作符等待模式
  此派生模式指普通模式中，执行一个操作命令后 VIM 等待一个”动作”来完成这个命令。VIM 也支持在操作符等待模式中使用”文本对象”作为动作，包括 ”aw” 一个单词(a word)、”as” 一个句子(a sentence)、”ap” 一个段落(a paragraph)等。如，在普通模式下 ”d2as” 删除当前和下一个句子。在可视模式下 ”apU” 把当前段落所有字母大写。
- 插入普通模式
  这个模式是在插入模式下按下ctrl-o键的时候进入。这个时候暂时进入普通模式，执行完一个命令之后，VIM 返回插入模式
- 插入可视模式
  这个模式是在插入模式下按下ctrl-o键并且开始一个可视选择的时候开始。在可视区域选择取消的时候，VIM 返回插入模式。
- 插入选择模式
  通常这个模式由插入模式下鼠标拖拽或者shift方向键来进入。当选择区域取消的时候，VIM 返回插入模式。
- 替换模式
  这是一个特殊的插入模式，在这个模式中可以做和插入模式一样的操作，但是每个输入的字符都会覆盖文本缓冲中已经存在的字符。在普通模式下按 ”R” 键进入。

进入/退出 VIM

- 进入 VIM
  $ vim file
- 退出 VIM
  - :q：退出未被编辑或已保存的文件
  - :q!：强行退出 VIM，放弃未保存的修改
  - :x：存盘退出 VIM
  - :wq：存盘退出 VIM

查询帮助

在命令模式下 :help 或 help keyword 可以进入在线帮助，移动光标到对应主题（用 || 括起部分）按 <ctrl−]>，进入此主题，按 <ctrl−t 返回上一级，按 q 退出帮助

文本输入

以下操作在命令模式下

- 插入命令：
  - i：插在当前光标之前
  - I：插在当前行的最开始
- 附加命令：
  - a：插在当前光标之后
  - A：插在当前行的最后
- 新行命令：
  - o：在当前行的下一行另开新空白行开始输入
  - O：在当前行的上一行另开新空白行开始输入

移动光标

以下操作在命令模式下
建议用 hjkl 移动光标，以保证手不离开键盘的主输入区输入更快

<up_arrow> ↑
<ctrl-P>

\- k

← <left-arrow> h l <right-arrow> →
Backspace Space

\+ j

<ctrl-N>
<down_arrow> ↓

行列移动光标

以下操作在命令模式下

- 移至行首：^、0（数字零）
- 移至行尾：$
- 移至指定行：:[行号]回车 或 [行号]G
- 移至指定百分比的行：[百分比]%回车
- 移至指定列：[列号]|

上下翻屏

以下操作在命令模式下

- 滚屏命令：<ctrl-u> 和 <ctrl-d>，分别向上和向下滚动半个窗口
- 分页命令：<ctrl-f> 和 <ctrl-b>，分别向前和向后分页

文本删除

以下操作在命令模式下

- 删除字符
  - x 或 nx：从光标所在的位置删除一个或 n 个字符
  - X 或 nX：删除光标前的一个或 n 个字符
- 删除文本对象
  - dd：删除光标所在的行
  - D：删除从光标位置开始至行尾
  - dw：删除从光标位置至该词末的所有字符
  - d0：删除从光标位置开始至行首
  - d5G：将光标所在行至第 5 行删除
  - dG：将光标所在行到文件末尾删除

修改命令

以下操作在命令模式下

- r：替换一个字符
- R：替换本行光标出字符以后的
- c：替换掉光标处字符并添加
- C：修改光标后的到行尾
- ~：大小写转换：
- guu：本行全部转化为小写
- gUU：本行全部转化为大写

搜索和替换命令

以下操作在命令模式下

- 搜索：/所要搜索的内容
- 替换：:[range]s/所要替换的内容/替换成的内容/[flags]，如果[range]为 %，则对每一行都执行替换，如果 [flag] 为 g，则对同一行内所有匹配的都执行替换，否则只对第一次匹配的替换。支持正则表达式，比如把 2009-01-31 或 2009-1-3 格式的日期全部替换为 01/31/2009 和 1/3/2009 格式：
  %s~\v(\d{4})-(\d{1,2})-(\d{1,2})~\2/\3/\1~g

复原和重复命令

以下操作在命令模式下

- u：若插入后用此命令，就删除刚插入的正文；若删除后用它，则插入刚删除的正文
- U：把当前行恢复成它被编辑之前的状态
- .：重复实现刚才的插入命令或删除命令

可视模式

在命令模式下输入 <ctrl-v> 进入可视模式，可以移动光标选择行列块，并对此块内的内容进行处理

文件比较：vimdiff

vim −d 多个文件或 vimdiff 多个文件，可以对多个文件进行比较编辑

要遵守的一般步骤提要

进入 VIM	键入 vim 并按回车
到输入模式	按 a 或 i
输入文本	将文本键入缓冲区
到命令模式	按 <ESC>
保存缓冲区到文件	键入 :w file 并按回车
退出 VIM	键入 :q 并按回车

gvim 或 vim −g 将启动图形界面的 VIM，并支持鼠标操作

键盘映射

:map 可以查看当前键盘映射，也可以设置键盘映射，如本人利用 beamer 写幻灯片时为了简化操作的映射：

```
map <F4> :set paste<CR>o\begin{frame}{<++>}<CR><++><CR>\end{frame}<Esc>2k<C-j>
```

按 <F4> 后将自动输入以下内容，并将光标以插入模式定位到{}中，以方便输入内容：

```
\begin{frame}{}
<++>
\end{frame}
```

VIM 插件

VIM 支持插件，可以针对不同的类型的文件自动完成某种操作或键盘映射，以方便操作
比如，利用 C 语言的一个插件，运行 vim myprog.c 将自动在 myprog.c 中添加以下内容，还支持函数补全等

```
/*
 * ===========================================================================
 *
 *         Filename:  myprog.c
 *
 *      Description:
 *
 *          Version:  1.0
 *          Created:  2009年12月16日21时00分15秒
 *         Revision:  none
 *         Compiler:  gcc
 *
 *           Author:  YOUR NAME (),
 *          Company:
 *
 * ===========================================================================
 */
```

VIM 设置

用户可以在自己的~/.vimrc 中配置需要的 VIM 选项，” 后的为注释

```
syntax on           ” 打开语法高亮
set hlsearch ”搜索时高亮显示匹配的字符
filetype plugin on ”自动判断文件类型，比如判断 C、Fortran 源文件等
filetype indent on ”根据文件类型自动缩进
set nocompatible      ”不使用兼容模式，功能强
set backspace=indent,eol,start    ”设置backspace可以回退删除字符
set autoindent        ”设置自动缩进
set textwidth=0       ”设置行宽，0，表示不自动换行
set backup          ” 设置使用备份
set bdir=~/tmp/vim ”设置备份目录，如果不设置，将自动对编辑的文件以在原文件名后添加 的文件做为备份文件
set viminfo='20,\”50       ”设置启动时读取历史信息的长度
set history=50        ”设置命令历史长度
set ruler           ”设置显示当前光标位置
set ts=4 ”tabstop: 设置 tab 长度
set sw=4 ”shiftwidth: 设置自动使用每层缩进的空格数
” The following are commented out as they cause vim to behave a lot
” different from regular vi. They are highly recommended though.
set showcmd       ”显示命令
set showmatch         ”显示匹配
set ignorecase        ”忽略大小写
set incsearch         ”输入搜索命令时，显示目前输入的模式的匹配位置
set autowrite         ”执行外部命令时自动保存
set encoding=2byte-cp936 ”设置使用中文2byte-cp936编码
let fortran_have_tabs=1 ”设置不要高亮 Fortran 77 源文件中的 tab
```

VIM 入门教程命令：vimtutor

在终端中输入 vimtutor 可以调出自带的《VIM教程》，直接按照所说的在启动的说明文件中进行操作即可，无需另打开其它文件进行学习。

- VIM 是一个具有很多命令的功能非常强大的编辑器。限于篇幅，在本教程当中就不详细介绍了。本教程的设计目标是讲述一些必要的基本命令，而掌握好这些命令，您就能够很容易将vim当作一个通用的万能编辑器来使用了。
- 完成本教程的内容大约需要 25-30 分钟，取决于您训练的时间。
- 每一节的命令操作将会更改本文。推荐您复制本文的一个副本，然后在副本上进行训练(如果您是通过”vimtutor”来启动教程的，那么本文就已经是副本了)。
- 切记一点：本教程的设计思路是在使用中进行学习的。也就是说，您需要通过执行命令来学习它们本身的正确用法。如果您只是阅读而不操作，那么您可能会很快遗忘这些命令的！

awk 简介

- awk 名称得于它的创始人 Alfred Aho、Peter Weinberger 和 Brian Kernighan 姓氏的首字母
- awk 是 Unix/Linux 环境中现有的功能最强大的数据处理引擎之一
- awk 设计思想来源于 SNOBOL4、sed、Marc Rochkind 设计的有效性语言、语言工具 yacc 和 lex，还从 C 语言中获取了一些优秀的思想
- awk 具有完全属于其本身的语法，在很多方面类似 Unix Shell 编程语言
- awk 提供了极其强大的功能：可以进行正则表达式的匹配，样式装入、流控制、数学运算符、进程控制语句甚至于内置的变量和函数
- awk 的目的是用于文本处理，只要在输入数据中有模式匹配，就执行一系列指令，扫描文件中的每一行，查找与命令行中所给定内容相匹配的模式，如发现匹配内容，则进行下一个编程步骤，如果找不到匹配内容，则继续处理下一行

awk 语法

awk 命令的语法基本是：

```
awk '{pattern + action}' {filenames}
```

- pattern 表示 awk 在数据中查找的内容，支持正则表达式
- action 是在找到匹配内容时所执行的一系列命令
- 花括号（{}）不需要在程序中始终出现，但它们用于根据特定的模式对一系列指令进行分组
- pattern 和 action 整体之外需用 '' 括起来

第一个 awk

```
awk –F: '{ print $1 }' /etc/passwd
```

- 执行此 awk 命令时，它依次对输入文件 /etc/passwd 文件中的每一行执行 print 命令，所有输出都发送到标准输出（stdout）
- 花括号用于将几块代码组合到一起，类似于 C 语言
- print $1 命令，打印当前行的第一个字段

sed 简介

在 UNIX/Linux 世界中有很多文本编辑器，如 VI、VIM、Emacs 和 jed 等，尽管其交互式特性可以成为强项，但也有其不足之处，如需对一组文件执行类似更改的情形，使用者也许会本能地运行自己所喜爱的编辑器，然后手工执行一组烦琐、重复和耗时的编辑任务，非常繁琐。
如可使编辑文件的过程自动化，以便用“批处理”方式编辑文件，甚至编写可以对现有文件进行复杂更改的脚本，那将太好了，有一种更好的方法 sed(stream editor for filtering and transforming text)。
sed 是几乎包括在所有 UNIX/Linux 平台的轻量级流编辑器：

- 相当小巧，通常要比其它脚本语言小很多倍
- sed 是流编辑器，它可对从如管道这样的标准输入接收的数据进行编辑，无需将要编辑的数据存储在磁盘上的文件中
- 可以轻易将数据管道输出到 sed，所以将 sed 用作强大的 shell 脚本中长而复杂的管道很容易

GNU sed

- 对 Linux 用户来说最好的 sed 版本之一恰好是 GNU sed
- 每一个 Linux 发行版都有（或至少应该有）GNU sed
- GNU sed 之所以流行不仅因为可以自由分发其源代码，还因为它恰巧有许多对 POSIX sed 标准便利、省时的扩展
- GNU sed 没有 sed 早期专门版本的很多限制，如行长度限制，可处理任意长度的行

sed 处理方式

- 通过对输入数据执行任意数量指定的编辑操作（“命令”）来工作
- 基于行，因此按顺序对每一行执行命令
- 默认将其结果写入标准输出（stdout），不修改任何输入文件
- 如果添加 -i 参数则直接修改输入文件

sed 与 Unix/Linux 命令等价代码

Unix/Linux 命令	sed 等价代码
cat	sed ':'
cat -s	sed '/./,/^$/!d'
tac	sed '1!G;h;$!d'
grep	sed '/patt/!d'
grep -v	sed '/patt/d'
head	sed '10q'
head -1	sed 'q'
tail	sed -e ':a' -e '$q;N;11,$D;ba'
tail -1	sed '$!d'
tail -f	sed -u '/./!d'
cut -c 10	sed 's/(.){10}.*/1/'
cut -d: -f4	sed 's /(([^:]*):){4}.*/2/'
tr A-Z a-z	sed 'y/ABCDEFGHIJKLMNOPQRSTUVWXYZ/abcdefghijklmnopqrstuvwxyz/'
tr a-z A-Z	sed 'y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/'
tr -s ' '	sed 's/ +/ /g'
tr -d '\012'	sed 'H;$!d;g;s/\n//g'
wc -l	sed -n '$='
uniq	sed 'N;/^(.*)n1$/!P;D'
rev	sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1/;//D;s/.//'
basename	sed 's ,.*/,,'
dirname	sed 's ,[^/]*$,,'
xargs	sed -e ':a' -e '$!N;s/\n/ /;ta'
paste -sd:	sed -e ':a' -e '$!N;s/\n/:/;ta'
cat -n	sed '=' | sed '$!N;s/\n/ /'
grep -n	sed -n '/patt/{=;p;}' | sed '$!N;s/\n/:/'
cp orig new	sed 'w new' orig[/code:1:739eb4cef5]

科学可视化：gnuplot

Gnuplot 是一种免费分发的绘图工具，可以移植到各种主流平台。它可以下列两种模式之一进行操作：当需要调整和修饰图表使其正常显示时，通过在 gnuplot 提示符中发出命令，可以在交互模式下操作该工具。或者，gnuplot 可以从文件中读取命令，以批处理模式生成图表。例如，如果您正在运行一系列的实验，需要每次运行后都查看结果图表；或者当您在图表最初生成很久以后需要返回图表修改某些内容时，批处理模式能力会特别有用。当在 WYSIWIG 编辑器中很难捕获用于修饰图表的鼠标单击事件时，您可以很容易地将 gnuplot 命令保存在文件中，六个月后将其加载到交互式会话中重新执行。
Gnuplot 是在 1986 年由 Colin Kelley 和 Thomas Williams 最初开发的。许多参与者都在为不同的“终端”创建变种方面做出了贡献。在 1989 和 1990 年，这些变种被合并到 gnuplot 2.0 中。最新版本为 2009 年 9 月发布的 4.2.6。

- http://www.ibm.com/developerworks/cn/linux/l-gnuplot/index.html
- http://www.gnuplot.info/

简单实例

电学中的幅频响应代码，取自 gnuplot 自带的 electron.dem：

```
A(jw) = ({0,1}*jw/({0,1}*jw+p1)) * (1/(1+{0,1}*jw/p2))
p1 = 10
p2 = 10000
set dummy jw
set grid x y2
set key default
set logscale xy
set log x2
unset log y2
set title "Amplitude and Phase Frequency Response"
set xlabel "jw (radians)"
set xrange [1.1 : 90000.0]
set x2range [1.1 : 90000.0]
set ylabel "magnitude of A(jw)"
set y2label "Phase of A(jw) (degrees)"
set ytics nomirror
set y2tics
set tics out
set autoscale  y
set autoscale y2
plot abs(A(jw)), 180/pi*arg(A(jw)) axes x2y2
```

运行：

```
$ gnuplot electron.dem
```

科学可视化：GDL-GNU Data Language

开源的 IDL (Interactive Data Language) 兼容编译器，与 IDL 6.0 语法兼容。一个简单例子，利用 gdl 启动 GDL，后输入以下命令：

```
device,decomposed=1
plot,sin(findgen(360)*!dtor),color='ff00ff'xl,$
title='Sin(x)',xtitle='x',ytitle='sin(x)',linestyle=2
```

简单说明：

- device,decomposed=1 设置显示设备使用颜色分解
- sin：正弦函数
- findgen(360)：产生一个从 0 递增到 359 浮点序列
- !dtor：系统变量，角度变弧度
- color：设置也颜色
- title：设置标题
- xtitle：设置 X 轴标题
- ytitle：设置 Y 轴标题
- linestyle：设置曲线类型

主页：http://gnudatalanguage.sourceforge.net/

另外一些推荐软件

- C/C++、Fortran 编译器：GCC
- 开源的 matlab 替代品：octave
- 元素周期表：gelemental 与 gperiodic
- 具有画图等功能强大的科学计算器：Qalculate!
- 强大的命令行计算器：wcale
- 类似 Windows 下 windig 的数据图像取数据：g3data

附录 网络文档

- 常用论坛：
  - 科大 BBS Linux 系统：
    http://bbs.ustc.edu.cn/cgi/bbsdoc?board=Linux
  - 清华水木社区Linux 系统与应用：
    http://www.newsmth.net/bbsdoc.php?board=LinuxApp
  - 清华水木社区Linux 开发与高级讨论：
    http://www.newsmth.net/bbsdoc.php?board=LinuxDev
  - LinuxSir：http://www.linuxsir.org/
  - ChinaUnix：http://www.chinaunix.net/
- Linux 常用软件推荐集合汇总：
  http://www.linuxsir.org/bbs/showthread.php?t=199479
- Linux 指令大全：
  http://www.linuxsir.org/bbs/showthread.php?t=204304

联系信息

- 中国科大超算中心：
  - 主页：http://scc.ustc.edu.cn
  - 电话：0551-3602248
  - 信箱：sccadmin@ustc.edu.cn
- 青能所超算中心：
  - 当前主页：http://124.16.151.186
  - 将来域名：http://scc.qibebt.cas.cn
  - 电话：0532-80662613
  - 信箱：scc@qibebt.ac.cn
- 李会民：
  - 主页：http://staff.ustc.edu.cn/~hmli/
  - 电话：0532-80662613
  - 信箱：hmli@ustc.edu.cn、lihm@qibebt.ac.cn

欢迎指出错误和改进意见。