

###课程目标

- ==掌握for循环语句的基本语法结构==
- ==掌握while和until循环语句的基本语法结构==
- 能会使用RANDOM产生随机数
- 理解嵌套循环

一、随机数

bash默认有一个\$RANDOM的变量 默认是0~32767。使用set |grep RANDOM 查看上一次产生的随机数

```
echo $RANDOM
```

产生0~1之间的随机数

```
echo $[${RANDOM}%2]
```

产生0~2之间的随机数

```
echo $[${RANDOM}%3]
```

产生0~3之间的随机数

```
echo $[${RANDOM}%4]
```

。。。。

产生0~9内的随机数

```
echo $[${RANDOM}%10]
```

产生0~100内的随机数

```
echo $[${RANDOM}%101]
```

产生50~100之内的随机数

```
echo $[${RANDOM}%51+50]
```

产生三位数的随机数

```
echo $[${RANDOM}%900+100]
```

实战案例1

1. 写一个脚本，产生一个phonenum.txt文件，随机产生以139开头的手机号1000个，每个一行。

分析：

1. 产生1000个电话号码，脚本需要循环1000次
2. 139+8位，后8位随机产生，可以让每一位数字都随机产生，\${RANDOM%10} 0-9
3. 将随机产生的数字分别保存到变量里，然后加上139保存到文件里

```
#!/bin/bash
```

```
# random phonenum
```

```
# 循环1000次产生电话号码并保存到文件
```

```
for i in {1..1000}
```

```
do
```

```
    n1=${RANDOM%10}
```

```
    n2=${RANDOM%10}
```

```
    n3=${RANDOM%10}
```

```
    n4=${RANDOM%10}
```

```
    n5=${RANDOM%10}
```

```
n6=$((RANDOM%10))
n7=$((RANDOM%10))
n8=$((RANDOM%10))
echo "139$n1$n2$n3$n4$n5$n6$n7$n8" >> phonenum.txt
done
```

```
#!/bin/bash
# random phonenum
# 循环1000次产生电话号码
for ((i=1;i<=1000;i++))
do
    n1=$((RANDOM%10))
    n2=$((RANDOM%10))
    n3=$((RANDOM%10))
    n4=$((RANDOM%10))
    n5=$((RANDOM%10))
    n6=$((RANDOM%10))
    n7=$((RANDOM%10))
    n8=$((RANDOM%10))
    echo "139$n1$n2$n3$n4$n5$n6$n7$n8" >> phonenum.txt
done
```

```
#!/bin/bash
i=1
while [ $i -le 1000 ]
do
    n1=$((RANDOM%10))
    n2=$((RANDOM%10))
    n3=$((RANDOM%10))
    n4=$((RANDOM%10))
    n5=$((RANDOM%10))
    n6=$((RANDOM%10))
    n7=$((RANDOM%10))
    n8=$((RANDOM%10))
    echo "139$n1$n2$n3$n4$n5$n6$n7$n8" >> phonenum.txt
    let i++
done
```

continue:继续，跳过本次循环，执行下一次循环

break:打断，执行循环体外的代码**do..done**外

exit:退出程序

```
#!/bin/bash
for i in {1..1000}
do
    n1=$((RANDOM%10))
    n2=$((RANDOM%10))
    n3=$((RANDOM%10))
    n4=$((RANDOM%10))
    n5=$((RANDOM%10))
    n6=$((RANDOM%10))
    n7=$((RANDOM%10))
    n8=$((RANDOM%10))
    echo "139$n1$n2$n3$n4$n5$n6$n7$n8" >> phonenum.txt
done
```

```
#!/bin/bash
```

```

#create phone num file
for ((i=1;i<=1000;i++))
do
    n1=${RANDOM%10}
    n2=${RANDOM%10}
    n3=${RANDOM%10}
    n4=${RANDOM%10}
    n5=${RANDOM%10}
    n6=${RANDOM%10}
    n7=${RANDOM%10}
    n8=${RANDOM%10}
    echo "139$n1$n2$n3$n4$n5$n6$n7$n8" | tee -a phonenum.txt
done

#!/bin/bash
count=0
while true
do
    n1=${RANDOM%10}
    n2=${RANDOM%10}
    n3=${RANDOM%10}
    n4=${RANDOM%10}
    n5=${RANDOM%10}
    n6=${RANDOM%10}
    n7=${RANDOM%10}
    n8=${RANDOM%10}
    echo "139$n1$n2$n3$n4$n5$n6$n7$n8" | tee -a phonenum.txt && let count++
    if [ $count -eq 1000 ];then
        break
    fi
done

```

2. 在上面的1000个手机号里抽奖5个幸运观众，显示出这5个幸运观众。但只显示头3个数和尾号的4个数，中间的都用*代替

思路：

- 确定幸运观众所在的行 随机生成 RANDOM \${RANDOM%1000+1}
- 将电话号码提取出来 head 和 tail
- 显示前3个和后4个数到屏幕 最后将电话号码输出到屏幕 echo \${电话号码部分}

```

#!/bin/bash
#定义变量
phone=/shell04/phonenum.txt
for ((i=1;i<=5;i++))
do
    #定位幸运观众所在行号
    line=`wc -l $phone | cut -d' ' -f1`
    luck_line=${RANDOM%$line+1}
    #取出幸运观众所在行的电话号码
    luck_num=`head -$luck_line $phone|tail -1`
    #显示到屏幕
    echo "139****${luck_num:7:4}"
    echo $luck_num >> luck.txt
    #删除已经被抽取的幸运观众号码
    sed -i "/$luck_num/d" $phone
done

```

```
#!/bin/bash
file=/shell04/phonenum.txt
for i in {1..5}
do
    file_num=`wc -l $file |cut -d' ' -f1`
    line=`echo [$RANDOM%$file_num+1]`
    luck=`head -n $line $file|tail -1`
    echo "139****${luck:7:4}" && echo $luck >> /shell04/luck_num.txt
done
```

```
#!/bin/bash
for ((i=1;i<=5;i++))
do
    file=phonenum.txt
    line=`cat phonenum.txt |wc -l` 1000
    luckline=$((RANDOM%line+1))
    phone=`cat $file|head -$luckline|tail -1`
    echo "幸运观众为:139****${phone:7:4}"
done
```

或者

```
#!/bin/bash
# choujiang
phone=phonenum.txt
for ((i=1;i<=5;i++))
do
    num=`wc -l phonenum.txt |cut -d' ' -f1`
    line=`echo [$RANDOM$num+1]`
    luck=`head -$line $phone |tail -1`
    sed -i "/$luck/d" $phone
    echo "幸运观众是:139****${luck:7:4}"
done
```

3. 批量创建5个用户，每个用户的密码为一个随机数

思路：

- 循环5次创建用户
- 产生一个密码文件来保存用户的随机密码
- 从密码文件中取出随机密码赋值给用户

```
#!/bin/bash
#crate user and set passwd
#产生一个保存用户名和密码的文件
echo user0{1..3}:itcast[$RANDOM%9000+1000]#@~|tr ' ' '\n'>> user_pass.file
#循环创建5个用户
for ((i=1;i<=5;i++))
do
    user=`head -$i user_pass.file|tail -1|cut -d: -f1`
    pass=`head -$i user_pass.file|tail -1|cut -d: -f2`
    useradd $user
```

```

    echo $pass|passwd --stdin $user
done

或者
for i in `cat user_pass.file`
do
    user=`echo $i|cut -d: -f1`
    pass=`echo $i|cut -d: -f2`
    useradd $user
    echo $pass|passwd --stdin $user
done

#!/bin/bash
#crate user and set passwd
#产生一个保存用户名和密码的文件
echo user0{1..3}:itcast[$(RANDOM%9000+1000)]#@~|tr ' ' '\n'|tr ':' ' ' >>
user_pass.file
#循环创建5个用户
while read user pass
do
    useradd $user
    echo $pass|passwd --stdin $user
done < user_pass.file

pwgen工具产生随机密码:
[root@server shell104]# pwgen -cn1 12
Meep5ob1aesa
[root@server shell104]# echo user0{1..3}:$(pwgen -cn1 12)
user01:Baqu9haipho user02:Feiphoh7moo4 user03:eilahj5eth2R
[root@server shell104]# echo user0{1..3}:$(pwgen -cn1 12)|tr ' ' '\n'
user01:eiwashuZo5hi
user02:eiDeih7aim9k
user03:aeBahwien8co

```

二、嵌套循环

一个==循环体==内又包含另一个**完整**的循环结构，称为循环的嵌套。在外部循环的每次执行过程中都会触发内部循环，直至内部完成一次循环，才接着执行下一次的外部循环。for循环、while循环和until循环可以**相互**嵌套。

demo1: 打印如下图案

```

1
12
123
1234
12345

X轴:
for ((i=1;i<=5;i++));do echo -n $i;done
Y轴:
负责打印换行

#!/bin/bash
for ((y=1;y<=5;y++))

```

```

do
    for ((x=1;x<=$y;x++))
    do
        echo -n $x
    done
done
echo
done

#!/bin/bash
for ((y=1;y<=5;y++))
do
    x=1
    while [ $x -le $y ]
    do
        echo -n $x
        let x++
    done
done
echo
done

```

demo2: 打印如下图案

```

5
54
543
5432
54321

Y轴: 打印换行
X轴: 打印数字 5-1

#!/bin/bash
y=5
while (( $y >= 1 ))
do
    for ((x=5;x>=$y;x--))
    do
        echo -n $x
    done
    echo
    let y--
done

#!/bin/bash
for (( y=5;y>=1;y--))
do
    for (( x=5;x>=$y;x--))
    do
        echo -n $x
    done
    echo
done

#!/bin/bash

```

```

y=5
while [ $y -ge 1 ]
do
    for ((x=5;x>=$y;x--))
    do
        echo -n $x
    done
    echo
    let y--
done

```

```

#!/bin/bash
y=1
until (( $y >5 ))
do
    x=1
    while (( $x <= $y ))
    do
        echo -n ${6-$x}
        let x++
    done
    echo
    let y++
done

```

课后打印:

```

54321
5432
543
54
5

```

课堂练习：打印九九乘法表（三种方法）

```

1
12
123
1234
12345

for ((y=1;y<=5;y++))
do
    for ((x=1;x<=$y;x++))
    do
        echo -n $x
    done
    echo
done

1*1=1

1*2=2   2*2=4

```

1*3=3 2*3=6 3*3=9

1*4=4 2*4=8 3*4=12 4*4=16

1*5=5 2*5=10 3*5=15 4*5=20 5*5=25

1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36

1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49

1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64

1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81

Y轴：循环9次，打印9行空行

X轴：循环次数和Y轴相关；打印的是X和Y轴乘积 `$[] $()`

```
#!/bin/bash
for ((y=1;y<=9;y++))
do
    for ((x=1;x<=$y;x++))
    do
        echo -ne "$x*$y=${x*$y}\t"
    done
    echo
    echo
done
```

```
#!/bin/bash
y=1
while [ $y -le 9 ]
do
    x=1
    while [ $x -le $y ]
    do
        echo -ne "$x*$y=${x*$y}\t"
        let x++
    done
    echo
    echo
    let y++
done
```

或者

```
#!/bin/bash
for i in `seq 9`
do
    for j in `seq $i`
    do
        echo -ne "$j*$i=${i*$j}\t"
    done
    echo
    echo
done
```

或者


```
#!/bin/bash
y=1
until [ $y -gt 9 ]
do
    x=1
    until [ $x -gt $y ]
    do
        echo -ne "$x*$y=$[ $x*$y ]\t"
        let x++
    done
    echo
    let y++
done
```

三、阶段性总结

1. 变量定义

普通变量定义：

变量名=值 shell变量默认可以赋予任何类型
 \$变量名 \${变量名} \${变量名:从第几个字符开始:截取几个字符}
 unset 变量名

交互式：

read 变量名
 -p
 -t
 -s
 -n

数组定义：

array=(var1 var2 var3 ...)
 array[0]=var1
 array[1]=var2
 array[2]=var3

普通数组：数组的索引是整数

定义关联数组

关联数组：索引是字符串

获取数组里的元素：

\${array[*]}
 \${array[2]}
 \${array[@]:1:2}
 \${!array[@]} 获取数组的索引号（下标）
 \${#array[@]} 获取数组索引号的个数

定义有类型的变量：

declare
 -i
 -x
 -a
 -A

2. 循环语句

for:

列表循环、非列表循环、类C风格 循环次数已知

while:

条件为真，进入循环，条件为假，退出循环 循环次数跟条件有关

until:

条件为假，进入循环，条件为真，退出循环 循环次数跟条件有关

3. 影响shell程序的内置命令

exit 退出整个程序

break 结束当前循环，或跳出本层循环

continue 忽略本次循环剩余的代码，直接进行下一次循环

shift 使位置参数向左移动，默认移动1位，可以使用**shift 2**

以下脚本都能够实现用户自定义输入数字，然后脚本计算和：

```
[root@MissHou shell104]# cat shift.sh
```

```
#!/bin/bash
```

```
sum=0
```

```
while [ $# -ne 0 ]
```

```
do
```

```
let sum=$sum+$1
```

```
shift
```

```
done
```

```
echo sum=$sum
```

```
[root@MissHou shell104]# cat for3.sh
```

```
#!/bin/bash
```

```
sum=0
```

```
for i
```

```
do
```

```
let sum=$sum+$i
```

```
done
```

```
echo sum=$sum
```

```
:
```

```
true
```

```
false
```

4. 补充扩展expect

expect 自动应答 tcl语言

需求1: A远程登录到server上什么都不做

```
#!/usr/bin/expect
```

```
# 开启一个程序
```

```
spawn ssh root@10.1.1.1
```

```
# 捕获相关内容
expect {
    "(yes/no)?" { send "yes\r";exp_continue }
    "password:" { send "123456\r" }
}
interact    //交互
```

脚本执行方式:

```
# ./expect1.sh
# /shell04/expect1.sh
# expect -f expect1.sh
```

1) 定义变量

```
#!/usr/bin/expect
set ip 10.1.1.2
set pass 123456
set timeout 5
spawn ssh root@$ip
expect {
    "yes/no" { send "yes\r";exp_continue }
    "password:" { send "$pass\r" }
}
interact
```

2) 使用位置参数

```
#!/usr/bin/expect
set ip [ lindex $argv 0 ]
set pass [ lindex $argv 1 ]
set timeout 5
spawn ssh root@$ip
expect {
    "yes/no" { send "yes\r";exp_continue }
    "password:" { send "$pass\r" }
}
interact
```

需求2: A远程登录到server上操作

```
#!/usr/bin/expect
set ip 10.1.1.1
set pass 123456
set timeout 5
spawn ssh root@$ip
expect {
    "yes/no" { send "yes\r";exp_continue }
    "password:" { send "$pass\r" }
}

expect "#"
send "rm -rf /tmp/*\r"
send "touch /tmp/file{1..3}\r"
send "date\r"
send "exit\r"
expect eof
```

需求3: shell脚本和expect结合使用，在多台服务器上创建1个用户

```
[root@server shell04]# cat ip.txt
10.1.1.1 123456
10.1.1.2 123456
```

1. 循环
2. 登录远程主机-->ssh-->从ip.txt文件里获取IP和密码分别赋值给两个变量
3. 使用expect程序来解决交互问题

```
#!/bin/bash
# 循环在指定的服务器上创建用户和文件
while read ip pass
do
    /usr/bin/expect <<-END &>/dev/null
    spawn ssh root@$ip
    expect {
        "yes/no" { send "yes\r";exp_continue }
        "password:" { send "$pass\r" }
    }
    expect "#" { send "useradd yy1;rm -rf /tmp/*;exit\r" }
    expect eof
END
done < ip.txt
```

```
#!/bin/bash
cat ip.txt|while read ip pass
do
    {

        /usr/bin/expect <<-HOU
        spawn ssh root@$ip
        expect {
            "yes/no" { send "yes\r";exp_continue }
            "password:" { send "$pass\r" }
        }
        expect "#"
        send "hostname\r"
        send "exit\r"
        expect eof
        HOU

    }&

done
wait
echo "user is ok...."
```

或者

```
#!/bin/bash
while read ip pass
do
    {
```

```

/usr/bin/expect <<-HOU
spawn ssh root@$ip
expect {
    "yes/no" { send "yes\r";exp_continue }
    "password:" { send "$pass\r" }
}
expect "#"
send "hostname\r"
send "exit\r"
expect eof
HOU

}&
done<ip.txt
wait
echo "user is ok...."

```

四、综合案例

实战案例2

写一个脚本，将跳板机上yunwei用户的公钥推送到局域网内可以ping通的所有机器上

说明：主机和密码文件已经提供

10.1.1.1:123456

10.1.1.2:123456

案例分析

- 关闭防火墙和selinux
- 判断ssh服务是否开启（默认ok）
- ==循环判断给定密码文件里的哪些IP是可以ping通== ip pass
- ==判断IP是否可以ping通——>\$?——>流程控制语句==
- ==密码文件里获取主机的IP和密码保存变量== ip pass
- ==判断公钥是否存在——>不存在创建它==
- ==ssh-copy-id 将跳板机上的yunwei用户的公钥推送到远程主机——>expect解决交互==
- ==将ping通的主机IP单独保存到一个文件==
- ==测试验证==

代码拆分

```

1. 判断yunwei用户的公钥是否存在
[ ! -f /hoem/yunwei/.ssh/id_rsa ] && ssh-keygen -P '' -f ./id_rsa

2. 获取IP并且判断是否可以ping通
1) 主机密码文件ip.txt
10.1.1.1:123456
10.1.1.2:123456
2) 循环判断主机是否ping通
tr ':' ' ' < ip.txt|while read ip pass
do
    ping -c1 $ip &>/dev/null
    if [ $? -eq 0 ];then
        推送公钥
    fi
done

```

```
fi
done
```

3. 非交互式推送公钥

```
/usr/bin/expect <<-END &>/dev/null
spawn ssh-copy-id root@$ip
expect {
    "yes/no" { send "yes\r";exp_continue }
    "password:" { send "$pass\r" }
}
expect eof
END
```

最终实现

环境准备:

jumper-server 有yunwei用户

yunwei用户sudo授权:

visudo

Allow root to run any commands anywhere

root ALL=(ALL) ALL

yunwei ALL=(root) NOPASSWD:ALL,!/sbin/shutdown,!/sbin/init,!/bin/rm -rf /

解释说明:

- 1) 第一个字段yunwei指定的是用户: 可以是用户名, 也可以是别名。每个用户设置一行, 多个用户设置多行, 也可以将多个用户设置成一个别名后再进行设置。
- 2) 第二个字段ALL指定的是用户所在的主机: 可以是ip, 也可以是主机名, 表示该sudo设置只在该主机上生效, ALL表示在所有主机上都生效! 限制的一般都是本机, 也就是限制使用这个文件的主机; 一般都指定为"ALL"表示所有的主机, 不管文件拷到那里都可以用。比如: 10.1.1.1=...则表示只在当前主机生效。
- 3) 第三个字段(root)括号里指定的也是用户: 指定以什么用户身份执行sudo, 即使用sudo后可以享有所有root账号下的权限。如果要排除个别用户, 可以在括号内设置, 比如ALL=(ALL,!oracle,!pos)。
- 4) 第四个字段ALL指定的是执行的命令: 即使用sudo后可以执行所有的命令。除了关机和删除根内容以外; 也可以设置别名。NOPASSWD: ALL表示使用sudo的不需要输入密码。
- 5) 也可以授权给一个用户组

%admin ALL=(ALL) ALL 表示admin组里的所有成员可以在任何主机上以任何用户身份执行任何命令

+++++脚本实现:

```
#!/bin/bash
```

```
#判断公钥是否存在
```

```
[ ! -f /home/yunwei/.ssh/id_rsa ] && ssh-keygen -P '' -f ~/.ssh/id_rsa
```

```
#循环判断主机是否ping通, 如果ping通推送公钥
```

```
tr ':' ' ' < /shell04/ip.txt|while read ip pass
```

```
do
```

```
{
```

```
    ping -c1 $ip &>/dev/null
```

```
    if [ $? -eq 0 ];then
```

```
        echo $ip >> ~/ip_up.txt
```

```
        /usr/bin/expect <<-END &>/dev/null
```

```
        spawn ssh-copy-id root@$ip
```

```

        expect {
            "yes/no" { send "yes\r";exp_continue }
            "password:" { send "$pass\r" }
        }
    expect eof
END
fi

}&
done
wait
echo "公钥已经推送完毕，正在测试...."
#测试验证
remote_ip=`tail -1 ~/ip_up.txt`
ssh root@$remote_ip hostname &>/dev/null
test $? -eq 0 && echo "公钥成功推送完毕"

```

实战案例3

写一个脚本，统计web服务的不同==连接状态==个数

```

#!/bin/bash
#count_http_80_state
#统计每个状态的个数
declare -A array1
states=`ss -ant|grep 80|cut -d' ' -f1`
for i in $states
do
    let array1[$i]++
done
#通过遍历数组里的索引和元素打印出来
for j in ${!array1[@]}
do
    echo $j:${array1[$j]}
done

```

五、课后作业

- 1、将/etc/passwd里的用户名分类，分为管理员用户，系统用户，普通用户。
- 2、写一个倒计时脚本，要求显示离2019年1月1日（元旦）的凌晨0点，还有多少天，多少时，多少分，多少秒。
- 3、写一个脚本把一个目录内的所有==空文件==都删除，最后输出删除的文件的个数。