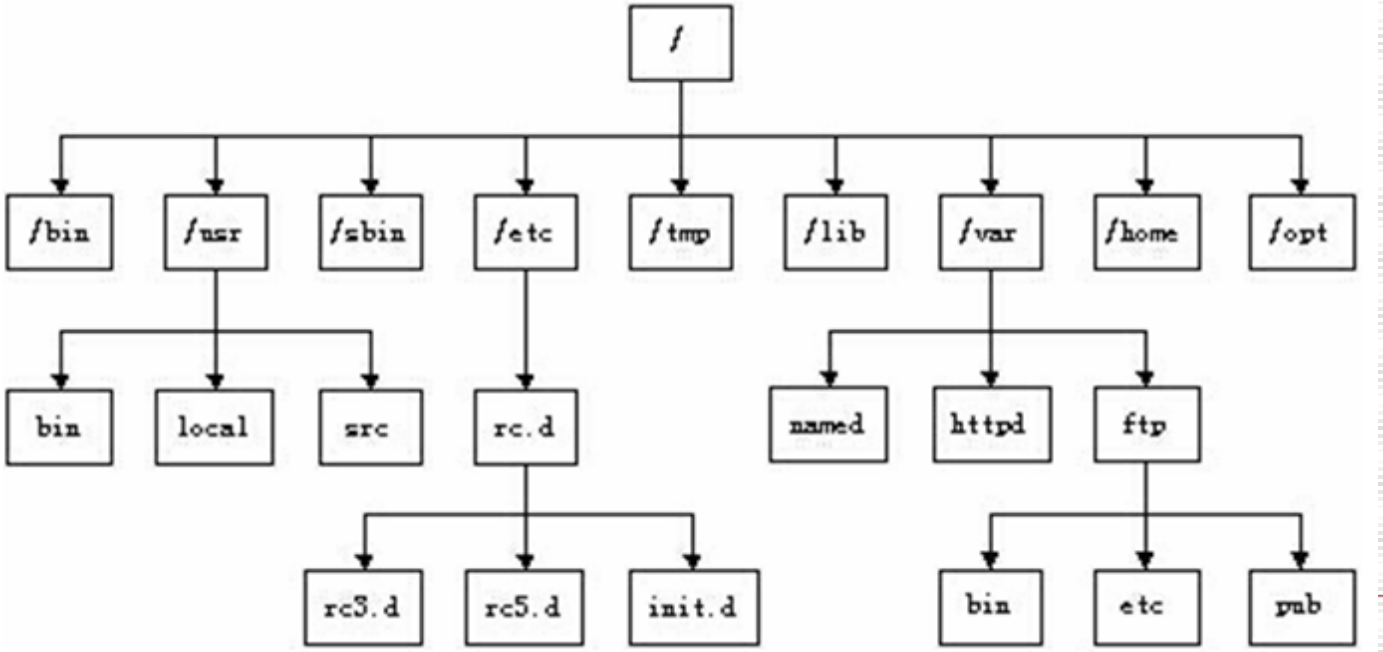
Linux Shell

文件系统结构

从用户角度理解文件系统

- 文件系统是Linux下的所有文件和目录的集合，这些文件和目录结构是以一个树状的结构来组织的， 这个树状结构构成了Linux中的文件系统。



- 根目录
- 起始目录
  - 当用户登录系统时，Linux将用户至于一个特殊的目录下
  - 可以用代字符“~”来指定起始目录
- 当前工作目录
  - 某个时刻所处的目录
  - 用字符“.”表示
  - 当前工作目录的父目录用“..”表示

路径名

- 在文件系统的层次结构中，文件或目录使用路径（pathname）来指定
- 如果一个路径用从根目录开始的名字来表示，则称它为*绝对路径名*（***absolute pathname***）。

  /users/faculty/sarwar/courses/ee446
- 从当前目录，或用户的起始目录开始的路径名叫相对路径名（***relative pathname***）。

  ./courses/ee446

  courses/ee446

确定起始目录的绝对路径名

echo string	
作用	将string内容显示在屏幕上

echo $HOME

pwd

浏览文件系统

cd [directory]	
作用	将当前工作目录切换到directory所代表的目录下，或者，在不指定参数情况下切换到起始目录下。

cd

cd mybin

cd ~

cd ..

ls [option] [pathname-list]	
作用	显示由pathname-list指定的文件和目录下的文件名
常用选项	
-F	在目录后显示“/”，在二进制文件后显示“*”，在符号连接后显示“@”
-a	显示所有的文件，包含隐含文件等
-i	显示索引节点号
-l	显示文件的详细列表
-R	递归显示当前目录和子目录下的文件和目录
-d	如果参数是目录，仅列出名称（不列内容）

```
$ ls -l
total 4232
-rwxr-xr-x   1 root   root      3756  Oct 14 04:44 dmesg
-r-xr-xr-x   1 root   root      12708 Oct  3 05:40 ps
-rwxr-xr-x   1 root   root      5388  Aug  5  1998 pwd
....
```

d 目录。
l 符号链接(指向另一个文件)。
s 套接字文件。
b 块设备文件。
c 字符设备文件。
p 命名管道文件。
- 普通文件，或者更准确地说，不属于以上几种类型的文件。

```
ls -l ~/courses/ee446/lab[!5]*.c

ls ~/[!0-9]*.[c,C]

ls [a-zA-Z]??[1-5].html
```

创建与删除目录

mkdir [option] [dirnames]	
目标	创建一个由dirnames指定的目录
常用选项	
-m MODE	创建一个目录并使它具有指定的访问权限
-p	创建“dir”时，首先创建所有不存在的父目录。

mkdir /lhome/lrj/mybin

mkdir -p mydoc/FAQ

rmdir [option] dirnames	
目标	删除由dirnames指定的空目录
常用选项	
-p	删除目录时同时删除为空的父目录。

rmdir somedir/

创建、删除和管理文件

cp [option] file1 file2 cp [option] file…file targetdir cp [option] sourcedir… targetdir	
目标	1）将文件file1的内容复制到file2,2）将文件复制到目录3）复制目录
常用选项	
-i	如果目标文件存在，则在覆盖前提是用户
-p	在复制的文件中保留源文件的访问权限和修改时间
-r	递归复制文件和子目录

```
cp /bin/?sh .

cp /bin/cpio mybin

cp abc bcd mydoc

cp /usr/bin/[yz]* .

cp –r /etc/skel .
```

mv [option] ‘source’ ‘target file’ mv [option] ‘source…’target dir’	
目标	mv命令来为文件或目录改名或将文件由一个目录移入另一个目录中
常用选项	
-f	忽略文件的访问权限，强制移动
-i	在文件移动时提示用户

mv /some/dir/file1 /someother/dir/

$ mv /some/dir/file1 /someother/dir/file2

$ mv /some/dir/files /someother/dir/

$ mv /some/dir/ /someother/dir/

$ mv file newname_file

$ mv dir newname_dir

rm [option] ‘file-list’	
目标	删除文件或目录
常用选项	
-f	忽略文件的访问权限，强制删除
-i	在文件删除时提示用户
-r	如果‘file-list’是一个目录，递归删除’file-list’中的文件和子目录。

显示文本文件内容

- cat 和 tac
- 功能：滚屏显示文本文件内容
  - cat 用于从文件头到文件尾显示
  - tac 用于从文件尾到文件头显示

$ cat file

$ tac file

- more 和 less
- 功能：分屏显示文本文件内容
  - more 只能从文件头到文件尾显示
  - less 可以使用PgUp和PgDn双向显示
- 用法：

  $ more file

  $ less file

- head 和 tail
- 功能：默认显示10行内容
  - head 显示文本文件的前部的若干行
  - tail 显示文本文件的后部的若干行
- 用法：（n为数字）

$ head file

$ head -n file

$ tail file

$ tail -n file

$ tail +n file

- wc
- 功能：统计指定文本文件的行数、字数、字符数
- 用法：

  $ wc file

  $ wc -l file

  $ wc -w file

  $ wc -c file

常用的信息显示命令

- date
- 功能：显示和设置日期时间
- 用法：

  $ date

  # date -s MM/DD/YYYY

  # date -s hh:mm:ss

- cal
- 功能：显示日历
- 用法：

  $ cal

  $ cal -y

  $ cal year

  $ cal mouth year

- locale
- 功能：显示当前语言环境
- 用法：

  $ locale

- file
  - 功能：显示指定文件的类型
  - 用法：

$ file filename

- stat
  - 功能：显示指定文件的各种相关信息
  - 用法：

$ stat filename

- dmesg
  - 功能：显示系统启动信息
  - 用法：

$ dmesg

- uname
  - 功能：显示操作系统信息
  - 用法：

$ uname

$ uname -r

$ uname -a

文件压缩命令

compress [options] [file-list]	
作用	压缩file-list中的文件，压缩后文件后缀为.Z。
常用选项	
-c	将压缩文件显示到屏幕上，而不是写入.Z文件中
-f	强制压缩（不带提示）
-v	显示压缩百分比以及被压缩的文件名字
d	将压缩档解压缩

$ compress somefilename

$ compress -d somefilename.Z

$ uncompress /mnt/lgx/a1.doc.Z

gzip [options] [file-list]	
作用	压缩file-list中的文件，压缩后文件后缀为.gz。
常用选项	
-c	输出到标准输出设备，输入文件保持不变
-f	强制压缩（不带提示）
-v	显示压缩百分比以及被压缩的文件名字
-N	根据N值控制压缩速度
-d	对压缩文件（.gz）进行解压
-r	递归压缩指定目录下的文件

```
$ gzip somefilename

$ gzip -d somefilename.gz

$ gunzip /mnt/lgx/a1.doc.gz
```

- **$ bzip2 somefilename**
- **$ bzip2 -d somefilename.bz2**

文件系统备份

tar [options] [file-list]	
作用	创建归档并添加或提取归档
常用选项	
-c	建立一个压缩文件的参数指令
-x	解开一个压缩文件的参数指令
-t	查看 tarfile 里面的文件
-z	是否需要用 gzip 压缩
-j	是否需要用 bzip2 压缩？
-v	压缩的过程中显示文件
-f	使用档名（在 f 之后要立即接档名 ）

-p	使用原文件的原来属性
-P	可以使用绝对路径来压缩
-N	比后面接的日期(yyyy/mm/dd)还要新的才会被打包进新建的文件中
--exclude FILE:	在压缩的过程中，不要将 FILE 打包

$ tar -cvf myball.tar somedirname

$ tar -tf myball.tar

$ tar -xvf myball.tar

$ tar -zcvf myball.tar.gz somedirname

$ tar -ztf myball.tar.gz

$ tar -zxvf myball.tar.gz

$ tar -Zcvf myball.tar.Z somedirname

$ tar -Ztf myball.tar.Z

$ tar -Zxvf myball.tar.Z

$ tar -jcvf myball.tar.bz2 somedirname

$ tar -jtf myball.tar.bz2

$ tar -jxvf myball.tar.bz2

```
tar -N "2011/09/28" -zcvf
    home.tar.gz /home

tar --exclude /home/dmtsai -zcvf
    myfile.tar.gz /home/* /etc

tar -jxv -f /root/etc.tar.bz2
    etc/shadow
```

重定向

- Linux下的标准设备

名称	代号	代表意思	设备	说明
STDIN	0	标准输入	键盘	命令在执行时所要的输入数据通过它来取得
STDOUT	1	标准输出	显示器	命令执行后的输出结果从该端口送出
STDERR	2	标准错误	显示器	命令执行时的错误信息通过该端口送出

- 重定向就是不使用系统的标准输入设备、标准输出设备或标准错误设备，而进行重新指定。
- 重定向分为输出重定向、输入重定向和错误重定向。
  - 输入重定向：不使用标准输入作为数据的输入，而是使用其它设备或文件获得输入数据
  - 输出重定向：不使用标准输出作为数据的输出，而是使用其它设备或文件作为数据输出
  - 错误重定向：不使用标准错误作为错误的输出，而是使用其它设备或文件作为错误输出

重定向符

- 输入重定向
  - <
  - <<!.....!
- 输出重定向
  - >
  - >>

合并输入和输出重定向

command<input-file>output-file

command>output-file<input-file

cat<lab1>lab2

带文件描述符的输入和输出重定向

- 利用文件描述符，可以分别使用操作符“0<”和“1>”来重定向标准输入和标准输出。

cat 1>

- 错误重定向
  - 2>
  - 2>>
  - &>（包括输出重定向）

在一个命令中重定向标准输出和标准错误

cat lab1 lab2 lab3 1>cat.output.errors 2>&1

字符串“2>&1”告诉命令shell将描述符2作为描述1的一个副本

cat lab1 lab2 lab3 2>cat.output.errors 1>&2

$ ls –l /tmp >mydir

$ ls –l /etc >>mydir

$ echo “Please call me : 62228899”>message

```
$wc < /etc/passwd

$wc <<!
>mytext
> This text forms the content of the
   heredocument ,
> which continues until the end of text
   delimiter
> !
```

```
$cat <<! >mytext
> This text forms the content of the
  heredocument ,
> which continues until the end of
  text delimiter
> !
```

```
$ myprogram 2> err_file

$ myprogram &> output_file
```

管道

- 管道：将一个命令的输出传送给令一个命令，作为另一个命令的输入
- 使用方法：

  命令1|命令2|命令3……|命令n

$ ls -Rl /etc |more

***R**：递归列出所有子目录

$ cat /etc/passwd | more

$ cat /etc/passwd | wc

命令执行顺序

- 命令间隔符
  - ；——用；间隔的各命令按顺序依次执行
  - &&——前后命令的执行存在“逻辑与”关系，只有&&前面的命令执行成功后，它后面的命令才被执行
  - || ——前后命令的执行存在“逻辑或”关系，只有||前面的命令执行失败后，它后面的命令才被执行
- 命令执行优先级
  - ；的优先级最低
  - ||和&&具有相同的优先级
  - 同优先级，按从左到右的结合原则执行命令行
  - 使用( )可以组合命令行中的命令，改变执行顺序

$ date ; pwd ; ls

顺序执行date、pwd和ls命令。

$ mail jjh < message && rm message

若文件message被mail发送出去，就把它删除，否则不删除。

$ write jjh < report || mail jjh < report

若对方拒绝对话，就将信息发送到他的信箱里。

$ date ; cat file |wc

只有cat命令的信息通过管道送给wc命令。

$ (date; cat file) |wc

date和cat命令的信息都通过管道送给wc命令。

正则表达式

- 正则表达式（regular expression）是一种字符模式，用于在查找过程中匹配指定的字符。
- 正则表达式元字符

元字符	功能	示例	匹配对象
^	行首定位符	‘^the’	匹配所有以the开头的行
$	行尾定位符	‘the$’	匹配所有以the结尾头的行
.	匹配单个字符	‘l..e’	匹配l开头，中间两个任意字符，e结尾
*	匹配0个或多个位于*号前的字符	‘ *the’	匹配跟在零个或多个空格后的the
[]	匹配一组字符中任一个	[tT]he	匹配包含the和The的行
[x-y]	匹配范围内的一字符	[A-Z]he	匹配后面紧跟he的一个A到Z之间的字符
[^]	匹配不再组内的字符	[^A-Z]	匹配不再A到Z之间的任意一字符
\	转义字符	‘love\.’	匹配love后跟一个句号
\{n,m\}	连续n到m个字符	o\{2,5\}	连续2到5个o

\{n\}

连续n个前一字符

\{n,\}

连续n个以上前一字符

\(..\)

grep

- grep家族由命令grep，egrep和fgrep组成
- grep命令在文件中全局查找指定的正则表达式，并且打印所有包含该表达式的行。
- 命令格式
  - grep [options] *基本正则表达式* [filename]

    这里*基本正则表达式*可为字符串。

- 主要参数

  -c 只输出匹配行的计数。
  -i 不区分大小写（只适用于单字符）。
  -h 查询多文件时不显示文件名。
  -l 查询多文件时只输出包含匹配字符的文件名。
  -n 显示匹配行及行号。
  -s 不显示不存在或无匹配文本的错误信息。
  -v 显示不包含匹配文本的所有行。。

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep NW datafile
northwest           NW      Charles Main        3.0   .98   3      34
```

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep '^n' datafile
northwest           NW      Charles Main        3.0   .98   3    34
northeast           NE      AM Main Jr.         5.1   .94   3    13
north               NO      Margot Weber        4.5   .89   5     9
```

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep '4$' datafile
northwest           NW      Charles Main          3.0   .98    3     34
```

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep '5\..' datafile
western          WE      Sharon Gray      5.3   .97   5    23
southern         SO      Suan Chin        5.1   .95   4    15
northeast        NE      AM Main Jr.      5.1   .94   3    13
central          CT      Ann Stephens     5.7   .94   5    13
```

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep '^[we]' datafile
western         WE      Sharon Gray     5.3   .97   5   23
eastern         EA      TB Savage       4.4   .84   5   20
```

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep 'ss* ' datafile
northwest          NW      Charles Main        3.0   .98    3     34
southwest          SW      Lewis Dalsass       2.7   .8     2     18
```

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep '[a-z]\{9\}' datafile
northwest           NW      Charles Main        3.0   .98    3     34
southwest           SW      Lewis Dalsass       2.7   .8     2     18
southeast           SE      Patricia Hemenway   4.0   .7     4     17
northeast           NE      AM Main Jr.         5.1   .94    3     13
```

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep '\(3\)\.[0-9].*\1    *\1' datafile
northwest           NW      Charles Main          3.0   .98    3     34
```

% cat datafile

northwest	NW	Charles Main	3.0	.98	3	34
western	WE	Sharon Gray	5.3	.97	5	23
southwest	SW	Lewis Dalsass	2.7	.8	2	18
southern	SO	Suan Chin	5.1	.95	4	15
southeast	SE	Patricia Hemenway	4.0	.7	4	17
eastern	EA	TB Savage	4.4	.84	5	20
northeast	NE	AM Main Jr.	5.1	.94	3	13
north	NO	Margot Weber	4.5	.89	5	9
central	CT	Ann Stephens	5.7	.94	5	13

```
% grep -n '^south' datafile
3:southwest          SW      Lewis Dalsass          2.7   .8    2   18
4:southern           SO      Suan Chin              5.1   .95   4   15
5:southeast          SE      Patricia Hemenway      4.0   .7    4   17
```

find命令

- find命令以遍历当前目录甚至于整个文件系统来查找某些文件或目录。
- find命令的一般形式为：
  - **find pathname -options [-print -exec -ok]**
    - pathname find命令所查找的目录路径。
    - -print find命令将匹配的文件输出到标准输出。
    - -exec find命令对匹配的文件执行该参数所给出的shell命令。
    - -ok 和-exec的作用相同，只不过以一种更为安全的模式来执行该参数所给出的shell命令，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

- find命令选项
  - -name 按照文件名查找文件。
  - -user　按照文件属主来查找文件。
  - -type 查找某一类型的文件
  - -group 按照文件所属的组来查找文件。
  - ……

```
find ~ -name "*.txt" –print

find . -name "[A-Z]*" –print

find ~ -user dave –print

find /etc -type d -print
```

```
$ cat /etc/passwd | grep lrj

$ dmesg | grep eth0

$ gzip -dc xyz.tar.gz | tar -xvf
```

```
$ man bash | col -b > bash.txt

$ls –F|grep /$

$ ls -l * | grep "^d"

$ ls -l * | grep "^-" | wc -l

$ ls -l * | grep "^d" | wc -l
```

- **(cd /source/directory && tar cf - .) | (cd /dest/directory && tar xvfp -)**

$ du -S | sort -n

$ sort abc|uniq

命令行快捷方式

- 命令补全和文件名补全
  - 使用<TAB>键
- 命令别名
  - alias命令和unalias命令

- alias [alias_name='original_command']
- unalias alias_name

```
$ alias
$ alias type='cat'
$ unalias type
```

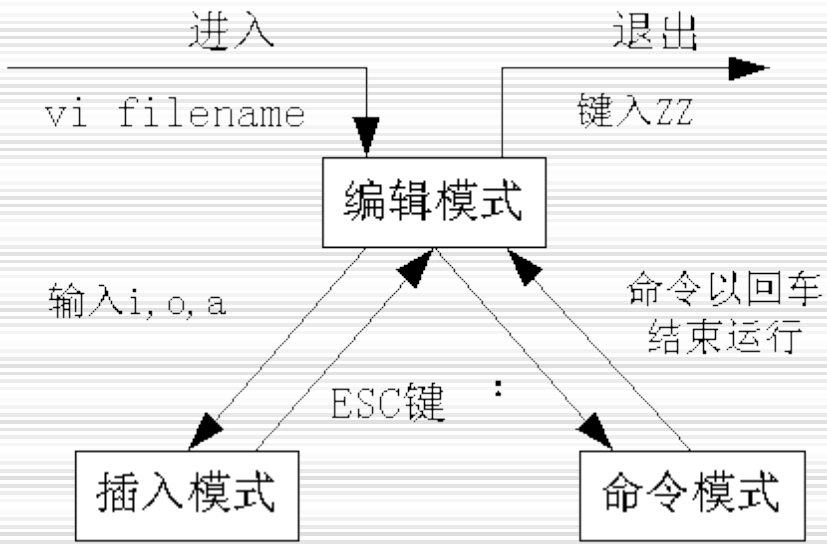
- 命令历史
  - 用上下方向键、PgUp和PgDn键来查看历史命令
  - 可以使用键盘上的编辑功能键对显示在命令行上的命令进行编辑
  - 使用history命令查看命令历史

- 使用如下方法引用命令历史
  - **$！<命令事件号>**
  - **$！<已经使用过的命令前面的部分>**

vi及其运行模式

- vi 简介
  - vi是“visual interface”的简称。
  - vi可以执行输出、删除、查找、替换、块操作等众多文本操作 。
  - vi不是一个排版程序，它可以对字体、格式、段落等其他属性进行编排，它只是一个文本编辑程序。
  - vi是全屏幕文本编辑器，它没有菜单，只有命令。

vi 的运行模式



vi 的基本操作

- 从编辑模式进入插入模式
  - i
  - a
  - o
- 从插入模式返回编辑模式
  - <ESC>
- 编辑模式下退出vi
  - ZZ
  - ZQ
- 命令模式下退出vi
  - :wq
  - :q
  - :q!

命令替换

- 功能：命令参数可以由另一个命令执行的结果来替代。
- 使用方法：

  $ cmd1 `cmd2 arguments`

  或

  $ cmd1 $(cmd2 arguments)
- 使用举例：

  $ echo The present time is `date`

  # mkbootdisk $(uname -r)

  *单引号 算引号 反引号

Shell脚本

- Shell脚本简介
  - Shell是一个功能强大的脚本编程语言。
  - 用Shell编写的批处理文件称为Shell脚本。
  - Shell脚本可以将若干条命令浓缩成一条命令来使用。
  - Shell脚本在系统管理和维护方面大有用处。

Shell脚本的组成

- Shell脚本的成分
  - 注释部分：注释部分以#开头的行。
  - 命令：在Shell脚本中可以出现任何在交互方式下可以使用的命令。
  - 变量：在Shell脚本中既可以使用用户自定义的变量，也可以使用系统环境变量。
  - 流程控制：流程控制语句对命令的执行流程进行控制 （分支、循环、子Shell调用）。

Shell变量

- 变量赋值（定义变量）

  varName = Value

  export varName = Value

  在定义变量时，若string中包含空格、制表符和换行符， 则string必须用‘string’或“string”的形式，即用单（双）引号将其括起来。

- 引用变量
  - $varName

特殊变量

变量名	含义
$0	shell或shell脚本的名字
$*	以一对双引号给出参数列表 （所有参数看作一个单词）
$@	将各个参数分别加双引号返回 （所有参数当作一个字符串中的几个单词）
$#	参数的个数
$_	代表上一个命令的最后一个参数
$$	代表所在命令的PID
$!	代表最后执行的后台命令的PID
$?	代表上一个命令执行后的退出状态

- Shell变量的作用域
  - Shell变量分为局部变量和全局变量。
  - 局部变量的作用范围仅仅限制在其命令行所在的Shell或Shell脚本文件中
  - 全局变量的作用范围则包括本Shell进程及其所有子进程
  - 可以使用export内置命令将局部变量设置为全局变量

Shell 脚本的建立与执行

- Shell脚本的建立
  - 使用文本编辑器编辑脚本文件
- Shell脚本的执行
  - 方法1

$ bash ./script-file

  - 方法2

$ chmod +x script-file

$ ./script-file

  - 方法3

$. script-file

流程控制语句

```
#!/bin/sh
# make a directory
mkdir /home/dave/mydocs
# copy all doc files
cp *.docs /home/dave/docs
# delete all doc files
rm *.docs
```

简单的if语句

最普通的i f语句是：

```
if 条件
then 命令
fi
```

```
if 条件；then
命令
fi
```

条件测试

- 测试文件状态

  test一般有两种格式，即：

  test condition

  或

  [condition]

  使用方括号时，要注意在条件两边加上空格。

-d	目录	-s	文件长度大于0、非空
-f	正规文件	-w	可写
-L	符号连接	-u	文件有suid位设置
-r	可读	-x	可执行

```
$ ls -l scores.txt
-rw-r--r--   1 dave   admin  0 May 15 11:29 scores.txt
$ [ -w scores.txt ]
$ echo $?
0


$ test -w scores.txt
$ echo $?
0
```

测试时使用逻辑操作符

-a 逻辑与，操作符两边均为真，结果为真，否则为假。

-o 逻辑或，操作符两边一边为真，结果为真，否则为假。

! 逻辑否，条件为假，结果为真。

```
$ [ -w results.txt -a -w scores.txt ]
$ echo $?
0
```

```
$ [ -x results.txt -o -x scores.txt ]
$ echo $?
0
```

字符串测试

- 字符串测试有5种格式。
  string_operator可为：
  = 两个字符串相等。
  ！= 两个字符串不等。
  -z 空串。
  -n 非空串。

```
test "string"
test string_operator "string"
test "string" string_operator "string"
[ string_operator string ]
[ string string_operator string ]
```

```
$ [ -z $EDITOR ]
$ echo $?
1
```

```
$ TAPE="/dev/rmt0"
$ TAPE2="/dev/rmt1"
$ [ "$TAPE" = "$TAPE2" ]
$ echo $?
1
```

测试数值

- 测试数值可以使用许多操作符，一般格式如下：

  " number" numeric_operator " number "

  或者

  ["number" numeric_operator "number"]

numeric_operator可为：

-eq 数值相等。

-ne 数值不相等。

-gt 第一个数大于第二个数。

-lt 第一个数小于第二个数。

-le 第一个数小于等于第二个数。

-ge 第一个数大于等于第二个数。

```
$ NUMBER=130
$ [ "$NUMBER" -eq "130" ]
$ echo $?
0
```

```
$ [ "$NUMBER" -gt "100" ]
$ echo $?
0
```

```
#!/bin/sh
# if test2
echo -n "Enter your name :"
read NAME
# did the user just hit return ????
if [ "$NAME" = "" ] ; then
  echo "You did not enter any information"
fi
$ iftest2
Enter your name :
You did not enter any information
```

```
#!/bin/sh
# ifcp
if cp myfile myfile.bak; then
  echo "good copy"
else
  echo "`basename $0`: error could not copy the files" >&2
fi
```

```
#!/bin/sh
# ifparam
if [ $# -lt 3 ]; then
# less than 3 parameters called, echo a usage message and exit
  echo "Usage: `basename $0`arg1 arg2 arg3" >&2
  exit 1
fi
# good, received 3 params, let's echo them
echo "arg1: $1"
echo "arg2: $2"
echo "arg3: $3"
```

完整的if

if 条件
t h e n
命令1
e l s e
命令2
f i

```
#!/bin/sh
# ifeditor
if [ -z $EDITOR ]; then
# the variable has not been set
  echo "Your EDITOR environment is not set"
else
# let's see what it is
  echo "Using $EDITOR as the default editor"
fi
```

case语句

```
case 值 i n
模式1 )
命令1
. . .
; ;
模式2）
命令2
. . .
; ;
e s a c
```

```
case $ANS in
1) echo "you select 1"
   ;;
2) echo "you select 2"
   ;;
3) echo "you select 3"
   ;;
4) echo "you select 4"
   ;;
5) echo "you select 5"
   ;;
*) echo "`basename $0`: This is not between 1 and 5" >&2
   exit 1
   ;;
esac
```

for循环

for 变量名 i n 列表
d o
    命令1
    命令2
d o n e

```
#!/bin/sh
# for_i
for loop in 1 2 3 4 5
do
  echo $loop
done
```

对for循环使用参数

for params in"$@"

或

for params in"$*"

```
#!/bin/sh
# forparam3
for params in "$@"
do
  echo "You supplied $params as a command line option"
done
  echo $params
done
```

```
for param in "$@"
do
    echo "$param"
done
```

```
#!/bin/sh
# forping
HOSTS="itserv dnssevr acctsmain ladpd ladware"
for loop in $HOSTS
do
  ping -c 2 $loop
done
```

until循环

until 条件

命令**1**

. . .

d o n e

while循环

while 命令
d o
命令1
命令2
. . .
d o n e

使用break和continue控制循环

- 有时需要基于某些准则退出循环或跳过循环步。ｓｈｅｌｌ提供两个命令实现此功能。
  - break。
  - continue。