本机课程目标

- 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

一、for循环语句

关键词：爱的魔力转圈圈😇

1. for循环语法结构

㈠ 列表循环

> 列表for循环：用于将一组命令执行**已知的次数**

- **基本语法格式**

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

- **举例说明**

```
# for var in {1..10};do echo $var;done
# for var in 1 2 3 4 5;do echo $var;done
# for var in `seq 10`;do echo $var;done
# for var in $(seq 10);do echo $var;done
# for var in {0..10..2};do echo $var;done
# for var in {2..10..2};do echo $var;done
# for var in {10..1};do echo $var;done
# for var in {10..1..-2};do echo $var;done
# for var in `seq 10 -2 1`;do echo $var;done
```

㈡ 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**

- **基本语法格式**

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

- **举例说明**

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

echo "脚本后面有$#个参数"
```

㈢ 类C风格的for循环

- **基本语法结构**

```
for(( expr1;expr2;expr3 ))
   do
      command
      command
      …
   done
for (( i=1;i<=5;i++))
   do
      echo $i
   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
```

- **举例说明**

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

2. 应用案例

㈠ 脚本计算1-100奇数和

① 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=遍历出来的奇数
3. 从1-100中找出奇数后，再相加，然后将和赋值给变量 循环变量 for
4. 遍历完毕后，将sum的值打印出来

② 落地实现（条条大路通罗马）

```
#!/bin/env bash
# 计算1-100的奇数和
# 定义变量来保存奇数和
sum=0

#for循环遍历1-100的奇数，并且相加，把结果重新赋值给sum

for i in {1..100..2}
do
   let sum=$sum+$i
done
#打印所有奇数的和
echo "1-100的奇数和是:$sum"


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"
```

③ 循环控制语句

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

㈡ 判断所输整数是否为质数

质数(素数)： 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

① 思路

1. 让用户输入一个数，保存到一个变量里 `read -p "请输入一个正整数:" num`
2. 如果能被其他数整除就不是质数——> `$num%$i` 是否等于0 `$i=2到$num-1`
3. 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
4. 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`

② 落地实现

```
#!/bin/env bash
#定义变量来保存用户所输入数字
read -p "请输入一个正整数字:" number

#先排除用户输入的数字1和2
[ $number -eq 1 ] && echo "$number不是质数" && exit
[ $number -eq 2 ] && echo "$number是质数" && exit

#循环判断用户所输入的数字是否质数

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"

优化思路：没有必要全部产生2~$[$number-1]序列，只需要产生一半即可。

更好解决办法：类C风格完美避开了生成序列的坑
for (( i=2;i<=$[$number-1];i++))
do
        [ $[$number%$i] -eq 0 ] && echo "$number不是质数" && exit

done
echo "$number是质数"
```

㈢ 批量创建用户

需求：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

① 思路

1. 添加用户的命令 `useradd -G class`
2. 判断class组是否存在 `grep -w ^class /etc/group` 或者 `groupadd class`
3. 根据题意，判断该脚本循环5次来添加用户 `for`
4. 给用户设置密码，应该放到循环体里面

② 落地实现

```
#!/bin/env bash
#判断class组是否存在
grep -w ^class /etc/group &>/dev/null
test $? -ne 0 && groupadd class

#循环创建用户
for ((i=1;i<=5;i++))
do
   useradd -G class u$i
   echo 123|passwd --stdin u$i
done
#用户创建信息保存日志文件
```

方法一：

```
#!/bin/bash
#判断class组是否存在
grep -w class /etc/group &>/dev/null
[ $? -ne 0 ] && groupadd class
#批量创建5个用户
for i in {1..5}
do
   useradd -G class u$i
   echo 123|passwd --stdin u$i
done
```

方法二：

```
#!/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
```

方法三：

```
#!/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

㈠ 批量创建用户

需求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.

```
#!/bin/bash
#判断/rhome是否存在
[ -f /rhome ] && mv /rhome /rhome.bak
test ! -d /rhome && mkdir /rhome
或者
[ -f /rhome ] && mv /rhome /rhome.bak || [ ! -d /rhome ] && mkdir /rhome

#创建用户，循环5次
for ((i=1;i<=5;i++))
do
   useradd -d /rhome/stu$i stu$i
   echo 123|passwd --stdin stu$i
done
```

㈡ 局域网内脚本检查主机网络通讯

需求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

以10.1.1.1~10.1.1.10为例

```
10.1.1.1~10.1.1.254

#!/bin/bash
#定义变量
ip=10.1.1
#循环去ping主机的IP
for ((i=1;i<=10;i++))
do
   ping -c1 $ip.$i &>/dev/null
   if [ $? -eq 0 ];then
      echo "$ip.$i is ok" >> /tmp/ip_up.txt
   else
      echo "$ip.$i is down" >> /tmp/ip_down.txt
   fi
   或者
   [ $? -eq 0 ] && echo "$ip.$i is ok" >> /tmp/ip_up.txt || echo "$ip.$i is down" >>
/tmp/ip_down.txt
done

[root@server shell03]# time ./ping.sh

real    0m24.129s
user    0m0.006s
sys     0m0.005s
```

延伸扩展：shell脚本并发

```
并行执行:
{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完毕再进行下面内容，需要加wait

#!/bin/bash
#定义变量
ip=10.1.1
#循环去ping主机的IP
for ((i=1;i<=10;i++))
do
{

        ping -c1 $ip.$i &>/dev/null
        if [ $? -eq 0 ];then
                echo "$ip.$i is ok" >> /tmp/ip_up.txt
        else
                echo "$ip.$i is down" >> /tmp/ip_down.txt
        fi
}&
done
wait
echo "ip is ok...."

[root@server ~]# time ./ping.sh
ip is ok...

real    0m3.091s
user    0m0.001s
sys     0m0.008s
```

㈢ 判断闰年

需求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
```

4. 总结

- FOR循环语法结构
- FOR循环可以结合条件判断和流程控制语句

    - do ……done 循环体
    - 循环体里可以是命令集合，再加上条件判断以及流程控制
- 控制循环语句
    - continue 继续，跳过本次循环，继续下一次循环
    - break 打断，跳出循环，执行循环体外的代码
    - exit 退出，直接退出程序

二、while循环语句

特点：条件为真就进入循环；条件为假就退出循环

1. while循环语法结构

```
while 表达式
   do
      command...
   done

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

循环打印1-5数字

```
FOR循环打印:
for ((i=1;i<=5;i++))
do
   echo $i
done

while循环打印:
i=1
while [ $i -le 5 ]
do
   echo $i
   let i++
done
```

2. 应用案例

㈠ 脚本计算1-50偶数和

```
#!/bin/env bash
sum=0
for ((i=0;i<=50;i+=2))
```

```
do
   let sum=$sum+$i  (let sum=sum+i)
done
echo "1-50的偶数和为:$sum"


#!/bin/bash
#定义变量
sum=0
i=2
#循环打印1-50的偶数和并且计算后重新赋值给sum
while [ $i -le 50 ]
do
   let sum=$sum+$i
   let i+=2  或者 $[$i+2]
done
#打印sum的值
echo "1-50的偶数和为:$sum"
```

㈡ 脚本同步系统时间

① 具体需求

1. 写一个脚本，30秒同步一次系统时间，时间同步服务器10.1.1.1
2. 如果同步失败，则进行邮件报警,每次失败都报警
3. 如果同步成功,也进行邮件通知,但是成功100次才通知一次

② 思路

1. 每隔30s同步一次时间，该脚本是一个死循环 while 循环
2. 同步失败发送邮件 1) ntpdate 10.1.1.1 2) rdate -s 10.1.1.1
3. 同步成功100次发送邮件 定义变量保存成功次数

③ 落地实现

```
#!/bin/env bash
# 该脚本用于时间同步
NTP=10.1.1.1
count=0
while true
do
   ntpdate $NTP &>/dev/null
   if [ $? -ne 0 ];then
      echo "system date failed" |mail -s "check system date"  root@localhost
   else
      let count++
      if [ $count -eq 100 ];then
      echo "systemc date success" |mail -s "check system date"  root@localhost && count=0
      fi
   fi
sleep 30
done
```

```
#!/bin/bash
#定义变量
count=0
ntp_server=10.1.1.1
while true
do
   rdate -s $ntp-server &>/dev/null
   if [ $? -ne 0 ];then
      echo "system date failed" |mail -s 'check system date'  root@localhost
   else
      let count++
      if [ $[$count%100] -eq 0 ];then
      echo "system date successfull" |mail -s 'check system date'  root@localhost &&
count=0
      fi
   fi
sleep 3
done
```

以上脚本还有更多的写法，课后自己完成

三、until循环

特点：条件为假就进入循环；条件为真就退出循环

1. until语法结构

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

打印1-5数字

```
i=1
while [ $i -le 5 ]
do
   echo $i
   let i++
done

i=1
until [ $i -gt 5 ]
do
   echo $i
   let i++
done
```

2. 应用案例

㈠ 具体需求

1. 使用until语句批量创建10个用户，要求stu1—stu5用户的UID分别为1001—1005；
2. stu6~stu10用户的家目录分别在/rhome/stu6—/rhome/stu10

㈡ 思路

1. 创建用户语句 `useradd -u|useradd -d`
2. 使用循环语句(until)批量创建用户 `until循环语句结构`
3. 判断用户前5个和后5个 `条件判断语句`

㈢ 落地实现

```
#! /bin/env bash
if [ -d /rhome ];then
    echo "/rhome目录已存在"
else
    mkdir /rhome
    echo "/rhome不存在，已完成创建"
fi

i=1
until [ $i -gt 10 ]
do
        if [ $i -le 5 ];then
                useradd -u $[1000+$i] stu$i
                echo 123|passwd --stdin stu$i

        else
                useradd -d /rhome/stu$i stu$i
                echo 123|passwd --stdin stu$i
        fi
let i++
done
```

```
================================================

#!/bin/bash
i=1
until [ $i -gt 10 ]
do
   if [ $i -le 5 ];then
      useradd -u $[1000+$i] stu$i && echo 123|passwd --stdin stu$i
   else
      [ ! -d /rhome ] && mkdir /rhome
      useradd -d /rhome/stu$i stu$i && echo 123|passwd --stdin stu$i
   fi
let i++
done
```

四、课后作业

1. 判断/tmp/run目录是否存在，如果不存在就建立，如果存在就删除目录里所有文件
2. 输入一个路径，判断路径是否存在，而且输出是文件还是目录，如果是链接文件，还得输出是 有效的连接还是无效的连接
3. 交互模式要求输入一个ip，然后脚本判断这个IP 对应的主机是否 能ping 通，输出结果类似于： Server 10.1.1.20 is Down! 最后要求把结果邮件到本地管理员root@localhost mail01@localhost
4. 写一个脚本/home/program，要求当给脚本输入参数hello时，脚本返回world,给脚本输入参数world时，脚本返回hello。而脚本没有参数或者参数错误时，屏幕上输出“usage:/home/program hello or world”
5. 写一个脚本自动搭建nfs服务