

linux基础与shell编程

博客选编



目 录

前言

linux基础(一)-----登录以及文件系统的了解

linux基础(二)----linux常用命令积累

linux基础(三)----linux命令系统学习----安装和登录命令

linux基础(四)----linux命令系统学习----文件处理命令

linux基础(五)----linux命令系统学习----系统管理命令

linux基础(六)----linux命令系统学习----网络操作命令

linux基础(七)----linux命令系统学习----系统安全相关命令

linux基础(八)----linux命令系统学习----其它命令

linux基础(九)----linux性能监测

linux基础(十)----linux网络配置详细步骤---桥接模式和两台机子的远程通信

linux基础(十一)----linux编程基础----变量

linux基础(十二)----linux编程基础----与用户交互

linux基础(十三)----linux编程基础----linux运算符

linux基础(十四)----linux编程基础----linux条件控制语句----if else语句

linux基础(十五)----linux编程基础----linux条件控制语句----case语句

linux基础(十六)----linux编程基础----linux条件控制语句----多层嵌套控制结构

linux基础(十七)----linux编程基础----linux循环控制语句----while循环

linux基础(十八)----linux编程基础----linux循环控制语句----for in循环

linux基础(十九)----linux编程基础----linux循环控制语句----break中断和continue继续

linux基础(二十)----linux编程基础----子程序----函数

前言

原文出处：[linux基础与shell编程](#)

作者：[zzq900503](#)

本系列文章经作者授权在看云整理发布，未经作者允许，请勿转载！

linux基础与shell编程

linux系统入门管理的详细步骤与常用命令以及shell编程的基础知识。

linux基础(一)-----登录以及文件系统的了解

如果你使用的是window的系统 并且没有linux系统的机子，可以尝试安装虚拟机，在虚拟机中进行linux系统的操作。

虚拟机相关可见: [hadoop基础虚拟机第二篇---虚拟机安装以及安装linux系统](#)

##登录

在安装时 会提示输入 用户名和密码 并 提示 根用户root的密码与此密码相同。

启动系统后我们尝试登录。

```
CentOS release 6.4 (Final)
Kernel 2.6.32-358.el6.i686 on an i686

localhost login: root
Password:
Login incorrect

login: joe
Password:
Last login: Sat Sep 28 20:07:10 on tty4
[joe@localhost ~]$ su -
Password:
[root@localhost ~]# su joe
[joe@localhost root]$ _
```

登录时 密码不显示。在系统命令行模式中，为了保护密码安全，是不会显示任何输入的密码字符的。所以说，你可能没看见任何东西，但是密码却已经输进去了。输完密码回车就行啦！

centos 不能用 根用户登录 只能用 用户名登录后 再切换到 根用户root

账户切换命令：

root 用户

```
su -
```

其他用户

```
su username
```

文件系统的层次结构

登录完成后，需要熟悉linux文件系统的结构。

普通文件

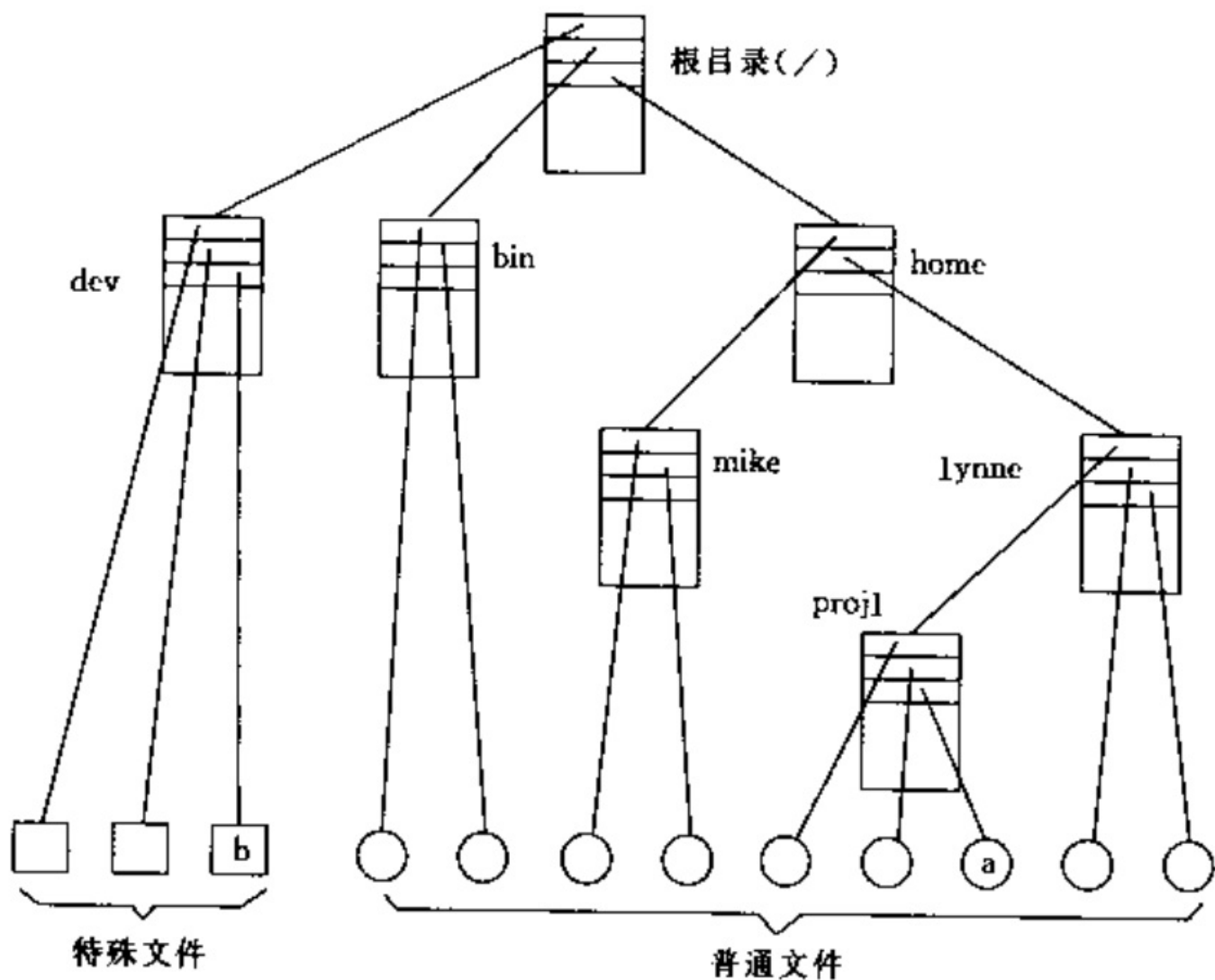
这些文件只是字节的集合。系统没有在文件中加入特定的结构。它们用作文本文件(包括源程序文件)，程序使用的数据文件，以及程序本身的可执行的二进制文件。

目录文件

目录是一种结构。目录是一个容器，可以用来存放其他文件和目录。事实上，目录本身只包含其他文件的名称和一些类似如何从磁盘上找到这此文件的简单信息。由于一个目录可以包含子目录名称，文件系统形成一个层次结构。

特殊文件

许多不同的文件类型属于这一范围。特殊文件与进程之间的通信以及进程和连接到机器的各种各样的外部设备之间的通信有关。所有这些类型的文件均放在一个大的树形层次结构中。树的顶部是一个单独的目录，称为根(root)目录(请勿与登录名root相混淆)。并且用斜杠符号/表示根目录。在根目录下，有一些用于不同目的的标准子目录和文件。这些高层的目录和文件结构只是从传统的意义讲是标准的，但并非一定要那样去做。



主要目录清单

- /bin 二进制可执行命令
- /dev 设备特殊文件
- /etc 系统管理和配置文件
- /home 用户起始目录的基点
- /lib 标准程序设计库
- /sbin 系统管理命令
- /tmp 公用的临时文件存贮点
- /usr/X11 X-windows系统文件
- /usr/adm 系统管理。数据文件
- /usr/bin其他的可执行命令
- /usr/lib库和软件包的配置文件
- /usr/local/bin本地增加的命令
- /usr/local/lib本地增加的库
- /usr/local/src本地命令的源文件

/usr/man系统联机手册页

/usr/src/linux Linux内核源程序文件

/var某些大文件的溢出区

关于目录的命令

显示当前目录

```
$pwd
```

前往目录 home

```
$cd /home
```

查看目录中的文件

```
$ls
```

```
CentOS release 6.4 (Final)
Kernel 2.6.32-358.el6.x86_64 on an x86_64

localhost login:

CentOS release 6.4 (Final)
Kernel 2.6.32-358.el6.x86_64 on an x86_64

localhost login: joe
Password:
Login incorrect

login: joe
Password:
[joe@localhost ~]$ pwd
/home/joe
[joe@localhost ~]$ cd /home
[joe@localhost home]$ ls
joe
[joe@localhost home]$ _
```

新建目录

```
$mkdir /home/joe
```

把目录分配给用户

```
$chown joe /home/joe
```

更改目的的读写权限

```
chmod 700 /home/joe
```

密码文件

密码(password)文件是系统的主要文件之一。

用户信息一般保存在 /etc/passwd

查看密码文件

```
$cat /etc/passwd
```

密码文件中的每一行是一个用户登录名的所有有关信息的记录。

每一条记录用冒号:分隔成7个字段(field)，具体格式如下

name:password:uid:gid:comment:home:shell

自左至右，7个字段的用途如下：

name 此字段包含用户登录名。这是用户登录时必须正确地敲入的名称。

password 这是用户的密码。该密码显示已经过加密，如果为空则表示该用户不需要密码。

uid 这是系统用来分配用户识别号的字段。一旦用户登录后，系统将用uid而不是用登录名来查找用户。

gid 有时候，一批用户需要在一个组内共同完成同一个项目。在这种情况下，允许他们共同访问一组特定的目录和文件是很有用的。这可以在这个字段内给小组的全体成员分配同一个组织别号(gid)来实现。

comment 这是注释字段。常用来保存用户的真实姓名和个人细节。

home 这一字段用来保存用户的起始目录的绝对路径名。当用户登录时，系统从这一字段取得用户起始目录路径名。

shell 如果这一用户登录成功，要执行的命令的绝对路径名就放在这一字段。这可以是任何命令。但是对普通用户帐号讲，这将是shell的路径名。如果此字段没有给出路径名，它的默认值是/bin/sh。

修改密码

```
$passwd
```

然后输入旧密码

再输入两次新密码

文件压缩解压

压缩成gz

```
$gzip 123.txt
```

解压gz

```
$gunzip 123.gz
```

压缩状态下读取内容

```
$zcat 123.gz | more
```

tar

c: 建立压缩档案

-x : 解压

-t : 查看内容

-r : 向压缩归档文件末尾追加文件

-u : 更新原压缩包中的文件

这五个是独立的命令，压缩解压都要用到其中一个，可以和别的命令连用但只能用其中一个。下面的参数是根据需要在压缩或解压档案时可选的。

-z : 有gzip属性的

-j : 有bz2属性的

-Z : 有compress属性的

-v : 显示所有过程

-O : 将文件解开到标准输出

下面的参数-f是必须的

-f: 使用档案名字，切记，这个参数是最后一个参数，后面只能接档案名。

解包 到当前路径

```
$tar xvf 123.tar
```

将当前目录的123.txt打包成tar

```
$tar cvf 123.tar ./123.txt
```

linux基础(二)----linux常用命令积累

才开始玩Linux 一进入看到黑漆漆的画面 感觉无从下手 试试下面的命令吧

查找文件

find

绝对强悍

```
find . -maxdepth 1 -name "@*"
```

这个命令意思是，查找当前目录下以@开头的文件或者目录，搜索深度为一级也就是只在当前目录找(.是 当前目录的意思)，不进入子目录。

如果你要从/目录开始找就：

```
find / -maxdepth 1 -name "@*"
```

如果想搜全盘，就把-maxdepth 1 去掉

编辑文件

vim

在 shell 模式下,键入vi 及需要编辑的文件名,即可进入vi. 例如:

```
vi example.txt
```

即可编辑 example.txt 文件.

如果该文件存在,则编辑界面中会显示该文件的内容,并将光标定位在文件的第一行;

如果文件不存在，则编辑界面中无任何内容。

如果需要在进入vi 编辑界面后，将光标置于文件的第n 行，则在vi命令后面加上 “+n” 参数即可。例如需要从 example.txt 文件的第5 行开始显示，则使用如下命令：

```
vi +5 example.txt
```

退出 vi 时，需要在末行模式中输入退出命令 “q” 。

如果在文本输入模式下，首先按 “ESC” 键进入命令模式，然后输入 “:” 进入末行模式在末行模式下，可使用如下退出命令：

:q 直接退出。如果在文本输入模式下修改了文档内容，则不能退出。

:wq 保存后退出。

:x 同 “wq”。

:q! – 不保存内容，强制退出。

1.写文件的话，可以用vi或者vim命令。例如：`$touch a.txt $vim a.txt`然后按i来编辑文档a.txt,编辑完成后，按Esc进入命令行，按shift键，然后键入冒号wq(:wq),表示保存你所作的修改并退出，如果不想保存所作的修改可以键入冒号q!(:q!),表示强制退出。随后可以用 cat命令来查看你所做的修改。

2. 修改文件名用 mv.例如：`mv a.txt b.txt`便把a.txt文件名修改成了b.txt.

清空文件内容：

用vi 打开 删了抓出来`cat /dev/null > filename`

部分快捷键命令

ctrl-f 在文件中前移一页(相当于page down)

ctrl-b 在文件中后移一页(相当于page up)

H 将光标移到屏幕上的起始行(或最上行);

M 将光标移到屏幕中间;

L 将光标移到屏幕最后一行。

/string 向前搜索给定的字符串string;

?string 向后搜索给定的字符串string;

n 向前或向后搜索，找出字符串下次出现的位置。

rc 用c替换当前光标指示的字符;

x 删除当前光标位置的字符;

dw 删除光标右面的字;

db 删除光标前面的字;

dd 删除光标所在的行，并去掉空隙。

在上面的任何命令前面加上数字，它们的功能扩充如下:

nrc 从光标位置开始用c替换n个字符;

nx 从光标位置开始删除n个字符;

ndw 在光标右面删除n个字;

ndb 在光标前面删除n个字;

ndd 删除n行，并去掉空隙。

其他常用的删除命令(前面不能加数字)是:

d\$ 从当前光标起删除字符直到行的结束;

d0 从当前光标起删除字符直到行的开始;

J 删除本行的回车字符(CR) , 并和下一行合并。

p(小写)将缓冲区的内容粘贴到当前光标的后面;

P(大写)将缓冲区的内容粘贴到当前光标的前面。

yy将当前行复制到剪切缓冲区;

nyy将n行复制到剪切缓冲区。

u 撤销前一条命令的结果;

. 重复最后一条修改正文的命令。

i 在光标左面插入正文;

a 在光标右面插入正文。

o在光标所在行下面增加新行;

O在光标所在行上面增加新行。

I 在光标行的开头插入;

A 在光标行的末尾插入;

:n 将光标移到第n行。

:a,b w file 将a行到b行的内容写到file中。

查看文件内容的方法及分页显示的办法

cat /etc/sysconfig/network

-n : 查看行号信息

more : 空格翻页 回车换行

less : 上下键翻页 可以反复看 (可以回翻页)

head : 显示前几行

head - 5 install.log 显示前5行

tail : 显示后几行

tail -f 动态更新

grep:查看文件中包含关键字的一行

grep --color=tty dump install.log 彩色显示

正则表达式 : ^root 以root为开头 (^)

root\$,以root结尾 (\$)

-v : 不包括的几行 (取反)

```
grep -n -v ^# /etc/vsftpd/vsftpd.conf
```

切换用户命令 : su - x1

ssh远程登录 : ssh 192.168.1.254

telnet服务远程登录：telnet 192.168.1.254(telnet不安全，默认不允许root用户登录)

在linux下一ping就ping个没完，怎么让它停下来？

Ctrl+c 停止

Ctrl+z 暂停

新建文件夹

mkdir filename

-m 用于对新建目录设置存取权限，也可以用 chmod 命令进行设置。

```
mkdir -m 777 test
```

-p 需要时创建上层文件夹(或目录)，如果文件夹(或目录)已经存在，则不视为错误。

删除文件 文件夹

rm

-i 删除前逐一询问确认。

-f 即使原档案属性设为唯读，亦直接删除，无需逐一确认。

-r 将目录及以下之档案亦逐一删除。

很多人还是习惯用rmdir，不过一旦目录非空，就陷入深深的苦恼之中，现在使用rm -rf命令即可。

直接rm就可以了，不过要加两个参数-rf 即：rm -rf 目录名字

-r 就是向下递归，不管有多少级目录，一并删除

-f 就是直接强行删除，不作任何提示的意思

注：在linux没有回收站，在试用rm命令的时候，一定要小心些，删除之后就无法再恢复了。

设置文件权限

r(Read，读取)：对文件而言，具有读取文件内容的权限；对目录来说，具有浏览目录的权限。

w(Write,写入)：对文件而言，具有新增、修改文件内容的权限；对目录来说，具有删除、移动目录内文件的权限。

x(eXecute，执行)：对文件而言，具有执行文件的权限；对目录来说该用户具有进入目录的权限。

r: 对应数值4

w: 对应数值2

x：对应数值1

-：对应数值0

数字设定的关键是mode的取值，一开始许多初学者会被搞糊涂，其实很简单，我们将rwx看成二进制数，如果有

则有1表示，没有则有0表示，那么rwx r-x r- 则可以表示成为：

111 101 100

再将其每三位转换成为一个十进制数，就是754。

例如，我们想让a.txt这个文件的权限为：

	可读	可写	可执行	
自己	是	是	-	110
同组用户	是	是	-	110
其他用户	是	-	-	100

那么，我们先根据上表得到权限串为：rw-rw-r--，那么转换成二进制数就是110 110 100，再每三位转换成为一个十进制数，就得到664，因此我们执行命令：

```
[root@localhost ~]# chmod 664 a.txt
```

查看系统版本号

登录到服务器执行 lsb_release -a ,即可列出所有版本信息,例如:

```
[root@t ~]# lsb_release -a
```

```
LSB Version: :core-4.0-amd64:core-4.0-noarch:graphics-4.0-amd64:graphics-4.0-noarch:printing-4.0-amd64:printing-4.0-noarch
```

```
Distributor ID: CentOS
```

```
Description: CentOS Linux release 6.0 (Final)
```

```
Release: 6.0
```

```
Codename: Final
```

2. 登录到linux执行cat /etc/redhat-release ，例如如下：

```
[root@3.5.5Biz-46 ~]# cat /etc/redhat-release
```

```
Red Hat Enterprise Linux AS release 4 (Nahant Update 1)
```

```
[root@3.5.5Biz-46 ~]#
```

这种方式下可以直接看到具体的版本号，比如 AS4 Update 1

3) 登录到linux执行rpm -q redhat-release ，例如如下

```
[root@3.5.5Biz-46 ~]# rpm -q redhat-release
```

redhat-release-4AS-2.4

[root@3.5.5Biz-46 ~]#

Linux命令行访问网页

curl <http://iframe.ip138.com/ic.asp>

(此命令可用于查询外网ip)

查看端口

```
netstat -apn
netstat -ntlp
```

常见参数

- a (all)显示所有选项，默认不显示LISTEN相关
- t (tcp)仅显示tcp相关选项
- u (udp)仅显示udp相关选项
- n 拒绝显示别名，能显示数字的全部转化成数字。
- l 仅列出有在 Listen (监听) 的服务状态
- p 显示建立相关链接的程序名
- r 显示路由信息，路由表
- e 显示扩展信息，例如uid等
- s 按各个协议进行统计
- c 每隔一个固定时间，执行该netstat命令。

复制文件

CP命令

格式: CP [选项] 源文件或目录 目的文件或目录

选项说明:-b 同名,备份原来的文件

- f 强制覆盖同名文件
- r 按递归方式保留原目录结构复制文件

```
cp -r /tmp/a /root/a
```

防火墙设置

Linux还是比较常用的，于是我研究了一下Linux关闭防火墙命令，在这里拿出来和大家分享一下，希望你能学会Linux关闭防火墙命令。

1) 永久性生效, 重启后不会复原

开启: `chkconfig iptables on`

关闭: `chkconfig iptables off`

2) 即时生效, 重启后复原

开启: `service iptables start`

关闭: `service iptables stop`

需要说明的是对于Linux下的其它服务都可以用以上命令执行开启和关闭操作。

在开启了防火墙时, 做如下设置, 开启相关端口,

修改/etc/sysconfig/iptables 文件, 添加以下内容:

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
```

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
```

查进程

ps命令查找与进程相关的PID号:

ps a 显示现行终端机下的所有程序, 包括其他用户的程序。

ps -A 显示所有程序。

ps c 列出程序时, 显示每个程序真正的指令名称, 而不包含路径, 参数或常驻服务的标示。

ps -e 此参数的效果和指定"A"参数相同。

ps e 列出程序时, 显示每个程序所使用的环境变量。

ps f 用ASCII字符显示树状结构, 表达程序间的相互关系。

ps -H 显示树状结构, 表示程序间的相互关系。

ps -N 显示所有的程序, 除了执行ps指令终端机下的程序之外。

ps s 采用程序信号的格式显示程序状况。

ps S 列出程序时, 包括已中断的子程序资料。

ps -t<终端机编号> 指定终端机编号, 并列出属于该终端机的程序的状况。

ps u 以用户为主的格式来显示程序状况。

ps x 显示所有程序, 不以终端机来区分。

最常用的方法是ps aux,然后再通过管道使用grep命令过滤查找特定的进程,然后再对特定的进程进行操作。

```
ps aux | grep program_filter_word,ps -ef |grep tomcat
```

ps -ef|grep java|grep -v grep 显示出所有的java进程, 去处掉当前的grep进程。

杀进程

使用kill命令结束进程: `kill xxx`

常用: `kill -9 324`

Linux下还提供了—killall命令，可以直接使用进程的名字而不是进程标识号，例如：# killall -9 NAME

tomcat管理

查找tomcat的目录

```
whereis tomcat
```

如果没有whereis 命令则是系统版本问题,可使用find查找文件夹

启动停止查看tomcat目录

```
service tomcat status  
service tomcat restart  
service tomcat start  
service tomcat stop
```

如果没有service命令

则需要进入到tomcat目录进行操作

进入bin目录

启动

```
./catalina.sh start
```

停止

```
./shutdown.sh
```

查看日志

进入logs目录

```
cat catalina.out
```

linux基础(三)----linux命令系统学习----安装和登录命令

不同Linux发行版的命令数量不一样，但Linux发行版本最少的命令也有200多个。这里把比较重要和使用频率最多的命令，按照它们在系统中的作用分成下面六个部分——介绍。

安装和登录命令：login、shutdown、halt、reboot、install、mount、umount、chsh、exit、last;

login

作用

login的作用是登录系统，它的使用权限是所有用户。

格式

login [name][- p][- h 主机名称]

主要参数

- p:通知login保持现在的环境参数。
- h:用来向远程登录的之间传输用户名。

步骤

如果选择用命令行模式登录Linux的话，那么看到的第一个Linux命令就是login：

一般界面是这样的：

```
Mandrake Linux release 9.1(Bamboo) for i586
```

```
renrel 2.4.21 - 0.13mdk on i686 / tty1
```

```
localhost login:root
```

```
password:
```

上面代码中，第一行是Linux发行版本号，第二行是内核版本号和登录的虚拟控制台，我们在第三行输入登录名，按“Enter”键在Password后输入账户密码，即可登录系统。

出于安全考虑，输入账户密码时字符不会在屏幕上回显，光标也不移动。

登录后会看到下面这个界面（以超级用户为例）：

```
[root@localhost root]#
```

```
last login:Tue ,Nov 18 10:00:55 on vc/1
```

上面显示的是登录星期、月、日、时间和使用的虚拟控制台。

应用技巧

Linux是一个真正的多用户操作系统，可以同时接受多个用户登录，还允许一个用户进行多次登录。

这是因为Linux和许多版本的Unix一样，提供了虚拟控制台的访问方式，允许用户在同一时间从控制台（系统的控制台是与系统直接相连的监视器和键盘）进行多次登录。

每个虚拟控制台可以看作是一个独立的工作站，工作台之间可以切换。虚拟控制台的切换可以通过按下Alt键和一个功能键来实现，通常使用F1-F6。

例如，用户登录后，按一下“Alt+F2”键，用户就可以看到上面出现的“login:”提示符，说明用户看到了第二个虚拟控制台。

然后只需按“Alt+F1”键，就可以回到第一个虚拟控制台。

一个新安装的Linux系统允许用户使用“Alt+F1”到“Alt+F6”键来访问前六个虚拟控制台。

虚拟控制台最有用的是，当一个程序出错造成系统死锁时，可以切换到其它虚拟控制台工作，关闭这个程序。

shutdown

作用

shutdown命令的作用是关闭计算机，它的使用权限是超级用户。

格式

```
shutdown [ - h ][ - i ][ - k ][ - m ][ - t ]
```

重要参数

- t：在改变到其它运行级别之前，告诉init程序多久以后关机。
- k：并不真正关机，只是送警告信号给每位登录者。
- h：关机后关闭电源。
- c：cancel current process取消目前正在执行的关机程序。所以这个选项当然没有时间参数，但是可以输入一个用来解释的讯息，而这信息将会送到每位使用者。
- F：在重启计算机时强迫fsck。
- time：设定关机前的时间。
- m：将系统改为单用户模式。
- i：关机时显示系统信息。

命令说明

shutdown命令可以安全地将系统关机。

有些用户会使用直接断掉电源的方式来关闭Linux系统，这是十分危险的。

因为Linux与Windows不同，其后台运行着许多进程，所以强制关机可能会导致进程的数据丢失，使系统处于不稳定的状态，甚至在有的系统中会损坏硬件设备（硬盘）。

在系统关机前使用shutdown命令，系统管理员会通知所有登录的用户系统将要关闭，并且login指令会被冻结，即新的用户不能再登录。

halt

作用

halt命令的作用是关闭系统，它的使用权限是超级用户。

格式

halt [- n] [- w] [- d] [- f] [- i] [- p]

主要参数说明

- n：防止sync系统调用，它用在用fsck修补根分区之后，以阻止内核用老版本的超级块覆盖修补过的超级块。
- w：并不是真正的重启或关机,只是写wtmp (/var/log/wtmp) 纪录。
- f：没有调用shutdown，而强制关机或重启。
- i：关机（或重启）前，关掉所有的网络接口。
- f：强迫关机，不呼叫shutdown这个指令。
- p: 当关机的时候顺便做关闭电源的动作。
- d：关闭系统，但不留下纪录。

命令说明

halt就是调用shutdown - h。

halt执行时，杀死应用进程，执行sync(将存于buffer中的资料强制写入硬盘中)系统调用，文件系统写操作完成后就会停止内核。

若系统的运行级别为0或6，则关闭系统；否则以shutdown指令（加上 - h参数）来取代。

reboot

作用

reboot命令的作用是重新启动计算机，它的使用权限是系统管理者。

格式

reboot [- n] [- w] [- d] [- f] [- i]

主要参数

- n: 在重开机前不做将记忆体资料写回硬盘的动作。
- w: 并不会真的重开机，只是把记录写到/var/log/wtmp文件里。
- d: 不把记录写到/var/log/wtmp文件里（ - n这个参数包含了 - d ）。
- i: 在重开机之前先把所有与网络相关的装置停止。

install

作用

install命令的作用是安装或升级软件或备份数据，它的使用权限是所有用户。

格式

(1)install [选项]... 来源目的地

(2)install [选项]... 来源... 目录

(3)install -d [选项]... 目录...

主要参数

- - backup[=CONTROL] : 为每个已存在的目的地文件进行备份。
- b : 类似 - - backup , 但不接受任何参数。
- c : (此选项不作处理)。
- d , - - directory : 所有参数都作为目录处理 , 而且会创建指定目录的所有主目录。
- D : 创建前的所有主目录 , 然后将复制至 ; 在第一种使用格式中 useful 。
- g , - - group=组 : 自行设定所属组 , 而不是进程目前的所属组。
- m , - - mode=模式 : 自行设定权限模式 (像chmod) , 而不是rwxr - xr - x。
- o , - - owner=所有者 : 自行设定所有者 (只适用于超级用户)。
- p , - - preserve - timestamps : 以文件的访问/修改时间作为相应的目的地文件的时间属性。
- s , - - strip : 用strip命令删除symbol table , 只适用于第一及第二种使用格式。
- S , - - suffix=后缀 : 自行指定备份文件的。
- v , - - verbose : 处理每个文件/目录时印出名称。
- - help : 显示此帮助信息并离开。
- - version : 显示版本信息并离开。

命令说明

在前两种格式中 , 会将复制至或将多个文件复制至已存在的 , 同时设定权限模式及所有者/所属组。

在第三种格式中 , 会创建所有指定的目录及它们的主目录。长选项必须用的参数在使用短选项时也是必须的。

mount

作用

mount命令的作用是加载文件系统 , 它的用权限是超级用户或/etc/fstab中允许的使用者。

格式

mount -a [- fv] [- t vfstype] [- n] [- rw] [- F] device dir

主要参数

- h : 显示辅助信息。
- v : 显示信息 , 通常和 - f用来除错。
- a : 将/etc/fstab中定义的所有文件系统挂上。
- F : 这个命令通常和 - a一起使用 , 它会为每一个mount的动作产生一个行程负责执行。在系统需要挂上大量

NFS文件系统时可以加快加载的速度。

- f：通常用于除错。它会使mount不执行实际挂上的动作，而是模拟整个挂上的过程，通常会和 - v一起使用。
- t vfstype：显示被加载文件系统的类型。
- n：一般而言，mount挂上后会在/etc/mstab中写入一笔资料，在系统中没有可写入文件系统的情况下，可以用这个选项取消这个动作。

应用技巧

在Linux和Unix系统上，所有文件都是作为一个大型树（以/为根）的一部分访问的。

要访问CD-ROM上的文件，需要将CD-ROM设备挂装在文件树中的某个挂装点。

如果发行版安装了自动挂装包，那么这个步骤可自动进行。

在Linux中，如果要使用硬盘、光驱等储存设备，就得先将它加载，当储存设备挂上了之后，就可以把它当成一个目录来访问。

挂上一个设备使用mount命令。

在使用mount这个指令时，至少要先知道下列三种信息：要加载对象的文件系统类型、要加载对象的设备名称及要将设备加载到哪个目录下。

Linux可以识别的文件系统

- ◆ Windows 95/98常用的FAT 32文件系统：vfat；
- ◆ Win NT/2000 的文件系统：ntfs；
- ◆ OS/2用的文件系统：hpfs；
- ◆ Linux用的文件系统：ext2、ext3；
- ◆ CD-ROM光盘用的文件系统：iso9660。

虽然vfat是指FAT 32系统，但事实上它也兼容FAT 16的文件系统类型。

确定设备的名称

在Linux中，设备名称通常都存在/dev里。

这些设备名称的命名都是有规则的，可以用“推理”的方式把设备名称找出来。

例如，/dev/hda1这个IDE设备，hd是Hard Disk(硬盘)的，sd是SCSI Device，fd是Floppy Device(或是Floppy Disk)。

a代表第一个设备，通常IDE接口可以接上4个IDE设备(比如4块硬盘)。

所以要识别IDE硬盘的方法分别就是hda、hdb、hdc、hdd。hda1中的“1”代表hda的第一个硬盘分区(partition)，hda2代表hda的第二主分区，第一个逻辑分区从hda5开始，依此类推。

此外，可以直接检查/var/log/messages文件，在该文件中可以找到计算机开机后系统已辨认出来的设备代号。

查找挂载点

在决定将设备挂载之前，先要查看一下计算机是不是有个/mnt的空目录，该目录就是专门用来当作挂载点

(Mount Point)的目录。

建议在/mnt里建几个/mnt/cdrom、/mnt/floppy、/mnt/mo等目录，当作目录的专用挂载点。

举例而言，如要挂载下列5个设备，其执行指令可能如下（假设都是Linux的ext2系统，如果是Windows XX请将ext2改成vfat）：

软盘 ==>mount -t ext2 /dev/fd0 /mnt/floppy

cdrom ==>mount -t iso9660 /dev/hdc/mnt/cdrom

SCSI cdrom ==>mount -t iso9660 /dev/sdb/mnt/scdrom

SCSI cdr ==>mount -t iso9660 /dev/sdc/mnt/scdr

不过目前大多数较新的Linux发行版本（包括红旗 Linux、中软Linux、Mandrake Linux等）都可以自动挂装文件系统，但Red Hat Linux除外。

umount

作用

umount命令的作用是卸载一个文件系统，它的使用权限是超级用户或/etc/fstab中允许的使用者。

格式

umount -a [- fNrsvw] [- t vfstype] [- n] [- rw] [- F] device dir

使用说明

umount命令是mount命令的逆操作，它的参数和使用方法和mount命令是一样的。

Linux挂装CD-ROM后，会锁定CD—ROM，这样就不能用CD-ROM面板上的Eject按钮弹出它。

但是，当不再需要光盘时，如果已将/cdrom作为符号链接，请使用umount/cdrom来卸装它。

仅当无用户正在使用光盘时，该命令才会成功。该命令包括了将带有当前工作目录当作该光盘中的目录的终端窗口。

chsh

作用

chsh命令的作用是更改使用者shell设定，它的使用权限是所有使用者。

格式

chsh [- s] [- list] [- - help] [- v] [username]

主要参数

- l：显示系统所有Shell类型。
- v：显示Shell版本号。

应用技巧

Linux下有多种Shell，一般缺省的是Bash，如果想更换Shell类型可以使用chsh命令。

先输入账户密码，然后输入新Shell类型，如果操作正确系统会显示“Shell change”。其界面一般如下：

```
Changing fihsng shell for cao
```

```
Password:
```

```
New shell [/bin/bash]: /bin/tcsh
```

上面代码中，[]内是目前使用的Shell。普通用户只能修改自己的Shell，超级用户可以修改全体用户的Shell。

要想查询系统提供哪些Shell，可以使用chsh -l 命令。

可以看到，系统中可以使用的Shell有bash（缺省）、csh、sh、tcsh四种。

exit

作用

exit命令的作用是退出系统，它的使用权限是所有用户。

格式

exit

参数

exit命令没有参数，运行后退出系统进入登录界面。

last

作用

last命令的作用是显示近期用户或终端的登录情况，它的使用权限是所有用户。通过last命令查看该程序的log，管理员可以获知谁曾经或企图连接系统。

格式

```
last[—n][ - f file][ - t tty] [—h 节点][ - I —IP][—1][ - y][1D]
```

主要参数

- n：指定输出记录的条数。
- f file：指定用文件file作为查询用的log文件。
- t tty：只显示指定的虚拟控制台上登录情况。
- h 节点：只显示指定的节点上的登录情况。
- i IP：只显示指定的IP上登录的情况。
- 1：用IP来显示远端地址。
- y：显示记录的年、月、日。
- ID：知道查询的用户名。

- x:显示系统关闭、用户登录和退出的历史。

linux基础(四)----linux命令系统学习----文件处理命令

文件处理命令：file、mkdir、grep、dd、find、mv、ls、diff、cat、ln；

Linux系统信息存放在文件里，文件与普通的公务文件类似。

每个文件都有自己的名字、内容、存放地址及其它一些管理信息，如文件的用户、文件的大小等。

文件可以是一封信、一个通讯录，或者是程序的源语句、程序的数据，可以包括可执行的程序和其它非正文内容。

Linux文件系统具有良好的结构，系统提供了很多文件处理程序。

这里主要介绍常用的文件处理命令。

file

作用

通过探测文件内容判断文件类型，使用权限是所有用户。

格式

file [options] 文件名

[options]主要参数

-v：在标准输出后显示版本信息，并且退出。

-z：探测压缩过的文件类型。

-L：允许符合连接。

-f name：从文件名file中读取要分析的文件名列表。

简单说明

使用file命令可以知道某个文件究竟是二进制（ELF格式）的可执行文件，还是Shell Script文件，或者是其它的格式。

file能识别的文件类型有目录、Shell脚本、英文文本、二进制可执行文件、C语言源文件、文本文件、DOS的可执行文件。

应用实例

如果我们看到一个没有后缀的文件grap，可以使用下面命令：

```
$ file grap
```

```
grap : English text
```

此时系统显示这是一个英文文本文件。需要说明的是，file命令不能探测包括图形、音频、视频等多媒体文件类

型。

mkdir

作用

mkdir命令的作用是建立名称为dirname的子目录，与MS DOS下的md命令类似，它的使用权限是所有用户。

格式

mkdir [options] 目录名

[options]主要参数

- m, - - mode=模式：设定权限，与chmod类似。
- p, - - parents：需要时创建上层目录；如果目录早已存在，则不当作错误。
- v, - - verbose：每次创建新目录都显示信息。
- - version：显示版本信息后离开。

应用实例

在进行目录创建时可以设置目录的权限，此时使用的参数是 “ - m ” 。

假设要创建的目录名是 “tsk” ，让所有用户都有rwx(即读、写、执行的权限)，那么可以使用以下命令：

```
$ mkdir -m 777 tsk
```

grep

作用

grep命令可以指定文件中搜索特定的内容，并将含有这些内容的行标准输出。

grep全称是Global Regular Expression Print，表示全局正则表达式版本，它的使用权限是所有用户。

格式

grep [options]

[options]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。

pattern正则表达式主要参数：

\：忽略正则表达式中特殊字符的原有含义。

^：匹配正则表达式的开始行。

\$：匹配正则表达式的结束行。

\：到匹配正则表达式的行结束。

[]：单个字符，如[A]即A符合要求。

[-]：范围，如[A-Z]，即A、B、C一直到Z都符合要求。

。：所有的单个字符。

- ：有字符，长度可以为0。

正则表达式是Linux/Unix系统中非常重要的概念。正则表达式（也称为“regex”或“regexp”）是一个可以描述一类字符串的模式（Pattern）。

如果一个字符串可以用某个正则表达式来描述，我们就说这个字符和该正则表达式匹配（Match）。

这和DOS中用户可以使用通配符“*”代表任意字符类似。

在Linux系统上，正则表达式通常被用来查找文本的模式，以及对文本执行“搜索 - 替换”操作和其它功能。

应用实例

查询DNS服务是日常工作之一，这意味着要维护覆盖不同网络的大量IP地址。

有时IP地址会超过2000个。如果要查看nnn.nnn网络地址，但是却忘了第二部分中的其余部分，只知到有两个句点，例如nnn nn..。

要抽取其中所有nnn.nnn IP地址，使用[0 - 9]{3}.[0 - 0{3}]\。含义是任意数字出现3次，后跟句点，接着是任意数字出现3次，后跟句点。

```
$grep '[0 - 9 ]{3}].[0 - 0{3}' ipfile
```

补充说明，grep家族还包括fgrep和egrep。

fgrep是fix grep，允许查找字符串而不是一个模式；

egrep是扩展grep，支持基本及扩展的正则表达式，但不支持\q模式范围的应用及与之相对应的一些更加规范的模式。

dd

作用

dd命令用来复制文件，并根据参数将数据转换和格式化。

格式

```
dd [options]
```

[opitions]主要参数

bs=字节：强迫 ibs=及obs=。

cbs=字节：每次转换指定的。

conv=关键字：根据以逗号分隔的关键字表示的方式来转换文件。

count=块数目：只复制指定的输入数据。

ibs=字节：每次读取指定的。

if=文件：读取内容，而非标准输入的数据。

obs=字节：每次写入指定的。

of=文件：将数据写入，而不在标准输出显示。

seek=块数目：先略过以obs为单位的指定的输出数据。

skip=块数目：先略过以ibs为单位的指定的输入数据。

应用实例

dd命令常常用来制作Linux启动盘。

先找一个可引导内核，令它的根设备指向正确的根分区，然后使用dd命令将其写入软盘：

```
$ rdev vmlinuz /dev/hda
```

```
$dd if = vmlinuz of = /dev/fd0
```

上面代码说明，使用rdev命令将可引导内核vmlinuz中的根设备指向/dev/hda，请把“hda”换成自己的根分区，接下来用dd命令将该内核写入软盘。

find

作用

find命令的作用是在目录中搜索文件，它的使用权限是所有用户。

格式

```
find [path][options][expression]
```

path指定目录路径，系统从这里开始沿着目录树向下查找文件。它是一个路径列表，相互用空格分离，如果不写path，那么默认为当前目录。

[options]参数：

- depth：使用深度级别的查找过程方式，在某层指定目录中优先查找文件内容。
- maxdepth levels：表示至多查找到开始目录的第level层子目录。level是一个非负数，如果level是0的话表示仅在当前目录中查找。
- mindepth levels：表示至少查找到开始目录的第level层子目录。
- mount：不在其它文件系统（如Msdos、Vfat等）的目录和文件中查找。
- version：打印版本。

[expression]是匹配表达式，是find命令接受的表达式，find命令的所有操作都是针对表达式的。它的参数非常多，这里只介绍一些常用的参数。

- name : 支持通配符*和?。
- atime n : 搜索在过去n天读取过的文件。
- ctime n : 搜索在过去n天修改过的文件。
- group grpoupname : 搜索所有组为grpoupname的文件。
- user 用户名 : 搜索所有文件属主为用户名 (ID或名称) 的文件。
- size n : 搜索文件大小是n个block的文件。
- print : 输出搜索结果, 并且打印。

应用技巧

find命令查找文件的几种方法：

根据文件名查找

例如, 我们想要查找一个文件名是lilo.conf的文件, 可以使用如下命令：

```
find / - name lilo.conf
```

find命令后的 “/” 表示搜索整个硬盘。

快速查找文件

根据文件名查找文件会遇到一个实际问题, 就是要花费相当长的一段时间, 特别是大型Linux文件系统和大容量硬盘文件放在很深的子目录中时。

如果我们知道了这个文件存放在某个目录中, 那么只要在这个目录中往下寻找就能节省很多时间。

比如smb.conf文件, 从它的文件后缀 “.conf” 可以判断这是一个配置文件, 那么它应该在/etc目录内, 此时可以使用下面命令：

```
find /etc - name smb.conf
```

这样, 使用 “快速查找文件” 方式可以缩短时间。

根据部分文件名查找方法

有时我们知道只某个文件包含有abvd这4个字, 那么要查找系统中所有包含有这4个字符的文件可以输入下面命令：

```
find / - name 'abvd'
```

输入这个命令以后, Linux系统会将在/目录中查找所有的包含有abvd这4个字符的文件 (其中*是通配符), 比如abvdrmyz等符合条件的文件都能显示出来。

使用混合查找方式查找文件

find命令可以使用混合查找的方法, 例如, 我们想在/etc目录中查找大于500000字节, 并且在24小时内修改的某个文件, 则可以使用-and (与)把两个查找参数链接起来组合成一个混合的查找方式。

```
find /etc -size +500000c -and -mtime +1
```

mv

作用

mv命令用来为文件或目录改名，或者将文件由一个目录移入另一个目录中，它的使用权限是所有用户。
该命令如同DOS命令中的ren和move的组合。

格式

mv[options] 源文件或目录 目标文件或目录

[options]主要参数

- i：交互方式操作。如果mv操作将导致对已存在的目标文件的覆盖，此时系统询问是否重写，要求用户回答“y”或“n”，这样可以避免误覆盖文件。
- f：禁止交互操作。mv操作要覆盖某个已有的目标文件时不给任何指示，指定此参数后i参数将不再起作用。

应用实例

(1) 将/usr/cbu中的所有文件移到当前目录（用“.”表示）中：

```
$ mv /usr/cbu/ * .
```

(2) 将文件cjh.txt重命名为wjz.txt：

```
$ mv cjh.txt wjz.txt
```

ls

作用

ls命令用于显示目录内容，类似DOS下的dir命令，它的使用权限是所有用户。

格式

ls [options][filename]

options主要参数

- a, - - all：不隐藏任何以“.”字符开始的项目。
- A, - - almost - all：列出除了“.”及“..”以外的任何项目。
- - author：印出每个文件著作者。
- b, - - escape：以八进制溢出序列表示不可打印的字符。
- - block - size=大小：块以指定的字节为单位。
- B, - - ignore - backups：不列出任何以~字符结束的项目。
- f：不进行排序，- aU参数生效，- lst参数失效。
- F, - - classify：加上文件类型的指示符号(*/=|@ 其中一个)。
- g：like - l, but do not list owner。

- G, - - no - group : inhibit display of group information.
- i, - - inode : 列出每个文件的inode号。
- I, - - ignore=样式 : 不印出任何符合Shell万用字符的项目。
- k : 即 - - block - size=1K。
- l : 使用较长格式列出信息。
- L, - - dereference : 当显示符号链接的文件信息时，显示符号链接所指示的对象，而并非符号链接本身的信息。
- m : 所有项目以逗号分隔，并填满整行行宽。
- n, - - numeric - uid - gid : 类似 - l，但列出UID及GID号。
- N, - - literal : 列出未经处理的项目名称，例如不特别处理控制字符。
- p, - - file - type : 加上文件类型的指示符号 (/=@| 其中一个)。
- Q, - - quote - name : 将项目名称括上双引号。
- r, - - reverse : 依相反次序排列。
- R, - - recursive : 同时列出所有子目录层。
- s, - - size : 以块大小为序。

应用举例

ls命令是Linux系统使用频率最多的命令，它的参数也是Linux命令中最多的。

使用ls命令时会有几种不同的颜色，其中蓝色表示是目录，绿色表示是可执行文件，红色表示是压缩文件，浅蓝色表示是链接文件，加粗的黑色表示符号链接，灰色表示是其它格式文件。

ls最常使用的是ls - l。

使用ls - l命令文件类型开头是由10个字符构成的字符串。

其中第一个字符表示文件类型，它可以是下述类型之一：-（普通文件）、d（目录）、l（符号链接）、b（块设备文件）、c（字符设备文件）。

后面的9个字符表示文件的访问权限，分为3组，每组3位。

第一组表示文件属主的权限，第二组表示同组用户的权限，第三组表示其他用户的权限。

每一组的三个字符分别表示对文件的读（r）、写（w）和执行权限（x）。

对于目录，表示进入权限。s表示当文件被执行时，把该文件的UID或GID赋予执行进程的UID（用户ID）或GID（组ID）。

t表示设置标志位（留在内存，不被换出）。

如果该文件是目录，那么在该目录中的文件只能被超级用户、目录拥有者或文件属主删除。

如果它是可执行文件，那么在该文件执行后，指向其正文段的指针仍留在内存。

这样再次执行它时，系统就能更快地装入该文件。接着显示的是文件大小、生成时间、文件或命令名称。

diff

作用

diff命令用于两个文件之间的比较，并指出两者的不同，它的使用权限是所有用户。

格式

diff [options] 源文件目标文件

[options]主要参数

- a：将所有文件当作文本文件来处理。
- b：忽略空格造成的不同。
- B：忽略空行造成的不同。
- c：使用纲要输出格式。
- H：利用试探法加速对大文件的搜索。
- I：忽略大小写的变化。
- n --rcs：输出RCS格式。

cmp

作用

cmp（“compare”的缩写）命令用来简要指出两个文件是否存在差异，它的使用权限是所有用户。

格式

cmp[options] 文件名

[options]主要参数

- l：将字节以十进制的方式输出，并方便将两个文件中不同的以八进制的方式输出。

cat

作用

cat（“concatenate”的缩写）命令用于连接并显示指定的一个和多个文件的有关信息，它的使用权限是所有用户。

格式

cat [options] 文件1 文件2.....

[options]主要参数

- n：由第一行开始对所有输出的行数编号。
- b：和-n相似，只不过对于空白行不编号。
- s：当遇到有连续两行以上的空白行时，就代换为一行的空白行。

应用举例

(1) cat命令一个最简单的用处是显示文本文件的内容。例如，我们想在命令行看一下README文件的内容，可以使用命令：

```
$ cat README
```

(2)有时需要将几个文件处理成一个文件，并将这种处理的结果保存到一个单独的输出文件。

cat命令在其输入上接受一个或多个文件，并将它们作为一个单独的文件打印到它的输出。

例如，把README和INSTALL的文件内容加上行号（空白行不加）之后，将内容附加到一个新文本文件File1中：

```
$ cat README INSTALL File1
```

(3) cat还有一个重要的功能就是可以对行进行编号。

这种功能对于程序文档的编制，以及法律和科学文档的编制很方便，打印在左边的行号使得参考文档的某一部分变得容易，这些在编程、科学研究、业务报告甚至是立法工作中都是非常重要的。

使用cat命令/etc/named.conf文件进行编号

对行进行编号功能有-b（只能对非空白行进行编号）和-n（可以对所有行进行编号）两个参数：

```
$ cat -b /etc/named.conf
```

ln

作用

ln命令用来在文件之间创建链接，它的使用权限是所有用户。

格式

ln [options] 源文件 [链接名]

参数

- f：链结时先将源文件删除。
- d：允许系统管理者硬链结自己的目录。
- s：进行软链结(Symbolic Link)。
- b：将在链结时会被覆盖或删除的文件进行备份。

链接有两种，一种被称为硬链接（Hard Link），另一种被称为符号链接（Symbolic Link）。

默认情况下，ln命令产生硬链接。

硬连接指通过索引节点来进行的连接。

在Linux的文件系统中，保存在磁盘分区中的文件不管是什么类型都给它分配一个编号，称为索引节点号(Inode Index)。

在Linux中，多个文件名指向同一索引节点是存在的。

一般这种连接就是硬连接。

硬连接的作用是允许一个文件拥有多个有效路径名，这样用户就可以建立硬连接到重要文件，以防止“误删”的功能。

其原因如上所述，因为对应该目录的索引节点有一个以上的连接。

只删除一个连接并不影响索引节点本身和其它的连接，只有当最后一个连接被删除后，文件的数据块及目录的连接才会被释放。

也就是说，文件才会被真正删除。

与硬连接相对应，Linux系统中还存在另一种连接，称为符号连接（Symbolic Link），也叫软连接。

软链接文件有点类似于Windows的快捷方式。

它实际上是特殊文件的一种。

在符号连接中，文件实际上是一个文本文件，其中包含的有另一文件的位置信息。

linux基础(五)----linux命令系统学习----系统管理命令

系统管理命令：df、top、free、quota、at、lp、adduser、groupadd、kill、crontab；

对于Linux系统来说，无论是中央处理器、内存、磁盘驱动器、键盘、鼠标，还是用户等都是文件，Linux系统管理的命令是它正常运行的核心。

熟悉了Linux常用的文件处理命令以后，这次了解对系统和用户进行管理的命令。

df

作用

df命令用来检查文件系统的磁盘空间占用情况，使用权限是所有用户。

格式

df [options]

主要参数

- s：对每个Names参数只给出占用的数据块总数。
- a：递归地显示指定目录中各文件及子目录中各文件占用的数据块数。若既不指定 - s，也不指定 - a，则只显示Names中的每一个目录及其中的各子目录所占的磁盘块数。
- k：以1024字节为单位列出磁盘空间使用情况。
- x：跳过在不同文件系统上的目录不予统计。
- l：计算所有的文件大小，对硬链接文件则计算多次。
- i：显示inode信息而非块使用量。
- h：以容易理解的格式印出文件系统大小，例如136KB、254MB、21GB。
- P：使用POSIX输出格式。
- T：显示文件系统类型。

说明

df命令被广泛地用来生成文件系统的使用统计数据，它能显示系统中所有的文件系统的信息，包括总容量、可用的空闲空间、目前的安装点等。

超级权限用户使用df命令时会发现这样的情况：某个分区的容量超过了100%。

这是因为Linux系统为超级用户保留了10%的空间，由其单独支配。

也就是说，对于超级用户而言，他所见到的硬盘容量将是110%。

这样的安排对于系统管理而言是有好处的，当硬盘被使用的容量接近100%时系统管理员还可以正常工作。

应用实例

Linux支持的文件系统非常多，包括JFS、ReiserFS、ext、ext2、ext3、ISO9660、XFS、Minx、vfat、MSDOS等。

使用df -T命令查看磁盘空间时还可以得到文件系统的信息：

```
# df -T
```

文件系统类型 容量 已用 可用 已用% 挂载点

```
/dev/hda7 reiserfs 5.2G 1.6G 3.7G 30% /
```

```
/dev/hda1 vfat 2.4G 1.6G 827M 66% /windows/C
```

```
/dev/hda5 vfat 3.0G 1.7G 1.3G 57% /windows/D
```

```
/dev/hda9 vfat 3.0G 2.4G 566M 82% /windows/E
```

```
/dev/hda10 NTFS 3.2G 573M 2.6G 18% /windows/F
```

```
/dev/hda11 vfat 1.6G 1.5G 23M 99% /windows/G
```

从上面除了可以看到磁盘空间的容量、使用情况外，分区的文件系统类型、挂载点等信息也一览无遗。

top

作用

top命令用来显示执行中的程序进程，使用权限是所有用户。

格式

```
top [-] [d delay] [q] [c] [S] [s] [n]
```

主要参数

d：指定更新的间隔，以秒计算。

q：没有任何延迟的更新。如果使用者有超级用户，则top命令将会以最高的优先序执行。

c：显示进程完整的路径与名称。

S：累积模式，会将已完成或消失的子行程的CPU时间累积起来。

s：安全模式。

i：不显示任何闲置(Idle)或无用(Zombie)的行程。

n：显示更新的次数，完成后将会退出top。

说明

top命令是Linux系统管理的一个主要命令，通过它可以获得许多信息。

top命令的显示第一行表示的项目依次为当前时间、系统启动时间、当前系统登录用户数目、平均负载。

第二行显示的是所有启动的进程、目前运行的、挂起(Sleeping)的和无用(Zombie)的进程。

第三行显示的是目前CPU的使用情况，包括系统占用的比例、用户使用比例、闲置(Idle)比例。

第四行显示物理内存的使用情况，包括总的可以使用的内存、已用内存、空闲内存、缓冲区占用的内存。

第五行显示交换分区使用情况，包括总的交换分区、使用的、空闲的和用于高速缓存的大小。

第六行显示的项目最多，下面列出了详细解释。

PID (Process ID)：进程标示号。

USER：进程所有者的用户名。

PR：进程的优先级别。

NI：进程的优先级别数值。

VIRT：进程占用的虚拟内存值。

RES：进程占用的物理内存值。

SHR：进程使用的共享内存值。

S：进程的状态，其中S表示休眠，R表示正在运行，Z表示僵死状态，N表示该进程优先值是负数。

%CPU：该进程占用的CPU使用率。

%MEM：该进程占用的物理内存和总内存的百分比。

TIME +：该进程启动后占用的总的CPU时间。

Command：进程启动的启动命令名称，如果这一行显示不下，进程会有一个完整的命令行。

top命令使用过程中，还可以使用一些交互的命令来完成其它参数的功能。这些命令是通过快捷键启动的。

：立刻刷新。

P：根据CPU使用大小进行排序。

T：根据时间、累计时间排序。

q：退出top命令。

m：切换显示内存信息。

t：切换显示进程和CPU状态信息。

c：切换显示命令名称和完整命令行。

M：根据使用内存大小进行排序。

W：将当前设置写入 ~/.toprc 文件中。这是写top配置文件的推荐方法。

可以看到，top命令是一个功能十分强大的监控系统的工具，对于系统管理员而言尤其重要。但是，它的缺点是会消耗很多系统资源。

应用实例

使用top命令可以监视指定用户，缺省情况是监视所有用户的进程。

如果想查看指定用户的情况，在终端中按“U”键，然后输入用户名，系统就会切换为指定用户的进程运行界面。

free

作用

free命令用来显示内存的使用情况，使用权限是所有用户。

格式

`free [- b | - k | - m] [- o] [- s delay] [- t] [- V]`

主要参数

- b - k - m : 分别以字节 (KB、MB) 为单位显示内存使用情况。
- s delay : 显示每隔多少秒数来显示一次内存使用情况。
- t : 显示内存总和列。
- o : 不显示缓冲区调节列。

应用实例

free命令是用来查看内存使用情况的主要命令。

和top命令相比，它的优点是使用简单，并且只占用很少的系统资源。

通过-s参数可以使用free命令不间断地监视有多少内存在使用，这样可以把它当作一个方便实时监控器。

```
#free -b -s5
```

使用这个命令后终端会连续不断地报告内存使用情况（以字节为单位），每5秒更新一次。

quota

作用

quota命令用来显示磁盘使用情况和限制情况，使用权限超级用户。

格式

`quota [- g] [- u] [- v] [- p]`用户名组名

参数

- g : 显示用户所在组的磁盘使用限制。
- u : 显示用户的磁盘使用限制。
- v : 显示没有分配空间的文件系统的分配情况。
- p : 显示简化信息。

应用实例

在企业应用中磁盘配额非常重要，普通用户要学会看懂自己的磁盘使用情况。

要查询自己的磁盘配额可以使用下面命令（下例中用户账号是caojh）：

```
# quota caojh
```

```
Disk quotas for user caojh(uid 502):
```

```
Filesystem blocks quota limit grace files quota limit grace
```

```
/dev/hda3 58 200000 400000 41 500 1000
```

以上显示ID号为502的caojh账号，文件个数设置为500~1000个，硬盘空间限制设置为200MB~400MB。

一旦磁盘配额要用完时，就需要删除一些垃圾文件或向系统管理员请求追加配额。

at

作用

at命令用来在指定时刻执行指定的命令序列。

格式

at [-v [-q x] [-f file] [-m] time

主要参数

-V：显示标准错误输出。

-q：许多队列输出。

-f：从文件中读取作业。

-m：执行完作业后发送电子邮件到用户。

time：设定作业执行的时间。

time格式有严格的要求，由小时、分钟、日期和时间的偏移量组成，其中日期的格式为MM.DD.YY，MM是分钟，DD是日期，YY是指年份。偏移量的格式为时间 + 偏移量，单位是minutes、hours和days。

应用实例

```
#at - f data 15:30 +2 days
```

上面命令表示让系统在两天后的15：30执行文件data中指明的作业。

lp

作用

lp是打印文件的命令，使用权限是所有用户。

格式

lp [-c] [-d] [-m] [-number] [-title] [-p]

主要参数

-c：先拷贝文件再打印。

-d：打印队列文件。

-m：打印结束后发送电子邮件到用户。

-number：打印份数。

-title：打印标题。

-p：设定打印的优先级别，最高为100。

应用实例

(1) 使用lp命令打印多个文件

```
#lp 2 3 4
```

request id is 11 (3 file(s))

其中2、3、4分别是文件名；“request id is 11 (3 file(s))”表示这是第11个打印命令，依次打印这三个文件。

(2) 设定打印优先级别

```
#lp lp -d LaserJet -p 90 /etc/aliases
```

通过添加“-p 90”，规定了打印作业的优先级为90。它将在优先级低于90的打印作业之前打印，包括没有设置优先级的作业，缺省优先级是50

useradd

作用

useradd命令用来建立用户帐号和创建用户的起始目录，使用权限是超级用户。

格式

```
useradd [ - d home] [ - s shell] [ - c comment] [ - m [ - k template]] [ - f inactive] [ - e expire ] [ - p  
passwd] [ - r] name
```

主要参数

- c：加上备注文字，备注文字保存在passwd的备注栏中。
- d：指定用户登入时的起始目录。
- D：变更预设值。
- e：指定账号的有效期限，缺省表示永久有效。
- f：指定在密码过期后多少天即关闭该账号。
- g：指定用户所属的群组。
- G：指定用户所属的附加群组。
- m：自动建立用户的登入目录。
- M：不要自动建立用户的登入目录。
- n：取消建立以用户名称为名的群组。
- r：建立系统账号。
- s：指定用户登入后所使用的shell。
- u：指定用户ID号。

说明

useradd可用来建立用户账号，它和adduser命令是相同的。

账号建好之后，再用passwd设定账号的密码。

使用useradd命令所建立的账号，实际上是保存在/etc/passwd文本文件中。

应用实例

建立一个新用户账户，并设置ID：

```
# useradd caojh - u 544
```

需要说明的是，设定ID值时尽量要大于500，以免冲突。因为Linux安装后会建立一些特殊用户，一般0到499之间的值留给bin、mail这样的系统账号。

groupadd

作用

groupadd命令用于将新组加入系统。

格式

```
groupadd [ - g gid ] [ - o ] [ - r ] [ - f ] groupname
```

主要参数

- g gid：指定组ID号。
- o：允许组ID号，不必惟一。
- r：加入组ID号，低于499系统账号。
- f：加入已经有的组时，发展程序退出。

应用实例

建立一个新组，并设置组ID加入系统：

```
# groupadd - g 344 cjh
```

此时在/etc/passwd文件中产生一个组ID（GID）是344的项目。

kill

作用

kill命令用来中止一个进程。

格式

```
kill [ - s signal | - p ] [ - a ] pid ...
```

```
kill - l [ signal ]
```

参数

- s：指定发送的信号。
- p：模拟发送信号。
- l：指定信号的名称列表。

pid：要中止进程的ID号。

Signal : 表示信号。

说明

进程是Linux系统中一个非常重要的概念。

Linux是一个多任务的操作系统，系统上经常同时运行着多个进程。

我们不关心这些进程究竟是如何分配的，或者是内核如何管理分配时间片的，所关心的是如何去控制这些进程，让它们能够很好地为用户服务。

Linux操作系统包括三种不同类型的进程，每种进程都有自己的特点和属性。

交互进程是由一个Shell启动的进程。

交互进程既可以在前台运行，也可以在后台运行。

批处理进程和终端没有联系，是一个进程序列。

监控进程（也称系统守护进程）是Linux系统启动时启动的进程，并在后台运行。

例如，httpd是著名的Apache服务器的监控进程。

kill命令的工作原理是，向Linux系统的内核发送一个系统操作信号和某个程序的进程标识号，然后系统内核就可以对进程标识号指定的进程进行操作。

比如在top命令中，我们看到系统运行许多进程，有时就需要使用kill中止某些进程来提高系统资源。

在讲解安装和登陆命令时，曾提到系统多个虚拟控制台的作用是当一个程序出错造成系统死锁时，可以切换到其它虚拟控制台工作关闭这个程序。

此时使用的命令就是kill，因为kill是大多数Shell内部命令可以直接调用的。

应用实例

（1）强行中止（经常使用杀掉）一个进程标识号为324的进程：

```
# kill - 9 324
```

（2）解除Linux系统的死锁

在Linux中有时会发生这样一种情况：一个程序崩溃，并且处于死锁的状态。

此时一般不用重新启动计算机，只需要中止(或者说是关闭)这个有问题的程序即可。

当kill处于X-Window界面时，主要的程序(除了崩溃的程序之外)一般都已经正常启动了。

此时打开一个终端，在那里中止有问题的程序。

比如，如果Mozilla浏览器程序出现了锁死的情况，可以使用kill命令来中止所有包含有Mozilla浏览器的程序。

首先用top命令查处该程序的PID，然后使用kill命令停止这个程序：

```
# kill - SIGKILL XXX
```

其中，XXX是包含有Mozilla浏览器的程序的进程标识号。

（3）使用命令回收内存

我们知道内存对于系统是非常重要的，回收内存可以提高系统资源。

kill命令可以及时地中止一些“越轨”的程序或很长时间没有相应的程序。

例如，使用top命令发现一个无用 (Zombie)的进程，此时可以使用下面命令：

```
# kill -9 XXX
```

其中，XXX是无用的进程标识号。

然后使用下面命令：

```
# free
```

此时会发现可用内存容量增加了。

(4) killall命令

Linux下还提供了一个killall命令，可以直接使用进程的名字而不是进程标识号，例如：

```
# killall -HUP inetd
```

crontab

作用

使用crontab命令可以修改crontab配置文件，然后该配置由cron公用程序在适当的时间执行，该命令使用权限是所有用户。

格式

crontab [-u user]文件

crontab [-u user] { -l | -r | -e }

主要参数

-e：执行文字编辑器来设定日程表，内定的文字编辑器是vi。

-r：删除目前的日程表。

-l：列出目前的日程表。

crontab文件的格式为“M H D m d cmd”。其中，M代表分钟（0~59），H代表小时（0~23），D代表天（1~31），m代表月（1~12），d代表一星期内的天（0~6，0为星期天）。cmd表示要运行的程序，它被送入sh执行，这个Shell只有USER、HOME、SHELL三个环境变量。

说明

和at命令相比，crontab命令适合完成固定周期的任务。

应用实例

设置一个定时、定期的系统提示：

```
[cao @www cao]#crontab -e
```

此时系统会打开一个vi编辑器。

如果输入以下内容：35 17 * * 5 wall"Tomorrow is Saturday I will go CS"，然后存盘退出。这时在/var/spool/cron/目录下会生产一个cao的文件，内容如下：

```
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.2707 installed on Thu Jan 1 22:01:51 2004)
# (Cron version -- $Id: crontab.c,v 2.131994/01/17 03:20:37 vixie Exp $)
35 17 * * 5 wall "Tomorrow is Saturday I will play CS "
```

这样每个星期五17：35系统就会弹出一个终端，提醒星期六可以打打CS了！。

linux基础(六)----linux命令系统学习----网络操作命令

网络操作命令：ifconfig、ip、ping、netstat、telnet、ftp、route、rlogin、rcp、finger、mail、nslookup；因为Linux系统是在Internet上起源和发展的，它与生俱来拥有强大的网络功能和丰富的网络应用软件，尤其是TCP/IP网络协议的实现尤为成熟。

Linux的网络命令比较多，其中一些命令像ping、ftp、telnet、route、netstat等在其它操作系统上也能看到，但也有一些Unix/Linux系统独有的命令，如ifconfig、finger、mail等。

Linux网络操作命令的一个特点是，命令参数选项和功能很多，一个命令往往还可以实现其它命令的功能。

ifconfig

作用

ifconfig用于查看和更改网络接口的地址和参数，包括IP地址、网络掩码、广播地址，使用权限是超级用户。

格式

ifconfig -interface [options] address

主要参数

-interface：指定的网络接口名，如eth0和eth1。

up：激活指定的网络接口卡。

down：关闭指定的网络接口。

broadcast address：设置接口的广播地址。

pointopoint：启用点对点方式。

address：设置指定接口设备的IP地址。

netmask address：设置接口的子网掩码。

应用说明

ifconfig是用来设置和配置网卡的命令行工具。

为了手工配置网络，这是一个必须掌握的命令。

使用该命令的好处是无须重新启动机器。

要赋给eth0接口IP地址207.164.186.2，并且马上激活它，使用下面命令：

```
#ifconfig eth0 210.34.6.89 netmask 255.255.255.128 broadcast 210.34.6.127
```

该命令的作用是设置网卡eth0的IP地址、网络掩码和网络的本地广播地址。

若运行不带任何参数的ifconfig命令，这个命令将显示机器所有激活接口的信息。

带有“-a”参数的命令则显示所有接口的信息，包括没有激活的接口。

注意，用ifconfig命令配置的网络设备参数，机器重新启动以后将会丢失。

如果要暂停某个网络接口的工作，可以使用down参数：

```
# ifconfig eth0 down
```

ip

作用

ip是iproute2软件包里面的一个强大的网络配置工具，它能够替代一些传统的网络管理工具，例如ifconfig、route等，使用权限为超级用户。

几乎所有的Linux发行版本都支持该命令。

格式

```
ip [OPTIONS] OBJECT [COMMAND [ARGUMENTS]]
```

主要参数

OPTIONS是修改ip行为或改变其输出的选项。所有的选项都是以-字符开头，分为长、短两种形式。目前，ip支持如表1所示选项。

OBJECT是要管理者获取信息的对象。目前ip认识的对象见表2所示。

表1 ip支持的选项

-V,-Version 打印ip的版本并退出。

-s,-stats,-statistics 输出更为详尽的信息。如果这个选项出现两次或多次，则输出的信息将更为详尽。

-f,-family 这个选项后面接协议种类，包括inet、inet6或link，强调使用的协议种类。

如果没有足够的信息告诉ip使用的协议种类，ip就会使用默认值inet或any。link比较特殊，它表示不涉及任何网络协议。

-4是-family inet的简写。

-6 是-family inet6的简写。

-0 是-family link的简写。

-o,-oneline 对每行记录都使用单行输出，回行用字符代替。如果需要使用wc、grep等工具处理ip的输出，则会用到这个选项。

-r,-resolve 查询域名解析系统，用获得的主机名代替主机IP地址

COMMAND

设置针对指定对象执行的操作，它和对象的类型有关。

一般情况下，ip支持对象的增加(add)、删除(delete)和展示(show或list)。

有些对象不支持这些操作，或者有其它的一些命令。

对于所有的对象，用户可以使用help命令获得帮助。

这个命令会列出这个对象支持的命令和参数的语法。

如果没有指定对象的操作命令，ip会使用默认的命令。

一般情况下，默认命令是list，如果对象不能列出，就会执行help命令。

ARGUMENTS

是命令的一些参数，它们依赖于对象和命令。

ip支持两种类型的参数：flag和parameter。

flag由一个关键词组成；parameter由一个关键词加一个数值组成。

为了方便，每个命令都有一个可以忽略的默认参数。

例如，参数dev是ip link命令的默认参数，因此ip link ls eth0等于ip link ls dev eth0。

我们将在后面的详细介绍每个命令的使用，命令的默认参数将使用default标出。

应用实例

添加IP地址192.168.2.2/24到eth0网卡上：

```
#ip addr add 192.168.1.1/24 dev eth0
```

丢弃源地址属于192.168.2.0/24网络的所有数据报：

```
#ip rule add from 192.168.2.0/24 prio 32777 reject
```

ping

作用

ping检测主机网络接口状态，使用权限是所有用户。

格式

ping [-dfnqrRv][-c][-i][-I][-l][-p][-s][-t] IP地址

主要参数

- d：使用Socket的SO_DEBUG功能。
- c：设置完成要求回应的次数。
- f：极限检测。
- i：指定收发信息的间隔秒数。
- I：网络界面使用指定的网络界面送出数据包。
- l：前置载入，设置在送出要求信息之前，先行发出的数据包。
- n：只输出数值。
- p：设置填满数据包的范本样式。
- q：不显示指令执行过程，开头和结尾的相关信息除外。
- r：忽略普通的Routing Table，直接将数据包送到远端主机上。
- R：记录路由过程。
- s：设置数据包的大小。
- t：设置存活数值TTL的大小。
- v：详细显示指令的执行过程。

ping命令是使用最多的网络指令，通常我们使用它检测网络是否连通，它使用ICMP协议。

但是有时会有这样的情况，我们可以浏览器查看一个网页，但是却无法ping通，这是因为一些网站处于安全考虑安装了防火墙。

另外，也可以在自己计算机上试一试，通过下面的方法使系统对ping没有反应：

```
# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

netstat

作用

检查整个Linux网络状态。

格式

netstat [-acCeFghilMnNoprstuvVwx][-A][--ip]

主要参数

- a--all：显示所有连线中的Socket。
- A：列出该网络类型连线中的IP相关地址和网络类型。
- c--continuous：持续列出网络状态。
- C--cache：显示路由器配置的快取信息。
- e--extend：显示网络其它相关信息。
- F--fib：显示FIB。
- g--groups：显示多重广播功能群组组员名单。
- h--help：在线帮助。
- i--interfaces：显示网络界面信息表单。
- l--listening：显示监控中的服务器的Socket。
- M--masquerade：显示伪装的网络连线。
- n--numeric：直接使用IP地址，而不通过域名服务器。
- N--netlink--symbolic：显示网络硬件外围设备的符号连接名称。
- o--timers：显示计时器。
- p--programs：显示正在使用Socket的程序识别码和程序名称。
- r--route：显示Routing Table。
- s--statistic：显示网络工作信息统计表。
- t--tcp：显示TCP传输协议的连线状况。
- u--udp：显示UDP传输协议的连线状况。
- v--verbose：显示指令执行过程。

-V--version : 显示版本信息。

-w--raw : 显示RAW传输协议的连线状况。

-x--unix : 和指定 “-A unix” 参数相同。

--ip--inet : 和指定 “-A inet” 参数相同。

应用实例

netstat

主要用于Linux察看自身的网络状况，如开启的端口、在为哪些用户服务，以及服务的状态等。此外，它还显示系统路由表、网络接口状态等。

可以说，它是一个综合性的网络状态的察看工具。

在默认情况下，netstat只显示已建立连接的端口。

如果要显示处于监听状态的所有端口，使用-a参数即可：

```
#netstat -a
```

Active Internet connections (only servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
-------	--------	--------	---------------	-----------------	-------

tcp	0	0	*:32768		LISTEN
-----	---	---	---------	--	--------

tcp	0	0	*:32769		LISTEN
-----	---	---	---------	--	--------

tcp	0	0	*:nfs		LISTEN
-----	---	---	-------	--	--------

tcp	0	0	*:32770		LISTEN
-----	---	---	---------	--	--------

tcp	0	0	*:868		LISTEN
-----	---	---	-------	--	--------

tcp	0	0	*:617		LISTEN
-----	---	---	-------	--	--------

tcp	0	0	*:mysql		LISTEN
-----	---	---	---------	--	--------

tcp	0	0	*:netbios-ssn		LISTEN
-----	---	---	---------------	--	--------

tcp	0	0	*:sunrpc		LISTEN
-----	---	---	----------	--	--------

tcp	0	0	*:10000		LISTEN
-----	---	---	---------	--	--------

tcp	0	0	*:http		LISTEN
-----	---	---	--------	--	--------

.....

上面显示出，这台主机同时提供HTTP、FTP、NFS、MySQL等服务。

telnet

作用

telnet表示开启终端机阶段作业，并登入远端主机。telnet是一个Linux命令，同时也是一个协议（远程登陆协议）。

格式

telnet [-8acdEfFKLrx][-b][-e][-k][-l][-n][-S][-X][主机名称IP地址]

主要参数

- 8：允许使用8位字符资料，包括输入与输出。
- a：尝试自动登入远端系统。
- b：使用别名指定远端主机名称。
- c：不读取用户专属目录里的.telnetrc文件。
- d：启动排错模式。
- e：设置脱离字符。
- E：滤除脱离字符。
- f：此参数的效果和指定“-F”参数相同。
- F：使用Kerberos V5认证时，加上此参数可把本地主机的认证数据上传到远端主机。
- k：使用Kerberos认证时，加上此参数让远端主机采用指定的领域名，而非该主机的域名。
- K：不自动登入远端主机。
- l：指定要登入远端主机的用户名称。
- L：允许输出8位字符资料。
- n：指定文件记录相关信息。
- r：使用类似rlogin指令的用户界面。
- S：服务类型，设置telnet连线所需的IP TOS信息。
- x：假设主机有支持数据加密的功能，就使用它。
- X：关闭指定的认证形态。

应用说明

用户使用telnet命令可以进行远程登录，并在远程计算机之间进行通信。

用户通过网络在远程计算机上登录，就像登录到本地机上执行命令一样。

为了通过telnet登录到远程计算机上，必须知道远程机上的合法用户名和口令。

虽然有些系统确实为远程用户提供登录功能，但出于对安全的考虑，要限制来宾的操作权限，因此，这种情况下能使用的功能是很少的。

telnet只为普通终端提供终端仿真，而不支持X-Window等图形环境。

当允许远程用户登录时，系统通常把这些用户放在一个受限制的Shell中，以防系统被怀有恶意的或不小心的用户破坏。

用户还可以使用telnet从远程站点登录到自己的计算机上，检查电子邮件、编辑文件和运行程序，就像在本地登录一样。

ftp

作用

ftp命令进行远程文件传输。FTP是ARPANet的标准文件传输协议，该网络就是现今Internet的前身，所以ftp既是协议又是一个命令。

格式

ftp [-dignv][主机名称IP地址]

主要参数

-d：详细显示指令执行过程，便于排错分析程序执行的情形。

-i：关闭互动模式，不询问任何问题。

-g：关闭本地主机文件名称支持特殊字符的扩充特性。

-n：不使用自动登陆。

-v：显示指令执行过程。

应用说明

ftp命令是标准的文件传输协议的用户接口，是在TCP/IP网络计算机之间传输文件简单有效的方法，它允许用户传输ASCII文件和二进制文件。

为了使用ftp来传输文件，用户必须知道远程计算机上的合法用户名和口令。

这个用户名/口令的组合用来确认ftp会话，并用来确定用户对要传输的文件进行什么样的访问。

另外，用户需要知道对其进行ftp会话的计算机名字的IP地址。

用户可以通过使用ftp客户程序，连接到另一台计算机上；

可以在目录中上下移动、列出目录内容；

可以把文件从远程计算机机拷贝到本地机上；

还可以把文件从本地机传输到远程系统中。

ftp内部命令有72个，下面列出主要几个内部命令：

ls：列出远程机的当前目录。

cd：在远程机上改变工作目录。

lcd：在本地机上改变工作目录。

close：终止当前的ftp会话。

hash：每次传输完数据缓冲区中的数据后就显示一个#号。

get (mget)：从远程机传送指定文件到本地机。

put (mput)：从本地机传送指定文件到远程机。

quit：断开与远程机的连接，并退出ftp。

##route

作用

route表示手工产生、修改和查看路由表。

格式

```
#route [-add][-net|-host] targetaddress [-netmask Nm][dev]If]
#route [-delete][-net|-host] targetaddress [gw Gw][-netmask Nm] [dev]If]
```

主要参数

-add：增加路由。

-delete：删除路由。

-net：路由到达的是一个网络，而不是一台主机。

-host：路由到达的是一台主机。

-netmask Nm：指定路由的子网掩码。

gw：指定路由的网关。

[dev]If：强迫路由链指定接口。

应用实例

route命令是用来查看和设置Linux系统的路由信息，以实现与其它网络的通信。

要实现两个不同的子网之间的通信，需要一台连接两个网络的路由器，或者同时位于两个网络的网关来实现。

在Linux系统中，设置路由通常是为了解决以下问题：该Linux系统在一个局域网中，局域网中有一个网关，能够让机器访问Internet，那么就需要将这台机器的IP地址设置为Linux机器的默认路由。

使用下面命令可以增加一个默认路由：

```
route add 0.0.0.0 192.168.1.1
```

rlogin

作用

rlogin用来进行远程注册。

格式

```
rlogin [-8EKldx] [-e char] [-k realm] [-l username] host
```

主要参数

-8：此选项始终允许8位输入数据通道。该选项允许发送格式化的ANSI字符和特殊的代码。如果不用这个选项，除非远端的不是终止和启动字符，否则就去掉奇偶校验位。

-E：停止把任何字符当作转义字符。当和-8选项一起使用时，它提供一个完全的透明连接。

-K：关闭所有的Kerberos确认。只有与使用Kerberos确认协议的主机连接时才使用这个选项。

-L：允许rlogin会话在litout模式中运行。要了解更多信息，请查阅tty联机帮助。

-d：打开与远程主机进行通信的TCP sockets的socket调试。要了解更多信息，请查阅setsockopt的联机帮助。

-e：为rlogin会话设置转义字符，默认的转义字符是“~”。

-k：请求rlogin获得在指定区域内远程主机的Kerberos许可，而不是获得由krb_realmofhost(3)确定的远程主机区域内的远程主机的Kerberos许可。

-x：为所有通过rlogin会话传送的数据打开DES加密。这会影响响应时间和CPU利用率，但是可以提高安全性。

使用说明

如果在网络中的不同系统上都有账号，或者可以访问别人在另一个系统上的账号，那么要访问别的系统中的账号，首先就要注册到系统中，接着通过网络远程注册到账号所在的系统中。

rlogin可以远程注册到别的系统中，它的参数应是一个系统名。

rcp

作用

rcp代表远程文件拷贝，用于计算机之间文件拷贝，使用权限是所有用户。

格式

rcp [-px] [-k realm] file1 file2 rcp [-px] [-r] [-k realm] file

主要参数

-r：递归地把源目录中的所有内容拷贝到目的目录中。要使用这个选项，目的必须是一个目录。

-p：试图保留源文件的修改时间和模式，忽略umask。

-k：请求rcp获得在指定区域内的远程主机的Kerberos许可，而不是获得由krb_realmofhost(3)确定的远程主机区域内的远程主机的Kerberos许可。

-x：为传送的所有数据打开DES加密。

finger

作用

finger用来查询一台主机上的登录账号的信息，通常会显示用户名、主目录、停滞时间、登录时间、登录Shell等信息，使用权限为所有用户。

格式

finger [选项] [使用者] [用户@主机]

主要参数

-s：显示用户注册名、实际姓名、终端名称、写状态、停滞时间、登录时间等信息。

-l：除了用-s选项显示的信息外，还显示用户主目录、登录Shell、邮件状态等信息，以及用户主目录下的.plan、.project和.forward文件的内容。

-p：除了不显示.plan文件和.project文件以外，与-l选项相同。

应用实例

在计算机上使用finger：

```
[root@localhost root]# Finger
```

```
Login Name Tty Idle Login Time Office Office Phone
```

```
root root tty1 2 Dec 15 11
```

```
root root pts/0 1 Dec 15 11
```

```
root root *pts/1 Dec 15 11
```

应用说明

如果要查询远程机上的用户信息，需要在用户名后面接“@主机名”，采用[用户名@主机名]的格式，不过要查询的网络主机需要运行finger守护进程的支持。

mail

作用

mail作用是发送电子邮件，使用权限是所有用户。此外，mail还是一个电子邮件程序。

格式

```
mail [-s subject] [-c address] [-b address]
```

```
mail -f [mailbox]mail [-u user]
```

主要参数

-b address：表示输出信息的匿名收信人地址清单。

-c address：表示输出信息的抄送（）收信人地址清单。

-f [mailbox]：从收件箱者指定邮箱读取邮件。

-s subject：指定输出信息的主体行。

[-u user]：端口指定优化的收件箱读取邮件。

nslookup

作用

nslookup命令的功能是查询一台机器的IP地址和其对应的域名。使用权限所有用户。

它通常需要一台域名服务器来提供域名服务。如果用户已经设置好域名服务器，就可以用这个命令查看不同主机的IP地址对应的域名。

格式

```
nslookup [ IP地址/域名 ]
```

应用实例

(1) 在本地计算机上使用nslookup命令

```
$ nslookup
```

```
Default Server: name.cao.com.cn
```

```
Address: 192.168.1.9
```

■

在符号 ">" 后面输入要查询的IP地址域名，并回车即可。如果要退出该命令，输入 "exit" ，并回车即可。

(2) 使用nslookup命令测试named

输入下面命令：

```
nslookup
```

然后就进入交互式nslookup环境。

如果named正常启动，则nslookup会显示当前DNS服务器的地址和域名，否则表示named没能正常启动。

下面简单介绍一些基本的DNS诊断。

◆检查正向DNS解析，在nslookup提示符下输入带域名的主机名，如[hp712.my.com](#)，nslookup应能显示该主机名对应的IP地址。

如果只输入hp712，nslookup会根据/etc/resolv.conf的定义，自动添加my.com域名，并回答对应的IP地址。

◆检查反向DNS解析，在nslookup提示符下输入某个IP地址，如192.22.33.20，nslookup应能回答该IP地址所对应的主机名。

◆检查MX邮件地址记录在nslookup提示符下输入：

```
set q=mx
```

然后输入某个域名，输入[my.com](#)和[mail.my.com](#)，nslookup应能够回答对应的邮件服务器地址，即[support.my.com](#)和[support2.my.com](#)。

linux基础(七)----linux命令系统学习----系统安全相关命令

系统安全相关命令：passwd、su、umask、chgrp、chmod、chown、chattr、sudo、ps、who；

虽然Linux和Windows NT/2000系统一样是一个多用户的系统，但是它们之间有不少重要的差别。

对于很多习惯了Windows系统的管理员来讲，如何保证Linux操作系统安全、可靠将会面临许多新的挑战。本文将重点介绍Linux系统安全的命令。

passwd

作用

passwd命令原来修改账户的登陆密码，使用权限是所有用户。

格式

passwd [选项]账户名称

主要参数

-l：锁定已经命名的账户名称，只有具备超级用户权限的使用者方可使用。

-u：解开账户锁定状态，只有具备超级用户权限的使用者方可使用。

-x, --maximum=DAYS：最大密码使用时间（天），只有具备超级用户权限的使用者方可使用。

-n, --minimum=DAYS：最小密码使用时间（天），只有具备超级用户权限的使用者方可使用。

-d：删除使用者的密码,只有具备超级用户权限的使用者方可使用。

-S：检查指定使用者的密码认证种类,只有具备超级用户权限的使用者方可使用。

应用实例

```
$ passwd
```

```
Changing password for user cao.
```

```
Changing password for cao
```

```
(current) UNIX password:
```

```
New UNIX password:
```

```
Retype new UNIX password:
```

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

从上面可以看到，使用passwd命令需要输入旧的密码，然后再输入两次新密码。

SU

作用

su的作用是变更为其它使用者的身份，超级用户除外，需要键入该使用者的密码。

格式

su [选项]... [-] [USER [ARG]...]

主要参数

-f , --fast : 不必读启动文件（如 csh.cshrc等），仅用于csh或tcsh两种Shell。

-l , --login : 加了这个参数之后，就好像是重新登陆为该使用者一样，大部分环境变量（例如HOME、SHELL和USER等）都是以该使用者（USER）为主，并且工作目录也会改变。如果没有指定USER，缺省情况是root。

-m , -p , --preserve-environment : 执行su时不改变环境变数。

-c command : 变更账号为USER的使用者，并执行指令（command）后再变回原来使用者。

USER : 欲变更的使用者账号，ARG传入新的Shell参数。

应用实例

变更账号为超级用户，并在执行df命令后还原使用者。 su -c df root

umask

作用

umask设置用户文件和目录的文件创建缺省屏蔽值，若将此命令放入profile文件，就可控制该用户后续所建文件的存取许可。

它告诉系统在创建文件时不给谁存取许可。使用权限是所有用户。

格式

umask [-p] [-S] [mode]

参数

- S : 确定当前的umask设置。

- p : 修改umask设置。

[mode] : 修改数值。

说明

传统Unix的umask值是022，这样就可以防止同属于该组的其它用户及别的组的用户修改该用户的文件。

既然每个用户都拥有并属于一个自己的私有组，那么这种“组保护模式”就不在需要了。

严密的权限设定构成了Linux安全的基础，在权限上犯错误是致命的。

需要注意的是，umask命令用来设置进程所创建的文件的读写权限，最保险的值是0077，即关闭创建文件的进程以外的所有进程的读写权限，表示为-rw-----。

在~/.bash_profile中，加上一行命令umask 0077可以保证每次启动Shell后,进程的umask权限都可以被正确设

定。

应用实例

```
umask -S
```

```
u=rwx,g=rx,o=rx
```

```
umask -p 177
```

```
umask -S
```

```
u=rw,g=,o=
```

上述5行命令，首先显示当前状态，然后把umask值改为177，结果只有文件所有者具有读写文件的权限，其它用户不能访问该文件。这显然是一种非常安全的设置。

chgrp

作用

chgrp表示修改一个或多个文件或目录所属的组。使用权限是超级用户。

格式

```
chgrp [选项]...组文件...
```

或

```
chgrp [选项]... --reference=参考文件文件...
```

将每个的所属组设定为。

参数

-c, --changes : 像 --verbose，但只有在有更改时才显示结果。

--dereference : 会影响符号链接所指示的对象，而非符号链接本身。

-h, --no-dereference : 会影响符号链接本身，而非符号链接所指示的目的地(当系统支持更改符号链接的所有者，此选项才有效)。

-f, --silent, --quiet : 去除大部分的错误信息。

--reference=参考文件 : 使用的所属组，而非指定的。

-R, --recursive : 递归处理所有的文件及子目录。

-v, --verbose : 处理任何文件都会显示信息。

应用说明

该命令改变指定指定文件所属的用户组。

其中group可以是用户组ID，也可以是/etc/group文件中用户组的组名。

文件名是以空格分开的要改变属组的文件列表，支持通配符。

如果用户不是该文件的属主或超级用户，则不能改变该文件的组。

应用实例

改变/opt/local /book/及其子目录下的所有文件的属组为book，命令如下：

```
$ chgrp - R book /opt/local /book
```

chmod

作用

chmod命令是非常重要的，用于改变文件或目录的访问权限，用户可以用它控制文件或目录的访问权限，使用权限是超级用户。

格式

chmod命令有两种用法。一种是包含字母和操作符表达式的字符设定法（相对权限设定）；另一种是包含数字的数字设定法（绝对权限设定）。

（1）字符设定法

chmod [who] [+ | - | =] [mode] 文件名

◆操作对象who可以是下述字母中的任一个或它们的组合

u：表示用户，即文件或目录的所有者。

g：表示同组用户，即与文件属主有相同组ID的所有用户。

o：表示其它用户。

a：表示所有用户，它是系统默认值。

◆操作符号

＋：添加某个权限。

－：取消某个权限。

=：赋予给定权限，并取消其它所有权限（如果有的话）。

◆设置mode的权限可用下述字母的任意组合

r：可读。

w：可写。

x：可执行。

X：只有目标文件对某些用户是可执行的或该目标文件是目录时才追加x属性。

s：文件执行时把进程的属主或组ID置为该文件的文件属主。方式“u＋s”设置文件的用户ID位，“g＋s”设置组ID位。

t：保存程序的文本到交换设备上。

u：与文件属主拥有一样的权限。

g：与和文件属主同组的用户拥有一样的权限。

o：与其它用户拥有一样的权限。

文件名：以空格分开的要改变权限的文件列表，支持通配符。

一个命令行中可以给出多个权限方式，其间用逗号隔开。

（2）数字设定法

数字设定法的一般形式为： `chmod [mode]文件名`

数字属性的格式应为3个0到7的八进制数，其顺序是(u)(g)(o)文件名，以空格分开的要改变权限的文件列表，支持通配符。

数字表示的权限的含义如下：

0001为所有者的执行权限；

0002为所有者的写权限；

0004为所有者的读权限；

0010为组的执行权限；

0020为组的写权限；

0040为组的读权限；

0100为其他人的执行权限；

0200为其他人的写权限；

0400为其他人的读权限；

1000为粘贴位置位；

2000表示假

如这个文件是可执行文件，则为组ID为位置位，否则其中文件锁定位置位；

4000表示假如这个文件是可执行文件，则为用户ID为位置位。

实例

如果一个系统管理员写了一个表格(tem)让所有用户填写，那么必须授权用户对这个文件有读写权限，可以使用命令：`# chmod 666 tem`

上面代码中，这个666数字是如何计算出来的呢？

0002为所有者的写权限，0004为所有者的读权限，0020为组的写权限，0040为组的读权限，0200为其他人的写权限，0400为其他人的读权限，这6个数字相加就是666（注以上数字都是八进制数），可以看出，tem文件的权限是-rw-rw-rw-，即用户对这个文件有读写权限。

如果用字符权限设定使用下面命令：

```
# chmod a =wx tem
```

chown

作用

更改一个或多个文件或目录的属主和属组。使用权限是超级用户。

格式

chown [选项]用户或组文件

主要参数

--dereference：受影响的是符号链接所指示的对象，而非符号链接本身。

-h, --no-dereference：会影响符号链接本身，而非符号链接所指示的目的地(当系统支持更改符号链接的所有者，此选项才有效)。

--from=目前所有者:目前组只当每个文件的所有者和组符合选项所指定的，才会更改所有者和组。其中一个可以省略，这已省略的属性就不需要符合原有的属性。

-f, --silent, --quiet：去除大部分的错误信息。

-R, --recursive：递归处理所有的文件及子目录。

-v, --verbose：处理任何文件都会显示信息。

说明

chown将指定文件的拥有者改为指定的用户或组，用户可以是用户名或用户ID；组可以是组名或组ID；文件是以空格分开的要改变权限的文件列表，支持通配符。

系统管理员经常使用chown命令，在将文件拷贝到另一个用户的目录下以后，让用户拥有使用该文件的权限。

应用实例

1.把文件shiyang.c的所有者改为wan

```
$ chown wan shiyang.c
```

2.把目录/hi及其下的所有文件和子目录的属主改成wan，属组改成users。

```
$ chown -R wan.users /hi
```

chattr

作用

修改ext2和ext3文件系统属性(attribute)，使用权限超级用户。

格式

chattr [-RV] [-+=AacDdijsSu] [-v version]文件或目录

主要参数

- R：递归处理所有的文件及子目录。

- V：详细显示修改内容，并打印输出。

- ：失效属性。

+ ：激活属性。

= ：指定属性。

A : Atime , 告诉系统不要修改对这个文件的最后访问时间。

S : Sync , 一旦应用程序对这个文件执行了写操作, 使系统立刻把修改的结果写到磁盘。

a : Append Only , 系统只允许在这个文件之后追加数据, 不允许任何进程覆盖或截断这个文件。如果目录具有这个属性, 系统将只允许在这个目录下建立和修改文件, 而不允许删除任何文件。

i : Immutable , 系统不允许对这个文件进行任何的修改。如果目录具有这个属性, 那么任何的进程只能修改目录之下的文件, 不允许建立和删除文件。

D : 检查压缩文件中的错误。

d : No dump , 在进行文件系统备份时, dump程序将忽略这个文件。

C : Compress , 系统以透明的方式压缩这个文件。从这个文件读取时, 返回的是解压之后的数据; 而向这个文件中写入数据时, 数据首先被压缩之后才写入磁盘。

s : Secure Delete , 让系统在删除这个文件时, 使用0填充文件所在的区域。

u : Undelete , 当一个应用程序请求删除这个文件, 系统会保留其数据块以便以后能够恢复删除这个文件。

说明

chattr命令的作用很大, 其中一些功能是由Linux内核版本来支持的, 如果Linux内核版本低于2.2, 那么许多功能不能实现。

同样 - D检查压缩文件中的错误的功能, 需要2.5.19以上内核才能支持。

另外, 通过chattr命令修改属性能够提高系统的安全性, 但是它并不适合所有的目录。

chattr命令不能保护/、/dev、/tmp、/var目录。

应用实例

1.恢复/root目录,即子目录的所有文件

```
# chattr -R +u/root
```

2.用chattr命令防止系统中某个关键文件被修改

在Linux下, 有些配置文件(passwd ,fatab)是不允许任何人修改的, 为了防止被误删除或修改, 可以设定该文件的“不可修改位(immutable)”, 命令如下:

```
# chattr +i /etc/fstab
```

sudo

作用

sudo是一种以限制配置文件中的命令为基础, 在有限时间内给用户使用, 并且记录到日志中的命令, 权限是所有用户。

格式

```
sudo [-bhHpV] [-s ] [-u ] [指令]
```

```
sudo [-klv]
```

主要参数

- b：在后台执行命令。
- h：显示帮助。
- H：将HOME环境变量设为新身份的HOME环境变量。
- k：结束密码的有效期，即下次将需要输入密码。
- l：列出当前用户可以使用的命令。
- p：改变询问密码的提示符号。
- s：执行指定的Shell。
- u：以指定的用户为新身份，不使用时默认为root。
- v：延长密码有效期5分钟。

说明

sudo命令的配置在/etc/sudoers文件中。

当用户使用sudo时，需要输入口令以验证使用者身份。

随后的一段时间内可以使用定义好的命令，当使用配置文件中没有的命令时，将会有报警的记录。

sudo是系统管理员用来允许某些用户以root身份运行部分/全部系统命令的程序。

一个明显的用途是增强了站点的安全性，如果需要每天以超级用户的身份做一些日常工作，经常执行一些固定的几个只有超级用户身份才能执行的命令，那么用sudo是非常适合的。

ps

作用

ps显示瞬间进程 (process)的动态，使用权限是所有使用者。

格式

```
ps [options] [--help]
```

主要参数

ps的参数非常多, 此出仅列出几个常用的参数。

- A：列出所有的进程。
- l：显示长列表。
- m：显示内存信息。
- w：显示加宽可以显示较多的信息。
- e：显示所有进程。
- a：显示终端上的所有进程,包括其它用户的进程。

-au：显示较详细的信息。

-aux：显示所有包含其它使用者的进程。

说明

要对进程进行监测和控制，首先要了解当前进程的情况，也就是需要查看当前进程。

ps命令就是最基本、也是非常强大的进程查看命令。

使用该命令可以确定有哪些进程正在运行、运行的状态、进程是否结束、进程有没有僵尸、哪些进程占用了过多的资源等。

大部分信息都可以通过执行该命令得到。

最常用的三个参数是u、a、x。

下面就结合这三个参数详细说明ps命令的作用：ps aux

ps-aux命令详解

第2行代码中，USER表示进程拥有者；PID表示进程标示符；%CPU表示占用的CPU使用率；%MEM占用的物理内存使用率；VSZ表示占用的虚拟内存大小；RSS为进程占用的物理内存值；TTY为终端的次要装置号码。

STAT表示进程的状态，其中D为不可中断的静止（I/O动作）；R正在执行中；S静止状态；T暂停执行；Z不存在，但暂时无法消除；W没有足够的内存分页可分配；高优先序的进程；N低优先序的进程；L有内存分页分配并锁在内存体内（实时系统或I/O）。

START为进程开始时间。TIME为执行的时间。COMMAND是所执行的指令。

应用实例

在进行系统维护时，经常会出现内存使用量惊人，而又不知道是哪一个进程占用了大量进程的情况。

除了可以使用top命令查看内存使用情况之外，还可以使用下面的命令：

```
ps aux | sort +5n
```

who

作用

who显示系统中有哪些用户登陆系统，显示的资料包含了使用者ID、使用的登陆终端、上线时间、呆滞时间、CPU占用，以及做了些什么。使用权限为所有用户。

格式

```
who - [husfV] [user]
```

主要参数

-h：不要显示标题列。

-u：不要显示使用者的动作/工作。

-s：使用简短的格式来显示。

-f：不要显示使用者的上线位置。

-V：显示程序版本。

说明

该命令主要用于查看当前在线上的用户情况。

如果用户想和其它用户建立即时通信，比如使用talk命令，那么首先要确定的就是该用户确实在线上,不然talk进程就无法建立起来。

又如，系统管理员希望监视每个登录的用户此时此刻的所作所为，也要使用who命令。who命令应用起来非常简单，可以比较准确地掌握用户的情况,所以使用非常广泛。

linux基础(八)----linux命令系统学习----其它命令

其它命令：tar、unzip、gunzip、unarj、mttools、man、unendcode、uudecode。

在前面几讲中，我们把Linux命令按照在系统中的作用分成几个部分分别予以介绍。但是，还有一些命令不好划分，然而学习它们同样是比较重要的。

tar

作用

tar命令是Unix/Linux系统中备份文件的可靠方法，几乎可以工作于任何环境中，它的使用权限是所有用户。

格式

tar [主选项+辅选项]文件或目录

主要参数

使用该命令时，主选项是必须要有的，它告诉tar要做什么事情，辅选项是辅助使用的，可以选用。

主选项：

- c 创建新的档案文件。如果用户想备份一个目录或是一些文件，就要选择这个选项。
- r 把要存档的文件追加到档案文件的末尾。例如用户已经做好备份文件，又发现还有一个目录或是一些文件忘记备份了，这时可以使用该选项，将忘记的目录或文件追加到备份文件中。
- t 列出档案文件的内容，查看已经备份了哪些文件。
- u 更新文件。就是说，用新增的文件取代原备份文件，如果在备份文件中找不到要更新的文件，则把它追加到备份文件的最后。
- x 从档案文件中释放文件。

辅助选项：

- b 该选项是为磁带机设定的，其后跟一数字，用来说明区块的大小，系统预设值为20（20×512 bytes）。
- f 使用档案文件或设备，这个选项通常是必选的。
- k 保存已经存在的文件。例如把某个文件还原，在还原的过程中遇到相同的文件，不会进行覆盖。
- m 在还原文件时，把所有文件的修改时间设定为现在。
- M 创建多卷的档案文件，以便在几个磁盘中存放。
- v 详细报告tar处理的文件信息。如无此选项，tar不报告文件信息。
- w 每一步都要求确认。
- z 用gzip来压缩/解压缩文件，加上该选项后可以将档案文件进行压缩，但还原时也一定要使用该选项进行解压缩。

应用说明

tar是Tape Archive（磁带归档）的缩写，最初设计用于将文件打包到磁带上。

如果下载过Linux的源代码，或许已经碰到过tar文件，
请注意，不要忘了Linux是区分大小写的。例如，tar命令应该总是以小写的形式执行。
命令行开关可以是大写、小写或大小写的混合。例如，-t和-T执行不同的功能。
文件或目录名称可以混合使用大小写，而且就像命令和命令行开关一样是区分大小写的。

应用实例

tar是一个命令行的工具，没有图形界面。

使用Konsole打开一个终端窗口，接下来是一个简单的备份命令（在/temp目录中创建一个back.tar的文件，/usr目录中所有内容都包含在其中）。

```
$tar cvf - /usr > /temp/back.tar
```

另外，tar命令支持前面第三讲中讲过的crontab命令，可以用crontab工具设置成基于时间的有规律地运行。

例如，每晚6点把/usr目录备份到hda—第一个IDE接口的主驱动器（总是位于第一个硬盘）中，只要将下面语句添加到root的crontab中即可：

```
$00 06 * * * tar cvf /dev/hda1/usrfiles.tar - /usr
```

一般情况下，以下这些目录是需要备份的：

- ◆/etc 包含所有核心配置文件，其中包括网络配置、系统名称、防火墙规则、用户、组，以及其它全局系统项。
- ◆ /var 包含系统守护进程（服务）所使用的信息，包括DNS配置、DHCP租期、邮件缓冲文件、HTTP服务器文件、dB2实例配置等。
- ◆/home 包含所有默认用户的主目录，包括个人设置、已下载的文件和用户不希望失去的其它信息。
- ◆/root 根（root）用户的主目录。
- ◆/opt 是安装许多非系统文件的地方。IBM软件就安装在这里。OpenOffice、JDK和其它软件在默认情况下也安装在这里。

有些目录是可以不备份的：

- ◆ /proc 应该永远不要备份这个目录。它不是一个真实的文件系统，而是运行内核和环境的虚拟化视图，包括诸如/proc/kcore这样的文件，这个文件是整个运行内存的虚拟视图。备份这些文件只是在浪费资源。
- ◆/dev 包含硬件设备的文件表示。如果计划还原到一个空白的系统，就可以备份/dev。然而，如果计划还原到一个已安装的Linux系统，那么备份/dev是没有必要的。

unzip

作用

unzip命令位于/usr/bin目录中，它们和MS DOS下的pkzip、pkunzip及MSWindows中的Winzip软件功能一样，将文件压缩成.zip文件，以节省硬盘空间，当需要的时候再将压缩文件用unzip命令解开。该命令使用权限是所有用户。

格式

`unzip [-cflptuvz][-agCjLMnoqsVX][-P][.zip文件][文件][-d][-x]`

主要参数

-c：将解压缩的结果显示到屏幕上，并对字符做适当的转换。

-f：更新现有的文件。

-l：显示压缩文件内所包含的文件。

-p：与-c参数类似，会将解压缩的结果显示到屏幕上，但不会执行任何的转换。

-t：检查压缩文件是否正确。

-u：与-f参数类似，但是除了更新现有的文件外，也会将压缩文件中的其它文件解压缩到目录中。

-v：执行是时显示详细的信息。

-z：仅显示压缩文件的备注文字。

-a：对文本文件进行必要的字符转换。

-b：不要对文本文件进行字符转换。

-C：压缩文件中的文件名称区分大小写。

-j：不处理压缩文件中原有的目录路径。

-L：将压缩文件中的全部文件名改为小写。

-M：将输出结果送到more程序处理。

-n：解压缩时不要覆盖原有的文件。

-o：不必先询问用户，unzip执行后覆盖原有文件。

-P：使用zip的密码选项。

-q：执行时不显示任何信息。

-s：将文件名中的空白字符转换为底线字符。

-V：保留VMS的文件版本信息。

-X：解压缩时同时回存文件原来的UID/GID。

[.zip文件]：指定.zip压缩文件。

[文件]：指定要处理.zip压缩文件中的哪些文件。

-d：指定文件解压缩后所要存储的目录。

-x：指定不要处理.zip压缩文件中的哪些文件。

-Z unzip：-Z等于执行zipinfo指令。在Linux中，还提供了一个叫zipinfo的工具，能够察看zip压缩文件的详细信息。unzip最新版本是5.50。

gunzip

作用

gunzip命令作用是解压文件，使用权限是所有用户。

格式

`gunzip [-acfhLnNqrtvV][-s][文件...]`

或者

`gunzip [-acfhLnNqrtvV][-s][目录]`

主要参数

-a或--ascii：使用ASCII文字模式。

-c或--stdout或--to-stdout：把解压后的文件输出到标准输出设备。

-f或-force：强行解开压缩文件，不理睬文件名称或硬连接是否存在，以及该文件是否为符号连接。

-h或--help：在线帮助。

-l或--list：列出压缩文件的相关信息。

-L或--license：显示版本与版权信息。

-n或--no-name：解压缩时，若压缩文件内含有原来的文件名称及时间戳记，则将其忽略不予处理。

-N或--name：解压缩时，若压缩文件内含有原来的文件名称及时间戳记，则将其回存到解开的文件上。

-q或--quiet：不显示警告信息。

-r或--recursive：递归处理，将指定目录下的所有文件及子目录一并处理。

-S或--suffix：更改压缩字尾字符串。

-t或--test：测试压缩文件是否正确无误。

-v或--verbose：显示指令执行过程。

-V或--version：显示版本信息。

说明

gunzip是个使用广泛的解压缩程序，它用于解开被gzip压缩过的文件，这些压缩文件预设最后的扩展名为“.gz”。事实上，gunzip就是gzip的硬连接，因此不论是压缩或解压缩，都可通过gzip指令单独完成。gunzip最新版本是1.3.3。

unarj

作用

unarj解压缩格式为.arj格式的文件，使用权限是所有用户。

格式

`unarj [eltx][.arj压缩文件]`

主要参数

e：解压缩.arj文件。

l：显示压缩文件内所包含的文件。

t：检查压缩文件是否正确。

x：解压缩时保留原有的路径。

说明

带有.arj扩展名的文件是由用于VIS DOS和Windows的ARJ实用程序创建的。

因为ARJ是一种不能免费获得源代码的共享件程序，所以在Linux平台上几乎不存在与其功能匹配的工具，要解压缩.ark文件，就要使用unarj实用程序。

unarj比ARJ慢，能力也不如ARJ但至少能够顺利地抽取大多数.ark文件。unarj只能将文件抽取到当前的工作目录、列出档案内容，或者测试档案。从ARJ Software的站点或携带所需Linux发行版的FTP服务器上可以下载unarj源码。

另外，unarj通常是基本Linux发行版的一部分，因此可以在主要发行版本的CD-ROM上找到它。

如果需要可到所有Linux发行版链接的列表下载，ARJ软件网址为<http://www.arjsoft.com>,

ARJ的下载页面为<http://www.arjsoft.com/files.htm>。unarj最新版本是2.65，注意unarj选项不是以减号。)开头的。

mtools

作用

mtools实际上是一个命令集合，是DOS文件系统的工具程序，它可以模拟许多DOS命令，使用起来非常方便。使用权限是所有用户。

Linux系统提供了一组称为mtools的可移植工具，可以让用户轻松地、从标准的DOS软盘上读、写文件和目录。它们对DOS和Linux环境之间交换文件非常有用。

mtools的使用非常简单，如果想把软盘里所有的文件都拷贝到硬盘上，那么就可以执行以下命令：

```
mcopy a:.
```

也就是说，只需要在相应的DOS命令之前加上一个字母“m”，就可以完成对应的功能了。

一般Linux发行版本中都有这个软件，可以使用下面命令检查一下。

```
rpm -qa|grep mtools
```

如果没有安装，也没有关系，可以从网上下载(<http://mtools.linux.lu/>)一个最新版本安装。目前可供下载的最新mtools版本是3.9.9，下载链接为

<http://mtools.linux.lu/mtools-3.9.9-3.i386.rpm>。下载后安装一下即可。

包括的命令

mcd 目录名：改变MS DOS下的目录。

mcopy 源文件 目标文件：在MS DOS和Unix之间复制文件。

mdel 文件名：删除MS DOS下的文件。

mdir 目录名：显示MS DOS下的目录。

mformat 驱动器号：在低级格式化的软盘上创建MS DOS文件系统。

rnlabel 驱动器号：产生MS DOS下的卷标。

mmd 目录名：建立MS DOS下的目录。

mrd 目录名：删除MS DOS下的目录。

mren 源文件目标文件：重新命名已存在的MS DOS文件。

mtype 文件名：显示MS DOS文件的内容。

请注意，这些命令和对应的MS DOS命令非常相似。在mtools命令中，“/”和“\”是可以混用的。因为文件列表的是DOS系统下的文档，对大小写并不敏感，所以“CDE”和“cde”在这里是一样的。

应用实例

(1)如果把软盘进行快速格式化，可以使用命令mformat：

mformat A：

mtools当初发展的目的是用来处理DOS文件系统的，所以只能用在FAT文件格式的分区上。

需要注意的是，如果用mount命令来挂载了FAT16/32分区，那么就不能使用mtools的指令来处理这些分区上的文件。

这是因为一旦FAT16/32分区挂到了Linux文件目录下，Linux就会将其视为文件系统本身的一部分，这时如果要对它操作就必须使用Linux本身所附带的指令集。

(2)将DOS盘上的文件htca.c复制到当前目录下，并用ls命令进行验证。

```
$ mcopy a:\htca.c
```

```
$ ls -l htca.c
```

```
-rw-r--r-- 1 xxq xxq 27136 Jan 1 01:80 htca.c
```

man

作用

man命令用来提供在线帮助，使用权限是所有用户。

在Linux系统中存储着一部联机使用的手册，以供用户在终端上查找。使用man命令可以调阅其中的帮助信息，非常方便和实用。

格式

man命令名称

```
man [-acdfhkKtwW] [-m system] [-p string] [-C config_file] [-M path] [-P pager] [-S section_list] [section]  
name ...
```

参数

-C config_file：指定设定文件man.conf，缺省值是/etc/man.conf。

-M path：指定了联机手册的搜寻路径，如果没有指定则使用环境变数MANPATH的设定；如果没有使用MANPATH，则会使用/usr/lib/man.conf内的设定；如果MANPATH是空字符串，则表示使用缺省值。

-P pager：指定使用何种pager.man会优先使用此选项设定，然后是依环境变数MANPAGER设定，然后是环境

变数PAGER；man缺省使用/usr/bin/less -is。

-S section_list man：所搜寻的章节列表(以冒号分隔)，此选项会覆盖环境变数MANSECT的设定。

-a man：缺省情况是在显示第一个找到的手册之后，就会停止搜寻，使用此选项会强迫man继续显示所有符合name的联机手册。

-c：即使有最新的cat page，也继续对联机手册重新作排版，本选项在屏幕的行列数改变时或已排版的联机手册损坏时特别有意义。

-d：不要真的显示联机手册，只显示除错讯息。

-D：同时显示联机手册与除错讯息。

-h：显示求助讯息然后结束程式。

-K：对所有的联机手册搜寻所指定的字串。请注意，本功能回应速度可能很慢，如果指定section（区域）会对速度有帮助。

-m system：依所指定的system名称而指定另一组的联机手册。

man：是manual（手册）的缩写。在输入命令有困难时，可以立刻得到这个文档。例如,如果使用ps命令时遇到困难，可以输入man ps得到帮助信息，此时会显示出ps的手册页（man page）。

由于手册页man page是用less程序来看的(可以方便地使屏幕上翻和下翻),所以在man page里可以使用less的所有选项。

less中比较重要的功能键有:

[q] 退出；

[Enter] 一行行地下翻；

[Space] 一页页地下翻；

上翻一页；

[/] 后跟一个字符串和[Enter]来查找字符串；

[n] 发现上一次查找的下一个匹配。

阅读手册页

手册页在很少的空间里提供了很多的信息，这里简单介绍一下大多数手册页中都有的部分内容。Linux手册页主要有九个部分：用户指令、系统调用、程序库、设备说明、文件格式、游戏、杂项、系统指令、内核。

应用实例

Linux命令中有一些基础的、重要的命令，例如ps、find、cat和ls等。

下面来举一个综合应用的例子，由此可以看出man的地位在Linux中可谓至关重要。

但是，man所显示的信息却不是普通的文本，如果直接将这些文字重定向到一个文本文件，就会发现在man中高亮显示的文字就变成了两个，而且有不计其数的制表符，使打印、编辑都变得非常不便。

不过，使用下面这样一条语句就能得到ps命令打印。

```
# man ps | col -b | lpr
```

这条命令同时运用了输出重定向和管道两种技巧，作用是将ps命令的帮助信息可以直接打印出来。

更多的Man文件可以查看Linux Man

unencode

作用

unencode命令可以把一个二进制文件表编码为一个文本文件，使用权限是所有用户。

格式

unencode [-hv] [源文件]目标文件

主要参数

- h：列出指令使用格式(help)。
- v：列出版本信息。

应用说明

unencode指令可以将二进制文件转化成可使用电子邮件发送的ASCII编码形式。

unencode编码后的资料都以 begin开始，以end作为结束，且通常其中的每一行的开始均为“M”，中间部分是编码过的文件，编码后的文件比源文件要大一些。

uudecode

作用

uudecode命令用来将uuencode编码后的档案还原，uudecode只会将begin与end标记之间的编码资料还原，程序会跳过标记以外的资料。它的使用权限为所有用户。

格式

uuencode [-hv] [file1 ...]

主要参数

- h：列出指令使用格式(help)。
- v：列出版本信息。

应用实例

使用下面命令一次还原几个文件：

uuencode file1.uud file2.uud file3.uud

linux基础(九)----linux性能监测

Linux下查看内存,CPU信息

内存信息

使用free查看内存信息：

```
$ free -m
```

	total	used	free	shared	buffers	cached
Mem:	222	136	86	0	29	60
-/+ buffers/cache:		47	175			
Swap:	1905	0	1905			

- total：总共的内存大小
- used：已经被使用的内存
- free：空闲的内存
- shared：共享的内存大小
- buffers：用来做缓冲的内存
- cached：用来做cache的内存

Mem这行是以操作系统的角度去看待内存的使用，可以看到我们总共的内存是222M(total1)，使用了136M(used1)，有86M的空闲(free1)，29M的缓冲(buffers1)，60M的缓存(cached1)。

-/+ buffers/cache这行是以应用程序的角度去看待内存的使用，对于应用来说buffers和cached的内存是就是空闲的内存，在需要的时候是 可以直接拿来用的，所以：

```
used = used1 - buffers1 - shared1 = 136 - 29 - 60 = 47,  
free = free1 + buffers1 + shared1 = 86 + 29 + 60 = 175。
```

Swap这行是交换区的使用情况，如果used很大的话，说明内存不够用了。

PS:跑的虚拟机，内存有些小

CPU信息

Linux系统中的CPU信息存在于/proc/cpuinfo文件中，如果想了解全部的信息，可以直接查看这个文件。
有多少个物理CPU？

```
cat /proc/cpuinfo | grep 'physical id' | sort | uniq | wc -l
```

```
[root@localhost ~]# cat /proc/cpuinfo | grep 'physical id' | sort | uniq | wc -l
2
```

有多少个虚拟CPU？

```
cat /proc/cpuinfo | grep ^processor | sort | uniq | wc -l
```

CPU是几个核心的？

```
cat /proc/cpuinfo | grep 'cpu cores' | uniq
```

如何查看每个CPU的使用情况？执行top指令，然后按1就可以看到CPU的使用情况了。

用TOP来作性能监控

```
[root@localhost ~]# top
top - 09:51:45 up 14 days, 17:53, 1 user, load average: 2.06, 2.01, 2.00
Tasks: 482 total, 1 running, 481 sleeping, 0 stopped, 0 zombie
Cpu(s): 8.3%us, 0.1%sy, 0.0%ni, 91.6%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 65938736k total, 41716412k used, 24222324k free, 145964k buffers
Swap: 33038328k total, 529744k used, 32508584k free, 9728256k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
9830	root	20	0	30.1g	29g	2988	S	199.1	46.6	1683:16	perl
364	root	39	19	0	0	0	S	1.3	0.0	229:51.18	kipmi0
6748	mysql	20	0	4328m	257m	4328	S	0.7	0.4	32:00.55	mysqld
100	root	20	0	0	0	0	S	0.3	0.0	3:10.15	events/1
1	root	20	0	19228	288	132	S	0.0	0.0	0:02.34	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.08	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.94	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:23.93	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:01.29	watchdog/0

在系统维护的过程中，随时可能有需要查看 CPU 使用率，并根据相应信息分析系统状况的需要。在 CentOS 中，可以通过 top 命令来查看 CPU 使用状况。运行 top 命令后，CPU 使用状态会以全屏的方式显示，并且会处在对话的模式 -- 用基于 top 的命令，可以控制显示方式等等。退出 top 的命令为 q（在 top 运行中敲 q 键一次）。

操作实例:

在命令行中输入 “top”

即可启动 top

top 的全屏对话模式可分为3部分：系统信息栏、命令输入栏、进程列表栏。

最上部的系统信息栏

第一行(top)

- “09:51:45” 为系统当前时刻；
- “up 14 days,17:53” 为系统启动后到现在的运作时间；

“1 users” 为当前登录到系统的用户，更确切的说是登录到用户的终端数 -- 同一个用户同一时间对系统多个终端的连接将被视为多个用户连接到系统，这里的用户数也将表现为终端的数目；

“load average” 为当前系统负载的平均值，后面的三个值分别为1分钟前、5分钟前、15分钟前进程的平均数，一般的可以认为这个数值超过 CPU 数目时，CPU 将比较吃力的负载当前系统所包含的进程；

第二行 (Tasks)

“482 total” 为当前系统进程总数；

“1 running” 为当前运行中的进程数；

“481 sleeping” 为当前处于等待状态中的进程数；

“0 stoped” 为被停止的系统进程数；

“0 zombie” 为被复原的进程数；

第三行 (Cpus)

分别表示了 CPU 当前的使用率；

第四行 (Mem)

分别表示了内存总量、当前使用量、空闲内存量、以及缓冲使用中的内存量；

第五行 (Swap)

表示类别同第四行 (Mem)，但此处反映着交换分区 (Swap) 的使用情况。通常，交换分区 (Swap) 被频繁使用的情况，将被视作物理内存不足而造成的。

第二部分中间部分的内部命令提示栏

top 运行中可以通过 top 的内部命令对进程的显示方式进行控制。内部命令如下表：

s - 改变画面更新频率

l - 关闭或开启第一部分第一行 top 信息的表示

t - 关闭或开启第一部分第二行 Tasks 和第三行 Cpus 信息的表示

m - 关闭或开启第一部分第四行 Mem 和 第五行 Swap 信息的表示

N - 以 PID 的大小的顺序排列表示进程列表 (第三部分后述)

P - 以 CPU 占用率大小的顺序排列进程列表 (第三部分后述)

M - 以内存占用率大小的顺序排列进程列表 (第三部分后述)

h - 显示帮助

n - 设置在进程列表所显示进程的数量

q - 退出 top

第三部分最下部分的进程列表栏

以 PID 区分的进程列表将根据所设定的画面更新时间定期的更新。

进程信息区统计信息区域的下方显示了各个进程的详细信息。首先来认识一下各列的含义。

序号 列名 含义

a PID 进程id

b PPID 父进程id

c RUSER Real user name

d UID 进程所有者的用户id

e USER 进程所有者的用户名

f GROUP 进程所有者的组名

g TTY 启动进程的终端名。不是从终端启动的进程则显示为？

h PR 优先级

i NI nice值。负值表示高优先级，正值表示低优先级

j P 最后使用的CPU，仅在多CPU环境下有意义

k %CPU 上次更新到现在的CPU时间占用百分比

l TIME 进程使用的CPU时间总计，单位秒

m TIME+ 进程使用的CPU时间总计，单位1/100秒

n %MEM 进程使用的物理内存 百分比

o VIRT 进程使用的虚拟内存总量，单位kb。VIRT=SWAP+RES

p SWAP 进程使用的虚拟内存中，被换出的大小，单位kb。

q RES 进程使用的、未被换出的物理内存大小，单位kb。RES=CODE+DATA

r CODE 可执行代码占用的物理 内存大小，单位kb

s DATA 可执行代码以外的部分(数据 段+栈)占用的物理 内存大小，单位kb

t SHR 共享内存大小，单位kb

u nFLT 页面错误次数

v nDRT 最后一次写入到现在，被修改过的页面数。

w S 进程状态。

D =不可中断的睡眠状态

R =运行

S =睡眠

T =跟踪/停止

Z =僵尸进程

x COMMAND 命令名/命令行

y WCHAN 若该进程在睡眠，则显示睡眠中的系统函数名

z Flags 任务标志，参考 sched.h

默认情况下仅显示比较重要的 PID、USER、PR、NI、VIRT、RES、SHR、S、%CPU、%MEM、TIME+、

COMMAND 列。

可以通过下面的快捷键来更改显示内容。

更改显示内容通过 f 键可以选择显示的内容。按 f 键之后会显示列的列表，按 a-z 即可显示或隐藏对应的列，最后按回车键确定。

按 o 键可以改变列的显示顺序。按小写的 a-z 可以将相应的列向右移动，而大写的 A-Z 可以将相应的列向左移动。最后按回车键确定。

按大写的 F 或 O 键，然后按 a-z 可以将进程按照相应的列进行排序。而大写的 R 键可以将当前的排序倒转。

快捷键

top命令使用过程中，还可以使用一些交互的命令来完成其它参数的功能。这些命令是通过快捷键启动的。

< 空格 >：立刻刷新。

P：根据CPU使用大小进行排序。

T：根据时间、累计时间排序。

q：退出top命令。

m：切换显示内存信息。

t：切换显示进程和CPU状态信息。

c：切换显示命令名称和完整命令行。

M：根据使用内存大小进行排序。

W：将当前设置写入 ~/.toprc 文件中。这是写top配置文件的推荐方法。

可以看到，top命令是一个功能十分强大的监控系统的工具，对于系统管理员而言尤其重要。但是，它的缺点是会消耗很多系统资源。

应用实例

使用top命令可以监视指定用户，缺省情况是监视所有用户的进程。如果想查看指定用户的情况，在终端中按 “U” 键，然后输入用户名，系统就会切换为指定用户的进程运行界面。

作用

free命令用来显示内存的使用情况，使用权限是所有用户。

格式

```
free [ - b - k - m ] [ - o ] [ - s delay ] [ - t ] [ - V ]
```

主要参数

- b - k - m：分别以字节（KB、MB）为单位显示内存使用情况。
- s delay：显示每隔多少秒数来显示一次内存使用情况。
- t：显示内存总和列。

- o : 不显示缓冲区调节列。

应用实例

free命令是用来查看内存使用情况的主要命令。和top命令相比，它的优点是使用简单，并且只占用很少的系统资源。通过 - S参数可以使用free命令不间断地监视有多少内存在使用，这样可以把它当作一个方便实时监控器。

```
#free -b -s5
```

使用这个命令后终端会连续不断地报告内存使用情况（以字节为单位），每5秒更新一次。

linux基础(十)----linux网络配置详细步骤---桥接模式和两台机子的远程通信

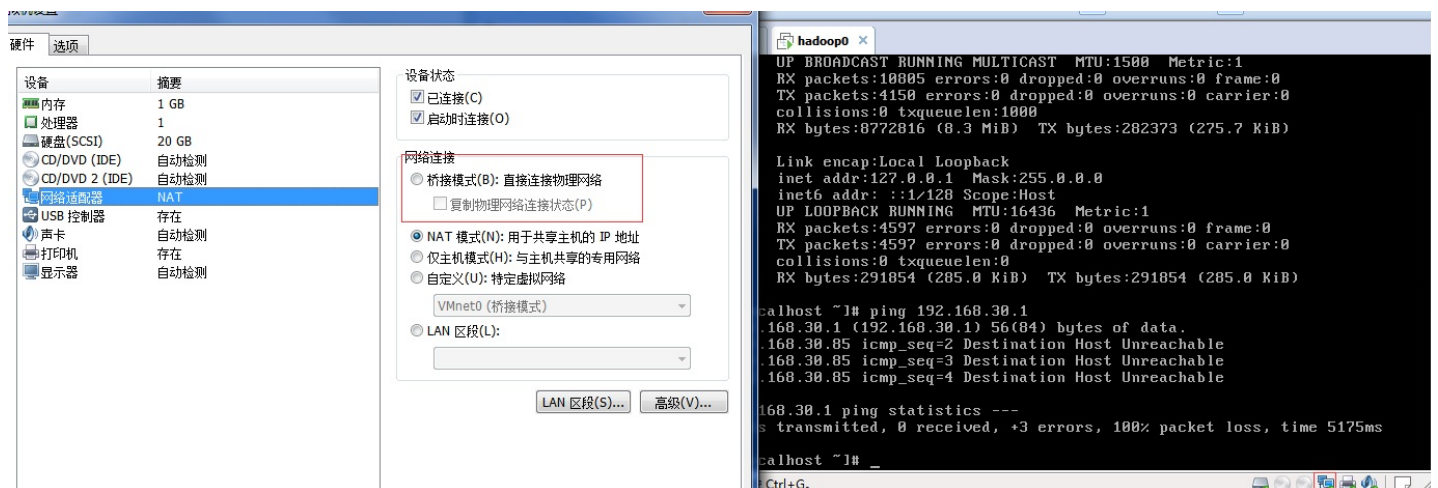
本篇记录如何给linux设置桥接网络配置 到达可以linux系统可以上网的目的。

配置linux网络配置

现在我有一台虚拟的linux如下: (可参考[hadoop基础虚拟机\(二\)---虚拟机安装以及安装linux系统](#))



虚拟机网络模式设置桥接



编辑配置文件

启动登录后开始编辑网络配置文件

```
$vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
[joe@localhost home]$ vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

需要改动的:

BOOTPROTO=static

默认是自动获取dhcp,这里我们改成static静态

需要增加的:

IPADDR=192.168.30.85

这个地址自己设置 因为我的windows系统是在30网段 所以 这里也用30网段, 这里才能连通。

NETMASK=255.255.255.0

网关最好也跟windows系统的保持一致

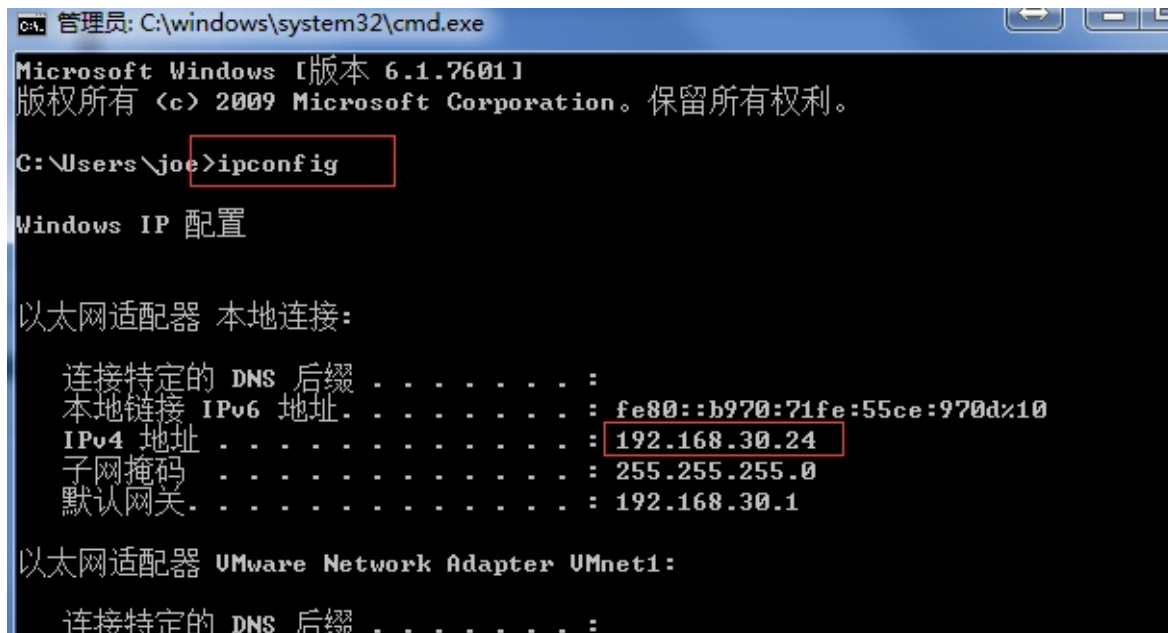
修改好之后保存退出。

使用root用户 重新启动网络配置

```
/etc/init.d/network restart
```

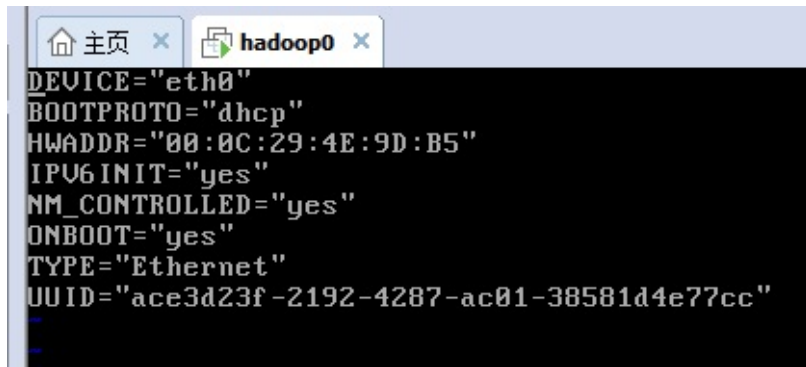
就可以上网了

windows中cmd查看网络配置

A screenshot of a Windows Command Prompt window. The title bar reads "管理员: C:\windows\system32\cmd.exe". The window content shows the output of the 'ipconfig' command. The text is as follows:
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
C:\Users\joe>ipconfig
Windows IP 配置
以太网适配器 本地连接:
 连接特定的 DNS 后缀 :
 本地连接 IPv6 地址. : fe80::b970:71fe:55ce:970d%10
 IPv4 地址 : 192.168.30.24
 子网掩码 : 255.255.255.0
 默认网关. : 192.168.30.1
以太网适配器 VMware Network Adapter VMnet1:
 连接特定的 DNS 后缀 :
In the screenshot, the command 'ipconfig' and the IP address '192.168.30.24' are highlighted with red rectangles.

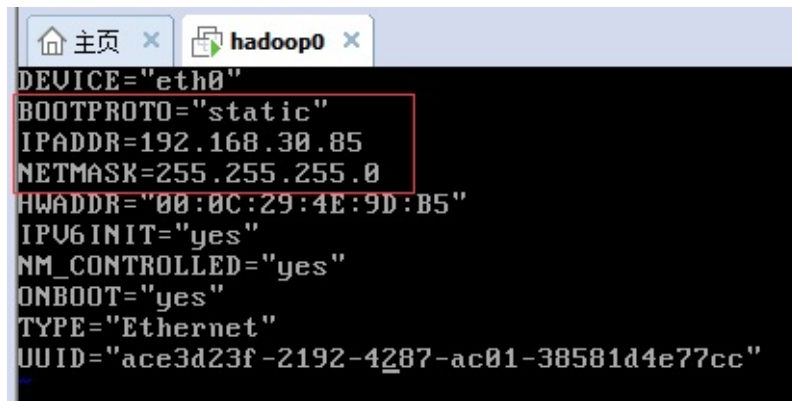
linux中的网段最好与windows中本地的相同,即前三个数字相同。方便我们后面测试网络,一般常见的是192.168.1.X 或者192.168.0.X

linux原来的网络配置



```
DEVICE="eth0"
BOOTPROTO="dhcp"
HWADDR="00:0C:29:4E:9D:B5"
IPV6INIT="yes"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="ace3d23f-2192-4287-ac01-38581d4e77cc"
```

修改后的linux网络配置



```
DEVICE="eth0"
BOOTPROTO="static"
IPADDR=192.168.30.85
NETMASK=255.255.255.0
HWADDR="00:0C:29:4E:9D:B5"
IPV6INIT="yes"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="ace3d23f-2192-4287-ac01-38581d4e77cc"
```

测试网络

我们可以使用

curl 来测试访问网址链接 也可以用ping 的方式 测试是否能ping 通，能ping 通说明已经能上网了。

我分别 ping 百度以及windows系统。

ping www.baidu.com

ping 192.168.30.24

结果如下:

```

[home 主页 x] [hadoop0 x]
[joe@localhost root]$ ping www.baidu.com
PING www.a.shifen.com (180.97.33.107) 56(84) bytes of data.
64 bytes from 180.97.33.107: icmp_seq=1 ttl=128 time=43.8 ms
64 bytes from 180.97.33.107: icmp_seq=2 ttl=128 time=41.5 ms
64 bytes from 180.97.33.107: icmp_seq=3 ttl=128 time=41.2 ms
64 bytes from 180.97.33.107: icmp_seq=4 ttl=128 time=41.6 ms
64 bytes from 180.97.33.107: icmp_seq=5 ttl=128 time=41.5 ms
64 bytes from 180.97.33.107: icmp_seq=6 ttl=128 time=41.6 ms
64 bytes from 180.97.33.107: icmp_seq=7 ttl=128 time=41.4 ms
^C
--- www.a.shifen.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6334ms
rtt min/avg/max/mdev = 41.277/41.866/43.882/0.866 ms
[joe@localhost root]$ ping 192.168.30.24
PING 192.168.30.24 (192.168.30.24) 56(84) bytes of data.
64 bytes from 192.168.30.24: icmp_seq=1 ttl=128 time=1.03 ms
64 bytes from 192.168.30.24: icmp_seq=2 ttl=128 time=0.692 ms
64 bytes from 192.168.30.24: icmp_seq=3 ttl=128 time=0.714 ms
64 bytes from 192.168.30.24: icmp_seq=4 ttl=128 time=0.801 ms
64 bytes from 192.168.30.24: icmp_seq=5 ttl=128 time=0.743 ms
^C
--- 192.168.30.24 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4918ms
rtt min/avg/max/mdev = 0.692/0.797/1.039/0.130 ms
[joe@localhost root]$ _

```

反过来 用windows系统的cmd界面 ping 192.168.30.85 (linux配置文件中设置的地址)

```

C:\Users\joe>ping 192.168.30.85

正在 Ping 192.168.30.85 具有 32 字节的数据:
来自 192.168.30.85 的回复: 字节=32 时间=54ms TTL=64
来自 192.168.30.85 的回复: 字节=32 时间=19ms TTL=64
来自 192.168.30.85 的回复: 字节=32 时间=14ms TTL=64
来自 192.168.30.85 的回复: 字节=32 时间=15ms TTL=64

192.168.30.85 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间<以毫秒为单位>:
        最短 = 14ms, 最长 = 54ms, 平均 = 25ms

C:\Users\joe>

```

如果能ping通 则说明 可以正常上网了。

可能遇到的问题

编辑网络配置文件时无操作权限

warning : changing a readonly file

'readonly' option is set (add ! to override)

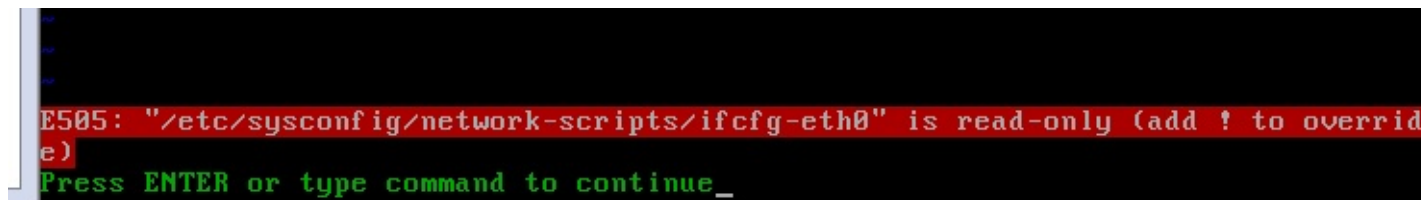
Can't open file writing

如图

```

"/etc/sysconfig/network-scripts/ifcfg-eth0"
"/etc/sysconfig/network-scripts/ifcfg-eth0" E212: Can't open file for writing
Press ENTER or type command to continue_

```



解决方法:

使用joe用户帐号发现给不了权限 也不在sudo的执行分组中

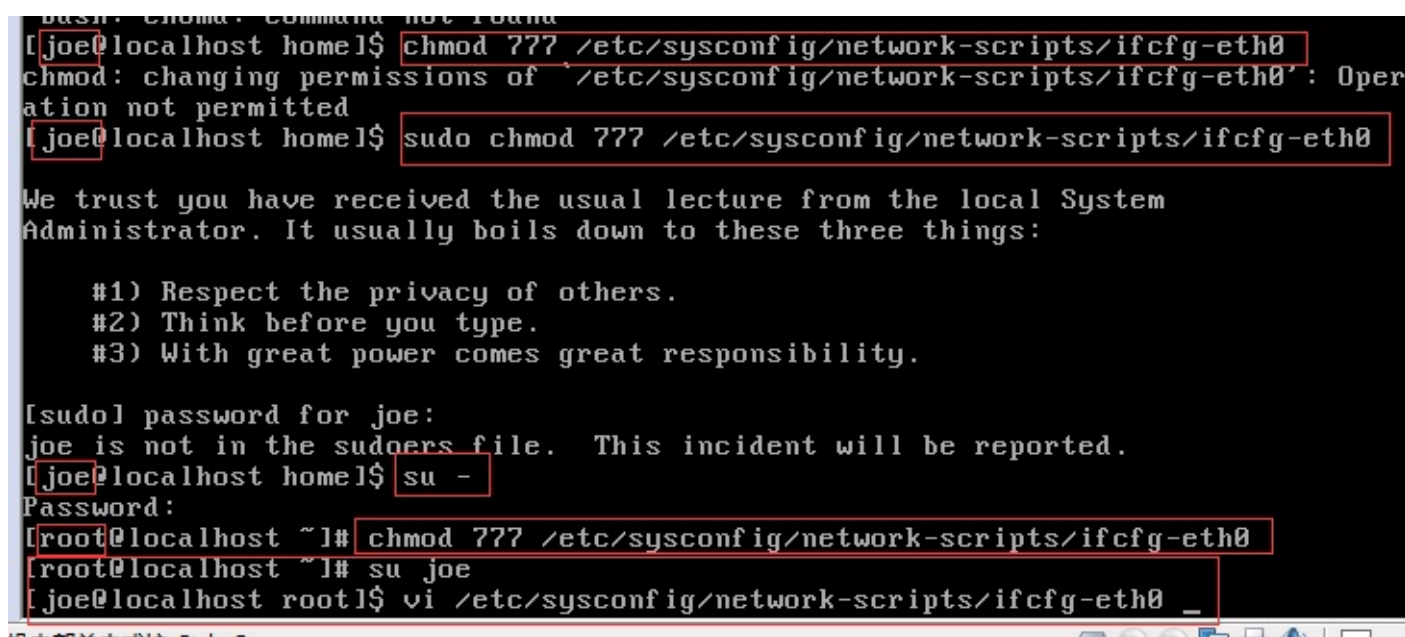
所以先用

su -

切换到root用户

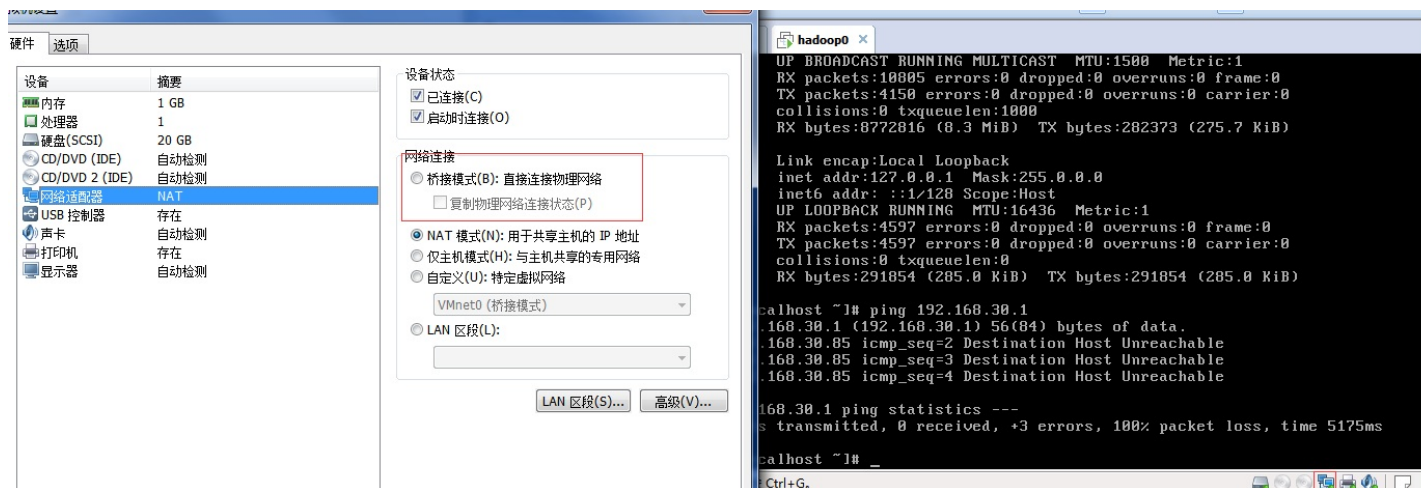
用root用户 chmod修改配置文件的权限为可写 我这里直接给最大的权限777

如图



网段已对应但是ping不通

注意虚拟机需要把网络模式更改为桥接。



远程登录

我们这里使用windows用过telnet远程登录linux系统。

安装windows的telnet功能

windows测试telnet

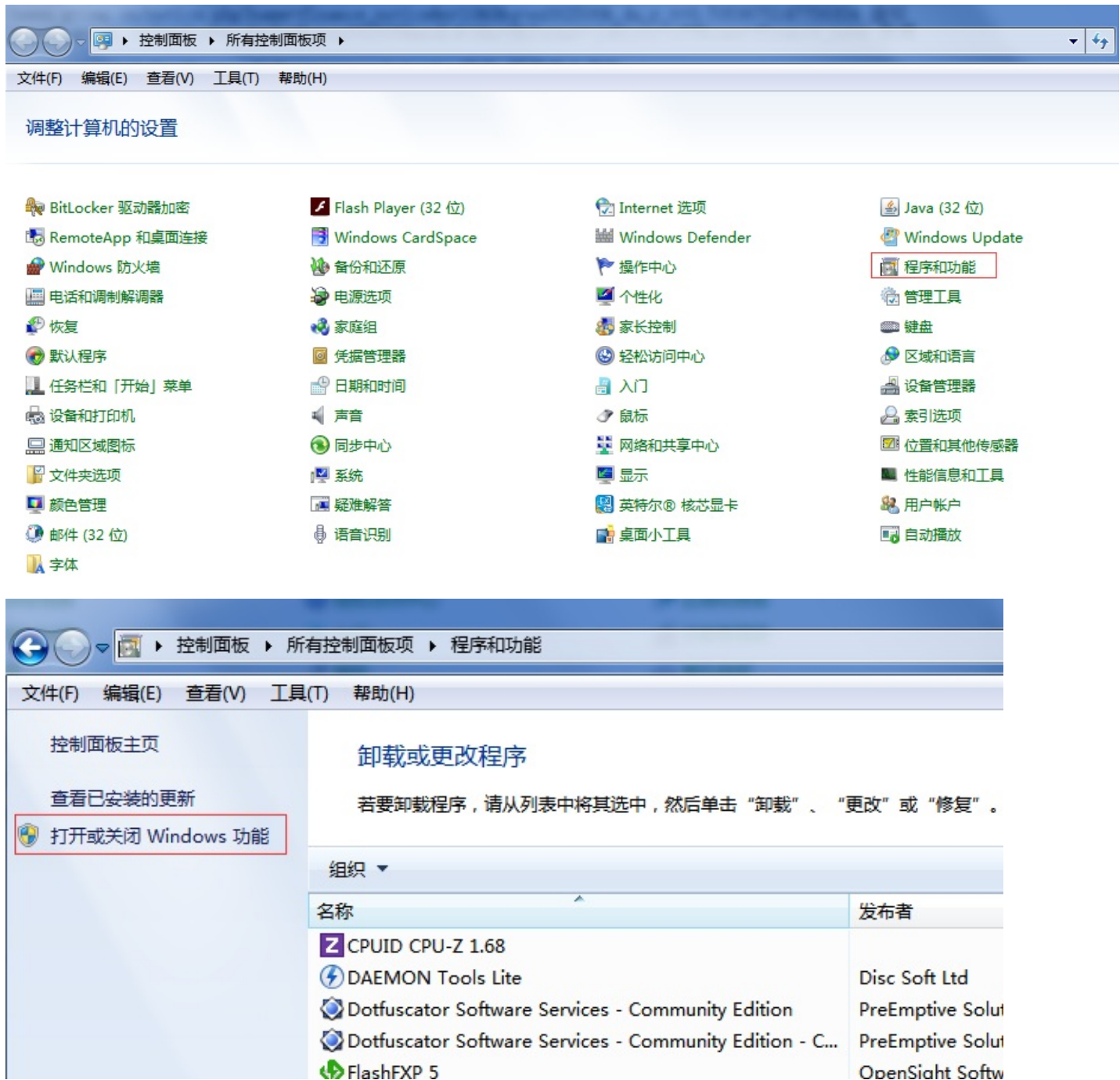
第一次使用不确定是否已经安装telnet可以使用 telnet 127.0.0.1 进行测试。

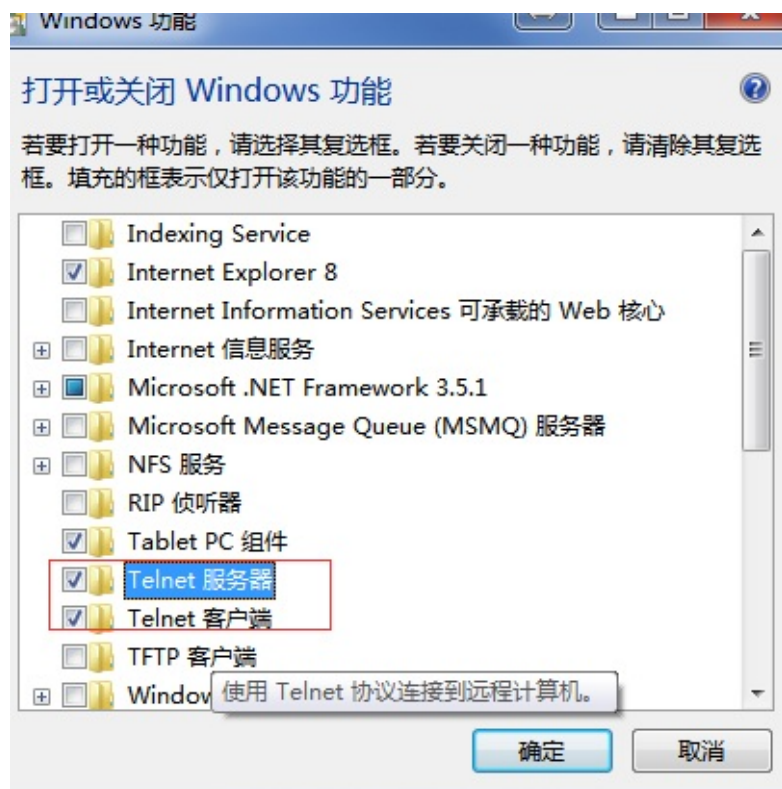
```
C:\Users\joe>telnet 127.0.0.1
正在连接127.0.0.1...无法打开到主机的连接。 在端口 23: 连接失败

C:\Users\joe>
```

打开windows的telnet服务

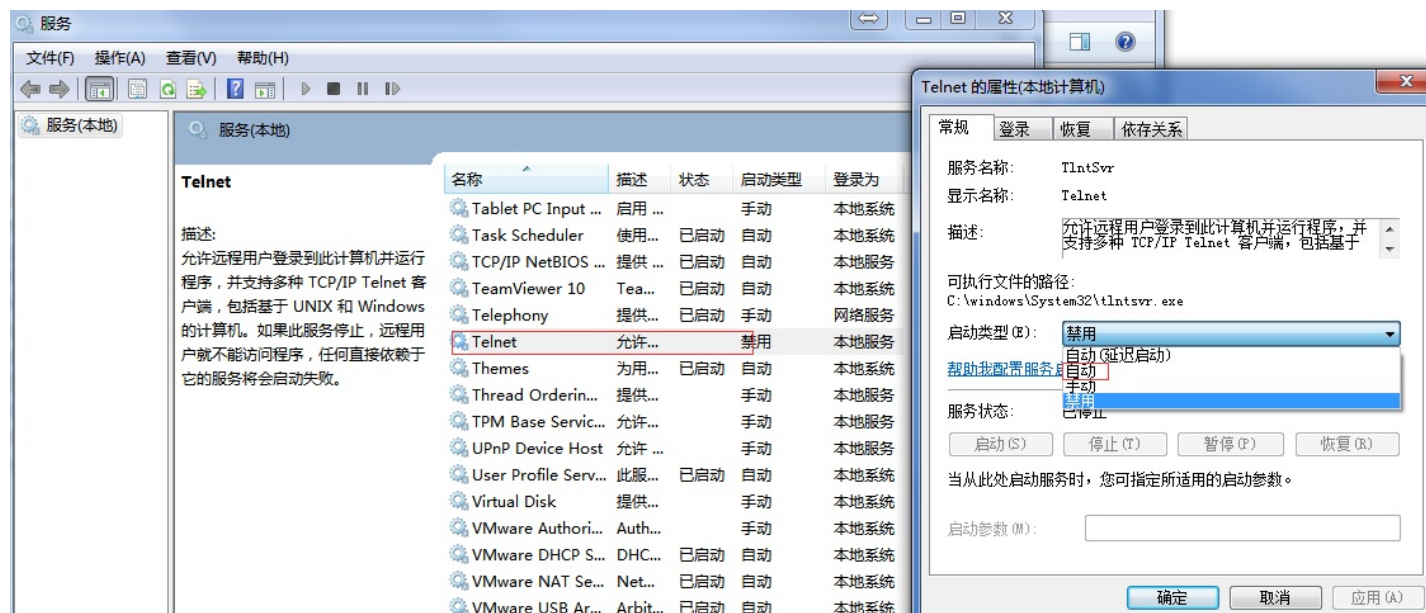
(win7)打开控制面板->程序和功能->打开或关闭windows功能->勾选Telnet服务器和Telnet服务端->确定





现在打开cmd，输入telnet 127.0.0.1测试一下，如果不成功,可能是telnet服务并没有开启(默认未开启)。(有的系统telnet功能虽然已经打开了，但是telnet依然连接不上，提示:connect refuse或者在端口 23: 连接失败)，也可能是这个原因，下面是开启的步骤。

(win7)打开控制面板->管理工具->服务->Telnet->右键属性改为 自动 ->然后在对着 telnet 右键 启动。





这时候再执行

telnet 127.0.0.1 成功后看到如下界面:

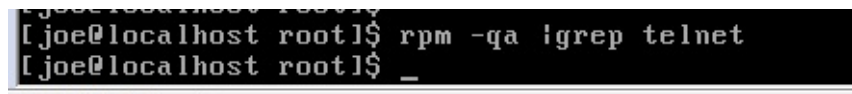


则windows端的 telnet 已可用。

CentOs安装telnet功能

检查是否已经安装telnet

```
$rpm -qa | grep telnet
```



未找到,则说明未安装。已安装则跳过安装步骤。

CentOS安装telnet功能

安装telnet及telnet-server，注意，需要root权限来安装。

```
$yum install telnet
```

```
$yum install telnet-server
```

等待一会会提示是否安装，输入y然后回车，一会就装好了。

```
Is this ok [y/N]: y
Downloading Packages:
telnet-0.17-48.el6.x86_64.rpm                               | 58 kB      00:00
warning: rpmts_HdrFromFdno: Header U3 RSA/SHA1 Signature, key ID c105b9de: NOKEY
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
Importing GPG key 0xC105B9DE:
  Userid : CentOS-6 Key (CentOS 6 Official Signing Key) <centos-6-key@centos.org>
  Package: centos-release-6-4.el6.centos.10.x86_64 (CentOS-2013030201)
  From    : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
Is this ok [y/N]: y
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : 1:telnet-0.17-48.el6.x86_64                  1/1
  Verifying  : 1:telnet-0.17-48.el6.x86_64                  1/1

Installed:
  telnet.x86_64 1:0.17-48.el6

Complete!
[root@localhost ~]#
```

```
(1/2): telnet-server-0.17-48.el6.x86_64.rpm                | 37 kB      00:00
(2/2): xinetd-2.3.14-39.el6_4.x86_64.rpm                  | 121 kB     00:00
-----
Total                                                    491 kB/s | 159 kB      00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : 2:xinetd-2.3.14-39.el6_4.x86_64
  Installing : 1:telnet-server-0.17-48.el6.x86_64
  Verifying  : 1:telnet-server-0.17-48.el6.x86_64
  Verifying  : 2:xinetd-2.3.14-39.el6_4.x86_64

Installed:
  telnet-server.x86_64 1:0.17-48.el6

Dependency Installed:
  xinetd.x86_64 2:2.3.14-39.el6_4

Complete!
[root@localhost ~]#
```

开启telnet服务

因为装好telnet服务之后，默认是不开启服务的，下面我们需要修改文件来开启服务。

```
vi /etc/xinetd.d/telnet
```

修改 disable = yes 为 disable = no

```
[root@localhost ~]#  
[root@localhost ~]# vi /etc/xinetd.d/telnet _
```

机内部单击或按 Ctrl+G

```

# default: on
# description: The telnet server serves telnet sessions; it uses \
#               unencrypted username/password pairs for authentication.
service telnet
{
    flags             = REUSE
    socket_type       = stream
    wait              = no
    user              = root
    server             = /usr/sbin/in.telnetd
    log_on_failure    += USERID
    disable           = yes
}

```

```

# default: on
# description: The telnet server serves telnet sessions; it uses \
#               unencrypted username/password pairs for authentication.
service telnet
{
    flags             = REUSE
    socket_type       = stream
    wait              = no
    user              = root
    server             = /usr/sbin/in.telnetd
    log_on_failure    += USERID
    disable           = no
}

```

激活xinetd服务

```
service xinetd restart
```

```

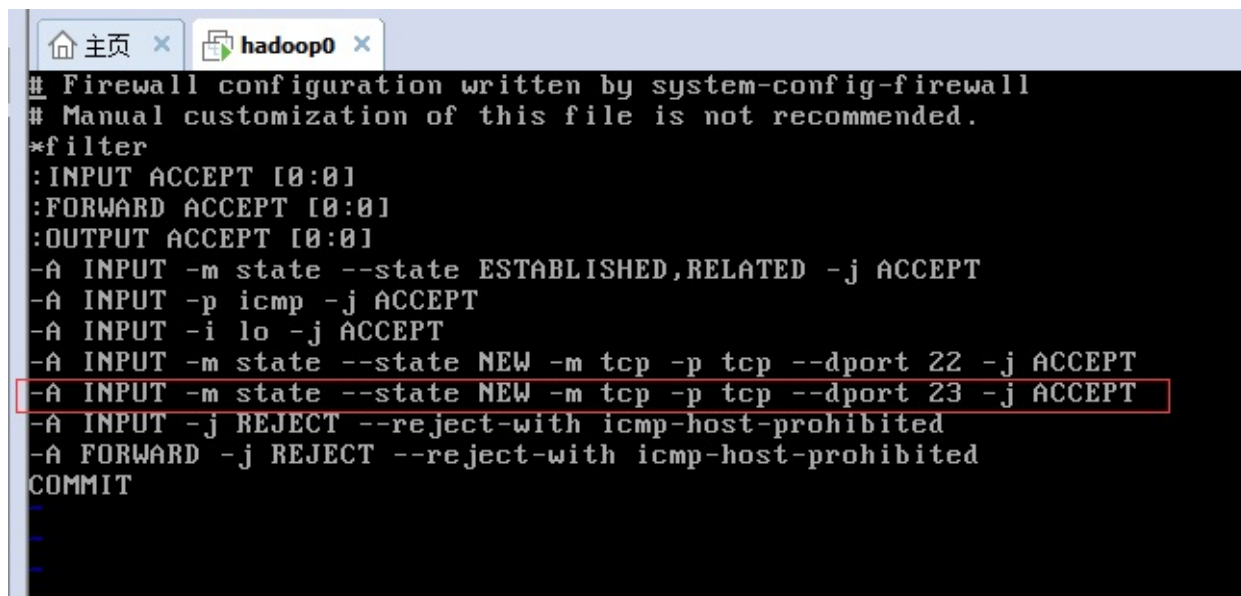
"/etc/xinetd.d/telnet" 14L, 304C written
[root@localhost ~]# service xinetd restart
Stopping xinetd: [FAILED]
Starting xinetd: [ OK ]
[root@localhost ~]# service xinetd restart_

```

防火墙中打开23端口

修改/etc/sysconfig/iptables 文件，添加以下内容：

-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 23 -j ACCEPT



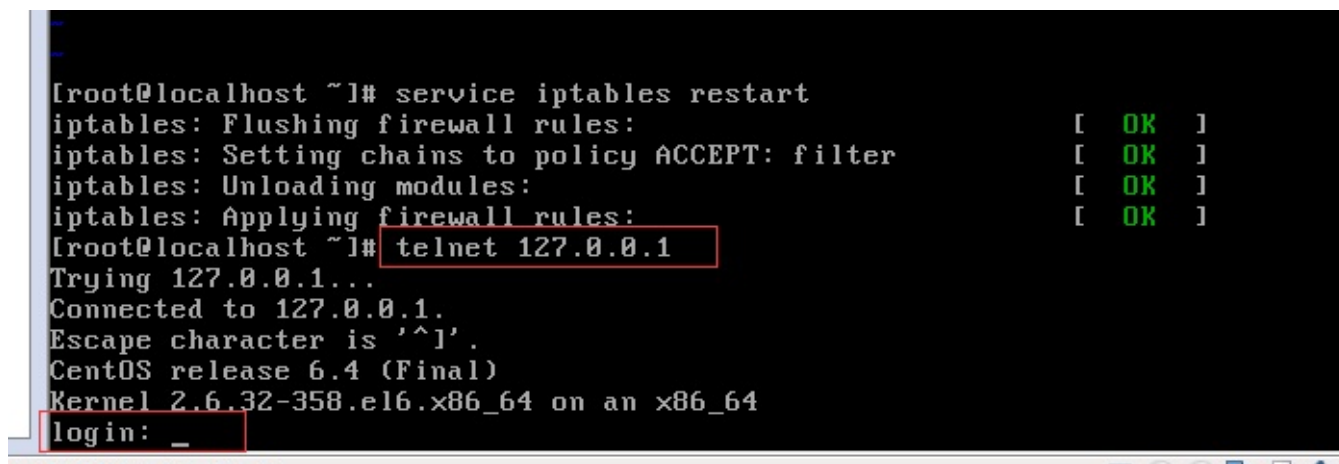
```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 23 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

然后service iptables restart重启防火墙。搞定！

```
service iptables restart
```

测试telnet

这时候telnet 127.0.0.1，可以成功访问。

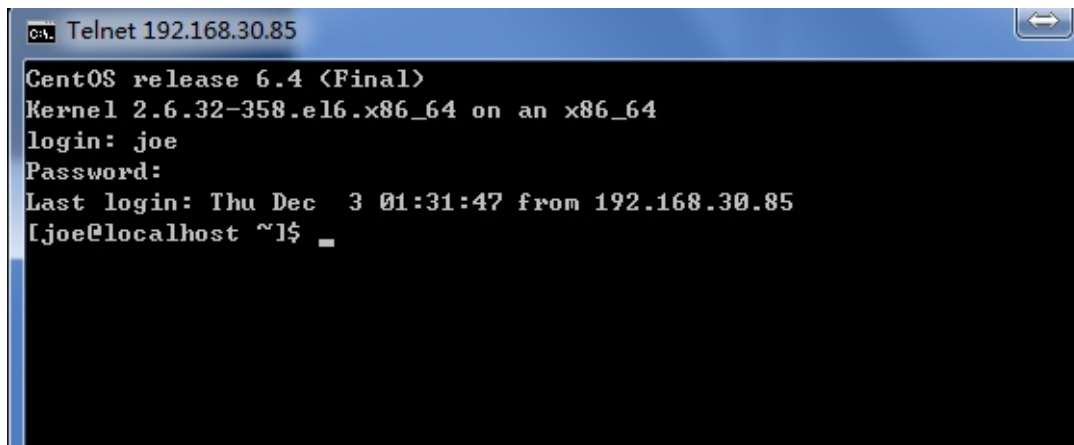


```
[root@localhost ~]# service iptables restart
iptables: Flushing firewall rules: [ OK ]
iptables: Setting chains to policy ACCEPT: filter [ OK ]
iptables: Unloading modules: [ OK ]
iptables: Applying firewall rules: [ OK ]
[root@localhost ~]# telnet 127.0.0.1
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
CentOS release 6.4 (Final)
Kernel 2.6.32-358.el6.x86_64 on an x86_64
login: _
```

windows与linux相互telnet

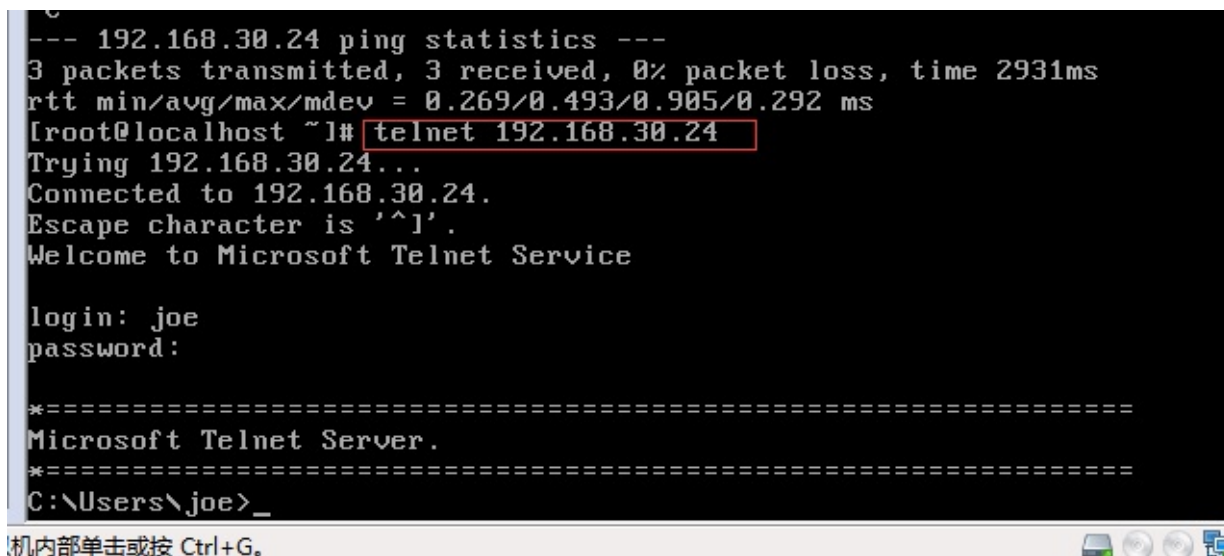
windows远程登录linux

```
telnet 192.168.30.85
```



linux远程登录windows

```
telnet 192.168.30.24
```



退出telnet

ctrl+] 然后在telnet 命令行输入 quit 就可以退出了

linux基础(十一)----linux编程基础----变量

linux编程基础 跟 其它语言的编程基础大同小异,涉及到变量,语法,流程控制等。

ps:

linux的程序编译解析时 是一行一行的编译解析,所以不写;也可以。

我们这里还是大概了解一下。

linux变量分类

Linux使用下列两种变量:

局部变量:由程序员建立,且仅供程序员所设计的程序使用。

环境变量:由程序员或他人建立,程序员和他人的程序都可以使用。

每当程序需要保存数据时,就可以建立(或声明)一个局部变量。这个局部变量仅能被声明者本人的程序使用。

每当登录Linux的时候,就可以会看到一连串的变量,这些变量就是环境变量。环境变量包含许多关于用户和计算机的信息。

例如,EDITOR就是一个通用环境变量。赋给这个环境变量的值,就是停驻在计算机上某文本编辑器的名字。

声明变量

在保留字declare和export后面加上一个变量名就可以创建一个变量,请看下面的例子:

```
declare FirstName
```

```
export editor
```

在这个例子中,共创建了两个变量:

保留字declare声明了一个环境变量FirstName。

保留字.export使得变量editor可由环境存取。

放置变量

在linux程序中,并没有规定要在何处声明一个变量。事实上,在程序中的任何位置都可以声明一个变量。但是,随意放置变量并不是一个好程序员所干的,因为这样可能会增大下次读程序查找它时的困难。

把所有的变量都放在程序的开头不失为一种良好的编程习惯,因为这样一来,在需要查找它们时也比较容易。

合并多个变量

在同一行上,一次可以同时声明一个或多个同类型的变量(比如说两个),如下例所示:

```
declare FirstName LastName
```

当然,只要该行能放下,还可以声明更多的变量。不过要注意,每个变量之间都要用一个空格隔开。

我们不必搜索程序中的每一行,只要看看程序开始部分的几行,就可以检查所有的变量

给变量命名

我们可以给变量任意命名。

例如，将一个变量命名为SSNum，并在其中存入一个电话号码。

但是，这个变量名相对于电话号码来说，并没有多少意义。

为了使编程变得更容易，应当给变量取一个合适的名字，让他人(包括我们自己)也能明白里面放的究竟是什么类型的数据。仅当打算在变量中存放一个社会福利号时，将变量命名为SSNum才算比较合适——就像用变量FirstName来存储某人的名字一样。

给变量命名，有以下几条规则：

任何变量都必须以一个字母为开头；

任何变量都只能由字母(包括大、小写)、数字和下划线(_)组成；

变量中不能有空格。

当然，我们不能把保留字用作变量。

将数赋给变量

在创建好变量之后，也许想给变量塞点什么东西。这种行为过程称之为给变量赋值。

给一个变量赋值，使用等号(=)和保留字let就足够了。

这两个符号(等于号和保留字let)的意思是，告诉计算机将某一个特定的数值赋给某一个特定的变量。

假设读者的薪金是一百万：下面举一个例子，来说明如何将这个值(1000000)赋给一个数字变量MySalary，如：

```
let MySalary=1000000
```

一个变量仅有一个值。不过，还可以通过给变量另赋一个值的方式来修改这个值。这时，计算机会把老值覆盖掉而用另一个新值来取代它。老值永远都是被抛弃者。

如果读者因为工作努力，老板给你薪金加倍，则这个例子变成：

```
let MySalary=1000000
```

```
let MySalary=2000000
```

将字符串赋给变量

给变量赋字符串就像给变量赋数值一样简单，但二者之间还是有一些区别的，如：

要用保留字declare或export声明一个字符串变量。

字符串两边必须带有引号，以便告诉计算机字符串从哪儿开始及从哪儿结束。

当在程序中引用变量时，必须在变量前加一个美元符号(\$)。

现在，可以把一个人的姓名(first name)赋值给一个字符串变量了，如：

```
declare FirstName ="Mary"
```

或者把一个人的名和姓都赋给一个字符串变量，如：

```
declare Name= "Mary Smith"
```

甚至还可以把一句完整的话赋给一个字符串变量，如：

```
declare MyGoal= "Buy out Bill Gates"
```

有时，字符串是由数字而不是由字母组成，但它们仍属于字符串，例如下面的一个电话号码：

```
declare Telephone="555-5555"
```

别忘了这条规则：在字符串两边用引号引起来。引号的目的是告诉Linux所有出现在引号之间的字符都应作为一个字符串处理。

将一个变量赋给其他变量

我们可以将数据存储在一个变量之中，然后再将它赋值给另外一个变量。这时，就有该数据的两个备份，而计算机并不会将数据从原来的变量中删去。

下面的例子说明，通过使用一个字符串变量，如何将字符串从一个变量复制到另外一个变量之中：

```
declare MyGoal = "Buy out Bill Gates"
```

```
declare OurGoal=" $MyGoal"
```

在这个例子中，计算机先找到变量MyGoal，并从它那儿复制数据；然后，计算机去取该数据，并把它放入变量OurGoal中。不过，在这里使用的是初始变量名MyGoal，并在其前面加上了一个美元符号(\$)。

当然，用数字变量也同样可以执行这个数据复制过程，如下例所示：

```
let MySalary =1000000
```

```
let OurSalaries=$MySalary
```

在这个例子中，计算机将变量Myalary的值拷贝给变量OurSalaries。此时，变量OurSalaries的值Myalary一样也是1000000。不过，变量MySalary中的值仍保持不变。

linux基础(十二)----linux编程基础----与用户交互

linux程序不像其它语言一样有界面上的按钮交互等，所以一般需要获取用户的键盘输入。

这次学习如何通过键盘获取用户输入的信息及将信息显示在屏幕上。

读取键盘输入

用户界面将信息显示在屏幕上，提示用户在键盘上按键。用户界面就是引导用户去做程序设计者想做的。

在举一个例子来说明如何指示计算机读入字符，并把它们赋给一个字符串变量，如下例所示：

```
#!/bin/bash
clear
echo " "
echo "Enter Yvur First Name:"
read FirstName
```

指令说明如下：

首先，#!/bin/bash指示计算机在清屏(clear)前，启动hash shel。

其次，计算机在屏幕上跳过一个空行(echo " ")，并告诉用户要输入的数据类型(echo "Enter Your FirstName:")

保留字read指示计算机读取用户在键盘上输入的所有字符，并把它们存放到字符串变量FirstName中。不过，这个命令仅在用户按了回车键之后才开始执行。

这里 无论用户输入 数字 字符串 或者混合输入 使用的代码都是一样的。

显示用户输入的数据

在程序获取用户输入的信息之后，就应当对它进行必要的处理。

程序既可以把数据用于计算，也可以把它存储在磁盘文件之中，甚至可以对它进行读者所能想像的任何处理。

对信息最常用的处理方式之一就是把它显示在屏幕上，另外就是把它与已有的信息进行比较。

但是，目前只需完成简单的信息显示工作就可以了。

一个典型的linux程序常叫用户输入一串字符，然后将它们存入一个变量中，接着在屏幕上把它们显示出来。

显示字符串的方式有两种，如：

将各个字符都显示在同一行上。

先将一些字符与其他的一些字符合并起来，再将它们显示在同一行上。

可以用指令echo "\$variable name"，让计算机仅仅显示用户输入程序中的字符(当然，应当用实际的变量来取代variable name)。

还可以把各个字符放在引号(" ")之间，将它们合并起来再显示，如下面的代码所示。在这段代码中，程序请求用户输入他们的名字，然后显示一条欢迎他们的信息。例如：


```
#!/bin/bash
clear
echo " "
echo "Enter Your First Name; "
read FirstName
echo "Hello, $FirstName"
```

尽管在这个例子中程序增加了一个文本Hello，但是仍可以将这个字符串存储在另一个字符串变量中(在下面的例子中就是这么做的)。利用下面的代码，可以先指示计算机存取变量\$Greeting并显示该字符串后，再存取变量\$FirstName并显示其中的那些字符。例如：

```
#!/bin/bash
clear
declare FirStName, Greeting
echo $Greeting ="Hello,"
echo " "
echo "Enter Your First Name; "
read FirstName
echo "$Greeting $FirstName"
```

把数据存入文件

将用户输入的数据存入磁盘文件是经常要做的事。

Linus Torralds (Linux) 的设计者)认为无论程序什么时候用保留字echo显示信息，它都要用到屏幕。

这对大多数程序而言确是如此，但是也可以改变程序，让它将数据显示到其他的地方，比如一个文件中。这种处理过程称之为重定向(redirection)，它的意思是指改变数据正常的流动方向。

覆盖写入

这种重定向工作其实一点也不难做。只要看看下面的例子，就会知道怎么做了。注意，除了最后一条语句外，其他的语句在前面都用过，如下例所示：

```
#!/bin/bash
clear
echo " "
echo "Enter Your First Name;"
read FirstName
echo "Enter Your Last Name;"
read LastName
echo "$FirstName $LastName" >employees.dat
```

大于号(>)指示计算机把变量的值存入一个名字为employees.dat的文件中，而不是显示在屏幕上。

关于重定向符大于号(>)有以下两点值得读者注意：

如果在程序中用大于号(>)向个文件中存信息，计算机将创建一个新的文件。

如果计算机中存在一个同名文件，它将用新的信息覆盖旧文件中的内容，文件中的原有信息将被丢失。所以，用它一定要小心。

向文件中添加数据

用双大于号(>>)可以将信息添加至文件的末尾而不覆盖文件中的已有数据。

下面的代码与上一个例子的代码几乎相同——只是它是将新的信息添加到文件中(假设该文件已经存在)。如果读者在程序中规定的文件不存在，它将会自动创建一个新文件。例：

```
#!/bin/bash
clear
echo  " "
echo "Enter Your First Name;"
read  FirstName
echo  "Enter Your Last Name;"
read  LastName
echo "$FirstName $LastName" >>employees. dat
```

显示存入文件中的数据

在程序中使用实用程序cat，可以把存入一个文件中的信息显示到计算机屏幕上，如下面的例子所示。

将一个实用程序名放入程序，就像在命令行上键入实用程序名一样，也可以让计算机运行它。

程序员常把这个过程说成是调用一个实用程序。例如：

```
#!/bin/bash
clear
echo "Employee Data"
echo  " "
cat employees. dat
```

注意，在最后一行cat命令之后，跟有一个文件名。就是我们之前写入的文件名。

linux基础(十三)----linux编程基础----linux运算符

算术运算符

算术运算符可以让计算机对数据或包含数据的变量进行加、减、乘、除等工作。

加运算符(+)

两数相加，用加运算符(+)，如下例所示:

```
let a=30
let b=10
let sum="$a + $b"
```

注意，等号(=)两边没有空格，而加号(+)两边有空格，且对所有的运算符而言两边都要求有空格。

减运算符(-)

两数相减用减运算符(-)

```
let Salary =3000
let Expenses =2500
let MyMoney="$Salary - $Expenses"
```

乘运算符(*)

两数相乘用乘运算符(*)

```
let Computers =100
let Commission=5
let TotalCom="$Computers * $Commission"
```

除运算符(/)

两数相除用运算符(/)，如下例所示:

```
let TotalCom =500
let Commission=5
let Computers="$TotalCom / $Commission"
```

取余运算符(%)

两数相除后取余用取余运算符(%), 如下例所示:

```
let  a=10
let  b=3
let  c="$a % $b"
```

逻辑运算符

我们可以用逻辑运算符来让计算机判断一个值的真(true)和假(false).真值是任何一个非零的值, 假值就是零。

非运算符(!)

非运算符可以把一个判断从真变成假, 也可以把一个判断从假变成真, 请看下面的例子:

```
$Salary!=3000
```

表达式\$Salary!= 3000让计算机把变量Salary的值和值3000进行比较(!是一个比较运算符, 它表示不等于)。

如果变量\$Salary的值不是3000, 则这个表达式是真。再看下面的

例子:

```
!$Salary!=3000
```

假设表达式\$Salary!=3000是真。这时, 由于在表达式前面有一个非运算符(!), 计算机就把这个表达式的值反过来, 即表达式!\$Salary!=3000的值是假。所以说非运算符(!)做出的是与逻辑相反的判断。

与运算符(&&)

让计算机比较两个变量的值, 其中每一个变量的值既可以是真也可以是假, 计算机要依据这两个变量的值才能做出该表达式的值是真还是假的判断。请看下面的例子:

```
$BuyCar=$LowestPrice && $LikeCar
```

计算机通过检查变量LowestPrice 和LikeCar的值, 来决定变量\$BuyCar的值是真还是假。

仅当变量LowestPrice 和LikeCar的值都是真时, 与运算符才让计算机返回一个真值。下面列出了各种可能的情况:

BuyCar	LowestPrice	LikeCar
真	真	真
假	假	假
假	真	假
假	假	真

或运算符(||)

或运算符(||)也是让计算机比较两个变量的值, 其中每一个变量的值既可以是真也可以是假。

计算机要依据这两个变量的值才能决定是返回一个真值还是一个假值。请看下面的例子:

```
$BuyCar=$LowestPrice || $LikeCar
```

计算机通过检查变量LowestPrice 和LikeCar的值，来决定变量\$BuyCar的值是真还是假。

只要这两个变量LowestPrice 和LikeCar之中有一个值是真，或运算符(||)就让计算机返回真值。下面列出了各种可能的情况:

BuyCar	LowestPrice	LikeCar
真	真	真
假	假	假
真	真	假
真	假	真

比较运算符

比较运算符让计算机比较两个数或者两个字符串的值，来决定它们之间的关系是等于、不等于、大于或者小于。

等于运算符(-eq)

要判断两个值是否相同，用等于运算符(-eq)。注意，在连字符(-)和eq之间没有空格。

下面的例子说明了这个运算符的用法:

```
let Salary =3000
let NewSalary =2000
test $Salary -eq $NewSalary
echo "$?"
```

指令说明如下:

let Salary =3000让计算机建立一个名为Salary的变量，并将值3000赋给它。

let NewSalary=2000 让计算机建立一个名为NewSalary的变量，并将值2000赋给它。

test \$Salary -eq \$NewSalary让计算机检验两个变量Salary和NewSalary的值是否相等。

如果相等，test返回一个真值，否则返回一个假值。最后，echo " \$?" 将test返回的值显示在屏幕上。

不等于运算符(-ne)

要判断两个值是否不相等，用不等于运算符(-ne)，请看下面的例子:

```
let Salary =3000
let NewSalary =2000
test $Salary -ne $NewSalary
echo "$?"
```

大于运算符(-gt)

为了判断第一个变量的值是否大于第二个变量的值，用大于运算符(-gt) ,请看下面的例子

```
let Salary =3000
let NewSalary =2000
test $Salary -gt $NewSalary
echo "$?"
```

大于或等于运算符(-ge)

为了判断第一个变量的值是否大于或等于第二个变量的值，使用大于或等于运算符(-ge)，请看下面的例子:

```
let Salary =3000
let NewSalary =2000
test $Salary -ge $NewSalary
echo "$?"
```

小于运算符(-lt)

为了判断第一个变量的值是否小于第二个变量的值，用小于运算符(-lt)，请看下面的例子:

```
let Salary =3000
let NewSalary =2000
test $Salary -lt $NewSalary
echo "$?"
```

小于或等于运算符(-le)

为了判断第一个变量的值是否小于或等于第二个变量的值，用小于或等于运算符(-le),请看下面的例子:

```
let Salary =3000
let NewSalary =2000
test $Salary -le $NewSalary
echo "$?"
```

linux基础(十四)----linux编程基础----linux条件控制语句----if else语句

if语句

当需要程序检测一个条件是真还是假的时候，就可以使用if语句。if语句仅仅是告诉程序：“如果条件为真，则执行这些指令，否则跳过这些指令。”

一个条件为真的语句可以让程序执行一组指令，一个条件为假的语句则跳过这些指令。

if语句的规则如下：

```
if[Condition]
```

```
then
```

```
Instruction
```

```
fi
```

一般说来，如果Condition存在(即条件Condition为真)，程序则执行下面的Instruction。

条件Condition必须是真或假，且永远如此。

下面举两个例子来说明if语句的用法。第一个例子是让程序判断一个盒子是否已装满，条件是盒子最多只能装下100个小甜饼。

```
if[$Quantity -eq 100]
then
echo "The box is full"
fi
```

这个语句让程序检测一下变量Quantity的值，看它是否是100。如果是，则在屏幕上显示：

```
The box is full.
```

否则，就跳过这条语句，并移到fi的下一条语句。注意，在这里，fi是一个保留字，它有特殊的含意(标志if语句的结束)。

再举一个例子，要求程序检测一下盒子是否已满，及是否还有其他的盒子可获得。当且仅当这两个条件同时是真时，程序才会报告无法再装甜饼了，如下例所示：

```
if [$Quantity -eq 100 ] &&[$AvailableBoxes -lt 1]
then
echo "The box is full and you have no more boxes,"
fi
```

这条语句要求程序检测一下变量Quantity的值，看它是否是100，及变量AvailableBoxes的值是否小于1。如果这两个条件都为真，则显示信息:The box is Full and you have no more boxes.

如果变量Quantity的值不是100，且盒子也有剩余，程序则跳过这条语句，移到保留字fi的下一条语句。

if else语句

if语句可以使程序依据一个条件来做出某种判断，如果条件是真则执行一组规定的指令。

但是，在使用if语句时也存在着一个问题，即条件是假时，可能还需要为程序提供另外一组指令。

当然，也可以再用一条if语句来解决这个问题，即在条件为假时让程序执行另一组指令。但是，有一种方法比这更简单，这就是if else语句。if else语句的语法规则是：

```
if[Condition]
then
    Instruction1
else
    Instruction2
fi
```

这条语句表示，如果Condition条件为真，程序则执行第一组指令(Instruction1);如果Condition(条件)为假，则程序执行第二组指令(Instruclian2)

例如：

```
if[$Quantity -eq 100]
then
    echo "The box is full"
else
    echo "The box is not full"
fi
```

if elif语句

if elif语句的语法规则如下：

```
if[Condition1]
then
    Instruction1
elif [Condition2 ]
then
    Instruction2
fi
```

在这条语句中，如果条件Condition1是真，程序则执行第一组指令Instruction1 ;否则，程序再判断条件Condition2是否是真。如果条件不为真，程序则执行第二组指令Instruction2;如果Condition2是假，则跳过第二组指令。

例子:

```
if[$Quantity -eq 100]
then
echo "The box is full"
elif [$Quantity -eq 95]
echo "You can add 5 cookies to the box."
fi
```

用if elif语句进行多重选择

在程序中使用多重if elif语句进行条件检测，程序能处理多种可能。多重if elif语句结构规则如下:

```
if[Condition1]
then
    Instruction1
elif[Condition2]
then
    Instruction2
elif[Condition3]
then
    Instruction3
fi
```

在这条语句中，如果Condition1是真，程序执行第一组指令Instruction1 ;否则，检测条件Condition2是否是真。如果条件Condition2是真，则执行第二组指令Instruction2 ;否则，检测Condition3是否是真。若条件Condition3是真，则执行第三组指令Instruction3;否则，跳过第三组指令Instruction3。

保证Linux至少执行一组指令

想像一下，如果我们写了一个结构庞大的if elif语句，却发现它什么也不干，会是什么感觉?先别笑，在实际编程中确实有这样的事发生。不过，要是在if elif语句的最后放一条elif fi语句，就可以避免这样的灾难，如下例所示:

```
if[Condition1]
then
    Instruction1
elif[Condition2]
then
    Instruction2
elif[Condition3]
then
    Instruction3
elif fi
```

```
elif  
  
    Instruction4  
  
fi
```

在这条语句中，如果Condition1是真，程序执行第一组指令Instruction1 ;否则，检测条件Condition2是否是真。如果条件Condition2是真，则执行第二组指令Instruction2 ;否则，检测Condition3是否是真。若条件Condition3是真，则执行第三组指令Instruction3;否则，执行第四组指令Instruction4。

linux基础(十五)----linux编程基础----linux条件控制语句----case语句

case语句

如果有很多层if else,最好使用case来代替。

case语句的功能是:把保留字case右边的值和闭括号 ")" 左边的值比较,其语法规则如下:

```
case VariableName in
value1)
    Instruction

    ;;
value2 )
    Instruction

    ;;

esac
```

从上面可以看出,case语句以保留字case开头,以esac结尾。它首先检查变量VariableName 的值,如果它与value1相等则执行第一组指令集,如果它与value2 相等则执行第二组指令集,以此类推。同时,在每组指令集的最后用两个分号(;;)。

在下面的例子中,将用case语句取代本章开头的if语句,请看:

```
case $region in
1)
echo "Hello, Gob. "
;;
2)
echo "Hello, Mary."
;;
3)
echo "Hello, Joan."
;;
4)
echo "Hello, Mike."
;;
5)
echo "Hello, Tom."
;;
esac
```

case语句中的默认情况

在case语句中，如果用户没有给程序提供所需要的匹配值，那么程序就找不到它所要匹配的对象。如果是由程序设计者本人提供程序所必需的匹配值，这种情况有时就不会发生。

我们完全可以用保留字符(*)来为此留一条后路，即提供一种默认情况，让程序在此情况下执行一些必要的操作。请看下面示例：

```
case $region in
1)
echo "Hello,Gob. "
;;
2)
echo "Hello,Mary."
;;
3)
echo "Hello,Joan."
;;
4)
echo "Hello,Mike."
;;
5)
echo "Hello,Tom."
;;
*)
echo "Sorry,your region is not on my list."
esac
```

case语句在用户界面中的应用

case语句最常和程序的用户界面一起使用。在一个程序显示菜单后，程序必须等待用户从键盘上输入一个字符。这通常由用户做出选择，输入选项，程序就用case语句把该选项与某些特定的值做比较。

假设读者想建立一个电子电话号码簿。为简单起见，在这里仅要求其具有两项功能：一是显示电话号码；二是可以添加电话号码。

首先，程序以菜单选项的形式将这两项功能显示在屏幕上。然后，由用户从键盘上输入相应的选项。

为实现这个目标，可以用case语句把用户输入的选项与已知的菜单项对比。

在下面的例子中，程序仅显示用户欲做何种选择的信息(当然，如果让程序再复杂一点，可以用其他的指令来取代这些信息)，请看：

```
#!/bin/bash
clear
echo ""
echo "The Telephone Book"
echo ""
```

```
echo "1.Display A Telephone Number"
echo "2.Add A New Telephone Number"
echo ""
echo "Q Quit"
echo ""
echo "Enter your selection;"
read selection
case $selection in
"1")
    echo "You want to display a telephone number."
    ;;
"2")
    echo "You want to add a new telephone number."
    ;;
"q")
exit 0
    ;;
"Q")
exit 0
    ;;
*)
echo "You made an invalid selection."
esac
```

为了避免用户退出程序时出现问题，最好像上面的例子那样，将字母的大、小写(如q和Q)都标上。

linux基础(十六)----linux编程基础----linux条件控制语句----多层嵌套控制结构

收到过装在一个大盒子中的礼物吗?当你迅速打开它时，里面还有一个稍小的盒子。再打开这个小盒子，发现里面又有一个小盒子.....直至找到礼物。

在编程中，盒子中又有盒子的现象称为嵌套(nesting)。如果把多个if语句或case语句组合到一起，就是嵌套了。

if嵌套结构

下面是一个使用if语句嵌套的例子:

```
if [$office -eq "1"]
then
  if [ $RegionalMgr -eq "Bob"]
  then
    echo "Hello,Bob."
  fi
fi
```

这个程序首先检查变量\$office的值。如果它是1，则再检查变量\$RegionalMgr的值。如果它的值是Bob，则显示欢迎Bob的问候语。

case嵌套结构

case语句同样可以嵌套(与上例的方法相同)，请看下面的例子:

```
case $office in
"1")
  case $RegionalMgr in
    "Bob")
      echo "Hello,Bob"
    ;;
  esac
;;
esac
```

这个程序首先检查变量\$office的值。若它等于1，程序再检查变量\$RegionalMgr的值。

若它的值是Bob,程序则执行后面的指令，显示对Bob的问候语。

缩进格式书写嵌套结构

计算机是不关心在if或case语句放了多少条if或Case语句的。但是，读者就不同了，因为嵌套语句一多，就不容易看懂程序。

建议用缩进格式书写每一个嵌套语句，如前面的例子所示。Linux本无所谓缩进格式的，但这样写便于阅读程序，故不失为是一种好的编程风格。

linux基础(十七)----linux编程基础----linux循环控制语句----while循环

while循环的一般格式如下:

```
while [Gondition ]
do
    Instruction
done
```

在while循环中，必须放入一个导致结果为真或假的变量或表达式，而在while循环体中则可以放入任意多的指令。

while循环在Linux代码中的使用

当程序遇到一个while循环时，首先检测它的条件，询问它"这个条件是真还是假?"，仅当条件为真时，程序才会读入while循环体中的指令。

请看下面演示while循环的例子:

```
declare raining ="1"
while["$raining" -eq "1"]
do
    echo "Still raining."
done
```

while循环在菜单中的应用

在创建菜单界面时，通常要用到while循环。在任何程序菜单中，程序用户都期望顺序是这样的:

- 1.显示菜单
- 2.选择菜单选项
- 3.运行有关程序
- 4.返回菜单进一步选择

设计这样一个菜单最简单的方法，就是把所有的这些指令都放在while循环中，在程序每次执行完用户的选择要求后，就返回到while循环，重新开始。请看下面的例子:

```
#!/bin/bash
declare flag ="1"
while ["$flag" -eq "1"]
```



```

do
    clear
    echo ""
    echo "The Telephone Book"
    echo ""
    echo "1.Display A Telephone Number"
    echo "2. Add A New Telephone Number"
    echo ""
    echo "Q Quit"
    echo " "
    echo "Enter your selection:"
    echo ""

read selection
case $selection in
    "1")
        #Run the subprogram to display a phone number
        getnum
        ;;
    "2")
        #Run the subprogram to add a new phone number
        addnum
        ;;
    "q")
        $flag="0"
        ;;
    "Q")
        $flag="0"
        ;;
    *)
        echo "You made an invalid selection, Try again."
esac
done

```

##while在计时循环中的应用

读者可能碰到过这样一种情况，就是想在程序执行下一组指令前，先让程序暂停一会儿。

例如，可能想让程序在屏幕上自动显示一系列信息，而在每显示下一条信息之前，让上一条信息在屏幕上停留一段时间——一种幻灯式的放映，与在Windows程序中见到的不一样。

要想让程序暂停足够长的时间以方便用户阅读信息，方法之一就是使用一种所谓的计时循环(timing loop)来创建这样一种暂停。计时循环其实是一种简单的while循环，仅仅是在循环体中用一条指令给一个变量加1。请看下面的例子：

```

declare counter =0
while ["$counter" -lt 1000]
do
let $counter="$counter+1"

```

```
done
```

在这个例子中，程序首先给变量counter赋值为0,然后就进入while循环。循环体中唯一一条指令的作用就是给计数器(counter)的值加1，并将新值重新赋给计数器变量(counter)。while循环一直计数到变量counter的值达到999为止--这时,程序将退出while循环。

可以通过条件表达式中值的加、减来调整暂停的时间。在本例中，条件表达式中counter的值是1000。若增加这个值，则暂停的时间将变长;若减少这个值，则暂停的时间会变短。

避免死循环

如果while循环的条件是假，程序就不会执行循环体中的任何指令。因此，while循环体中的指令也就从不重复。事实上，它们根本就未执行过。请看下面的例子:

```
declare raining="0"
while ["$raining -eq "1"]
do
    echo "Still raining"
clear
```

在这个例子中，程序根本不显示任何信息，因为\$raining的值不是1，程序跳过while循环。

如果while循环的条件是真，程序至少执行一次循环体中的所有指令。

如果while循环的条件永远是真，那么程序就一遍又一遍地执行循环体中的指令，程序永不结束运行。我们把这种情形称为死循环(endless loop)。

```
declare raining="1"
while ["$raining -eq "1"]
do
    echo "Still raining"
clear
```

没有一个程序员愿意在程序中放入一个死循环。为避免这种情况的出现，必须确保while循环体中至少有一个指令。可以修改while循环条件中所用的真值或假值。

```
declare raining="1"
while ["$raining -eq "1"]
do
    let $raining="0"
clear
```

linux基础(十八)----linux编程基础----linux循环控制语句----for in循环

什么时候使用for in循环

前面我们已经学习了while循环,那么一般什么情况下使用for in循环呢:

当想让程序在条件为真时执行一系列的指令,就使用while循环。

它会一直运行到程序中有一条指令将条件修改为假时终止。

但是,如果知道循环体中指令执行的具体次数,那么就使用for in循环。

for in循环一般格式如下:

```
for VariableName in wordlist
do
    instruction
done
```

怎样使用for in循环

请看下面for in循环写成的例子:

```
for friend in Mary Joe Sue
do
echo "Hello, $friend
done
```

在for in循环的第一行,在far friend in后面紧跟着三个人的名字,这样就建立了一个单词表,以后就可以把这些名字当作值赋给变量friend。

输出结果:

```
echo "Hello, Mary"
```

```
echo "Hello, Joe."
```

```
echo "Hello. Sue."
```

使用单词表

赋给变量的那些值统称为单词表(wordlist)。它是一个关于字符串值(字符和数字的组合)的列表,我们可以按顺序从中一次取一个值赋给变量。

再顺便提一下，除了for in循环之外，在其他的语句中也可以使用单词表。

关于单词表有以下几点值得注意：

1. 赋给单词表的值必须是一个字符串值。
2. 每一个值之间必须用空格隔开。
3. 如果空格也是字符串的一部分，则需要在字符串的两边加上引号。

为了说明含空格的字符串的用法，请看下面的例子：

```
for friend in "Mary Jones" "Joe Smith" "Sue Janes"
do
    echo "Hello, $friend."
done
```

linux基础(十九)----linux编程基础----linux循环控制语句----break中断和continue继续

break和continue的使用是紧密和while循环联系在一起的。

使用break快速退出

你曾碰到过想骑车跑一英里却在半路停下来事情吗?事实上,这样的事情也同样会发生在程序执行while循环的过程中。脚踏车有一个安全刹车来迫使它停下来,而在while循环中这个安全刹车就是保留字break。

while循环一直循环到它的条件变为假才停止。如果想在循环的条件未改变之前就退出循环,那该怎么办呢?不妨用break语句。

请看下面使用break语句的例子:

```
let n=1
while [ "$n" -eq 1 ]
do
    echo "Enter your name or type stop to end:"
    read name
    case $name in
        "stop")
            break
        ;;
    esac

done
echo "Good-bye!"
```

这个程序请求用户输入一个用户名或输入stop来退出循环。只要用户输入的不是stop,程序就显示一条问候语。用户输入了stop后,case语句就叫程序中断(break),它立即退出循环,并继续执行保留字done后的指令。在这个例子中,程序最后显示Good-bye!

使用continue语句重新循环

使用到达循环的开始有时,程序并不必执行放入循环体中的所有指令。如果使用continue语句,程序将跳过循环体中continue后面的其余指令。请看下面使用continue语句的例子:

```
declare n =1
while [ $n -eq 1 ]
do
    echo "Enter your name yr type stop to end:"
    read name
    echo "Enter your employee number;"
```

```
    read num
case $name in
    "stop")
        if [ "$num" -eq 1 ]
        then
            continue
        else
            break
        fi
        ;;
    *)
        echo "Hello,$name"
    esac
done
echo "Good-bye !"
```

计算机首先请求用户输入用户名和数字。

在用户输入了他的名字后，程序显示对他的问候语。

若用户输入的是stop和非1的数字，它用保留字break退出循环，并跳到程序的最后一条指令(即done之后的一条指令)，显示Good-bye！

continue语句使程序跳过循环体中它后面其余的语句，并直接跳到循环的开始(顶部)，而不显示问候信息。在到达循环的开始(顶部)之后，它重新核对条件，并决定是否再次执行循环体中的指令。

linux基础(二十)----linux编程基础----子程序----函数

写一个又大又复杂的程序的技巧之一，就是将该程序分解成一些称之为子程序的小程序，而在每一个子程序中，又可以把重复出现的代码组织到一起形成一个函数。

函数和子程序执行的是主程序某一特定的任务。我们要做的工作就是写一个主程序，当需要某一个函数和子程序的时候就调用它们。

本章先了解函数。

在编写Linux程序的时候，有时不得不一遍又一遍地重写某些相同的命令。例如，给朋友写一份邀请信，请他们来参加一个狂欢舞会。可把这些信息写在一个for in循环中，并将每个朋友的名字放在一个单词表中。当然，也可以把每一份邀请信都写进程序中，只不过是内容重写几次罢了。或者使用一个函数。

函数(function)是Linux程序的一部分，与程序的其余部分是分开的。试设想一下：一个函数就像程序中的一个分隔仓。我们先给一个函数取一个惟一的名称(即函数名)，然后就可以把经常要使用的一条或多条命令放入函数之中。

当linux需要使用存放在函数中的命令时，可以在程序中键入该函数的名称。因此，通过使用函数名就可以重复使用函数中的命令，而不必再重写这些命令。譬如，把那些邀请词放在一个函数中。

在每次调用函数时，可以给函数传递一定的信息。这个过程称之为给函数传递参数。因此，可以创建一个打印邀请信的函数，每调用它一次，就给函数传递一个名字，而函数则会自动地把名字赋给函数中的某个变量。以后，当涉及到那位朋友的时候，代表他的就是一个变量，而不是他的名字。

通常使用函数来处理传递给函数的某些信息，然后，将处理结果返回到程序的调用处。函数处理后返回的结果称之为返回值(return value)。我们可以在程序中利用这个返回值。

如果需要重写某段代码，就可以使用函数来代替它。

创建一个函数

在使用一个函数之前，必须先要在程序中定义一个函数。因此，创建一个函数最好的地方是在程序的开始位置。创建函数也称之为定义函数，定义的结果称为函数定义。

定义一个函数请遵循下述步骤：

1.首先在程序中键入一个关键字function。

它表示要创建一个函数。

2.给函数起一个名字。且它在程序中没有任何重名。

函数的名称最好能反映函数的功能。

3.在函数名和第一个命令之间要使用一个开大括号"{"。

它表示函数执行命令的入口。

4.在函数体中键入所需要的任何命令。

在函数体中使用在本书中讨论的任何命令都可以。

5.在函数体每一个命令之后，下一个命令之前用一个分号";"隔开。

注意，分号和命令之间不要留有空格。分号";"表示一个特定的命令结束。

6.在最后一个命令的分号";"之后用一个闭大括号"}"。

注意，在最后一个分号"}"和"|"之间不要留有空格。闭大括号"}"表示函数定义的结束。

下面是函数定义的一般格式:

```
function name
{
command;
command;
}
```

下面是一个典型的函数定义，它的作用是在屏幕上显示一条欢迎信息(此函数例子在本章后面还会引用到，请留心)。

```
function display
{
echo "Welcome to the world"
echo "of functios"
}
```

调用一个函数

在定义了一个函数以后，调用它是非常容易的。我们需要做的就是在需要使用该函数的地方用一下函数名即可。通过函数调用，Linux会自动寻找该函数的定义，并执行其中的每一条指令。

在程序执行到闭大括号"}"后，Linux返回到调用函数的下一行，并继续执行其他的指令。

下面是函数调用的一个典型例子:

```
#!/bin/bash
clear
function display
echo "Welcome to the world"
echo "of functions. "
}
display
```

给函数传递参数

例如，假设要写一个验证用户ID和密码(password)的函数，这个函数除了ID和密码之外，还包括所有验证用户身份所必需的命令和信息。在每次调用函数时，除了ID和密码可以不同之外，其余各部分都应相同。

如果读者想在程序中调用该函数，就把ID和密码作为参数传递给函数，linux把它们赋给某些变量。在这个程序中，这两个变量分别是\$1和\$2。每一个数字都指向它所对应的传递来的参数。注意，由于Linux在这里期望的是数字，所以不能用其他的字符如var1来取代这些变量中的数字的。

因此，可以用变量来取代函数定义中的信息。例如，ID是传递给函数的第一个参数，密码是传递给函数的第二个参数。因此，在函数定义中，当需要引用ID时就可以用变量\$1,需要引用密码时可以用变量\$2。

请看下面的例子:

```
#!/bin/bash
clear
function verify
{
if [$1 -eq "Bob" ]&&[$2 -eq "555"]
then
echo "Verified"
else
echo "Rejected"
fi
}
verify Bob 555
```

验证传递给函数的参数的个数

如果程序不能给一个函数传递它所需要的全部信息，可能会遇到麻烦。例如，一个函数可能要依据程序提供的某些信息来进行某种计算操作。不过，如果程序根本不给函数传递什么 信息，也可能会出现错误。

变量\$#包含了传递给函数的参数的个数。通过对变量\$#的值和函数实际需要的参数的个数进行比较，就可以决定是否阻止函数处理传递来的信息。请看下面的示例:

```
#!/bin/bash
clear
function verify
{
if [$# -ne 2]
then
echo "Wrong number of arguments!"
else
if[$1 -eq "Bob" ] &&[$2 -eq "555"]
then
echo "Verified"
else
echo "Rejected"
fi
fi
}
verify Bob 555
```

与子程序共享函数

子程序同样也可以使用程序员建立的函数。只要把子程序设想成是程序中的一个程序即可。

我们建立一个用函数验证用户ID和密码的程序。但这次却想在另一个不含有该函数的程序中使用该函数。当然，不是把该函数复制到第二个程序中，而是让第二个程序(子程序)也能分享第一个程序中的函数。

我们可以用export命令输出函数，实现子程序之间共享函数。然后，就可以在子程序中调用该子函数。请看下面的例子：

```
#!/bin/bash
clear
function verify
{
    if [$# -ne 2]
    then
        echo "Wrong number of arguments!"
    else
        if[$1 -eq "Bob"]&&[$2 -eq "555"]
        then
            echo "Verified"
        else
            echo "Rejected"
        fi
    fi
}
export verify
subprogram1
```

现在对加在函数定义之后的两行说明如下：

- 1.在函数定义结束符“}”之后，用export命令来说明子程序subprogram1可以访问函数verify。
- 2.程序中最后一行是调用子程序subprogram1。注意，子程序subprogram1并没有包含函数verify的定义，但子程序可以调用它，因为在调用子程序之前已用export命令说明了输出该函数verify。

子程序subprogram1如下：

```
#!/bin/bash
clear
verify "Bob" "555"
```

从函数返回信息

函数就像一条双行道，程序可以发送信息给函数，函数可以回送信息给程序。比如说，不是用函数verify验证一个用户的ID和密码是否正确，而是让函数将它处理后的结果返回给调用程序。

那么，调用程序就要决定它下面该如何操作。

函数的返回值通常存储在变量\$?中。但要注意，返回值必须是在0到256之间的一个整数。因此，是不能将字符串作为返回值返回的。

通常是用一个关键字return，再在其后加上一个要返回的值的方式，从函数返回一个值。

由于返回值是一个整数，因此必须给每一个要返回的整数赋予一定的含义。例如，常用0表示函数运行正常，用非0表示函数运行出错。

关键字return可以放在函数中的任意位置，但它通常放在函数验证某些值-----如比较ID和密码的位置之后。

Linux执行到return之后，函数就停止往下执行，返回到主程序的调用行。

若想知道如何从函数返回一个值，请看下面的例子：

```
#!/bin/bash
clear
function verify
{
    if [$# -ne 2]
    then
        return 1
    else
        if [$1 -eq "Bob"]&&[$2 -eq "555"]
        then
            return 0
        else
            return 2
        fi
        echo "Rejected"
    fi
fi
verify Bob 555

case $? in

do
0)
    echo "Verified"
    ;;
1)
    echo "Wrong number of arguments!"
    ;;
2)
    echo "Rejected"
    ;;
done
```