# Shell脚本编程

## 1、shell脚本运行

运行Shell脚本有两种方法：

1．作为可执行程序

将代码保存为 .sh 文件，

|  |
| --- |
| #!/bin/bash  echo "Hello World!" |

cd到相应目录：

|  |
| --- |
| bogon:Desktop apple$ chmod +x ./t.sh #使脚本具有执行权限  bogon:Desktop apple$ ./t.sh #执行脚本  Hello World! |

注意：

执行：**./t.sh** 告诉系统在当前目录下寻找运行程序t.sh 。

执行：**t.sh** 系统会去PATH里寻找有没有t.sh ，而只有/bin,/sbin,/usr/bin等在PATH里，你的当前目录通常不在PATH里。

|  |
| --- |
| bogon:Desktop apple$ ./t.sh  -bash: ./t.sh: Permission denied |

2．作为解释器参数

直接运行解释器，其参数就是shell脚本的文件名：

|  |
| --- |
| bogon:Desktop apple$ /bin/sh t.sh  Hello World! |

这种方式运行脚本，不需要在第一行指定解释器信息，写了也么有用。

## 2、Shell变量

定义变量：

|  |
| --- |
| your\_name=”mengyueping.com” |

注意：变量名和等号之间不能有空格。变量名命名规则：

* 首字符必须为字母（a-z， A-Z）。
* 中间不能有空格，可以使用下划线（\_） 。
* 不能使用标点符号。
* 不能使用bash里的关键字（可用help命令查看保留关键字）

可以用语句变量赋值：

|  |
| --- |
|  |

变量类型：

* 局部变量：在脚本或命令中定义，仅在当前shell实例中有效，其他shell启动的程序不能访问局部变量。
* 环境变量：所有的程序，包括shell启动的程序，都能访问环境变量，有些程序需要环境变量来保证其正常运行。必要的时候shell脚本也可以定义环境变量。
* shell变量：有shell程序设置的特殊变量。Shell变量中有一部分是环境变量，有一部分是局部变量，这些变量保证了shell的正常运行。

## 3、Shell字符串

### Shell传递参数

处理参数：

|  |  |
| --- | --- |
| **参数处理** | **说明** |
| **$#** | 传递到脚本的参数个数 |
| **$\*** | 以一个单字符串显示所有向脚本传递的参数。  如："$\*"用 " 括起来的情况、以"$1 $2 … $n"的形式输出所有参数。 |
| **$$** | 脚本运行的当前进程ID号 |
| **$!** | 后台运行的最后一个进程的ID号 |
| **$@** | 与 $\* 相同，但是使用时加引号，并在引号中返回每个参数。  如 "$@"用 " 括起来的情况、以"$1" "$2" … "$n" 的形式输出所有参数。 |
| **$-** | 显示Shell使用的当前选项，与set命令功能相同。 |
| **$?** | 显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。 |

### 算术运算符

|  |  |  |
| --- | --- | --- |
| **运算符** | **说明** | **举例** |
| **+** | 加法 | `expr $10 + $20` 结果为30 |
| **-** | 减法 | `expr $10 - $20` 结果为-10 |
| **\*** | 乘法 | `expr $10 \\* $20` 结果为200 |
| **/** | 除法 | `expr $20 / $10` 结果为2 |
| **%** | 取余 | `expr $20 % $10` 结果为0 |
| **=** | 赋值 | a=$b 把变量b的值赋给a |
| **==** | 相等。  用于比较两个数字，相同则返回true . | [$a == $b] 返回 false |
| **!=** | 不相等。用于比较两个数字，不相同则返回true。 | [$a != &b] 返回true |
|  |  |  |

注意：条件表达式要放在方括号之间，并且要有空格。

### 关系运算符

a=10 b=20

|  |  |  |
| --- | --- | --- |
| **运算符** | **说明** | **举例** |
| **-eq** | 检测两个数是否相等，相等返回true | [ $a –eq $b ] 返回 false |
| **-ne** | 检测两个数是否相等，不相等返回true | [ $a –ne $b ] 返回 true |
| **-gt** | 检测左边的数是否大于右边的，如果是，则返回true 。 | [ $a –gt $b ] 返回false |
| **-lt** | 检测左边的数是否小于右边的，如果是，则返回true。 | [ $a –lt $b ] 返回true |
| **-ge** | 检测左边的数是否大于等于右边的，如果是，则返回true | [ $a –ge $b ] 返回false |
| **-le** | 检测左边的数是否小于等于右边的，如果是，则返回true | [ $a –le $b ] 返回true |
|  |  |  |

### 布尔运算符

|  |  |  |
| --- | --- | --- |
| **运算符** | **说明** | **举例** |
| **!** | 非运算符，表达式为true 则返回false ,否则返回true | [ !false ]返回true |
| **-o** | 或运算，有一个表达式为true则返回true | [ $a –lt 20 –o $b –gt 100 ] 返回true |
| **-a** | 与运算，两个表达式都为true才返回true | [ $a –lt 20 –a $b –gt 100 ] 返回false |

### 逻辑运算符

|  |  |  |
| --- | --- | --- |
| **运算符** | **说明** | **举例** |
| **&&** | 逻辑 AND | [[ $a –lt 100 && $b –gt 100 ]] 返回false |
| **||** | 逻辑OR | [[ $a –lt 100 || $b –gt 100 ]] 返回true |
|  |  |  |

### 字符串运算符

|  |  |  |
| --- | --- | --- |
| **运算符** | **说明** | **举例** |
| **=** | 检测两个字符串是否相等，相等返回true | [ $a = $b ] 返回false |
| **!=** | 检测两个字符串是否相等，不相等返回true | [ $a != $b ] 返回true |
| **-z** | 检测字符串长度是否为0，为0返回true | [ -z $a ]返回false |
| **-n** | 检测字符串长度是否为0，不为0返回true | [ -n $a ] 返回true |
| **str** | 检测字符串是否为空，不为空返回true | [ $a ] 返回true |

### 文件测试运算符

文件测试运算符用于：检测Unix文件的各种属性。

|  |  |  |
| --- | --- | --- |
| **操作符** | **说明** | **举例** |
| **-b file** | 检测文件是否是**块设备文件**，如果是，则返回true | [ -b $file ] 返回false |
| **-c file** | 检测文件是否是**字符设备文件**，如果是，则返回true | [ -c $file ] 返回false |
| **-d file** | 检测文件是否是**目录**，如果是，则返回true | [ -d $file ] 返回false |
| **-f file** | 检测文件是否是**普通文件**（既不是目录，也不是设备文件），如果是，则返回true | [ -f $file ] 返回true |
| **-g file** | 检测文件是否**设置了SGID 位**，如果是，则返回true | [ -g $file ] 返回false |
| **-k file** | 检测文件是否设置了**粘着位（Sticky Bit）**,如果是，则返回true | [ -k $file ] 返回false |
| **-p file** | 检测文件是否是**有名管道**，如果是，则返回true | [ -p $file ] 返回false |
| **-u file** | 检测文件是否**设置了SUID位**，如果是，则返回true | [ -u $file ] 返回false |
| **-r file** | 检测文件是否**可读**，如果是，则返回true | [ -r $file ] 返回true |
| **-w file** | 检测文件是否**可写**，如果是，则返回true | [ -w $file ] 返回true |
| **-x file** | 检测文件是否**可执行**，如果是，则返回true | [ -x $file ] 返回true |
| **-s file** | 检测文件是否**为空**（文件大小是否大于0），不为空返回true | [ -s $file ] 返回true |
| **-e file** | 检测文件（包括目录）是否**存在**，如果是，则返回true. | [ -e $file ] 返回true |

### printf的转义序列

|  |  |
| --- | --- |
| **序列** | **说明** |
| **\a** | 警告字符，通常为ASCII的BEL字符 |
| **\b** | 后退 |
| **\c** | 抑制（不显示）输出结果中任何结尾的换行字符（只在%b格式指示符控制下的参数字符串中有效），而且，任何留在参数里的字符、任何接下来的参数以及任何留在格式字符串中的字符，都被忽略 |
| **\f** | 换页（formfeed） |
| **\n** | 换行 |
| **\r** | 回车（Carriage return） |
| **\t** | 水平制表符 |
| **\v** | 垂直制表符 |
| **\\** | 一个字面上的反斜杠字符 |
| **\ddd** | 表示1到3位数八进制值的字符。仅在格式字符串中有效。 |
| **\0ddd** | 表示1到3位的八进制值字符 |
|  |  |

### Shell test命令

数值测试：

|  |  |
| --- | --- |
| **参数** | **说明** |
| **-eq** | 等于 则为真 |
| **-ne** | 不等于 则为真 |
| **-gt** | 大于 则为真 |
| **-lt** | 小于 则为真 |
| **-ge** | 大于等于 则为真 |
| **-le** | 小于等于 则为真 |
|  |  |

字符串测试

|  |  |
| --- | --- |
| **运算符** | **说明** |
| **=** | 等于 则为真 |
| **!=** | 不等于 则为真 |
| **-z 字符串** | 字符串的长度为零 则为真 |
| **-n 字符串** | 字符串的长度不为零 则为真 |
|  |  |

文件测试：

|  |  |
| --- | --- |
| **操作符** | **说明** |
| **-e 文件名** | 如果文件存在 则为真 |
| **-r 文件名** | 如果文件存在且可读 则为真 |
| **-w 文件名** | 如果文件存在且可写 则为真 |
| **-x 文件名** | 如果文件存在且可执行 则为真 |
| **-s 文件名** | 如果文件存在且至少有一个字符 则为真 |
| **-d 文件名** | 如果文件存在且为目录 则为真 |
| **-f 文件名** | 如果文件存在且为普通文件 则为真 |
| **-c 文件名** | 如果文件存在且为字符型特殊文件 则为真 |
| **-b 文件名** | 如果文件存在且为块特殊文件 则为真 |

### Shell 输入/输出重定向

|  |  |
| --- | --- |
| **命令** | **说明** |
| **command>file** | 将输出重定向到file |
| **command<file** | 将输入重定向file |
| **command>>file** | 将输出以追加的方式重定向到file |
| **n>file** | 将文件描述符为n的文件重定向到file |
| **n>>file** | 将文件描述符为n的文件以追加的方式重定向到file |
| **n>&m** | 将输出文件m和n合并 |
| **n<&m** | 将输入文件m和n合并 |
| **<<tag** | 将开始标记tag和结束标记tag之间的内容作为输入 |
|  |  |

注意： 文件描述符0通常是标准输入（STDIN）, 1是标准输出（STDOUT）, 2是标准错误输出（STDERR）.

## 4、Shell中的关联数组

关联数组：

普通数组：只能使用整数作为数组索引。

关联数组：可以使用字符串作为数组索引。

申明关联数组变量：

**#declare –A ass\_array1**

数组名[索引]=变量值

**#ass\_array1[index1]=pear**

## 5、Shell典型应用

场景脚本结构：

Input

控制脚本

monitor\_man.sh

日志分析

提取系统信息

应用运行分析

Shell脚本应用场景：

提取Linux 操作系统信息

获取操作系统运行状态

分析应用状态

应用日志分析

shell能做什么？

Shell的开发环境及使用特性？

明白shell对于资源监控功能作用：总结出linux系统的资源使用，以及每个性能参数含义。

Shell将操作系统、应用、日志连接起来。

主控脚本实现

系统信息及系统状态分析

Nginx和mysql应用状态分析

应用日志分析

shell对于系统资源提取及运行状态分析

shell分析程序或者应用状态

shell日志提取分析

# Linux

Linux严格区分大小写；

Linux中所有内容以文件形式保存，包括硬件。一切内容皆文件。

Linux**不依靠扩展名区分文件类型**。

约定俗成的扩展名：(这些是给管理员使用的)

压缩包： **.gz** 、**.bz2** 、 **.tar.bz2** 、 **.tgz** 等

二进制软件包： **.rpm**

网页文件： **.html .php**

脚本文件： **.sh**

配置文件： **.conf**

字符界面优势：

字符界面占用的系统资源更少。

字符界面减少了出错、被攻击的可能性。

磁盘分区：

使用分区编辑器（partition editor）在磁盘上划分几个逻辑部分。 碟片一旦划分成数个分区（Partition），不同类的目录与文件可以存储进不同的分区。

分区类型：（分区号1234只能给主分区和扩展分区，不能给逻辑分区，逻辑分区是从5开始的。）

主分区：最多只能有4个。

扩展分区：

最多只能有1个。

主分区加扩展分区最多有4个。

不能写入数据，只能包含逻辑分区。

逻辑分区：

格式化：

（高级格式化）又称逻辑格式化，它是指根据用户选定的文件系统（如FAT16、FAT32、NTFS、EXT2、EXT3、EXT4等），在磁盘的特定区域写入特定数据，在分区中划出一片用于存放文件分配表、目录表等用于文件管理的磁盘空间。

硬件设备文件名：(/dev ：专门用来存放硬件文件)

|  |  |
| --- | --- |
| 硬件 | 设备文件名 |
| IDE硬盘 | /dev/hd[a-d] |
| SCSI/SATA/USB硬盘 | /dev/sd[a-p] |
| 光驱 | /dev/cdrom或/dev/hdc |
| 软盘 | /dev/fd[0-1] |
| 打印机（25针） | /dev/lp[0-2] |
| 打印机（USB） | /dev/usb/lp[0-15] |
| 鼠标 | /dev/mouse |

分区设备文件名

设备文件名：

/dev/hda1 (IDE硬件接口)

/dev/sda1 (SCSI硬盘接口、SATA硬盘接口)

挂载

必须分区：

/ （根分区）

swap分区（交换分区，内存2倍，不超过2GB。内存2G以上，swap分区和内存一样大）

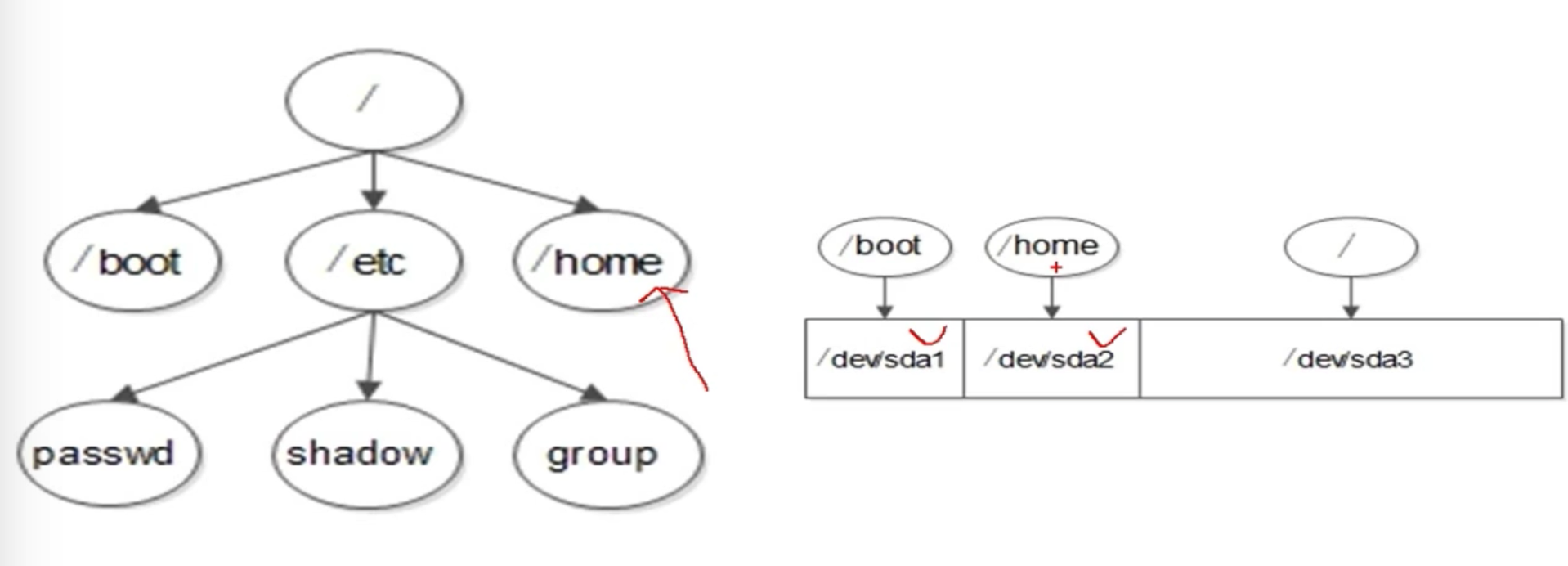
推荐分区：

/boot （启动分区，200MB）

文件系统结构：

从磁盘来看，属于不同分区。

从系统来看，/home、/etc、/boot 都属于/根目录下。



分区：把硬盘分为小的逻辑分区

格式化：写入文件系统

分区设备文件名：给每个分区定义设备文件名

挂载：给每个分区分配挂载点

centos安装日志：

/root/install.log : 存储了安装在系统中的软件包及其版本信息。

/root/install.log.syslog : 存储了安装过程中留下的事件记录

/root/anaconda-ks.cfg : 以Kickstart配置文件的格式记录安装过程中设置的选项信息。

## Linux常用命令

### 命令基本格式

**命令 [选项] [参数]**

注意：个别命令使用不遵循此格式

当有多个选项时，可以写在一起

简化选项与完整选项：

-a等于--all

**[root@localhost ~]#**

root 代表当前登录用户

localhost 主机名

~ 当前所在目录（家目录）

# 超级用户的提示符

普通用户的提示符是$

Linux中，管理员账号是root

查询目录中内容： **ls**

ls [选项] [文件或目录]

选项：

-a 显示所有文件，包括隐藏文件（liunx中以.开头的文件是隐藏文件）

-l 显示详细信息

-d 查看目录属性

-h 人性化显示文件大小

-i 显示inode（每一个文件都有一个id号）

命令： ll 相当于 ls -l

**-rw-r--r--**

**-** ：文件类型（第一位是 **-** 代表文件; 第一位是 **d** 代表目录; 第一位是 **|** 代表软链接文件）

**rw-** ：u所有者

**r--** ：g所属组

**r-- ：**o其他人

r读 w写 x执行

|  |
| --- |
| 注意：liunx中有七种文件类型，文件、目录、软连接、块设备文件、字符设备文件、套接字文件、管道文件。 |

### 目录处理命令

建立目录： **mkdir**

**mkdir -p [目录名]**

**-p递归创建**

**英文愿意：make directories**

切换所在目录： cd

cd [目录]

英文愿意：change directory

简化操作：

**cd ~** 进入当前用户的家目录

**cd**

**cd –** 进入上次目录

**cd ..** 进入上一级目录

**cd .** 进入当前目录

清屏快捷键： Ctrl + L

相对路径：参照当前所在目录，进行查找。

如： [root@imooc ~]# cd ../usr/local/src/

绝对路径：从根目录开始指定，一级一级递归。在任何目录下，都能进入指定位置。

如： [root@imooc ~]# cd /etc/

查询所在目录位置： pwd

pwd

英文愿意：print working directory

删除空目录： **rmdir**

**rmdir [目****录名]**

英文愿意：remove empty directories

注意：只能刪除空目录。

删除文件或目录：rm

rm -rf [文件或目录]

英文愿意：remove

选项：

-r 删除目录

-f 强制

复制命令：cp

cp [选项] [原文件或目录] [目标目录]

英文：copy

选项：

-r 复制目录

-p 连带文件属性复制

-d 若原文件是链接文件，则复制链接属性

-a 相当与 -pdr

剪切或改名命令： mv

mv [原文件或目录] [目标目录]

英文：move

原目录和目标目录在同一个目录中就是改名命令;

原目录和目标目录不在同一个目录中就是剪切命令。

常用目录的作用：

**/**  根目录

**/bin**  命令保存目录（普通用户就可以读取的命令）

**/boot** 启动目录，启动相关文件

**/dev**  设备文件保存目录

**/etc**  配置文件保存目录

**/home** 普通用户的家目录

**/lib**  系统库保存目录

**/mnt** 系统挂载目录（常用）

**/media** 挂载目录

根目录下的bin和sbin，usr目录下的bin和sbin，这四个目录都是用来保存系统命令的。

bin目录下是任何用户都能执行;

sbin目录下只有root超级用户才能够执行;

/root 超级用户的家目录

/tmp 临时目录

/proc 直接写入内存的

/sys

proc和sys目录不能直接操作，这两个目录保存的是内存的过载点。

/usr 系统软件资源目录

/var 系统相关文档内容

可以在家目录root或home，以及tmp目录下随便放内容。

链接命令：ln

ln -s [原文件] [目标文件]

英文：link

功能描述：生成链接文件

选项：-s 创建软链接

硬链接：

1、拥有相同的i节点和存储block块，可以看做是同一个文件。

2、可通过i节点识别。

3、不能跨分区。

4、不能针对目录使用。

5、对一个文件添加一个硬链接，它的引用计数会加1。

6、删除原文件，硬链接文件仍然可以使用。

软链接：

1、类似windows快捷方式

2、软链接拥有自己打i节点和Block块，但是数据块中只保存原文件打文件名和I节点号，并没有实际打文件数据。

3、lrwxrwxrwx l软链接

软链接文件权限都为：rwxrwxrwx

4、修改任意文件，另一个都改变。

5、删除原文件，软链接不能使用。

注意：软链接创建一定要写绝对路径。

### 文件处理命令

### 文件搜索命令

文件搜索命令： **locate**

locate 文件名

在后台数据库中按文件名搜索，搜索速度更快。

/var/lib/mlocate 是存放 #locate 命令所搜索的后台数据库 mlocate.db

updatedb ：更新数据库

只能按文件名搜索，遵循配置文件。

/etc/updatedb.com 配置文件

PRUNE\_BIND\_MOUNTS=”yes” ： #开启搜索限制

PRUNEFS = ： #搜索时，不搜索的文件系统

PRUNENAMES= ：#搜索时，不搜索打文件类型

PRUNEPATHS= ： #搜索时，不搜索打路径

搜索命令的命令 ：**whereis**与**which**

**whereis** 命令

#搜索命令所在**路径**及**帮助文档**所在位置。

选项：

-b ：只查找可执行文件

-m ：只查找帮助文件

**which** 命令

which 文件名

#搜索命令所在**路径**及**别名**

PATH环境变量：

PATH环境变量：定义的是系统搜索命令打路径

#**echo $PATH**

/home/ubuntu/bin:/home/ubuntu/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

文件搜索命令：**find**

find [搜索范围] [搜索条件] ： #搜索文件

find / -name install.log

#避免大范围搜索，会非常耗费系统资源

#完全匹配，不是模糊查询。

#find 是在系统当中搜索符合条件打文件名。如果需要匹配，使用通配符匹配，通配符是完全匹配。

**find /root -iname install.log**

#不区分大小写

**find /root -user root**

#按照所有者搜索

**find /root -nouser**

#查找没有所有者的文件

**find /var/log/ -mtime +10**

#查找10天前修改的文件

-10 10天内修改文件

10 10天当天修改的文件

+10 10天前修改的文件

**atime** ：文件访问时间

**ctime** ： 改变文件属性

**mtime** ： 修改文件内容

**find .(目录) -size 25k**

#查找文件大小是25KB的文件

-25k 小于25KB的文件

-25k 等于25KB的文件

+25k 大于25KB的文件

**find . -inum 262422**

#查找i节点是262422的文件

**find /etc -size +20k -a -size -50k**

#查找/etc/目录下，大于20KB并且小于50KB的文件

-a and 逻辑与， 两个条件都满足

-o or 逻辑或， 两个条件满足一个即可

f**ind /etc -size +20k -a -size -50k -exec ls -lh {} \;**

#查找/etc/目录下，大于20KB并且小于50KB的文件，并显示详细信息

#**-exec**/-ok 命令 **{} \;** 对搜索结果执行操作。

Linux中的通配符：

**\*** 匹配任意内容

**?** 匹配任意一个字符

**[]**  匹配任意一个中括号内的字符

grep命令：

grep [选项] 字符串 文件名

#在文件当中匹配符合 条件的字符串

选项：

-i 忽略大小写

-v 排除指定字符串

find命令与 grep命令的区别

find命令：在系统当中搜索符合条件打文件名，如果需要匹配，使用通配符匹配，通配符是**完全匹配**。

grep命令：在文件当中搜索符合条件打字符串，如果需要匹配，使用正则表达式匹配，正则表达式是**包含匹配**。

### 帮助命令

帮助命令 man：

**man 命令**

#获取指定命令的帮助

**man ls**

#查看ls的帮助

man的级别：

1 ：查看命令的帮助

2 ：查看可被内核调用的函数的帮助

3 ：查看函数和函数库的帮助

4 ：查看特殊文件的帮助（主要是/dev目录下的文件）

5 ：查看配置文件的帮助

6 ：查看游戏的帮助

7 ： 查看其它杂项的帮助

8 ： 查看系统管理员可用命令的帮助

9 ： 查看和内核相关文件的帮助

查看命令拥有哪个级别的帮助：

**man -f 命令**

相当于：

**whatis 命令**

举例：

man -5 passwd

man -4 null

man -8 ifconfig

查看和命令相关的所有帮助：

**man -k 命令**

相当于：

**apropos命令s**

例如：apropos passwd

其他帮助命令：

选项帮助：

命令 --help

#获取命令选项的帮助

例如： ls –help

shell 内部命令帮助：

help shell 内部命令

#获取shell内部命令的帮助

例如：

whereis cd #确定是否是shell内部命令

help cd #获取内部命令帮助

详细命令帮助info

info 命令：

- 回车 ： 进入子帮助页面（带有\*号标记）

- u ： 进入上层页面

- n ：进入下一个帮助小节

-p ：进入上一个帮助小节

-q ： 退出

### 压缩与解压缩命令

常用压缩格式：.zip .gz .bz2

常用压缩格式： .tar.gz .tar.bz2

.zip 格式压缩：

命令：zip 压缩文件名 原文件

#压缩文件

命令： zip -r 压缩文件名 原目录

#压缩目录

命令：unzip 压缩文件

#解压缩.zip文件

.gz格式压缩：

命令：gzip 原文件

#压缩为.gz格式打压缩文件，原文件会消失

命令：gzip -c 原文件 > 压缩文件

#压缩为.gz格式，原文件保留

例如：gzip -c testdir > testdir.gz

命令：gzip -r 目录

#压缩目录下所有的子文件，但是不能压缩目录

命令：gzip -d 压缩文件

#解压缩文件

命令：gunzip 压缩文件

#解压缩文件

.bz2格式压缩：

命令：bzip2 原文件

#压缩为.bz2格式，不保留原文件

命令：bzip2 -k 原文件

#压缩之后保留原文件

注意：bzip2命令不能压缩目录。

命令：bzip2 -d 压缩文件

#解压缩，-k保留压缩文件

命令：bunzip2 压缩文件

#解压缩， -k保留压缩文件

**.tar.gz 格式压缩**：

命令：tar -cvf 打包文件名 原文件

选项：

-c ：打包

-v ：显示过程

-f ：制定打包后的文件名

例如： tar -cvf test.tar test

解打包命令：

命令：tar -xvf 打包文件名

选项：

-x ： 解打包

例如：

tar -xvf test.tar

其实.tar.gz格式是先打包为.tar格式，再压缩为.gz格式

命令： tar -zcvf 压缩包名.tar.gz 原文件

选项：

-z ： 压缩为.tar.gz格式

命令： tar -zxvf 压缩包名.tar.gz

选项：

-x ：解压缩.tar.gz格式

**.tar.bz2 格式压缩**：

其实.tar.bz2格式是先打包为.tar格式，再压缩为.bz2格式

命令： tar -zcvf 压缩包名.tar.bz2 原文件

选项：

-z ： 压缩为.tar.bz2格式

命令： tar -zxvf 压缩包名.tar.bz2

选项：

-x ：解压缩.tar.bz2格式

-t ：只查看不解压

### 关机和重启命令

1》 **shutdown命令**

#shutdown [选项] 时间

选项：

-c ：取消当前一个关机命令

-h ： 关机

-r ：重启

2》 其他的关机命令

# halt

# poweroff

# init 0

3》 其他重启命令

# boot

# init 6

4》 系统运行级别

0 ： 关机

1 ：单用户

2 ：不完全多用户，不含NFS服务

3 ：完全多用户

4 ： 未分配

5 ：图形界面

6 ： 重启

#cat /etc/inittab

#修改系统默认运行级别

id:3:initdefault:

# runlevel

#查询系统运行级别

5》 退出登录命令

# logout

### 其他常用命令

挂载命令：

1》查询与自动挂载

**# mount**

#查询系统中已经挂载的设备

**# mount -a**

#依据配置文件 **/etc/fstab** 的内容 （自动挂载相关配置内容）

2》 挂载命令格式

**# mount [-t 文件系统] [-o 特殊选项] 设备文件名 挂载点**

选项：

-t 文件系统： 加入文件系统类型来指定挂载的类型，可以ext3、ext4、iso9660等文件系统

-o 特殊选项 ： 可以指定挂载的额外选项

特殊选项：

|  |  |
| --- | --- |
| 参数 | 说明 |
| atime / noatime | 更新访问时间/不更新访问时间。访问分区文件时，是否更新文件的访问时间，默认为更新 |
| async/sync | 异步/同步，默认为异步 |
| auto/noauto | 自动/手动，mount -a 命令执行时，是否会自动安装/etc/fstab文件内容挂载，默认为自动 |
| defaults | 定义默认值，相当于rw,suid,dev,exec,quto,nouser,async这七个选项 |
| exec/noexec | 执行/不执行，设定是否允许在文件系统中执行可执行文件，默认是exec允许 |
| remount | 重新挂载已经挂载的文件系统，一般用于指定修改特殊权限 |
| rw/ro | 读写/只读，文件系统挂载时，是否具有读写权限，默认时rw |
| suid/nosuid | 具有/不具有SUID权限，设定文件系统是否具有SUID和SGID的权限，默认时具有 |
| user/nouser | 允许/不允许普通用户挂载，设定文件系统是否允许普通用户挂载，默认是不允许，只有root可以挂载分区 |
| usrquota | 写入代表文件系统支持用户磁盘配额，默认不支持 |
| grpquota | 写入代表文件系统支持组磁盘配额，默认不支持 |

3》 挂载光盘

# mkdir /mnt/cdrom/

#建立挂载点

# mount -t iso9660 /dev/cdrom /mnt/cdrom/

#挂载光盘

# mount /dev/sr0 /mnt/cdrom/

4》 卸载命令

# umount 设备文件名或挂载点

# umount /mnt/cdrom

5》 挂载U盘

**# fdisk -l**

#查看U盘设备文件名

**# mount -t vfat /dev/sdb1 /mnt/usb/**

注意：Linux默认是不支持NTFS文件系统的

**用户登录查看和用户交互命令：**

1》命令：**w 用户名**

查看登录用户信息。

命令输出：

USER：登录的用户名

TTY：登录终端

FROM：从哪个IP地址登录

LOGIN@：登录时间；

IDLE：用户闲置时间；

JCPU：指的是和该终端连接的所有进程占用的时间。这个时间里并不包括过去的后台作业时间，但却包括当前正在运行的后台作业所占用的时间；

PCPU：是指当前进程所占用的时间；

WHAT：当前正在运行的命令

2》命令： **who 用户名**

查看登录用户信息。

命令输出：

- 用户名 - 登录终端 -登录时间（登录来源IP地址）

3》 命令 ： last

查询当前登录和过去登录的用户信息。

last命令默认时读取  **/var/log/wtmp** 文件数据

命令输出：

- 用户名 -登录终端 -登录IP -登录时间 -退出时间（在线时间）

4》命令：lastlog

查看所有用户的最后一次登录时间。

lastlog命令默认时读取 /var/log/lastlog 文件内容。

命令输出：

-用户名 -登录终端 -登录IP -最后一次登录时间

## Shell

### 1、shell是什么

shell是一个命令行解释器，它为用户提供了一个向Linux内核发送请求以便运行程序的界面系统程序，用户可以用shell来启动、挂起、停止甚至是编写一些程序。

shell还是一个功能相当强大的编程语言，易编写，易调试，灵活性较强。Shell是解释执行的脚本语言，在Shell中可以直接调用Linux系统命令。

打开文件，查看Linux支持的Shell： /etc/shells

Bash：(Bourne Again Shell)（/bin/bash）

#! ：一个约定的标记，它告诉系统这个脚本需要什么解释器来执行，即使用哪一种Shell，告诉系统其后路径所指定的程序即解释此脚本文件的Shell程序。

扩展名： .sh

注释：

以“#”开头的行是注释，会被解释器忽略。没有多行注释，只有单行注释。

遇到大段的代码需要临时注释起来，过一会儿又取消注释，每一行加个#符号太费力了，可以把这一段要注释的代码用一对花括号括起来，定义成一个函数，没有地方调用这个函数，这块代码就不会执行，达到了和注释一样的效果。

echo向窗口输出命令： **echo [选项] [输出内容]**

选项： -e 支持反斜杠控制的字符转换

|  |  |
| --- | --- |
| 控制字符 | 作用 |
| \a | 输出警告 |
| \b | 退格键，也就是向左删除键 |
| \n | 换行符 |
| \r | 回车键 |
| \t | 制表符，也就是Tab键 |
| \v | 垂直制表符 |
| \0mn | 按照八进制ASCII码表输出字符。其中0为数字零，nnn时三位八进制数 |
| \xhh | 按照十六进制ASCII码表输出字符。其中hh是两位十六进制数 |

# echo -e “hell\bo”

# 删除左侧字符

# echo -e “h\te\tl\nl\to”

#制表符与换行符

# echo -e ”\x68\t\x65\t\x6c\n\x6c\t\x6f\t”

#按照十六进制ASCII码也同样可以输出

# echo -e “\e[1;31m输出颜色\e[0m”

#输出颜色：

#30m=黑色， 31m=红色， 32m=绿色， 33m=黄色

#34m=蓝色， 35m=洋红， 36m=青色， 37m=白色

脚本赋予执行权限，直接运行：

命令： chmod 755 hello.sh

./hello.sh

通过Bash调用执行脚本：

bash hello.sh

### 2、Bash的基本功能

#### 1》命令别名与快捷键

# alias

#查看系统中所有的命令别名

# alias 别名= ‘原命令’

#设定命令别名

# vi ~/.bashrc

#写入环境变量配置文件 （在 家目录下 root 或 home）（别名永久生效）（Ubuntu目录： /etc/bash.bashrc）

# unalias 别名

#删除别名

命令生效顺序：

第一顺位：执行用绝对路径或相对路径执行的命令。

第二顺位：执行别名。

第三顺位：执行Bash的内部命令。

第四顺位：执行按照 $PATH 环境变量定义的目录查找顺序找到的第一个命令。

常用快捷键：

ctrl + c : 强制终止当前命令。

ctrl +l : 清屏。

ctrl + a : 光标移动到命令行首。

Ctrl + e ： 光标移动到命令行尾。

Ctrl + u ：从光标所在位置删除到行首。

Ctrl + z ：把命令放入后台。

ctrl + r ： 在历史命令中搜索。

#### 2》历史命令

# **history [选项] [历史命令保存文件]**

选项：

-c ：清空历史命令

-w ：把缓存中的历史命令写入历史命令保存文件 ~/.bash\_history

历史命令默认会保存1000条，可以在环境变量配置文件 /etc/profile 中进行修改。

历史命令的调用：

使用上、下箭头调用以前的历史命令

使用”!n” 重复执行第n条历史命令

使用”!!”重复执行上一条命令

使用”!字串”重复执行最后一条以该字串开头的命令

#### 3》输出重定向

标准输入输出：

|  |  |  |  |
| --- | --- | --- | --- |
| 设备 | 设备文件名 | 文件描述符 | 类型 |
| 键盘 | /dev/stdin | 0 | 标准输入 |
| 显示器 | /dev/sdtout | 1 | 标准输出 |
| 显示器 | /dev/sdterr | 2 | 标准错误输出 |

输出重定向：

|  |  |  |
| --- | --- | --- |
| 类型 | 符号 | 作用 |
| 标准输出重定向 | 命令 > 文件  （>两边加空格） | 以**覆盖**的方式，把命令的正确输出输出到指定的文件或设备当中 |
| 命令 >> 文件  （>>两边加空格） | 以**追加**的方式，把命令的正确输出输出到指定的文件或设备当中 |
| 标准错误输出重定向 | 错误命令 2>文件  （>两边不加空格） | 以**覆盖**的方式，把命令的错误输出输出到指定的文件或设备当中 |
| 错误命令 2>>文件  （>>两边不加空格） | 以**追加**的方式，把命令的错误输出输出到指定的文件或设备当中 |
|  |  |  |
| 正确输出和错误输出同时保存 | 命令 > 文件 2>&1 | 以**覆盖**的方式，把**正确输出**和**错误输出**都保存到同一个文档中。 |
| 命令 >> 文件 2>&1 | 以**追加**的方式，把**正确输出**和**错误输出**都保存到同一个文件当中。 |
| 命令&>文件 | 以**覆盖**的方式，把**正确输出**和**错误输出**都保存到同一个文件当中。 |
| 命令&>>文件 | 以**追加**的方式，把**正确输出**和**错误输出**都保存到同一个文件当中。 |
| 命令>>文件 1 2>>文件 2 | 把正确的输出追加到文件1中，把错误的输出追加到文件2中。 |
|  | 命令 &>/dev/null | 命令执行结果丢入黑洞 |

输入重定向：

# wc [选项] [文件名]

选项：

-c ：统计字节数

-w ：统计单词数

-l ： 统计行数

命令<文件 ：把文件作为命令的输入

命令<<标识符

标识符把标识符之间内容作为命令的输入。

#### 4》 管道符

多命令顺序执行：

|  |  |  |
| --- | --- | --- |
| 多命令执行符 | 格式 | 作用 |
| ; | 命令1 ; 命令2 | 多个命令顺序执行，命令之间没有任何逻辑联系 |
| && | 命令1 && 命令2 | 逻辑与  当命令1正确执行，则命令2才会执行；  当命令1执行不正确，则命令2不会执行。 |
| || | 命令1 || 命令2 | 逻辑或  当命令1执行不正确，则命令2才会执行；  当命令1正确执行，则命令2不会执行。 |

管道符：

命令格式： # 命令1 | 命令2

#命令1的正确输出作为命令2的操作对象。

# more : 分屏显示

# netstat -an : 所有连接

#### 5》通配符

|  |  |
| --- | --- |
| 通配符 | 作用 |
| ? | 匹配一个任意字符 |
| \* | 匹配0个或任意多个任意字符，也就是可以匹配任何内容 |
| [] | 匹配中括号中任意一个字符。例如：[abc]代表一定匹配一个字符，或者还是a,或者是b，或者是c . |
| [-] | 匹配中括号中任意一个字符，-代表一个范围。例如：[a-z]代表匹配一个小写字母 |
| [^] | 逻辑非，表示匹配不是中括号内的一个字符。例如：[^0-9]代表匹配一个不是数字的字符。 |

Bash中其他特殊符号：

|  |  |
| --- | --- |
| 符号 | 作用 |
| ‘’ | 单引号。在单引号中所有的特殊符号，如”$”和”`”都没有特殊含义。 |
| “” | 双引号。在双引号中特殊符号都没有特殊含义，但是“$”、“`”、“\”是例外，拥有“调用变量的值”、“引用命令”和“转义符”的特殊含义。 |
| `` | 反引号。反引号括起来的内容是系统命令，在Bash中会先执行它。和$()作用一样，不过推荐使用$()，因为反引号非常容易看错。 |
| $() | 和反引号作用一样，用来引用系统命令。 |
| # | 在Shell脚本中，#开头的行代表注释。 |
| $ | 用于调用变量的值，如需要调用变了name的值时，需要用$name的方式得到变量的值。 |
| \ | 转义符，跟在\之后的特殊符号将失去特殊含义，变为普通字符。如\$将输出“$”符号，而不当做时变了引用。 |

## VI编辑器

### 1、概述

**Visual Interface** ： 可视化接口

VIM升级：

- VIM支持多级撤销

- VIM可以跨平台运行

- VIM支持语法高亮

- VIM支持图形界面

操作模式：

Command Mode - 命令模式

Insert Mode - 输入模式

Last Line Mode - 底行模式（尾行，末行）

命令：**# vim +3 abc**

#打开一个文件，光标定位到第三行。

命令：**# vim +/mengyueping abc**

#打开一个文件，光标定位到mengyueping 第一次出现的那一行。

进入文件，点击字母 n 在mengyueping多次出现的位置，转换光标。

命令：**# vim aa bb cc**

#创建或打开多个文件

在底行模式，进行多文件之间的切换 ，输入 **:n** 查看下一个文件，输入 **:N** 或者  **:prev** 查看上一个文件。

### 2、底行模式，常用指令：

**# :w**

从内存写入硬盘，即：保存修改。

**# :q**

退出当前vim编辑器打开的文件。

**# :!**

强制执行。

**# :ls**

列出当前编辑器中打开的所有文件。

**# :n**

切换到下一个文件。

#:N 切换到上一个文件。

**# :15**

把光标快速定位到给定的数字那一行。

**# /xxx**

从光标位置向后搜索xxx这个字符串。搜索到xxx第一次出现的位置，光标就定位到那里。

# ?xxx

从光标位置向前搜索xxx这个字符串。

### 3、命令模式下，常用指令：

# h 光标左移

# j 光标下移

# k 光标上移

# l 光标右移

# ctrl + f 向下翻页 （front）

# ctrl + b 向上翻页 (back)

# ctrl + d 向下翻半页（down）

# ctrl + u 向上翻半页（up）

# dd 删除光标坐在行

# o 在光标所在行的下方，插入一行并切换到输入模式

# yy 复制光标所在的行

# p 在光标所在行的下方粘贴

# P 在光标所在行的上方粘贴

### 4、编辑器设置

v方式一：临时性质的设置

方式二：永久性质的设置（修改**vimrc**文件）

语法高亮： **syntax on**

显示行号： **set number**

自动缩进：

**set autoindent**

**set cindent**

自动加入文件头：

shell高亮显示：(centos中能够使用，mac终端中不起作用)

基本格式：

**echo –e 终端颜色 + 显示内容 + 结束后的颜色**

echo -e "\e[1;30m www.mengyueping.com Hi~ \e[1;0m"

echo -e "\e[1;35m" "www.mengyueping.com Hi~" $(tput sgr0)

\e[1;0m" ： 使用颜色闭合，将终端颜色重新设置成黑色0m.

$(tput sgr0) ：屏幕终端进行重置成默认色。

## 磁盘管理

# **df**  查看磁盘分区使用状况

- l 仅显示本地磁盘（默认）

- a 显示所有文件系统的磁盘使用情况，包含比如 /proc/

- h 以1024 进制计算最合适的单位显示磁盘容量

-H 以1000 进制计算最合适的单位显示磁盘容量

-T 显示磁盘分区类型

-t 显示指定类文件系统的磁盘分区

-x 不显示指定类型文件系统的磁盘分区

# **du** 统计磁盘上的文件大小

- b 以byte为单位统计文件

- k 以KB为单位统计文件

- m 以MB为单位统计文件

- h 按照1024 进制以最适合的单位统计文件

- H 按照1000 进制以最适合的单位统计文件

- s 指定统计目标

说明：

第一：Linux 中硬件设备都是以文件的形式存在于根目录下的dev目录下。

第二：硬件设备都是由Linux自动识别的。

第三：必须对硬盘进行分区、格式化、挂载后才能使用。

分区：

第一：主分区和扩展分区总数不能超过4个

第二：扩展分区最多只能有一个

第三：扩展分区不能直接存取数据（需要划分逻辑分区）。

分区模式：

MBR ： 主分区不超过4个；单个分区容量最大2TB；

GPT： 主分区个数“几乎”没有限制（128个主分区）；单个分区容量“几乎”没有限制；（18EB）

(1EB=1024PB, 1PB=1024TB, 1TB=1024GB)

GPT的主分区中，不适合安装X86架构的系统（32位）。

做MBR分区：

# **fdisk**

-l 列出当前分区表

# fdisk /dev/sdb（要进行磁盘路径） ： 进入分区模式

- n 添加一个新的分区

p primary 主分区

e extended 扩展分区

l logical 逻辑分区

做GPT分区：

# **parted**

# **select** /dev/sdc 选中一块要分区的硬盘

# **mklabel**  **gpt** 使用GPT分区表进行分区

# **mklabel** **msdos** 使用MBR分区表进行分区

# **mkpart** 添加一个新分区

# **print** 查看当前分区状况

（交互模式和命令模式进行分区）

命令模式：#**mkpart** test 2000 3000 (指定分区名字，分区范围)

# **unit GB**  指定分区磁盘大小单位

# **quit** 退出

parted分区会立即生效。

分区格式化：

# ls -l /dev/sdb\* 查看硬盘分区信息

格式化指定分区、指定格式：

# mkfs.ext3 /dev/sdb1 方法一

# mkfs -t ext4 /dev/sdb2 方法二

分区挂载：

分区默认的挂载目录是： /mnt 目录

挂载点必须存在。

# mkdir -p /mnt/test 创建要挂载的目录，挂载点

# mount /dev/sdb1 /mnt/test 挂载到创建的目录

# umount /mnt/test 卸载

# vim + /etc/fstab 分区永久性(系统重启，自动挂载)，需要编写/etc/fstab文件.

swap 交换分区：

第一：建立一个普通的Linux分区

第二： 修改分区类型的16进制编码

# fdisk /dev/sdb

# p

# t

# 输入分区标号

# L 查看16进制编码

83 Linux 普通的Linux分区

82 Linux swap swap分区

第三：格式化交换分区

第四：启用交换分区

# mkswap /dev/sdb6 格式化交换分区

# swapon /dev/sdb6 启用交换分区

# free 查看swap加载状况

# swapoff /dev/sdb6 停用交换分区

## 用户管理

用户：使用操作系统的人。

用户组：具有相同系统权限的一组用户。

# **cat /etc/group** 存储当前系统中所有用户组信息。

- Group : x : 123 : abc,def,xyz

- 组名称 ： 组密码占位符 ：　组编码　　：组中用户名列表

**# cat /etc/gshadow** 　　存储当前系统中用户组的密码信息

- Group : \* : : abc,def,xyz

- 组名称 : 组密码 : 组管理者 : 组中用户名列表

**# cat /etc/passwd**  存储当前系统中所有用户打信息

- user : x : 123 : 456 : xxxxxxxx : /home/user : /bin/bash

- 用户名 : 密码占位符:用户编号:用户组编号:用户注释信息 : 用户主目录 : shell类型

**# cat /etc/shadow** 存储当前系统中所有用户的密码信息

- user : vf;/Zu8sdf…..::::::

- 用户名 : 密码 ::::::

# groupadd 名字 :添加组

# groupmod -n 新名字 原名字 :修改组名字

# groupmod -g 指定组编号 组名字 :修改组编号

# groupadd -g 指定组编号 组名字 :创建组名字并指定组编号

# groupdel 用户组名字 :删除用户组

# useradd -g 用户组名字 用户名字 :创建用户到指定用户组

# useradd -d /home/xxx 用户名字 : 创建用户的时候,指定用户文件夹

# usermod -c 注释 用户名 :修改用户的注释

# usermod -l 新用户名 旧用户名 :将新用户的注释修改成和旧用户的一样

# usermod -d /home/xxx 用户名 :修改用户名的文件夹

# usermod -g 用户组组名 用户 : 更改用户的用户组

# userdel 用户名 :删除用户名

# userdel -r 用户名 : 删除用户名且删除用户文件夹

# touch /etc/nologin :创建一个nologin文件(空文件即可),暂时禁止普通账户(root之外的)登录服务器

高级命令:

# passwd -l 用户名 :锁定用户

# passwd -u 用户名 : 解锁用户

# passwd -d 用户名 : 清除用户密码

用户可以同时属于多个组:

- 一个主要组

- 多个附属组

# gpasswd -a 用户名 附属组名字 : 给用户添加附属组

# newgrp 附属用户组 : 用户自己切换到附属用户组

# gpasswd -d 用户名 用户组 : 去除用户的附属组

# useradd -g 主要组 -G 附属组1, 附属组2, ……. : 创建用户的时候,既要指定主要组,又指定附属组.

# gpasswd 用户组名字 : 设置用户组密码

# su 用户名 :当前用户,切换到其它用户

其它命令:

# Whoami

-显示当前登录用户名

# id 用户名

-显示指定用户信息,包括用户编号 \用户名

- 主要组编号及名称,附属组列表

# groups 用户名

- 显示用户所在的所有组

# chfn 用户名

- 设置用户资料,依次输入用户资料

# finger 用户名

- 显示用户详细资料

## Linux网络管理