Linux操作系统

9 shell流程控制

主讲：杨东平
中国矿大计算机学院

流程控制语句

- Ø单分支 if 语句
- Ø双分支 if 语句
- Ø多分支 if 语句
- Øcase 语句
- Øfor 循环
- Øwhile 循环和 until循环

单分支 if 语句

Ø语法格式1：

```
if [ 条件判断式 ]
then
   条件成立时执行的命令
   …
fi
```

Ø语法格式2(写成一行，适用于终端命令提示符) ：

```
if [ 条件表达式 ]; then 条件成立时执行的命令; fi
```

Ø 几个注意点：

- v if 语句使用 fi 结尾(其它流程控制语句存在类似情况)
- v [条件判断式] 就是使用 test 命令判断，所以中括号和条件判断式之间必须有空格
- v 写在一行上时的各部分之间需要用 “;” 分隔

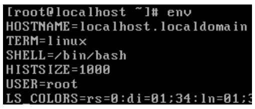
单分支 if 语句(续)

Ø例1(视频：18 流控：JudgeRoot.sh)：判断登录的用户是否 root

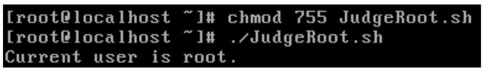
```
#! /bin/bash
# JudgeRoot.sh

test=$(env | grep "USER" | cut -d "=" -f2)

if [ "$test" == root ]
then
   echo "Current user is root."
fi
```



```
[root@localhost ~]# env
HOSTNAME=localhost.localdomain
TERM=linux
SHELL=/bin/bash
HISTSIZE=1000
USER=root
LS_COLORS=rs=0:di=01;34:ln=01;3
```



```
[root@localhost ~]# chmod 755 JudgeRoot.sh
[root@localhost ~]# ./JudgeRoot.sh
Current user is root.
```

单分支 if 语句(续)

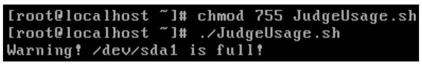
Ø例2(视频：19 流控：JudgeUsage.sh)：判断分区使用率

```
#! /bin/bash
# JudgeUsage.sh

rate=$(df -h | grep “/dev/sda1” | awk '{print $5}' | cut -d "%" -f1)
if [ $rate –ge 7 ]
then
   echo "Warning! /dev/sda1 is full!"
fi
```

awk 命令把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理

```
[root@localhost ~]# df -h
Filesystem                    Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root  [illegible]
tmpfs                         [illegible]                /dev/shm
/dev/sda1                     [illegible]                /boot
```



```
[root@localhost ~]# chmod 755 JudgeUsage.sh
[root@localhost ~]# ./JudgeUsage.sh
Warning! /dev/sda1 is full!
```

双分支 if 条件语句：if else 语句

Ø语法：

```
if [ 条件判断式 ]
then
   条件成立时执行的程序段
else
   条件不成立时执行的另一个程序段
fi
```

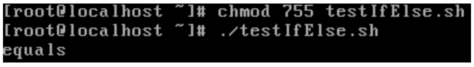
双分支 if 条件语句：if else 语句(续)

Ø例(视频：20 流控：testIfElse.sh)：

```
#! /bin/bash
# testIfElse.sh

num1=$[2*3]
num2=$[1+5]
if test $[num1] -eq $[num2]   # if [ $[num1] -eq $[num2] ]

then
    echo "equals"
else
    echo "not equals"
fi
```



```
[root@localhost ~]# chmod 755 testIfElse.sh
[root@localhost ~]# ./testIfElse.sh
equals
```

多分支 if 条件语句

Ø语法：

```
if [ 条件判断式1 ]
then
    当条件判断式1成立时执行的程序段1
elif [ 条件判断式2 ]
then
    当条件判断式2 成立时执行的程序段2
…省略更多条件…
else
    当所有条件都不成立时最后执行的程序段
fi
```

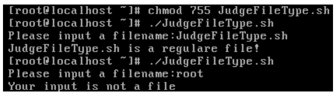
多分支 if 条件语句(续)

Ø 例(视频：21 流控：JudgeFileType.sh)：判断用户输入的是什么文件

```
#! /bin/bash
# JudgeFileType.sh

read -p "Please input a filename:" file
#接收键盘的输入并赋予变量file

if [ -z "$file" ]
#判断file变量是否为空
then
   echo "Error,Please input a filename"
   exit 1
elif [ ! -e "$file" ]
#判断file的值是否存在
then
   echo "Your input is not a file!"
   exit 2
elif [ -f  "$file" ]
# 判断file是否为普通文件
then
   echo "$file is a regulare file!"
elif [ -d "$file" ]
#判断file是否为目录文件
then
   echo "$file is a directory!"
else
   echo "$file is an other file!"
fi
```



```
[root@localhost ~]# chmod 755 JudgeFileType.sh
[root@localhost ~]# ./JudgeFileType.sh
Please input a filename:JudgeFileType.sh
JudgeFileType.sh is a regulare file!
[root@localhost ~]# ./JudgeFileType.sh
Please input a filename:root
Your input is not a file
```

多分支 case 条件语句

Øcase 语句匹配一个值或一个模式，如果匹配成功，执行相匹配的命令

Ø语法：

```
case 值 in
模式1)
   变量值匹配模式1时执行的命令序列
;;
模式2)
   变量值匹配模式2时执行的命令序列
;;
……
*)
   变量值与以上模式都不匹配时默认执行的命令序列
;;
esac
```

多分支 case 条件语句(续)

Ø语法说明：

v case 行尾必须为单词 in， 每个模式必须以右括号")" 结束

v 双分号 ;; 表示命令序列结束，类似与其他语言中的 break

v 匹配模式中可使用方括号表示一个连续的范围，如 [0-9]；使用竖杠符号 "|" 表示或

v 最后的 "*)" 表示默认模式，当使用前面的各种模式均无法匹配该变量时，将执行 "*)" 后的命令序列

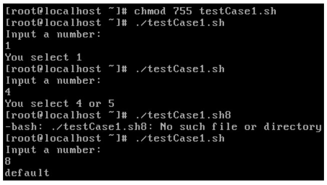
v case 后的值可以为变量或常数

多分支 case 条件语句(续)

Ø例(视频：22 流控：testCase1.sh)：提示输入 1 到 4，与每一种模式进行匹配

```
#! /bin/bash
# testCase1.sh

echo 'Input a number:'
read Num
case $Num in
   1)  echo 'You select 1'
   ;;
   2)  echo 'You select 2'
   ;;
   3)  echo 'You select 3'
   ;;
   4|5)  echo 'You select 4 or 5'
   ;;
   *)  echo 'default'
   ;;
esac
```



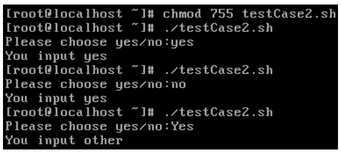
```
[root@localhost ~]# chmod 755 testCase1.sh
[root@localhost ~]# ./testCase1.sh
Input a number:
1
You select 1
[root@localhost ~]# ./testCase1.sh
Input a number:
4
You select 4 or 5
[root@localhost ~]# ./testCase1.sh8
-bash: ./testCase1.sh8: No such file or directory
[root@localhost ~]# ./testCase1.sh
Input a number:
8
default
```

多分支 case 条件语句(续)

Ø例(视频：23 流控：testCase2.sh)：判断用户输入

```
#! /bin/bash
# testCase2.sh

read -p "Please choose yes/no:"  -t 30 choose
case $choose in
"yes")
   echo "You input yes"
   ;;
"no") echo "you input no"
   ;;
*) echo "you input other"
   ;;
esac
```



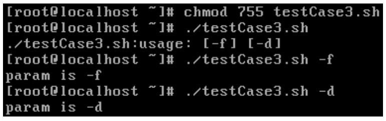
```
[root@localhost ~]# chmod 755 testCase2.sh
[root@localhost ~]# ./testCase2.sh
Please choose yes/no:yes
You input yes
[root@localhost ~]# ./testCase2.sh
Please choose yes/no:no
You input yes
[root@localhost ~]# ./testCase2.sh
Please choose yes/no:Yes
You input other
```

多分支 case 条件语句(续)

Ø例(视频：24 流控：testCase3.sh)：命令行输入

```
#!/bin/bash
# testCase3.sh

option=$1
case ${option} in
  -f) echo "param is -f"
     ;;
  -d) echo "param is -d"
     ;;
  *)
     echo "$0:usage: [-f ] | [ -d ]"
     exit 1  #退出码
     ;;
esac
```



```
[root@localhost ~]# chmod 755 testCase3.sh
[root@localhost ~]# ./testCase3.sh
./testCase3.sh:usage: [-f] [-d]
[root@localhost ~]# ./testCase3.sh -f
param is -f
[root@localhost ~]# ./testCase3.sh -d
param is -d
```

for 循环

Ø语法1：

```
  for 变量 in 值1 值2 值3…
     do
        程序块
     done
```

Ø 语法2：

```
  for 变量 in `命令`
     do
        程序块
     done
```

Ø 语法3：

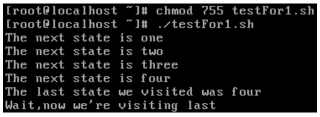
```
  for ((初始值; 循环控制条件; 变量变化))
     do
        程序块
     done
```

for 循环(续)

Ø例(视频：25 流控：testFor1.sh)：对列表中的值进行迭代

```
#! /bin/bash
# testFor1.sh

for t in one two three four
  do
     echo "The next state is $t"
  done
echo "The last state we visited was $t"
t=last
echo "Wait,now we're visiting $t"
```



```
[root@localhost ~]# chmod 755 testFor1.sh
[root@localhost ~]# ./testFor1.sh
The next state is one
The next state is two
The next state is three
The next state is four
The last state we visited was four
Wait,now we're visiting last
```

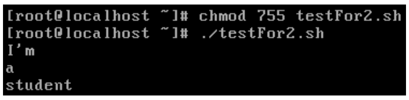
若遇到 \ ' , . 等特殊符号时，可以使用反斜杠 \ 来转义特殊字符，或者用双引号括起来

for 循环(续)

Ø例(视频：26 流控：testFor2.sh)：若遇到 \ ' , . 等特殊符号时，可以使用反斜杠 \ 来转义特殊字符，或者用双引号括起来

```
#! /bin/bash
# testFor2.sh

for t in "I'm" a student
  do
    echo "$t"
  done
```



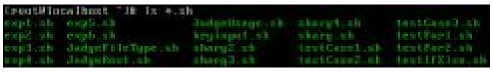
```
[root@localhost ~]# chmod 755 testFor2.sh
[root@localhost ~]# ./testFor2.sh
I'm
a
student
```

for 循环(续)

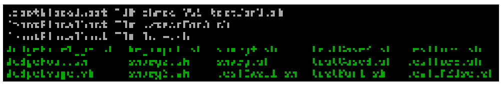
Ø例(视频:27 流控：testFor3.sh)：批量删除文件

```
#! /bin/bash
# testFor3.sh

for file in `ls exp?.sh`
 do
   rm -f $file
 done
```



```
[root@localhost ~]# ls *.sh
exp1.sh  exp5.sh               judgeUsage.sh  sharg1.sh     testCase3.sh
exp2.sh  exp6.sh               keyinput.sh    sharg.sh      testFor1.sh
exp3.sh  JudgeFileType.sh      sharg2.sh      testCase1.sh  testFor2.sh
exp4.sh  JudgeRoot.sh          sharg3.sh      testCase2.sh  testIfElse.sh
```



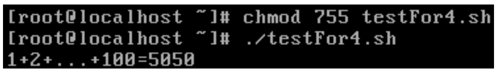
```
[root@localhost ~]# chmod 755 testFor3.sh
[root@localhost ~]# ./testFor3.sh
[root@localhost ~]# ls *.sh
JudgeFileType.sh  keyinput.sh  sharg1.sh     testCase2.sh  testFor2.sh
JudgeRoot.sh      sharg2.sh    sharg.sh      testCase3.sh  testFor3.sh
judgeUsage.sh     sharg3.sh    testCase1.sh  testFor1.sh   testIfElse.sh
```

for 循环(续)

Ø例(视频：28 流控：testFor4.sh)求1~100的和

```
#! /bin/bash
# testFor4.sh

sum=0
for (( i=1; i<=100; i++ ))
  do
      sum=$(( $sum + $i ))
  done
echo "1+2+…100=$sum"
```



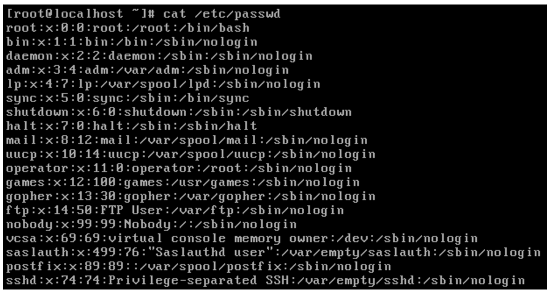
```
[root@localhost ~]# chmod 755 testFor4.sh
[root@localhost ~]# ./testFor4.sh
1+2+...+100=5050
```

综合示例

Ø例(视频：)批量添加指定数量的用户

```
#! /bin/bash
# testFor5.sh

read -p "Please input user: name:"  -t  30 name
read -p "Please input the number of users:" -t 30 num
read -p "Please input the password of users:" -t 30 pass
if [ ! -z  "$name" -a ! -z "$num" -a ! -z "$pass" ]
  then
      y=$(echo $num | sed 's/[0-9]//g')
      if [ -z "$y" ]
        then
            for (( i=1; i<=$num; i=i+1 ))
                do
                    /usr/sbin/useradd $name$i &> /dev/null
                    echo $pass | /usr/bin/passwd --stdin $name$i &> /dev/null
                 done
        fi
fi
```



```
[root@localhost ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
```

while 循环

Ø语法：

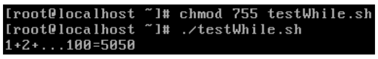
```
while [ 条件判断式 ]
  do
     条件判断式成立时执行的命令
  done
```

while 循环(续)

Ø例(视频：29 流控：testWhile.sh)：计算1+2+…100之和

```
#! /bin/bash
# testWhile.sh

i=1
s=0
while [ $i –le 100 ]
  do
      s=$(( $s + $i ))
      i=$(( $i +1 ))
  done
echo "1+2+…100=$s"
```



```
[root@localhost ~]# chmod 755 testWhile.sh
[root@localhost ~]# ./testWhile.sh
1+2+...100=5050
```

until 循环

Ø语法：

```
until [ 条件判断式 ]
do
     条件判断式不成立时执行的命令
done
```

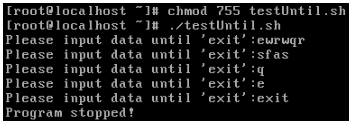
Ø注意：循环体执行到条件判断式成立时为止

until 循环(续)

- 例(视频：30 流控：testUntil.sh)从键盘输入数据，直到输入 ”exit” 为止

```
#! /bin/bash
# testUntil.sh

read -p "Please input data until 'exit':"  EXIT
until [ $EXIT = “exit” ]
  do
      read -p "Please input data until 'exit':"  EXIT
  done
echo "Program stopped!"
```



```
[root@localhost ~]# chmod 755 testUntil.sh
[root@localhost ~]# ./testUntil.sh
Please input data until 'exit':ewrwqr
Please input data until 'exit':sfas
Please input data until 'exit':q
Please input data until 'exit':e
Please input data until 'exit':exit
Program stopped!
```