Shell程序设计-循环

目录

目录

循环语句 - for

- for循环是最简单，也是最常用的循环语句。for循环通常用于遍历整个对象或者列表。
- 按照循环条件的不同，for循环语句可以分为以下三种：
  - 带列表的for循环
  - 不带列表的for循环
  - C风格的for循环

循环语句 - for

- 带列表的for循环语句

带列表的for循环通常用于将一组语句执行已知的次数，基本语法如下：

```
for variable in {list}
do
 statement1
 statement2
 ...
done
```

variable称为循环变量，list是一个列表，可以是一系列的数字或者字符串，元素之间使用空格隔开。

循环语句 - for

- 带列表的for循环语句

```
#! /bin/bash

#for 循环开始
for var in {1..8}
do
# 依次输出列表中的数字
echo "the number is $var"
done
```

```
1 #! /bin/bash
2
3 #for 循环开始
4 for var in 1 2 3 4 5 6 7 8
5 do
6 # 依次输出列表中的数字
7 echo "the number is $var"
8 done
~
```

```
[root@rocky shell-test]# bash ex5-1.sh
the number is 1
the number is 2
the number is 3
the number is 4
the number is 5
the number is 6
the number is 7
the number is 8
[root@rocky shell-test]#
```

循环语句 - for

- 带列表的for循环语句

带列表的for循环语句还可以使用{start...end...step}格式的循环列表，基本语法如下：

```
for varibale in {start..end..step}
do
 statement1
 statement2
 ...
done
```

其中，start表示起始的数值，end表示终止的数值，step表示步长。

循环语句 - for

- 带列表的for循环语句

```
#! /bin/bash

# 定义变量，并赋初值为 0
sum=0;
#for 循环开始，设置起始数值为 1 ，结束数值为 100 ，步□ 为 2
for i in {1..100..2}
do
# 将数累加
let "sum+=i"
done
echo "the sum is $sum"
```

```
[root@rocky shell-test]# bash ex5-3.sh
the sum is 2500
[root@rocky shell-test]#
```

循环语句 - for

- 在某些特殊情况下，for循环的条件列表可以完全省略，称为不带列表的for循环语句。如果没有为for循环提供条件列表，Shell将从命令行参数获取条件列表。

不带列表的for循环语句一般语法如下：

```
for variable
do
 statement1
 statement2
 ...
done
```

循环语句 - for

由于系统变量”$@“和”$*“同样可以获取所有的参数，所以上页的语法等价于以下两种语法：

```
for variable in $@
do
 statement1
 statement2
 ...
done
```

```
for variable in $*
do
 statement1
 statement2
 ...
done
```

循环语句 - for

- 不带列表的for循环语句

```
#! /bin/bash

# 不带条件列表
for arg
do
# 输出每个参数
echo "$arg"
done
~
```

```
[root@rocky shell-test]# bash ex5-4.sh a b c d
a
b
c
d
[root@rocky shell-test]#
```

循环语句 - for

- 在Linux或者UNIX上面，C或者C++是最主流的开发语言。为了适应这部分用户的习惯，bash也提供了类C风格的for循环语句。

类C风格的for循环语句的基本语法如下：

```
for ((expression1;expression2;expression3))
do
 statement1;
 statement2;
 ...
done
```

循环语句 - for

在上面的语法中，for循环语句的执行条件被两个圆括号包括起来。

执行条件分为3个部分，由两个分号隔开：第1部分expresson1通常是条件变量初始化的语句；第2部分expresson2是决定是否执行for循环的条件。

当expreson2的值为0时，执行整个循环体；当expresson2的值为非0时，退出for循环体。第3部分，即表达式expresson3通常改变条件变量的值，例如递减或者递增等。

循环语句 - for

- 类C风格的for循环语句

```
#! /bin/bash

#for 循环开始
for (( i=1;i<5;i++))
do
# 输出循环变量 i 的值
echo "$i"
done
~
```

```
[root@rocky shell-test]# bash ex5-5.sh
1
2
3
4
[root@rocky shell-test]#
```

目录

循环语句 - while

- while循环语句是另外一种常见的循环结构。使用while循环结构，可以使得用户重复执行一系列的操作，直到某个条件发生。

while循环语句的基本语法：

```
while expression
do
 statement1
 statement2
 ...
done
```

循环语句 - while

上面的语法中，expression表示while循环体执行时需要满足的条件。通常情况下，expression代表一个测试表达式。

当while循环结构在执行时，会首先判断expression表达式的值，如果表达式的值为0，则执行循环体中的语句；否则，退出while循环，执行done关键字后面的语句。

当循环体中的语句执行完成之后，会重新计算expression的值，如果仍然是0，则继续执行下一次的循环，直至expression的值为非0。

循环语句 - while

- 通过计数器控制while循环结构。

```
#! /bin/bash

# 定义循环变量
i=1
#while 循环开始
while [[ "$i" -lt 10 ]]
do
# 计算平方
let "square=i*i"
# 输出平方
echo "$i*$i=$square"
# 循环变量自增
let "i=i+1"
done
```

```
[root@rocky shell-test]# bash ex5-6.sh
1*1=1
2*2=4
3*3=9
4*4=16
5*5=25
6*6=36
7*7=49
8*8=64
9*9=81
[root@rocky shell-test]#
```

循环语句 - while

- 通过结束标记控制while循环结构。

```
#! /bin/bash
# 提示用户输入数字
echo "Please enter a number between 1 and 10.Enter 0 to
exit."
# 读取用户输入的数字
read var
#while 循环开始
while [[ "$var" != 0 ]]
do
# 提示用户输入的数字太小
if [ "$var" -lt 5 ]
then
echo "Too small. Try again."
read var
# 提示用户输入的数字太大
elif [ "$var" -gt 5 ]
then
echo "Too big. Try again."
read var;
else
echo "Congratulation! You are right."
exit 0;
fi
done
```

```
[root@rocky shell-test]# bash  ex5-7.sh
Please enter a number between 1 and 10.Enter 0 to
exit.
6
Too big. Try again.
5
Congratulation! You are right.
[root@rocky shell-test]#
```

目录

循环语句 - until

- until循环语句的功能是不断地重复执行循环体中的语句，直至某个条件成立。until语句的基本语法如下：

```
until expression
do
 statement1
 statement2
 ...
done
```

expression是一个条件表达式。当该表达式的值不为0时，将执行do和done之间的语句；当expression的值为0时，将退出until循环结构，继续执行done语句后面的其他语句。

循环语句 - until

在每次执行循环体之间，until语句都会先判断expression的值，如果第一次执行时expression的值为0，则后面的循环将一次也不会执行；

如果expression的值为非0，则until语句将执行do和done之间的所有语句。当执行完毕之后，until语句会再次判断expression的值，如果为0，则退出循环；否则，继续重复执行循环体。

循环语句 - until

until语句使用方法（1），代码如下：

```
#! /bin/bash

# 定义循环变量 i
i=1
# 当 i 的值小于 9 时执行循环
until [[ "$i" -gt 9 ]]
do
# 计算 i 的平方
let "square=i*i"
# 输出 i 的平□
echo "$i*$i=$square"
# 循环变量加 1
let "i=i+1"
done
~
```

```
[root@rocky shell-test]# bash ex5-8.sh
1*1=1
2*2=4
3*3=9
4*4=16
5*5=25
6*6=36
7*7=49
8*8=64
9*9=81
[root@rocky shell-test]#
```

循环语句 - until

until语句使用方法（2），代码如下：

```
#! /bin/bash

# 定义变量 i
i=1
# 一直循环到变量 i 的值为 21
until [ "$i" -eq 21 ]
do
# 执行 useradd 命令添加用户
useradd user$i
# 修改用户密码
echo "password" | passwd --stdin user$i > /dev/null
# 循环变量自增
let "i++"
done
```

```
user1:x:1006:1006::/home/user1:/bin/bash
user2:x:1007:1007::/home/user2:/bin/bash
user3:x:1008:1008::/home/user3:/bin/bash
user4:x:1009:1009::/home/user4:/bin/bash
user5:x:1010:1010::/home/user5:/bin/bash
user6:x:1011:1011::/home/user6:/bin/bash
user7:x:1012:1012::/home/user7:/bin/bash
user8:x:1013:1013::/home/user8:/bin/bash
user9:x:1014:1014::/home/user9:/bin/bash
user10:x:1015:1015::/home/user10:/bin/bash
user11:x:1016:1016::/home/user11:/bin/bash
user12:x:1017:1017::/home/user12:/bin/bash
user13:x:1018:1018::/home/user13:/bin/bash
user14:x:1019:1019::/home/user14:/bin/bash
user15:x:1020:1020::/home/user15:/bin/bash
user16:x:1021:1021::/home/user16:/bin/bash
user17:x:1022:1022::/home/user17:/bin/bash
user18:x:1023:1023::/home/user18:/bin/bash
user19:x:1024:1024::/home/user19:/bin/bash
user20:x:1025:1025::/home/user20:/bin/bash
```

目录

循环语句 - break & continue

- break和continue语句，前者用于立即从循环中退出；而后者则用来跳过循环体中的某些语句，继续执行下一次循环。
- break语句可以用在for、while或者until等循环语句的循环体中。
- continue语句的作用和不是退出循环体，而是跳过当前循环体中该语句后面语句，重新从循环语句开始的位置执行。

```
for i in $(seq 1 10)
do
        echo $i
        if [ $i -eq 5 ]
        then
        break
fi
i=`expr $i + 1`
done
```

```
for i in $(seq 1 10)
do
        echo $i
        if [ $i -eq 5 ]
        then
        continue
fi
i=`expr $i + 1`
done
```

循环语句 - break & continue

- 利用continue语句控制循环
  - continue语句的作用不是提出循环体，而是跳出当前循环体中该语句后面的语句，重新从循环语句开始的位置执行

```
#! /bin/bash

for var in {1..10}
do
# 如果当前数字为奇数
if [[ "$var%2" -eq 1 ]]
then
# 跳过后面的语句
continue
fi
echo "$var"
done
```

```
[root@rocky shell-test]# bash  ex5-11.sh
2
4
6
8
10
[root@rocky shell-test]#
```

目录

嵌套循环

- 在程序设计语言中，嵌套的循环也是一种很常见的结构。通过嵌套循环，可以完成更复杂的功能。

```
#! /bin/bash

# 外层循环
for ((i=1;i<=9;i++))
do
# 内层循环
        for ((j=1;j<=i;j++))
do
# 计算 2 个数的乘积
let "product=i*j"
# 输出乘积
printf "$i*$j=$product"
# 输出空格分隔符
if [[ "$product" -gt 9 ]]
then
        printf " "
else
        printf " "
fi
        done
        echo
done
```

```
oot@rocky shell-test]# bash  ex5-12.sh
1=1
1=2 2*2=4
1=3 3*2=6 3*3=9
1=4 4*2=8 4*3=12 4*4=16
1=5 5*2=10 5*3=15 5*4=20 5*5=25
1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
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

oot@rocky shell-test]#
```

谢 谢