https://www.cnblogs.com/clarke/p/5483214.html

**[Shell脚本的条件控制和循环语句](https://www.cnblogs.com/clarke/p/5483214.html)**

**条件判断：if语句**

语法格式：

if [ expression ]

then

Statement(s) to be executed if expression is true

fi

注意：expression 和方括号([ ])之间必须有空格，否则会有语法错误。

if 语句通过关系运算符判断表达式的真假来决定执行哪个分支。Shell 有三种 if ... else 语句：

if ... fi 语句

if ... else ... fi 语句

if ... elif ... else ... fi 语句

示例：

[复制代码](javascript:void(0);)

#!/bin/bash/

a=10

b=20

if [ $a == $b ]

then

echo "a is equal to b"

elif [ $a -gt $b ]

then

echo "a is greater to b"

else

echo "a is less to b"

fi

[复制代码](javascript:void(0);)

if ... else 语句也可以写成一行，以命令的方式来运行:

a=10;b=20;if [ $a == $b ];then echo "a is equal to b";else echo "a is not equal to b";fi;

if ... else 语句也经常与 test 命令结合使用，作用与上面一样：

[复制代码](javascript:void(0);)

#!/bin/bash/

a=10

b=20

if test $a == $b

then

echo "a is equal to b"

else

echo "a is not equal to b"

fi

[复制代码](javascript:void(0);)

**分支控制：case语句**

case ... esac 与其他语言中的 switch ... case 语句类似，是一种多分枝选择结构。

示例：

[复制代码](javascript:void(0);)

#!/bin/bash/

grade="B"

case $grade in

"A") echo "Very Good!";;

"B") echo "Good!";;

"C") echo "Come On!";;

\*)

echo "You Must Try!"

echo "Sorry!";;

esac

[复制代码](javascript:void(0);)

转换成C语言是：

[复制代码](javascript:void(0);)

#include <stdio.h>

int main(){

char grade = 'B';

switch(grade){

case 'A': printf("Very Good!");break;

case 'B': printf("Very Good!");break;

case 'C': printf("Very Good!");break;

default:

printf("You Must Try!");

printf("Sorry!");

break;

}

return 0;

}

[复制代码](javascript:void(0);)

对比看就很容易理解了。很相似，只是格式不一样。

需要注意的是：

取值后面必须为关键字 in，每一模式必须以右括号结束。取值可以为变量或常数。匹配发现取值符合某一模式后，其间所有命令开始执行直至 ;;。;; 与其他语言中的 break 类似，意思是跳到整个 case 语句的最后。

取值将检测匹配的每一个模式。一旦模式匹配，则执行完匹配模式相应命令后不再继续其他模式。如果无一匹配模式，使用星号 \* 捕获该值，再执行后面的命令。

再举一个例子：

[复制代码](javascript:void(0);)

#!/bin/bash

option="${1}"

case ${option} in

"-f") FILE="${2}"

echo "File name is $FILE"

;;

"-d") DIR="${2}"

echo "Dir name is $DIR"

;;

\*)

echo "`basename ${0}`:usage: [-f file] | [-d directory]"

exit 1 # Command to come out of the program with status 1

;;

esac

[复制代码](javascript:void(0);)

运行结果：

$./test.sh

test.sh: usage: [ -f filename ] | [ -d directory ]

./test.sh -f index.html

File name is index.html

这里用到了特殊变量${1},指的是获取命令行的第一个参数。

**for循环**

shell的for循环与c、php等语言不同，同Python很类似。下面是语法格式：

for 变量 in 列表

do

command1

command2

...

commandN

done

示例：

#!/bin/bash/

for value in 1 2 3 4 5

do

echo "The value is $value"

done

The value is 1

The value is 2

The value is 3

The value is 4

The value is 5

顺序输出字符串中的字符：

for str in 'This is a string'

do

echo $str

done

面的代码将遍历当前目录下所有的文件。在Linux下，可以改为其他目录试试。

遍历文件内容：

**city.txt**

[复制代码](javascript:void(0);)

beijing

tianjin

shanghai

#!/bin/bash

citys=`cat city.txt`

for city in $citys

echo $city

done

[复制代码](javascript:void(0);)

**while循环**

只要while后面的条件满足，就一直执行do里面的代码块。

其格式为：

while command  
do  
Statement(s) to be executed if command is true  
done

命令执行完毕，控制返回循环顶部，从头开始直至测试条件为假。

示例：

[复制代码](javascript:void(0);)

#!/bin/bash

c=0;

while [ $c -lt 3 ]

do

echo "Value c is $c"

c=`expr $c + 1`

done

[复制代码](javascript:void(0);)

输出：

Value c is 0  
Value c is 1  
Value c is 2

这里由于shell本身不支持算数运算，所以使用expr命令进行自增。

**until循环**

until 循环执行一系列命令直至条件为 true 时停止。until 循环与 while 循环在处理方式上刚好相反。一般while循环优于until循环，但在某些时候，也只是极少数情况下，until 循环更加有用。

将上面while循环的例子改改，就能达到一样的效果：

[复制代码](javascript:void(0);)

#!/bin/bash

c=0;

until [ $c -eq 3 ]

do

echo "Value c is $c"

c=`expr $c + 1`

done

[复制代码](javascript:void(0);)

首先do里面的语句块一直在运行，直到满足了until的条件就停止。

输出：

Value c is 0  
Value c is 1  
Value c is 2

**跳出循环**

在循环过程中，有时候需要在未达到循环结束条件时强制跳出循环，像大多数编程语言一样，Shell也使用 break 和 continue 来跳出循环。

**break**

break命令允许跳出所有循环（终止执行后面的所有循环）。

[复制代码](javascript:void(0);)

#!/bin/bash

i=0

while [ $i -lt 5 ]

do

i=`expr $i + 1`

if [ $i == 3 ]

then

break

fi

echo -e $i

done

[复制代码](javascript:void(0);)

运行结果：

1  
2

在嵌套循环中，break 命令后面还可以跟一个整数，表示跳出第几层循环。例如：

**break n**

表示跳出第 n 层循环。

continue

continue命令与break命令类似，只有一点差别，它不会跳出所有循环，仅仅跳出当前循环。

[复制代码](javascript:void(0);)

#!/bin/bash

i=0

while [ $i -lt 5 ]

do

i=`expr $i + 1`

if [ $i == 3 ]

then

continue

fi

echo -e $i

done

[复制代码](javascript:void(0);)

运行结果：

1  
2  
4  
5

以上内容是小编给大家介绍的Shell脚本的条件控制和循环语句的相关知识，希望对大家有所帮助！

分类: [shell](https://www.cnblogs.com/clarke/category/822773.html)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[https://common.cnblogs.com/images/icon_weibo_24.png](javascript:void(0);)** **[https://common.cnblogs.com/images/wechat.png](javascript:void(0);)**

[https://pic.cnblogs.com/face/515420/20150813105051.png](http://home.cnblogs.com/u/clarke/)

[smallQSH](http://home.cnblogs.com/u/clarke/)  
[关注 - 91](http://home.cnblogs.com/u/clarke/followees)  
[粉丝 - 16](http://home.cnblogs.com/u/clarke/followers)

[+加关注](javascript:void(0);)

0

0

[«](https://www.cnblogs.com/clarke/p/5474096.html)上一篇：[python web框架学习笔记](https://www.cnblogs.com/clarke/p/5474096.html)  
[»](https://www.cnblogs.com/clarke/p/5499091.html)下一篇：[检查linux网络状态的两个脚本](https://www.cnblogs.com/clarke/p/5499091.html)