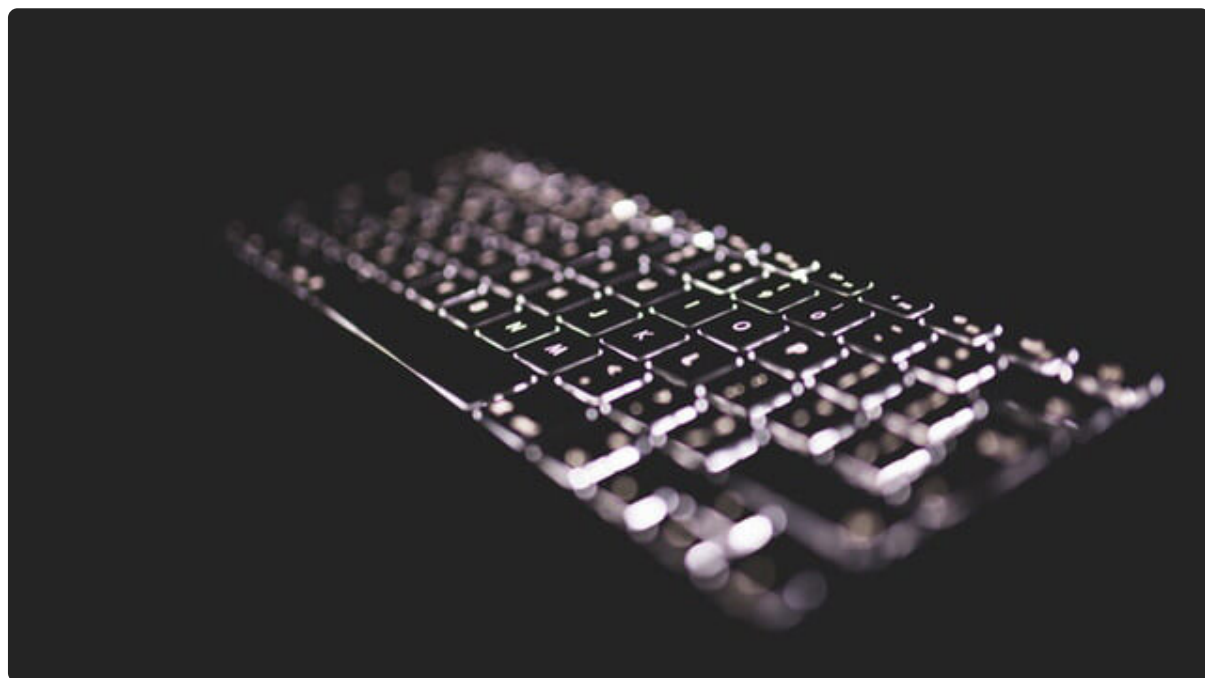


## 43 条件一出，Shell不服

更新时间：2019-08-26 11:04:47



“

每个人都是自己命运的主宰。

——斯蒂尔斯

”

### 内容简介

1. 前言
2. if：最简单的条件
3. 条件测试
4. case：测试多个条件
5. 总结

### 1. 前言

上一课 [带你玩转Linux和Shell编程 | 第五部分第四课：变量在手，Shell不愁](#) 有不少内容。这一课我们轻松一下，来聊聊条件语句。

今天的标题有点“任性”。主要是想到了一个对联：

说你行，你就行，不行也行。  
说不行，就不行，行也不行。  
横批：不服不行

因此“条件一出，Shell 不服不行”。

读者：“好冷的笑话...”

对于所有编程语言，做决定是很重要的。如果程序员不决定程序做什么，那么程序基本只会一直做同样的事情，就很无趣了。

而条件语句就是用于帮助程序做决定的。在我们的 **Shell** 脚本中，条件语句可以做以下“抉择”：

```
如果，变量的值等于 xxx，  
  
那么，这样做：  
  
否则，那样做。
```

如果你学过某一门编程语言，例如 **C** 语言，**C++** 或 **Java**，那么对于条件语句的原理应该很熟悉了。

即使你没学过其它编程语言，那也不必担心，跟着我们学完这课，保证你了然于胸。

## 2. if: 最简单的条件

最常用的条件语句就是 **if** 条件语句。

**if** 是英语“如果”的意思。

**if** 条件语句的基本格式是这样的：

```
if [ 条件测试 ]  
then  
    做这个  
fi
```

**fi** 是 **if** 的反转写法，表示“if 语句结束”。**then** 是英语“那么”的意思。

“做这个”只有在“条件测试”为真时，才会被执行。

注意：方括号 **[ ]** 中的 **条件测试** 两边必须要空一格。不能写成 **[test]**，而要写成 **[ test ]**。  
唉，**Shell** 就是这么任性，你不服都不行！

当然了，**if** 语句的基本写法还有一种，那就是把 **then** 写在 **if [ 条件测试 ]** 后面，如下：

```
if [ 条件测试 ]; then  
    做这个  
fi
```

用这种写法时，在 **if** 条件判断和 **then** 之间要加一个分号。

我们写 **Shell** 程序时，需要把“条件测试”换成我们真实要测试的条件，一般来说都是测试变量的值。

我们可以通过例子来学习，首先创建一个叫做 **condition.sh** 的文件（**condition** 是英语“条件”的意思）。

```
vim condition.sh
```

然后在里面输入以下内容：

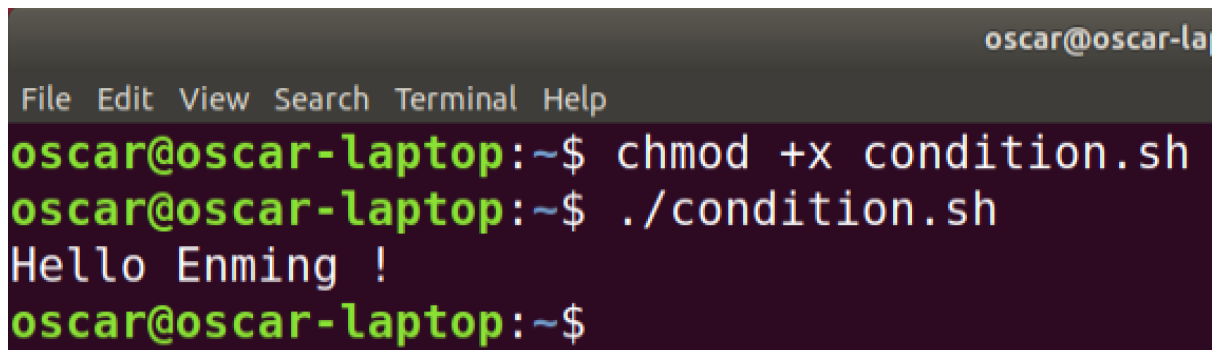
```
#!/bin/bash

name="Enming"

if [ $name = "Enming" ]
then
    echo "Hello $name !"
fi
```

在 **Shell** 语言中，“等于”是用一个等号（`=`）来表示的，这和大多数主流编程语言不同。**C** 语言中“等于”是用两个等号（`==`）来表示的。但 **Shell** 中用两个等号来表示“等于”的判断也是可以的。

上例中，因为变量 `name` 的值等于“Enming”，因此会输出“Hello Enming !”



```
oscar@oscar-laptop
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ chmod +x condition.sh
oscar@oscar-laptop:~$ ./condition.sh
Hello Enming !
oscar@oscar-laptop:~$
```

当然了，我们也可以测试两个变量，例如：

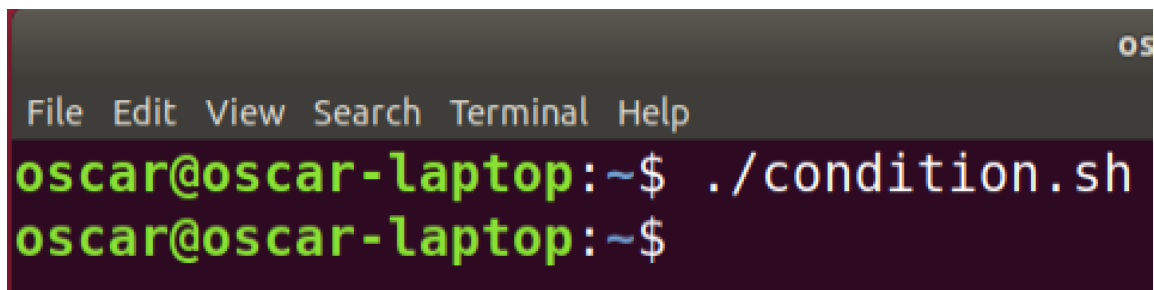
```
#!/bin/bash

name1="Enming"
name2="Oscar"

if [ $name1 = $name2 ]
then
    echo "You two have the same name !"
fi
```

“You two have the same name”是英语“你们俩有相同的名字”的意思。

运行上面的程序，可以看到什么也没有显示，因为 `Enming` 和 `Oscar` 这两个字符串不相同。



```
oscar@oscar-laptop
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ ./condition.sh
oscar@oscar-laptop:~$
```

## else : 否则

既然有“如果”的条件判断，那么也会存在条件不成立的时候，就好比上面那个程序中，`Enming` 和 `Oscar` 这两个字符串不相同的情况。

我们已经知道表示“如果”的关键字是 `if`，那么表示“否则”的关键字应该就是 `else` 咯？

没错。因为 **else** 就是英语“否则”的意思。

因此，**if** 和 **else** 两者配合的逻辑是这样的：

```
if [ 条件测试 ]
then
    做这个
else
    做那个
fi
```

也就是：如果“条件测试”为真，那么“做这个”被执行；否则，“做那个”被执行。

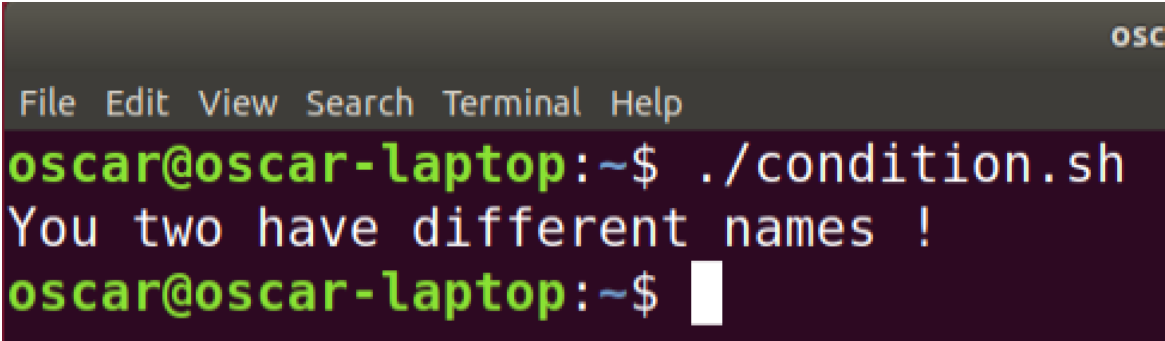
我们来看例子：

```
#!/bin/bash

name1="Enming"
name2="Oscar"

if [ $name1 = $name2 ]
then
    echo "You two have the same name !"
else
    echo "You two have different names !"
fi
```

运行：



```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ ./condition.sh
You two have different names !
oscar@oscar-laptop:~$
```

“You two have different names” 是英语“你们有不同的名字”的意思。

## **elif**：否则，如果

一般来说 **if** 和 **else** 已经能满足我们的大部分条件判断需要了，但有些时候，存在好几种情况。

光是 **if** 和 **else** 表示的两种对立的情况已经不足以满意要求了，因此我们再来学习一个关键字：**elif**。

**elif** 是 **else if** 的缩写，表示“否则 - 如果”。

**if**，**elif** 和 **else** 三者配合的逻辑是这样的：

```
if [ 条件测试 1 ]
then
    做事情 1
elif [ 条件测试 2 ]
then
    做事情 2
elif [ 条件测试 3 ]
then
    做事情 3
else
    做其他事情
fi
```

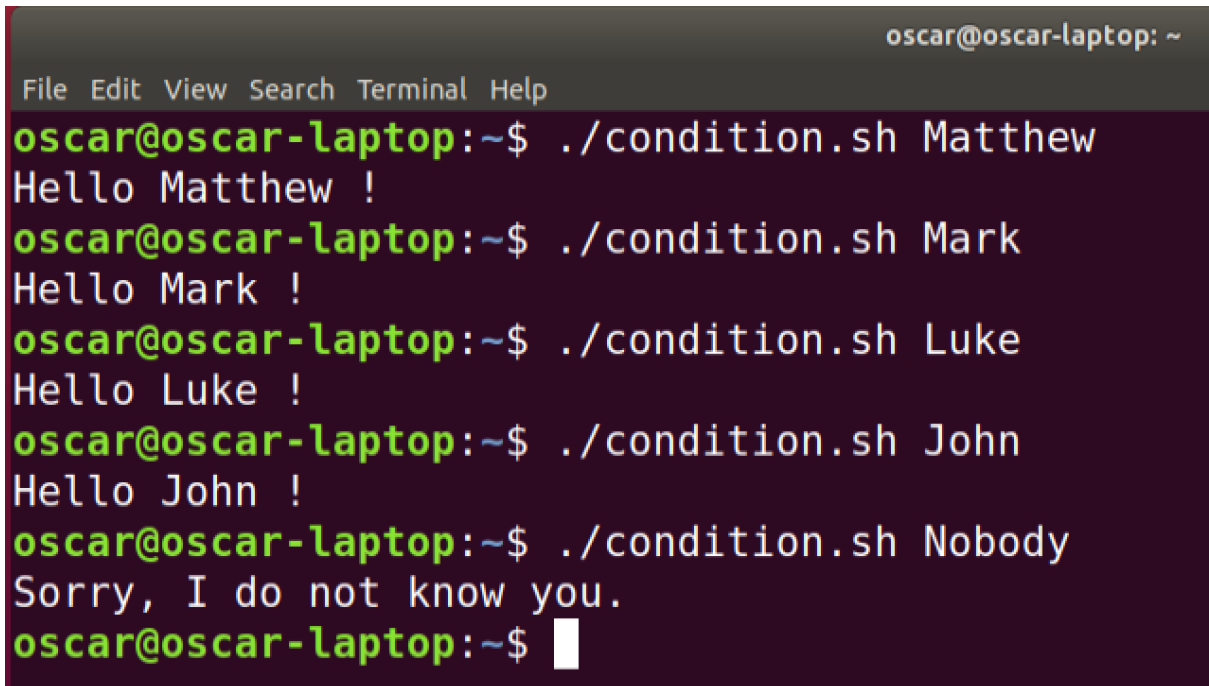
我们来看一个例子：

```
#!/bin/bash

if [ $1 = "Matthew" ]
then
    echo "Hello Matthew !"
elif [ $1 = "Mark" ]
then
    echo "Hello Mark !"
elif [ $1 = "Luke" ]
then
    echo "Hello Luke !"
elif [ $1 = "John" ]
then
    echo "Hello John !"
else
    echo "Sorry, I do not know you."
fi
```

“Sorry, I do not know you” 是英语“对不起，我不认识你”的意思。

运行：

A terminal window titled 'oscar@oscar-laptop: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of a script named 'condition.sh' with five different arguments: 'Matthew', 'Mark', 'Luke', 'John', and 'Nobody'. For the first four arguments, the script outputs a greeting: 'Hello Matthew !', 'Hello Mark !', 'Hello Luke !', and 'Hello John !'. For the 'Nobody' argument, it outputs 'Sorry, I do not know you.' The prompt 'oscar@oscar-laptop:~\$' is shown before each command and after the last one with a cursor.

```
oscar@oscar-laptop: ~
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ ./condition.sh Matthew
Hello Matthew !
oscar@oscar-laptop:~$ ./condition.sh Mark
Hello Mark !
oscar@oscar-laptop:~$ ./condition.sh Luke
Hello Luke !
oscar@oscar-laptop:~$ ./condition.sh John
Hello John !
oscar@oscar-laptop:~$ ./condition.sh Nobody
Sorry, I do not know you.
oscar@oscar-laptop:~$
```

我们可以写任意多个 `elif` 语句，但是 `if` 语句必须要有且只能有一个，`else` 语句不一定要有且最多只能有一个。

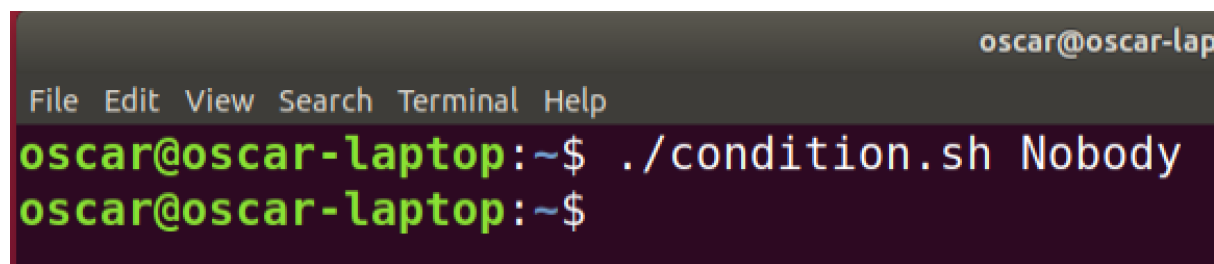
我们可以有单独由 `if` 语句组成的条件语句，也可以有 `if ... else ...` 组成的条件语句，也可以有 `if ... elif ... else...` 组成的条件语句，也可以有 `if ... elif ...` 组成的语句。

所以，你如果把 `else` 那个语句去掉，变成：

```
#!/bin/bash

if [ $1 = "Matthew" ]
then
    echo "Hello Matthew !"
elif [ $1 = "Mark" ]
then
    echo "Hello Mark !"
elif [ $1 = "Luke" ]
then
    echo "Hello Luke !"
elif [ $1 = "John" ]
then
    echo "Hello John !"
fi
```

也是可以的。只是在这种情况下，你输入除了那四个字符串以外的其他参数，`Shell` 脚本将没有任何输出：

A terminal window with a dark background. The title bar shows 'oscar@oscar-lap'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The prompt is 'oscar@oscar-laptop:~\$'. The user enters './condition.sh Nobody'. The prompt returns to 'oscar@oscar-laptop:~\$' without any output from the script.

### 3. 条件测试

现在我们来看看我们都能做哪些“条件测试”。

之前的例子，我们只比较了字符串，但其实我们远可以做比这更复杂的条件测试。

## 不同的测试类型

在 `Bash` 中我们可以做三种测试：

- 测试字符串
- 测试数字
- 测试文件

我们通过例子一一来学习。

## 测试字符串

我们之前的课程已经说过：在 `Shell` 中，所有的变量都是字符串。

因此，要做字符串的测试非常简单。记住以下表格：

条件	意义
<code>\$string1 = \$string2</code>	两个字符串是否相等。 <code>Shell</code> 大小写敏感，因此 <code>A</code> 和 <code>a</code> 是不一样的。
<code>\$string1 != \$string2</code>	两个字符串是否不同。

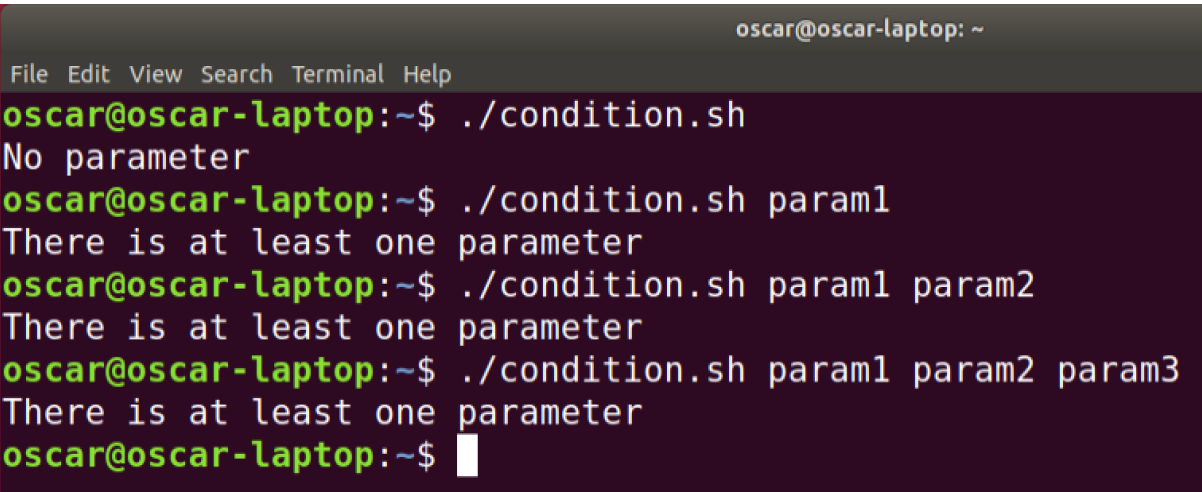
条件	意义
-z \$string	字符串 <code>string</code> 是否为空。 <code>z</code> 是 <code>zero</code> 的首字母，是英语“零”的意思。
-n \$string	字符串 <code>string</code> 是否不为空。 <code>n</code> 是英语 <code>not</code> 的首字母，是英语“不”的意思。

例如：

```
#!/bin/bash

if [ -z $1 ]
then
    echo "No parameter"
else
    echo "There is at least one parameter"
fi
```

“No parameter” 是英语“没有参数”的意思。  
“There is at least one parameter” 是英语“至少有一个参数”的意思。



测试数字

尽管 Shell 把所有变量都看成字符串，但是我们还是可以做数字的条件测试。记住以下表格：

条件	意义
\$num1 -eq \$num2	两个数字是否相等。和判断字符串所用的符号（=）不一样。eq 是 equal 的缩写，是英语“等于”的意思。
\$num1 -ne \$num2	两个数字是否不同。ne 是 not equal 的缩写，是英语“不等于”的意思。
\$num1 -lt \$num2	数字 num1 是否小于 num2。lt 是 lower than 的缩写，是英语“小于”的意思。
\$num1 -le \$num2	数字 num1 是否小于或等于 num2。le 是 lower or equal 的缩写，是英语“小于或等于”的意思。
\$num1 -gt \$num2	数字 num1 是否大于 num2。gt 是 greater than 的缩写，是英语“大于”的意思。
\$num1 -ge \$num2	数字 num1 是否大于或等于 num2。ge 是 greater or equal 的缩写，是英语“大于或等于”的意思。

看一个例子：

```
#!/bin/bash

if [ $1 -ge 10 ]
then
    echo "You have entered a number greater than 10 or equal to 10"
else
    echo "You have entered a number lower than 10"
fi
```

- “You have entered a number greater than 10 or equal to 10” 是英语“你输入了一个比 10 更大或等于 10 的数字”的意思。
- “You have entered a number lower than 10” 是英语“你输入了一个比 10 更小的数字”的意思。

```
oscar@oscar-laptop: ~
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ ./condition.sh 5
You have entered a number lower than 10
oscar@oscar-laptop:~$ ./condition.sh 10
You have entered a number greater than 10 or equal to 10
oscar@oscar-laptop:~$ ./condition.sh 20
You have entered a number greater than 10 or equal to 10
oscar@oscar-laptop:~$
```

## 测试文件

相比于主流编程语言，**Shell** 的一大优势就是可以非常方便地测试文件：文件存在吗？我们可以写入文件吗？这个文件比那个文件修改时间更早还是更晚？等等。

下表非常丰富：

条件	意义
-e \$file	文件是否存在。 <b>e</b> 是 <b>exist</b> 的首字母，表示“存在”。
-d \$file	文件是否是一个目录。因为 <b>Linux</b> 中一切都是文件，目录也是文件的一种。 <b>d</b> 是 <b>directory</b> 的首字母，表示“目录”。
-f \$file	文件是否是一个文件。 <b>f</b> 是 <b>file</b> 的首字母，表示“文件”。
-L \$file	文件是否是一个符号链接文件。 <b>L</b> 是 <b>link</b> 的首字母，表示“链接”。
-r \$file	文件是否可读。 <b>r</b> 是 <b>readable</b> 的首字母，表示“可读的”。
-w \$file	文件是否可写。 <b>w</b> 是 <b>writable</b> 的首字母，表示“可写的”。
-x \$file	文件是否可执行。 <b>x</b> 是 <b>executable</b> 的首字母，表示“可执行的”。
\$file1 -nt \$file2	文件 <b>file1</b> 是否比 <b>file2</b> 更新。 <b>nt</b> 是 <b>newer than</b> 的缩写，表示“更新的”。
\$file1 -ot \$file2	文件 <b>file1</b> 是否比 <b>file2</b> 更旧。 <b>ot</b> 是 <b>older than</b> 的缩写，表示“更旧的”。

来看一个例子：

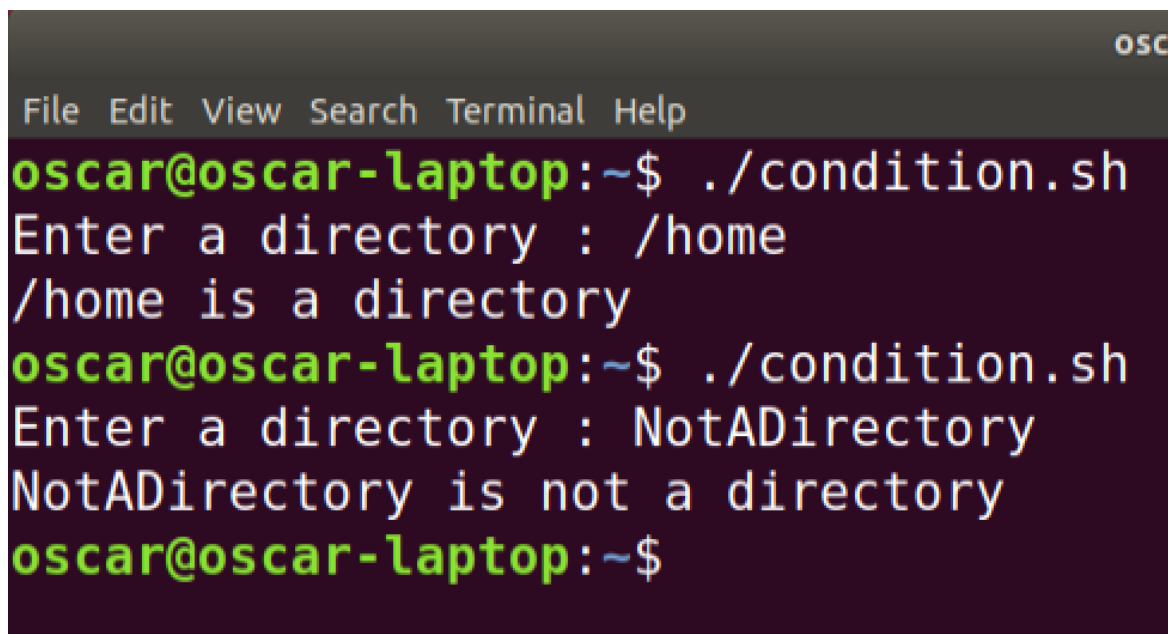
```
#!/bin/bash

read -p 'Enter a directory : ' file

if [ -d $file ]
then
    echo "$file is a directory"
else
    echo "$file is not a directory"
fi
```



运行：



```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ ./condition.sh
Enter a directory : /home
/home is a directory
oscar@oscar-laptop:~$ ./condition.sh
Enter a directory : NotADirectory
NotADirectory is not a directory
oscar@oscar-laptop:~$
```

上面的程序就是测试输入的参数是不是一个目录。

- ‘Enter a directory’ 表示“输入一个目录”。
- ‘is a directory’ 表示“是一个目录”。
- ‘is not a directory’ 表示“不是一个目录”。

## 一次测试多个条件

在一个条件测试中，我们可以同时测试多个条件。需要用到两种符号：

符号	意义
&&	两个 &。表示“逻辑与”。此符号两端的条件必须全为真，整个条件测试才为真；只要有一个不为真，整个条件测试为假。
	两个竖线。表示“逻辑或”。此符号两端的条件只要有一个为真，整个条件测试就为真；只有两个都为假，整个条件测试才为假。

来看一个例子：

```
#!/bin/bash

if [ $# -ge 1 ] && [ $1 = 'love' ]
then
    echo "Great !"
    echo "You know the password"
else
    echo "You do not know the password"
fi
```

“Great ! You know the password” 是英语“好极了！你知道密码”的意思。

“You do not know the password” 是英语“你不知道密码”的意思。

```
oscar@oscar
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ ./condition.sh love
Great !
You know the password
oscar@oscar-laptop:~$ ./condition.sh hate
You do not know the password
oscar@oscar-laptop:~$
```

上面的测试检验了两个条件：

1. 参数是否至少有 1 个（`$#` 大于或等于 1）
2. 第一个参数是否等于 love（`$1` 是否等于 love）。

love 表示“爱”，hate 表示“恨”。

在做多个条件的判断时，是按照从左到右的顺序判断的，如果前一个条件已经足以决定整个条件测试的真或假，那么后面的条件就不会被判断。

例如 `[ 2 -ge 1 ] || [ 3 -gt 2 ]` 中，2 是大于等于 1 的，因此，`2 -ge 1` 已经为真，整个条件语句肯定为真，`3 -gt 2` 就不需要被判断了。

而 `[ 2 -ge 4 ] && [ 5 -gt 2 ]` 中，2 是小于 4 的，因此，`2 -ge 4` 已经为假，整个条件语句肯定为假，`5 -gt 2` 就不需要被判断了。

## 反转测试

我们可以用“否定”来反转测试条件，要用到感叹号（`!`）。

来看一个例子：

```
#!/bin/bash

read -p 'Enter a file : ' file

if [ ! -e $file ]
then
    echo "$file does not exist"
else
    echo "$file exists"
fi
```

条件测试中我们写了 `! -e $file`，表示“如果文件 file 不存在”。

exist 是英语“存在”的意思。

运行：

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ ./condition.sh
Enter a file : myFile
myFile exists
oscar@oscar-laptop:~$ ./condition.sh
Enter a file : nothing
nothing does not exist
oscar@oscar-laptop:~$
```

#### 4. case : 测试多个条件

之前，我们写过一个例子，用来演示多个条件测试的情况：

```
#!/bin/bash

if [ $1 = "Matthew" ]
then
    echo "Hello Matthew !"
elif [ $1 = "Mark" ]
then
    echo "Hello Mark !"
elif [ $1 = "Luke" ]
then
    echo "Hello Luke !"
elif [ $1 = "John" ]
then
    echo "Hello John !"
else
    echo "Sorry, I do not know you."
fi
```

这个程序当然没有错，但是它只对同一个变量做测试（我们输入的参数 1），却用了这么多 `elif... then...` 未免略显繁琐，也不是很好理解。

像这样的情况，我们可以用 `case` 语句来实现。

`case` 是英语“情况”的意思。

我们上面的程序可以改写成这样：

```
#!/bin/bash

case $1 in
    "Matthew")
        echo "Hello Matthew !"
        ;;
    "Mark")
        echo "Hello Mark !"
        ;;
    "Luke")
        echo "Hello Luke !"
        ;;
    "John")
        echo "Hello John !"
        ;;
    *)
        echo "Sorry, I do not know you."
        ;;
esac
```

运行之后与之前的程序效果一样。

来分析一下上面的程序，因为有很多新的内容：

**case \$1 in** : **\$1** 表示我们要测试的变量是输入的的第一个参数。**in** 是英语“在...之中”的意思。

**"Matthew")** : 测试其中一个 **case**，也就是 **\$1** 是否等于 **"Matthew"**。当然，我们也可以用星号来做通配符来匹配多个字符，例如 **"M\*"** 可以匹配所有以 **M** 开头的字符串。

**::** : 类似于主流编程语言中的 **break**，表示结束 **case** 的读取，程序跳转到 **esac** 后面执行。

**\*)** : 相当于 **if** 条件语句的 **else**，表示“否则”，就是“假如不等于上面任何一种情况”。

**esac** : 是 **case** 的反写，表示 **case** 语句的结束。

其实 **case** 语句就相当于主流编程语言中的 **switch** 语句。

我们也可以在 **case** 语句的匹配项中做“或”的匹配，但是不是用两个竖线（逻辑或）了，而是用一个竖线。

来看一个例子：

```
#!/bin/bash

case $1 in
    "dog" | "cat" | "pig")
        echo "It is a mammal"
        ;;
    "pigeon" | "swallow")
        echo "It is a bird"
        ;;
    *)
        echo "I do not know what it is"
        ;;
esac
```

- “It is a mammal” 是英语“这是一只哺乳动物”的意思。**dog** 是“狗”，**cat** 是“猫”，**pig** 是“猪”。这三种动物都属于哺乳动物。
- “It is a bird” 是英语“这是一只鸟”的意思。**pigeon** 是“鸽子”，**swallow** 是“燕子”。这两种动物都属于鸟类。
- “I do not know what it is” 是英语“我不知道这是什么”的意思。

运行：

```
oscar@oscar-laptop
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ ./condition.sh dog
It is a mammal
oscar@oscar-laptop:~$ ./condition.sh cat
It is a mammal
oscar@oscar-laptop:~$ ./condition.sh pig
It is a mammal
oscar@oscar-laptop:~$ ./condition.sh pigeon
It is a bird
oscar@oscar-laptop:~$ ./condition.sh swallow
It is a bird
oscar@oscar-laptop:~$ ./condition.sh blabla
I do not know what it is
oscar@oscar-laptop:~$
```

## 5. 总结

我们可以用下面的语法来做条件测试：`if, then, [[ elif, then, fi ] else, ] fi`；

我们可以测试字符串，也可以测试数字，也可以测试文件。例如文件存在吗？文件是否可执行？等等；

如果需要，我们可以综合好几种测试，用 `&&`（逻辑与），`||`（逻辑或）符号；

感叹号（`!`）表示条件“否”，用于反转条件测试；

当我们对同一个变量做好多测试时，一般用 `case` 语句比 `if` 语句更直观。

今天的课就到这里，一起加油吧！

}