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==
- 能会使用随机数RANDOM
- 理解嵌套循环

一、for循环

1. 语法结构

1.1 列表循环

列表for循环：用于将一组命令执行**已知的次数**，下面给出了for循环语句的基本格式：

```
for variable in {list}
     do
          command
          command
          …
     done
或者
for variable in a b c
     do
         command
         command
     done
```

语法结构举例说明：

```
1045  for var in {1..10};do echo $var;done
1046  for var in 1 2 3 4 5;do echo $var;done
1047  for var in `seq 10`;do echo $var;done
1048  for var in $(seq 10);do echo $var;done
1049  for var in {0..10..2};do echo $var;done
1050  for var in {2..10..2};do echo $var;done
1051  for var in {10..1};do echo $var;done
1052  for var in {10..1..-2};do echo $var;done
1055  for var in `seq 10 -2 1`;do echo $var;done
```

1.2 不带列表循环

不带列表的for循环执行时由**用户指定参数和参数的个数**，下面给出了不带列表的for循环的基本格式：

```
for variable
    do
        command
        command
        …
   done
```

语法结构举例说明：

```
#!/bin/bash
for var
do
echo $var
done

echo "脚本后面有$#个参数"
```

1.3 类C风格的for循环

```
for(( expr1;expr2;expr3 ))
    do
        command
        command
        …
    done

expr1：定义变量并赋初值
expr2：决定是否进行循环（条件）
expr3：决定循环变量如何改变，决定循环什么时候退出
```

语法结构举例说明：

```
1068  for ((i=1;i<=5;i++));do echo $i;done
1069  for ((i=1;i<=10;i+=2));do echo $i;done
1070  for ((i=2;i<=10;i+=2));do echo $i;done
```

2. 示例说明

demo1：计算1到100的奇数之和，方法不止一种

```
思路：
1. 定义一个变量来保存奇数的和     sum=0
2. 找出1-100的奇数，保存到另一个变量里  i
3. 从1-100中找出奇数后，再相加，然后将和赋值给sum变量
4. 遍历完毕后，将sum的值打印出来

#!/bin/bash
```

```
# 计算1-100的奇数和
sum=0
for i in {1..100..2}
do
    sum=$[$sum+$i]
done
echo "1-100的奇数和是:$sum"


#!/bin/bash
sum=0
for ((i=1;i<=100;i++))
do
    if [ $[$i%2] -ne 0 ];then
    let sum=sum+$i
    fi
done
echo "1-100的奇数和是:$sum"


#!/bin/bash
sum=0
for ((i=1;i<=100;i++))
do
    [ $[$i%2] -eq 0 ] && true  || let sum=sum+$i
done
echo "1-100的奇数和是:$sum"
```

延伸:

```
true    真
:       真
false    假
```

方法1:

```
#!/bin/bash
sum=0
for i in {1..100..2}
do
    sum=$[$i+$sum]
done
echo "1-100的奇数和为:$sum"
```

方法2:

```
#!/bin/bash
sum=0
for ((i=1;i<=100;i+=2))
do
    let sum=$i+$sum
done

echo "1-100的奇数和为:$sum"
```

```
方法3:
#!/bin/bash
sum=0
for ((i=1;i<=100;i++))
do
    if [ $[$i%2] -ne 0 ];then
    let sum=$sum+$i
    fi
或者
test $[$i%2] -ne 0 && let sum=$sum+$i

done
echo "1-100的奇数和为:$sum"

方法4:
sum=0
for ((i=1;i<=100;i++))
do
    if [ $[$i%2] -eq 0 ];then
    continue
    else
    let sum=$sum+$i
    fi
done
echo "1-100的奇数和为:$sum"

#!/bin/bash
sum=0
for ((i=1;i<=100;i++))
do
    test $[$i%2] -eq 0 && continue || let sum=sum+$i
done
echo "1-100的奇数和是:$sum"
```

循环控制：

循环体： ==do....done==之间的内容

continue：继续。表示==循环体==内下面的代码不执行，重新开始下一次循环 break：打断。马上停止执行本次循环，执行==循环体==后面的代码 exit：表示直接跳出程序

```
[root@server shell03]# cat for5.sh
#!/bin/bash
for i in {1..5}
do
    test $i -eq 2 && break || touch /tmp/file$i
done
echo hello hahahah
```

demo2：输入一个正整数,判断是否为质数(素数) 质数：只能被1和它本身整除的数叫质数。 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

思路：
0. 让用户输入一个数，保存到一个变量里 read num
1、如果能被其他数整除就不是质数—>$num%$i 是否等于0
2、如果输入的数是1或者2取模根据上面判断又不符合，所以先排除1和2
3、测试序列从2开始，输入的数是4—>得出结果$num不能和$i相等，并且$num不能小于$i

```
#!/bin/bash
read -p "请输入一个正整数字:" number

[ $number -eq 1 ] && echo "$number不是质数" && exit
[ $number -eq 2 ] && echo "$number是质数" && exit

for i in `seq 2 $[$number-1]`
    do
     [ $[$number%$i] -eq 0 ] && echo "$number不是质数" && exit
    done
echo "$number是质数" && exit

bash -x for6.sh
```

demo3：批量加5个新用户，以u1到u5命名，并统一加一个新组，组名为class,统一改密码为123

思路：
1. 添加用户的命令 useradd
2. 根据题意，判断该脚本循环5次来添加用户 for
3. 给用户设置密码，应该放到循环体里面

```
#!/bin/bash
#判断class组是否存在
cut -d: -f1 /etc/group|grep -w class &>/dev/null
[ $? -ne 0 ] && groupadd class

#循环增加用户，循环次数5次，for循环,给用户设定密码
for ((i=1;i<=5;i++))
do
    useradd u$i -G class
    echo 123|passwd --stdin u$i
done



#!/bin/bash
grep -w class /etc/group &>/dev/null
test $? -ne 0 && groupadd class
或者
groupadd class &>/dev/null
```

```
for ((i=1;i<=5;i++))
do
useradd -G class u$i && echo 123|passwd --stdin u$i
done
```

3. 课堂练习

1、批量新建5个用户stu1~stu5，要求这几个用户的家目录都在/rhome.提示：需要判断该目录是否存在

```
#!/bin/bash
#判断/rhome家目录是否存在，如果不存在则创建它
[ -d /rhome ] && true || mkdir /rhome
或者
[ ! -d /rhome ] && mkdir /rhome
#批量创建用户
for ((i=1;i<=5;i++))
do
    useradd stu$i -d /rhome/stu$i && echo 123|passwd --stdin stu$i
done
echo -e "以下用户:stu1\nstu2\nstu3\nstu4\nstu5创建成功"

思考:
批量创建10个用户，前5个用户的uid是1001~1005;后5个用户的家目录/rhome/user01~user05
```

2、写一个脚本，局域网内，把能ping通的IP和不能ping通的IP分类，并保存到两个文本文件里（如果例举整个网段的254个IP花的时间比较长，可以只分类10个ip10.1.1.1~10） 这只是一个局域网内机器检查通讯的一个思路。

```
#!/bin/bash
#清空原来ip文件里的列表
>/tmp/ip_ok
>/tmp/ip_down
ip=10.1.1
#循环去ping局域网内的主机
for ((i=1;i<=10;i++))
do
ping -c1 $ip.$i &>/dev/null
test $? -eq 0 && echo "$ip.$i" |tee -a /tmp/ip_ok || echo "$ip.$i" |tee -a /tmp/ip_down
done

思考：以上方法可以实现，但是速度很慢，希望并行执行

#!/bin/bash
#清空原来ip文件里的列表
>/tmp/ip_ok
>/tmp/ip_down
ip=10.1.1
#循环去ping局域网内的主机

for ((i=1;i<=10;i++))
```

```
do
{
ping -c1 $ip.$i &>/dev/null
test $? -eq 0 && echo "$ip.$i" |tee -a /tmp/ip_ok || echo "$ip.$i" |tee -a /tmp/ip_down
}&
done
wait
echo "ip is ok..."

注意:
{程序}&表示将程序放到后台并行执行，如果需要等待程序执行王完毕再进行下面内容，需要加wait
```

3、输入一个年份，判断是否是润年（能被4整除但不能被100整除，或能被400整除的年份即为闰年。）

```
#!/bin/bash
read -p "Please input year:(2017)" year
if [ $[$year%4] -eq 0 -a $[$year%100] -ne 0 ];then
    echo "$year is leap year"
elif [ $[$year%400] -eq 0 ];then
    echo "$year is leap year"
else
    echo "$year is not leap year"
fi
```

二、until循环

条件为真就退出循环；条件为假就死循环

```
until expression   [ 1 -eq 1 ]  (( 1 >= 1 ))
    do
        command
        command
        ...
    done

until false
do
    echo hellalla
done

==========================================
补充:
true;echo $?
:;echo $?
以上代表真；以下代表假
false;echo $?
```

示例1：使用until打印1-5

```
//打印12345
[root@server shell03]# for ((i=1;i<=5;i++));do echo $i;done
[root@server shell03]# i=1;until [ $i -gt 5 ];do echo $i;let i++;done
//打印54321
[root@server shell03]# for ((i=5;i>=1;i--));do echo $i;done
[root@server shell03]# i=5;until (( $i < 1 ));do echo $i;let i--;done
[root@server shell03]# for ((i=1;i<=5;i++));do echo $[6-$i];done
[root@server shell03]# i=1;until [ $i -gt 5 ];do echo $[6-$i];let i++ ;done
```

示例2：使用until批量创建用户，用户user01~user03,并且家目录为/rhome/user01...

```
#!/bin/bash
#create users
#判断用户的家目录/rhome是否存在，不存在创建它
dir=/rhome
[ ! -d $dir ] && mkdir $dir -p
i=1
until [ $i -gt 3 ]
do
    useradd -d $dir/user0$i user0$i && echo 123|passwd --stdin user0$i && let i++
done
```

三、while循环

条件为真就进入死循环；条件为假就退出循环

```
while expression  [ 1 -eq 1 ] 或者 (( 1 > 2 ))
  do
     command
     command
     ...
 done
```

示例：

写一个30秒同步一次时间，向同步服务器10.1.1.250的脚本,如果同步失败，则进行邮件报警,每次失败都报警;同步成功,也进行邮件通知,但是成功100次才通知一次。

```
ntp-server:10.1.1.250
web-cluster:10.1.1.1 10.1.1.2 10.1.1.3
```

分析：
1. 计划任务时间最小是分钟，所以不能直接使用crontab完成，那么需要写脚本
2. 脚本每隔30s同步，该脚本是死循环，并且sleep 30来控制间隔

```
#!/bin/bash
#定义相关变量
ntp-server=10.1.1.250
count=0
until false
do
    ntpdate $ntp-server &>/dev/null
    if [ $? -ne 0 ];then
    echo "系统时间同步失败" |mail -s "check system date"  root@localhost
    else
        let count++
        if [ $count -eq 100 ];then
            echo "系统时间同步成功100次"|mail -s "check system date" root@localhost && count=0
        fi
    fi
sleep 30
done
```

```
#!/bin/bash
#ntpdate
count=0
ip=10.1.1.1
while true
do
    rdate -s $ip &>/dev/null
    if [ $? -ne 0 ];then
        echo "时间同步失败，请检查..." |mail -s "check system times" root@MissHou.itcast.cc
    else
        let count++
        if [ $[$count%100] -eq 0 ];then
        echo "时间同成功" |mail -s "check system times" root@MissHou.itcast.cc
        count=0
        fi
    fi
    sleep 30
done
```

或者

```
#!/bin/bash
#ntpdate
count=0
ip=10.1.1.1
while true
do
    rdate -s $ip &>/dev/null

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

```
        echo "时间同步失败，请检查..." |mail -s "check system times" root@MissHou.itcast.cc
    else
        let count++
        [ $[$count%100] -eq 0 ] && echo "时间同成功" |mail -s "check system times"
root@MissHou.itcast.cc && count=0
    fi
    sleep 30
done
```

总结：

用至少三种方法打印1~5和5-1

```

```

四、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。

```

#!/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

或者
#!/bin/bash
#create phone num file
for ((i=1;i<=1000;i++))
do
    n1=`echo $[$RANDOM%10]`
    n2=`echo $[$RANDOM%10]`
    n3=`echo $[$RANDOM%10]`
    n4=`echo $[$RANDOM%10]`
    n5=`echo $[$RANDOM%10]`
    n6=`echo $[$RANDOM%10]`
    n7=`echo $[$RANDOM%10]`
    n8=`echo $[$RANDOM%10]`
    echo "139$n1$n2$n3$n4$n5$n6$n7$n8" |tee -a phonenum.txt
done
或者
#!/bin/bash
count=0
while true
do
    n1=`echo $[$RANDOM%10]`
    n2=`echo $[$RANDOM%10]`
    n3=`echo $[$RANDOM%10]`
    n4=`echo $[$RANDOM%10]`
    n5=`echo $[$RANDOM%10]`
    n6=`echo $[$RANDOM%10]`
    n7=`echo $[$RANDOM%10]`
    n8=`echo $[$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

- 将电话号码提取出来
- 显示前3个和后4个数到屏幕

```
#!/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 shell04]# pwgen -cn1 12
Meep5ob1aesa
[root@server shell04]# echo user0{1..3}:$(pwgen -cn1 12)
user01:Bahqu9haipho user02:Feiphoh7moo4 user03:eilahj5eth2R
[root@server shell04]# echo user0{1..3}:$(pwgen -cn1 12)|tr ' ' '\n'
user01:eiwaShuZo5hi
user02:eiDeih7aim9k
user03:aeBahwien8co
```

五、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

分析：
1. 使用循环语句能够打印出12345
2. 外部循环只打印换行，内部循环打印12345数字
y轴
x轴

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

    do
```

```
        echo -n $x
    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
echo
done

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

demo2：打印如下图案

```
5
54
543
5432
54321

#!/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

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
```

```
#!/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
echo
let y++
done
```

六、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

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

定义关联数组
关联数组：索引是字符串
```

```
1185  declare -a
1186  declare -A
1187  declare -A array1
1188  array1=([linux]=1 [java]=2 [C]= )
1189  echo ${array1[@]}
1190  echo ${!array1[@]}
1191  declare -A array2
1192  array2[Aa]=1
1193  array2[Bb]=2
1194  echo ${array2[*]}
1195  let array1[linux]++
```

```
1196  echo ${array1[linux]}
1197  let array1[C]++
1198  echo ${array1[C]}
1199  let array1[C]++
1200  echo ${array1[C]}
1201  let array1[php]++
1202  echo ${array1[php]}
1203  let array1[php]++
1204  echo ${array1[php]}
1205  declare -A
1206  let array2[Cc]++
1207  declare -A
1208  let array2[Cc]++
1209  declare -A
1210  declare -a
1211  let array[3]++
1212  declare -a
1213  let array[3]++
1214  declare -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 shell03]# cat shift.sh
#!/bin/bash
sum=0
while [ $# -ne 0 ]
do
let sum=$sum+$1
shift
done
echo sum=$sum
[root@MissHou shell03]# 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

```
ssh 免密码登录

node1:yunwei
node2:root

node1:yunwei  免密码登录到node2机器上以root用户
ssh root@node2

1. 在node1机器上的yunwei用户加目录里生成一对秘钥
2. 将yunwei用户生成的公钥远程拷贝到node2的root家目录

都需要交互

jumper-server: yunwei    100台机器上的pos01

expect 自动应答  tcl语言  语法格式

需求1: server远程登录到node2上什么都不做
1) 定义变量
#!/usr/bin/expect
set ip 10.1.1.2
set pass 123
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：server远程登录到node2上操作

```
#!/usr/bin/expect
set ip 10.1.1.2
set pass 123
set timeout 5
spawn ssh root@$ip
expect {
    "yes/no" { send "yes\r";exp_continue }
    "password:" { send "$pass\r" }
}

#interact
expect "#"
send "hostname\r"
send "useradd stu01\r"
send "pwd\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
10.1.1.250 111111
10.1.1.2 123

#!/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...."
```

七、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

10.1.1.1~10.1.1.254

```
分析：
环境：
jumper-server   有yunwei用户
app1-appn  局域网内所有可以ping通的机器


1. 在跳板上创建yunwei用户，并且生成一对秘钥
2. 检测当前局域网中哪些ip是能ping通哪些是不能ping通          循环语句并发的去检查
3. 在脚本中所有的交互动作需要用到expect实现

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 /


#!/bin/bash
```

```
#检查局域网中哪些ip是可以ping通，并保存到一个文件
ip1=10.1.1
for ((i=1;i<=10;i++))
do
    {
      ping -c1 $ip1.$i &>/dev/null
      [ $? -eq 0 ] && echo "$ip1.$i" >> ip_up.txt
    }&
done
wait

#yunwei用户生成一对秘钥（有交互）
[ ! -f ~/.ssh/id_rsa ] && ssh-keygen -P '' -f ~/.ssh/id_rsa

#将yunwe用户的公钥远程拷贝到指定的服务器 100 循环
##判断expect程序是否安装
{
rpm -q expect
[ $? -ne 0 ] && sudo yum -y install expect
while read ip2
do
    /usr/bin/expect<<-EOF
    spawn ssh-copy-id  root@$ip2
    expect {
      "yes/no" {send "yes\r";exp_continue}
      "password:" {send "123\r"}
      }
    expect eof
    EOF
done<ip_up.txt
} &>/dev/null

#测试验证
remote_ip=`tail -1 ip_up.txt`
ssh root@$remote_ip hostname
[ $? -eq 0 ] && echo "公钥推送完毕...."



#!/bin/bash
#push publickey to aap-servers
#将局域网内可以ping通的主机ip保存到一个文件
> ip_up.txt
for i in {2..10}
do
    {
    ip=10.1.1.$i
    ping -c1 $ip &>/dev/null
    [ $? -eq 0 ] && echo $ip |tee -a ip_up.txt
    }&          //并行放到后台运行
done
wait            //等待进程结束

#将yunwei用户目录下的公钥推送到可以ping的服务器上
```

```
#1. 判断yunwei用户下有没有公钥
[ ! -f ~/.ssh/id_rsa.pub ] && ssh-keygen -P "" -f ~/.ssh/id_rsa

#2.将id_rsa.pub公钥远程推送到指定服务器
#2.1 判断expect程序是否安装，没安装则安装它
{
rpm -q expect
[ $? -ne 0 ] && sudo yum -y install expect

for remote_ip in `cat ip_up.txt`
do
    /usr/bin/expect <<-EOF
    spawn ssh-copy-id root@$remote_ip
    expect {
        "yes/no" { send "yes\r";exp_continue }
        "password:" { send "123\r" }
    }
     expect eof
    EOF
done
} &>/dev/null
#测试验证
test_ip=`tail -1 ip_up.txt`
ssh root@$test_ip hostname
test $? -eq 0 && echo "公钥推送成功。"
```

优化:

```
#!/bin/bash
[ ! -f ~/.ssh/id_rsa.pub ] && ssh-keygen -P "" -f ~/.ssh/id_rsa
rpm -q expect
[ $? -ne 0 ] && sudo yum -y install expect
for i in {130..140}
do
        ip=192.168.44.$i
        ping -c1 $ip
        [ $? -ne 0 ] && continue
        echo $ip >>ip_up.txt
        /usr/bin/expect <<-EOF
        spawn ssh-copy-id root@$ip
        expect {
        "yes/no" { send "yes\r";exp_continue }
        "password:" { send "123456\r" }
        }
        expect eof
        EOF
done
#测试验证
test_ip=`tail -1 ip_up.txt`
ssh root@$test_ip hostname
test $? -eq 0 && echo "公钥推送成功。"
```

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 STATE
states=`ss -ant|grep 80|cut -d' ' -f1`
for i in $states
do
        let STATE[$i]++
done
#通过遍历数组里的索引和元素打印出来
for j in ${!STATE[@]}
do
        echo $j:${STATE[$j]}
done
```

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2018年10月1日（国庆节）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。

```

```