[1 目录操作命令 2](#__RefHeading___Toc467066003)

[2 文件操作命令 2](#__RefHeading___Toc467066004)

[3 管道 4](#__RefHeading___Toc467066005)

[4 进程 5](#__RefHeading___Toc467066006)

[5 用户和组管理 5](#__RefHeading___Toc467066007)

[6 vi编辑器及shell编程 7](#__RefHeading___Toc467066008)

1. **目录**操作命令
2. **文件**操作命令
3. 变量、流、管道操作
4. **进程**管理
5. **用户**和**组**管理
6. **vi**编程
   * 1. 1 目录操作命令
   1. 查看当前工作目录: pwd;
   2. 改变目录: cd (e.g., 回到家目录：cd ~ ；回到上一层目录：cd ..)

1.3 创建目录

mkdir [-mp] 目录名 （e.g., mkdir –p test1/test2/test3; mkdir –m 711 test）

1.4 删除目录

rmdir [-p] 目录名 (e.g., rmdir –p test1/test2/test3; 注：-p的作用是将上层的空目录一起删除)

* + 1. 2 文件操作命令

2.1 ls：查看当前目录下的文件 (e.g., ls –l; -l表示以长格式列出当前目录的所有文件)

2.2 touch 创建文件/修改文件的修改时间

（e.g., 1, touch a.tmp 创建文件a.tmp；

2, touch –m “03/01/2014” a.tmp；将文件的访问时间修改为2014.03.01 ）

2.3 rm 删除文件/目录

（e.g., 1, rm ~/file; 删除当前目录下的文件file；

2, rm –r ~/directory 删除当前目录下的子目录directory, 其中“-r”的作用是递归删除，即删除目录directory下的所有文件及各级子目录）

2.4 cat 查看文件内容

(e.g., 1, cat hello.txt **查看**文件hello.txt的内容；

2, cat file1 file2 >> file3 将file1、file2文件的内容连接起来并存到文件file3中）

2.5 cp 复制文件

(e.g., 1, cp /etc/passwd /home/user/test； 将目录/etc/下的passwd文件复制到目录/home/user/下的test文件中；

2, cp -r /etc/ /tmp；将目录/etc/下的所有文件复制到目录/tmp下)

2.6 mv 移动文件

（e.g., 1, mv ~/test /home 将当前家目录下的test文件移动到/home目录中； 2， mv ./test ./test\_new 将文件test改名为test\_new）

2.7 ln 在文件之间建立链接

（e.g., ln file1 file2; 为文件file1指定另一个可用于访问的名称：file2; 建立好这个链接后， 使用cat file2命令便可查看file1中的内容，即相当于执行了命令“cat file1”）

2.8 chmod 设置文件的权限

(e.g., 1,“字符设定法” chmod ug+w, o-w my.txt；对文件my.txt，为文件的属主（user,即u）和同组用户（group,即g）增加‘写’的权限，为其他用户（other,即o）去掉对文件my.txt的‘写’权限

2 “数字设定法” chmod 744 my.txt; 即对文件my.txt来说，属主、同组的用户、其他用户的权限分别是rwx《读写执行’》、r《读》、 r《读》)

2.9 chown 改变文件/目录的拥有者

(e.g., 1. chown lenovo my.txt; 将文件my.txt的拥有者/所有者改成lenovo)

2.10 find 查找文件

（e.g., 1. sudo find /etc –size +1M –a –size -10M； 在目录下查找大于1M并且<对应命令中的‘-a’>小于10M的文件，并输出这些文件的信息

2. find . –type f –atime +60>/tmp/filelist； 在当前目录下<对应命令中的‘.’>查找访问时间在60天之前的文件，即查找60天以来没有被访问过的文件 ； 3 find . –type f -perm 644 –exec ls –l {} \; 在当前目录中查找文件所有者（即属主）具有读写<对应命令中的6>权限、文件所属组和其他用户具有读<对应命令中的4>权限的文件，并将这些文件列出来<对应命令中的ls -l>）

2.11 grep 在文件中查找内容

（e.g. 1, grep John students； 在文件中查找具有‘John’的元组《即具有‘John’的那些行》

2, grep -n ‘^root’ /etc/passwd； 在目录/etc/下的文件passwd中查找以root开头<对应命令中的’^root’>的那些元组，并显示它们在文件中的行号<对应命令中的 ‘-n’>）

2.12 which 按路径进行查找

（e.g., 1， which fdisk; 该命令的输入为： /sbin/fdisk； 即查找带有‘fdisk’的那些路径） 2， 又如：命令 which passwd的输出为：/usr/bin/passwd

2.13 diff 比较两个文件中的内容是否相同

（e.g.， diff file1 file2）

2.14 tar 将文件内容进行打包

（e.g.，1. tar -cvf myfile.tar myfile/; 将目录myfile中的文件打包到myfile.tar中;

2. tar –tvf myfile.tar 查看myfile.tar中的内容

3. tar –xvf myfile.tar 将文件从myfile.tar中解包出来）

* + 1. 3 管道

command1 | command2 | command 3

e.g. find /etc –name \*.conf>list1 2>list2

常用的管道命令

* + - 1. 3.1 cut

(e.g. echo $PATH | cut –d ‘:’ –f 3 输出结果是： /usr/local/bin)

* + - 1. 3.2 echo

(e.g., echo “this is a example for echo” )

* + - 1. 3.3 sort

(e.g., cat /etc/passwd | sort | head -5)

* + - 1. 3.4 wc [-lwm] 统计行、单词、字符数
      2. 3.5 head/tail
    1. 4 进程
       1. 4.1 ps 查看当前终端的系统进程

（e.g. 1. ps -ef）

* + - 1. 4.2 kill

(e.g., kill -9 进程号)

* + - 1. 4.3 top (和ps的作用基本相同，只是top动态的显示)
    1. 5 用户和组管理
       1. 5.1 useradd 添加用户

（e.g. 1, useradd –c zhangsan zhang; passwd zhang 添加一个用户账户为“zhang”的新用户，设置其用户名称为“zhangsan”【‘-c’：设置账号描述信息】，并使用passwd设置其密码

cat /etc/passwd | grep zhang; 通过查看/etc/passwd文件来验证是否已经添加zhang用户，输出结果： zhang:x:1001:1001:zhangsan:/home/zhang:/bin/bash

2, useradd –m zhang1; cat /etc/passwd | grep zhang1 【其中‘-m’用于自动建立用户的主目录】，输出结果： zhang1:x:1002:1002: :/home/zhang1:/bin/bash

3, useradd –d /home/uszhang zhang2; cat /etc/passwd | grep zhang2 【其中‘-d’用于设置用户的宿主目录】，输出结果：zhang2:x:1003:1003: :/home/uszhang:/bin/bash

4 useradd –s /bin/bash zhang3 【‘-s’用于设置用户的登录shell】

5 useradd –g zhang1 zhang4 创建用户zhang4，并将它设置为用户组zhang1的成员【‘-g’用户设置用户所属的基本组】）

* + - 1. 5.2 userdel 删除用户

（e.g. 1, userdel zhang 删除用户zhang； cat /etc/passwd | grep zhang 查看是否删除成功

2, userdel –r zhang1; 删除用户zhang1，并删除其主目录/home/zhang1. 【‘-r’：将用户目录下的文件全部删除】）

* + - 1. 5.3 usermod 修改用户信息

(e.g. 1 useradd –c 张四 –g workgroup -m zhang: 添加新用户zhang，设置其用户名称为‘张四’，所属组群为workgroup，并自动生成主目录

usermod –c 张三 zhang; cat /etc/passwd | grep zhang 将用户zhang原来的名称‘张四’修改为‘张三’，并查看是否修改成功，输出结果为：zhang:x:1002:1005:张三:/home/zhang:/bin/bash

1. usermod –d /home/zhangsan zhang 修改用户zhang的主目录为/home/zhangsan
2. usermod –g newgroup zhang 将用户zhang所属的组修改为newgroup
3. usermod –l zhangsan zhang 将用户zhang的账户名称修改为zhangsan
4. usermod –u 1003 zhangsan 将用户zhangsan的uid从原来的1002修改为1003)
   * + 1. 5.4 passwd 设置用户账户密码

（e.g. 1, passwd zhangsan 修改用户zhangsan的密码）

* + - 1. 5.5 su (‘substitute users’的简写) 切换用户命令

（**su** 可以让用户暂时变更登录的身份

e.g., 1. su - 切换到root用户的登录环境；输入‘su -’后提示输入登录密码

2. su user2 从当前用户‘user’切换到用户‘user2’）

* + - 1. 5.6 sudo 以管理员身份执行

（sudo也是以管理员身份执行的命令，当多个管理员参与同一个服务器的管理时，使用该命令可以根据每个管理员的技术特长和管理范围，来下放权限，并且不需要普通用户知道root密码。）

* + - 1. 5.7 groupadd 添加用户组

（e.g. 1, groupadd newgroup; cat /etc/group | grep newgroup 创建名为‘newgroup’的用户组，并查看是否创建成功； 输出结果：newgroup:x:1003:

2, groupadd –g 1006 mygroup 创建组名为‘mygroup’的用户组，并设置其id为1006）

* + - 1. 5.8 groupdel 删除用户组

（e.g. 1 groupdel mygroup; cat /etc/group | grep mygroup删除用户组mygroup，并查看是否将该用户组成功删除）

* + - 1. 5.9 groupmod 更改用户组的属性

(e.g. 1 groupmod -g 1002 newgroup 将用户组newgroup的GID更改为1002

2 groupmod -n workgroup newgroup; cat /etc/group | grep workgroup 将用户组newgroup的用户组名称更改为workgroup ，并查看是否修改成功，输出结果为：workgroup:x:1002:)

* + - 1. 5.10 who 查询有哪些用户登录到了当前系统中
      2. 5.11 groups 查看用户的用户组群成员身份信息

（e.g. groups zhang 显示用户zhang的用户组群成员身份信息）

* + - 1. 5.12 gpasswd 设置用户组的密码

（e.g. 1. gpasswd workgroup 设置用户组workgroup的密码

2. gpasswd -A zhangsan workgroup将zhangsan账户设置为用户组workgroup的管理员）

* + 1. 6 vi编辑器及shell编程
       1. 6.1 vi 文件名 显示文件内容、编辑文件

（e.g. 1, vi my.txt 在当前目录下，若文件my.txt存在则显示它的内容并等待用户的编辑命令，若my.txt不存在则创建该文件并显示一个空白的文件

相应的命令：

（1）‘i’：切换到插入文本模式，在光标左侧输入正文；

（2）‘a’：切换到插入文本模式，在光标右侧输入正文；

（3）‘o’：切换到插入文本模式，在光标所在行的下一行添加新行；

（4）按‘esc’从插入模式(i/a/o)进入vi命令模式 ）

（5） x：删除光标所在字 dw：删除光标后一个单词；dd：删除光标所在行

（6）yy：复制整行 yw：复制光标所在的单词 nyy：复制包括当前行在内的n行

（7）: s/str1/str2/ 用字符串str2替换当前行中首次出现的字符串str1

: s/str1/str2/gc 用字符串str2替换当前行中所有出现的字符串str1，并依次提示是否替换

: s/str1/str2/g 将文档中的所有str1替换成str2

: m,n s/str1/str2/g 将文档中第m到n行中的所有str1替换成str2

* + - 1. 6.2 shell脚本文件

1. 使用vi编辑文件
2. 保存文件 【:wq!】
3. 将文件赋予可执行的权限 【chomod 755 文件名】
4. 运行文件 【方式一： sh filename.sh 方式二： ./filename.sh】
   1. 一个shell脚本文件的例子:

文件名：abc.sh，其内容如下：

#!/bin/bash

# a simple shell script example

sayhello()

{

echo “enter your name:”

read name (读取来自键盘的输入变量信息)

echo “Hello $name”

}

echo “programme starts here”

sayhello

echo “programme ends.”

* + - 1. 6.4 shell 中的特殊符号

（1） # 注释符 【除了 **#!/bin/bash**中的‘#’以外，其他的‘#’都是起注释作用的】

（2） **$** 美元符

作用：使其后的普通字符成为变量名

例子：$a 表示变量a的值；$后的字符长度超过1个时，最好用{}括起来，如${value}

（3） **‘’** 单引号

作用：单引号括起来的字符 全部被看成普通字符；

例子： echo ‘my $shell’ ， 其输出为： my $shell

**（4） “”** 双引号

作用：双引号内的内容，除了$ 、\（转义符）、`（倒引号）保留特殊功能外，其它字符均作为普通字符输出

例子： echo “my home is $HOME” 命令的输出结果为：my home is /home/user

1. **``** 倒引号 （数字1键旁边的那个键）

作用：倒引号内的字符串均被当做shell命令行来解释执行

例子： echo “my home is **`pwd`**” 输出结果为： my home is /home/user

1. **\**  反斜线

作用：将特殊字符变成普通字符

例子： echo “Filename is No\$\\*” 输出结果： Filename is No$\*

* + - 1. 6.5 变量

1. 本地变量

说明：局部变量，只在创建它们的shell中使用，可在shell程序中任意使用和修改它们；

1. 变量的定义：变量=值 （例子，name=welcome）
2. 变量的引用： $变量 （例子，${name}）
3. 变量的清除：unset 变量名 （例子，unset name）
4. set： 显示本地所有的变量
5. readonly: 设置只读变量 （例子， free=work; echo $free 此时输出work

readonly free; free=www 由于将free设置成只读的变量，所以再将www赋给变量free的值时，会提示free是只读变量，不能对其值进行修改）

1. 环境变量

说明：系统环境的一部分，不必去定义它们

常用的环境变量：

1. HOME (使用者的家目录) b. SHELL （目前环境所使用的SHELL）
2. PWD （用户当前工作目录的路径） d. HISTSIZE PATH （下达过指令的数目）

修改PATH环境变量，使脚本不用加路径，直接输入文件名即可运行，例子：a) mkdir shdir && cd shdir b) vi hello c) chmod 755 hello d) cd

e) export PATH=$PATH:$HOME/shdir

由于将hello文件所在的目录添加到了环境变量PATH中，所以再任何目录下（即除目录shdir外的其它目录）输入sh hello即可执行该文件

1. 内部变量

说明：linux提供的一种特殊类型的变量，这类变量在程序中用来做出判断，在shell程序内这类变量的值是不能修改的。例子：

shell程序名: exam.sh 其内容如下：

#!/bin/bash

echo “当前程序的名字是：$0”;

echo “共输入了$#个参数，全部参数是：$@，第一个参数是：$1”

echo “当前进程的pid是：$$”

echo “执行完毕，返回$?”

输入： ./exam.sh hello world ok 回车后，输入结果为：

当前程序的名字是：./exam.sh

共输入了3个参数，全部参数是：hello world ok，第一个参数是：hello

当前进程的pid是：2425

执行完毕，返回0

* + - 1. 6.6 read, echo 输入输出

1. read –s –n1 –p “Yes (Y) or not (N)?” answer （用于从键盘读取一个字符，不回显； ‘-n1’表示一个字符，‘-s’表示输入的字符不会提示，即不回显）
2. read var1 var2 （输入变量var1和var2）
3. echo –e “a\tb\tc\nd\te\tf” （2行3列显示 ‘-e’：启用反斜线控制字符的转换 ‘\t’表格跳位键 ‘\n’换行符），输出结果为：

a b c

d e f

* + - 1. 6.7 {} 花括号的使用

1） echo {a,b,c} 输出： a b c

2) echo user{1,5,8} 输出： user1,user5,user8

3) echo {0..10} 输出：0 1 2 3 4 5 6 7 8 9 10

4） mkdir {dir1,dir2,dir3}; ls –ld dir{1,2,3}

* + - 1. 6.8 分支语句 if-then-else 和case

1. 使用test命令 （test 条件表达式）
2. 使用括号 （[ 条件表达式 ]）

**例子**： 删除文件的时候，为了避免出粗，可以先测试文件是否存在，如果确实存在，再执行删除操作

1. fname=file1
2. if test –e $fname 或（if [ -e $fname ]）
3. then
4. rm –f $fname
5. Echo “file $fname has been removed!”
6. Fi

**例子2** 检查从命令行输入的文件是否存在

check.sh 该程序的内容如下

1. #!/bin/bash
2. if [ $# -ne 1 ] 【$# 表示输入参数的数目】
3. then
4. echo “usage - $0 file-name”
5. exit 1
6. fi
7. if [ -f $1 ] 【$1 表示输入的第一个参数的值】
8. then echo “$1 file exists”
9. else
10. echo “sorry, $1 file does not exist”
11. fi

**例子3**

eg.sh 程序 比较输入的两个整数是否相等，其具体内容如下

#!/bin/bash

echo “enter the first integer:”

read firstnum

echo “enter the second integer:”

read secondnum

if [ “$firstnum” –gt “$secondnum” ]

then echo “$firstnum is greater than $secondnum”

elif [“$firstnum” –lt “$secondnum” ]

then echo “$firstnum is less than $secondnum”

else echo “$firstnum equals to $secondnum”

fi

**例子4**

eg2.sh 该程序根据当前系统中的时间（主要是根据小时）来给出一个问候（上午/下午/晚上好）；其具体内容如下

#!/bin/bash

hour=`date +%H` (注意**`**是倒引号)

case $hour in

0[1-9] | 1[01] )

echo “good morning!” ;; (注意：每个条件分支后必须有两个分号结尾)

1[2-7] )

echo “good afternoon!” ;;

\* )

echo “good evening!” ;;

esac

* + - 1. 6.9 循环结构 （for、 while、until）

例子1

输出100内10的倍数：

for i in {1..9}

do

echo `expr $i \\* 10` 或者 $((i\*10))

done

例子2 求从1加到100的和

sum=0

for ((i=1;i<=100;i++))

do

sum=$((sum+i))

done

echo $sum

**例子3**

for x in paper pencil pen

do

echo “the value of variable x is: $x”

sleep 1

done

例子4 （编写一个名为chname的程序，将当前目录下所有的.txt文件更名为.doc文件）

#!/bin/bash

for file in \*.txt

do

leftname=`basename $file .txt` 【basename：从后面的文件名中（即$file）剥去指定的后缀（即.txt）；例如 basename file1.txt .txt的执行结果是：file1】

mv $flie $leftname.doc

done

例子 5

i=1

sum=0

while [ $i –le 100 ]

do

sum= `expr $sum+$i` 【或者sum=$((sum+i))】

i=`expr $i+1` 【或者i=$((i+1))】

done

echo “the result is $sum”

**例子6**

批量增加20个用户，用户名称为userN，N的范围是1到20

#!/bin/bash

num=1

while [ $num –le 20 ]

do

useradd user${num}

num=$((num+1))

done

**例子7** 将上面创建的20个用户批量删除

num=20

until [ $num –eq 0 ]

do

userdel user${num}

num=$((num-1))

done