

for 命令：

for i in 的各种用法：

```
for i in "file1" "file2" "file3"
for i in /boot/
for i in /etc/*.conf
for i in $(seq -w 10) -- 等宽的01-10
for i in {1..10}
for i in $(ls)
for i in $(cat file)
for i in "$@" -- 取所有位置参数，可简写为for i
注意：bash shell支持C式for循环
```

```
#!/bin/bash
i=$1
for ((i=1;i<=j;i++))
do
touch file$i && echo file $i is ok
done
```

复习

```
$@: 所有位置变量的内容
$#: 位置变量的个数
$0: 文件名
$: 所有位置变量的内容
```

编写脚本应该注意的事项：

开头指定使用什么shell，例如：bash, ksh, csh等  
脚本功能描述，使用方法，作者，版本，日期等  
变量名，函数名要有实际意义，函数名以动名词形式，第二个单词首字母要大写。例如：updateConfig()  
缩进统一用4个空格，不用TAB  
取变量值使用大括号，如\${varname}  
删除文件时，如果路径有变量的，要判断变量有值，如rm -f \${abc}/  
如果变量abc没有值，则会把根目录下的文件删除  
脚本中尽量不要使用cd变换目录  
函数中也要有功能描述，使用依法，版本，日期等  
函数的功能要单一，不要太复杂  
\$( )比`更好  
尽量不要使用多层if语句，而应该以case语句替代  
如果需要执行确定次数的循环，应该用for语句替代while语句  
输入的参数要有正确性判断  
多加注释，方便自己或他人阅读。

练习1：编写脚本清空所有arp缓存记录：

```
#!/bin/bash
for i in $(arp | tail -n +2 | tr -s ' ' | cut -d ' ' -f1)
do
arp -d $i
done
```

练习2：产生十个随机数：

方法1：

```
#for i in {0..9};do echo $RANDOM;done
```

方法2：

```
#for i in $(seq 10);do echo $RANDOM;done
```

练习3：倒数五秒：

```
#!/bin/bash
echo "准备倒数5秒："
for i in $(seq 5 -1 1)
do
echo -en "$i";sleep 1
done
echo -e "开始"
```

方法2：

```
#!/bin/bash
echo "准备倒数5秒："
for i in $(seq 5 -1 1)
do
echo -en "\b$i";sleep 1
done
echo -e "\b开始"
```

练习4：批量添加用户：

```
#!/bin/bash
for i in $(cat /root/users.txt --) 从列表文件读取文件名
do
```

```
useradd $i
echo "123456" | passwd --stdin $i --》通过管道指定密码字符串
done
```

练习：

查找出uid大于10000的用户，然后删除，必须使用for循环。

```
#!/bin/bash
u_uid=(`cat /etc/passwd | awk -F: '{print $3}'`)
u_name=(`cat /etc/passwd | awk -F: '{print $1}'`)
for i in `seq ${#u_uid[@]}`
do
    if (( ${u_uid[i-1]} > 10000 ))
    then
        userdel -r ${u_name[i-1]}&&echo "${u_name[i-1]} delete ok"
    fi
done
```

方法2：用正则找出大于10000的用户：

```
#cat /etc/passwd | egrep "1[0-9][4] | [2-9][5]"
```

例子：根据ip地址检查网络中存活的主机ip。

## ❖ 根据IP地址检查网络中存活的主机IP

```
#!/bin/bash
for ip in 192.168.1.{1..254}
do
    ping -c 1 -w1 "${ip}" &> /dev/null
done
arp -n | grep ether | tr -s ' ' | cut -d' ' -f1
```

```
#!/bin/bash
for ip in 192.168.1.{1..254}
do
    ( ping -c 1 -w1 "${ip}" &> /dev/null ) &
    #放到后台执行可以并行计算
done
wait #等待后台子进程执行完毕，传回最后一个命令的执行状态
arp -n | grep ether | tr -s ' ' | cut -d' ' -f1
```

break语句：（跳出循环）

在for、while、until等循环语句中，用于跳出当前所在的循环体，执行循环体后的语句

continue语句：（跳出本次循环）

在for、while、until等循环语句中，用于跳出循环体内余下的语句，重新判断条件以便执行下一次循环。

练习：使用for循环实现批量添加用户

```
#!/bin/bash
for i in $(cat /root/users.txt --》从列表文件读取文件名)
do
    useradd $i
    echo "123456" | passwd $i --stdin --》通过管道指定密码UNAME
done
```

（ps：判断用户是否存在：id命令）

位置变量

位置变量：\$n，但是大于9的位置参数要用{}括起来：\${10}

位置变量的作用：其实就是传递参数到脚本里

\$0 --》代表的是脚本自己的名字

（位置变量的最常用用法：bash 1.sh 变量1 变量2...）

预定义变量：

\$#：命令行中位置变量的个数

\$\*：所有位置变量的内容（较少使用）

\$@：所有位置变量的内容

\$0：当前执行的进程/程序名

\$\$：当前shell的PID值，echo \$\$；ps \$\$，常用作临时变量的后缀

\$?：上一条命令执行后返回的状态，当返回状态值为0时表示执行正常，非0值表示执行异常或出错

\$RANDOM：随机数，可以作为临时文件名

例：输出0-9以内的随机数--》echo \$((RANDOM%10))

输出1-10以内的随机数--》echo \$((RANDOM%10+1))

！\$：代表上一条命令的参数

！！：执行上一条命令

练习： 输出

\$1 is aa,

\$2 is bb,

\$3 is cc,

\$4 is dd,

\$5 is ee

答案:

```
#!/bin/bash

echo "there are $# arguments in this scripts"
N=1 --》变量N用来计数
for i in $@
do
    echo "$N is $i"
    ((N++))
done
```

PS:

ping 命令

```
-c 1 --》只ping一次。
-i 0.2--》第一个包和第二个包之间间隔0.2s
-w 2 --》只等待2s
```

例:

ping 172.30.132.123 &>/dev/null

重定向对于ping命令无用，执行成功\$?就返回0，不成功则返回1

根据IP地址检查网络中存活的主机IP(大范围的扫描)

```
#!/bin/bash
for r in 192.168.1.{1..254}
do
    ping -c1 -w1 "${ip}" &>/dev/null
done
arp -n|grep ether|tr -s ' '|cut -d' ' -f1
```

关于ping命令的一个最经典的脚本:

```
for i in {1..193}
do
    ( ping -c1 -i0.2 -w1 172.16.30.$i &>/dev/null
    if (( $?==0 ))
    then
        echo "172.16.30.$i up" >>2.txt
    else
        echo "172.16.30.$i down" >>3.txt
    fi )& --》这样就把这一段放到后台去执行了，大大加快了速度。
done
sleep 2
live_pc_num=`cat 2.txt|wc -l`
down_pc_num=`cat 3.txt|wc -l`
echo "there are $down_pc_num is down"
echo "there are $live_pc_num is up"
echo "list:"
cat 2.txt
rm -rf 2.txt 3.txt
```

break语句

典型的while循环:

```
#!/bin/bash
i=1
while : --》:等价于true
do
    echo "$i"
    ((i++))
    sleep 0.3
done
注: 这是个死循环，会一直执行下去
```

加上break，可以跳出循环

```
#!/bin/bash
i=1
while :
do
    echo "$i"
    (( i++ ))
    if (( i==20000 )) --》输出的只有1-19999
    then
        break
```

```
fi
done
```

小结

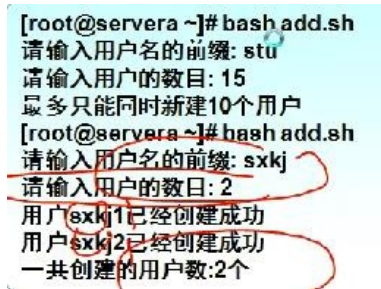
break: 跳出整个循环

exit: 跳出脚本

continue: 跳出本次循环，接着执行下一次循环

案例练习9:

批量添加用户并且满足以下要求:



```
[root@servera ~]# bash add.sh
请输入用户名的前缀: stu
请输入用户的数目: 15
最多只能同时新建10个用户
[root@servera ~]# bash add.sh
请输入用户名的前缀: sxxj
请输入用户的数目: 2
用户sxxj1已经创建成功
用户sxxj2已经创建成功
一共创建的用户数:2个
```

答案:

```
#!/bin/bash
read -p "请输入用户名的前缀: " a
read -p "请输入用户的数目: " num
if (( $num <= 10 ))
then
    n=0
    for i in `seq $num`
    do
        if useradd $a$i &>/dev/null
        then
            echo "用户$a$i创建成功!"
            (( n++ ))
            echo "123456"|passwd $a$i --stdin &>/dev/null
        fi
    done
    echo "一共创建的用户数: $n个"
else
    echo "最多只能创建10个用户啦!"
fi
```