软件测试开发职位内推Q群：485353510

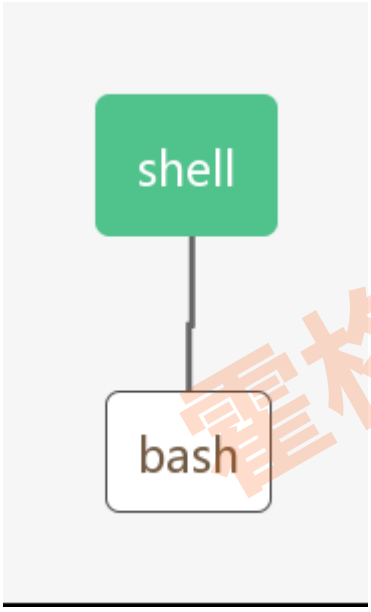
一 .shell

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510

bash和shell

shell是一个命令行解释器，它把用户命令**翻译**给操作系统。shell拥有自己的**命令集**，不论你何时键入一个命令，都被shell所解释。它虽然不是Unix/Linux系统内核的一部分，但它调用了系统核心的功能来执行程序、建立文件并协调程序的运行。熟练shell的使用方法，是做好**批处理**，**服务端测试**，**移动端测试**，**持续集成与自动化部署**的关键。
bash是borne again shell的缩写，它是shell的一种,Linux上默认采用的是bash。



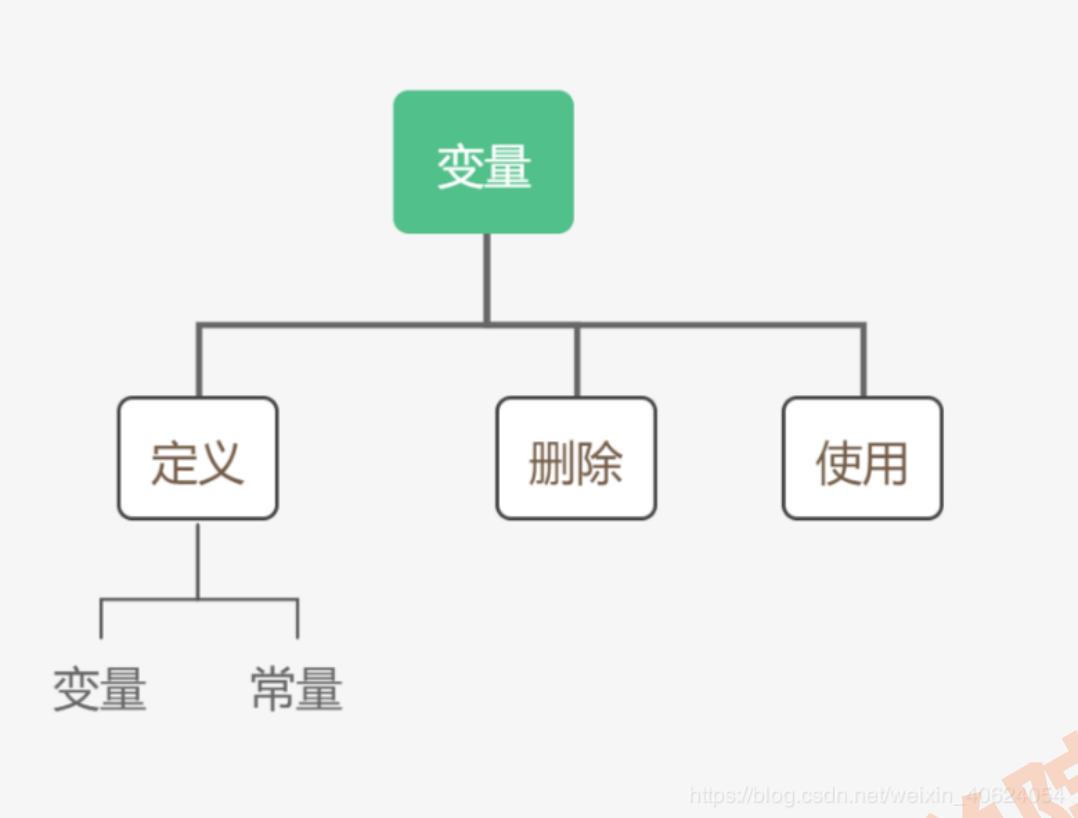
windows需要安装cygwin

bash语言

变量

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510



bash的变量很容易使用，在定义部分，我额外的加入了常量。

变量定义

变量的定义如下：

```
a=1
b=seveniruby
c="hello from testerhome"
d='hello from "霍格沃兹"'
f='ls'
```

要注意的有：

1. =左右不要有空格
2. 如果内容有空格，需要使用单引号或者双引号，b是seveniruby，如果变成hello from testerhome， 需要加引号，如c。
3. 双引号支持转义，$开头的变量会被自动替换，比如下面：

```
a=123
b=$a
echo $b
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
---
123
```

常量定义

使用readonly命令可以将变量定义为**只读变量**，只读变量的值不能被改变。
下面的例子尝试更改只读变量，结果报错：

```
myUrl="霍格沃兹"
readonly myUrl
myUrl="霍格沃兹学院"
---
/bin/sh: NAME: This variable is read only.
```

删除变量

使用unset命令可以删除变量。语法：

```
myUrl="霍格沃兹"
unset myUrl
echo $myUrl
---
以上实例执行将没有任何输出。
```

霍格沃兹测试学院

1. 变量被删除后不能**再次**使用。
2. unset 命令不能删除**只读变量**。
3. 很多人觉得删除变量没有用，你要养成删除**无用**变量的习惯，可以少占内存。

变量的使用

```
a=123
b=456
echo $a
echo ${b}
echo "$a"
---
```

软件测试开发职位内推Q群：485353510


```
123
456
123
```

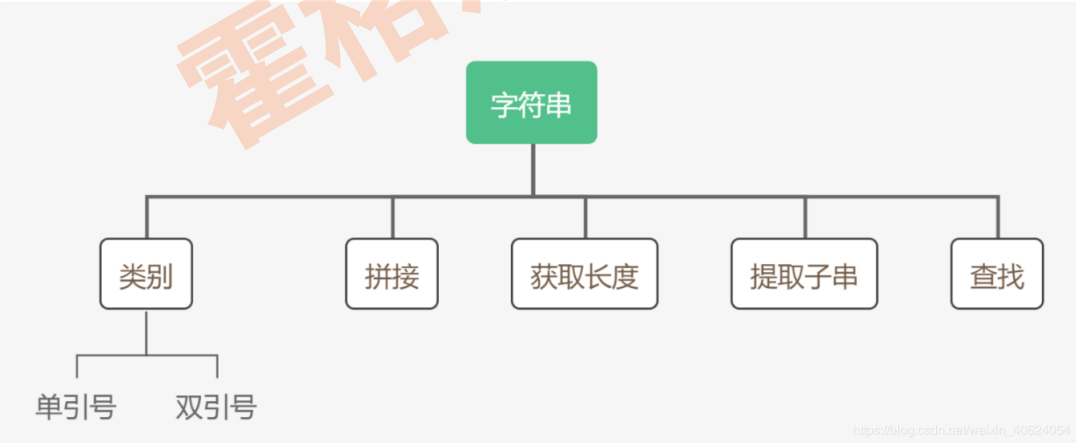
使用$var或${var}来访问变量，两者区别如下：

```
a=123
b={$a}123
c="$a 123"
echo $b
echo $c
---
123123
123 123
```

要把变量(a)和值(123)赋给新的变量b，可以用b={$a}123，也可以用b="$a 123"，显然前者更简洁。

> 变量不定义也能使用，引用未定义的变量，默认为空值。

字符串



字符串是shell编程中常的数据类型，字符串可以用**单引号**，也可以用**双引号**，要注意他们的区别，除此还要会操作字符串。

单引号

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
str1='霍格沃兹'
str2='$str1 学院'
school='学院'
str3='霍格沃兹'$school''
echo $str1
echo $str2
echo $str3
---
霍格沃兹
$str1 学院
霍格沃兹学院
```

1. 单引号里的任何字符都会原样输出，单引号字符串中的**变量是无效的**，比如上面输出了$str1，并不是'霍格沃兹'。
2. 单引号字串不能出现**单独**的单引号（使用转义符也不行），但可成对出现，作为字符串拼接使用（后面有例子）。

双引号

```
your_name='霍格沃兹小学弟'
str="Hello, I know you are \"$your_name\"!"
echo $str
---
Hello, I know you are "霍格沃兹小学弟"!
```

1. 双引号里可以有**变量**。
2. 双引号里可以出现**转义字符**。

拼接字符串

```
your_name="小学弟"
# 使用双引号拼接
greeting="hello, "$your_name" !"
greeting_1="hello, ${your_name} !"
echo $greeting  $greeting_1
# 使用单引号拼接
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
greeting_2='hello, '$your_name' !'
greeting_3='hello, ${your_name} !'
echo $greeting_2  $greeting_3
---
hello, 小学弟 ! hello, 小学弟 !
hello, 小学弟 ! hello, ${your_name} !
```

获取字符串长度

```
string="abcd"
echo ${#string}
---
4
```

提取子字符串

以下实例从字符串第0个字符**开始截取**10个字符：

```
string="testerhome is a great site"
echo ${string:0:10}
---
testerhome
```

查找子字符串

查找字符e或i的位置(哪个字母先出现就计算哪个)：

```
string="testerhome is a great site"
echo `expr index "$string" ei`
---
2
```

1. 以上脚本中 `是**反引号**，而不是单引号 '，不要看错了。
2. 输出的位置从1开始计数，比如：t处于"testerhome is a great site"的1处。

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510

数组

```
 array=(1 "abcd" 4 6)
echo ${array[0]}
echo ${array[*]}
echo ${#array[*]}

---
1
1 abcd 4 6
3
```

1. 使用 ()来定义数组变量，中间使用空格隔开。
2. 使用*来输出所有元素。
3. 使用#来获取数组中有多少元素。

注释

以#开头的行就是注释，会被解释器忽略。

```
#--------------------------------------------
# 这是一个注释
# author：霍格沃兹
# site：https://testing-studio.com/
# slogan：学的不仅是技术，更是梦想！
#--------------------------------------------
##### 用户配置区 开始 #####
#
#
# 这里可以添加脚本描述信息
#
#
##### 用户配置区 结束  #####
```

下面是多行注释：

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510

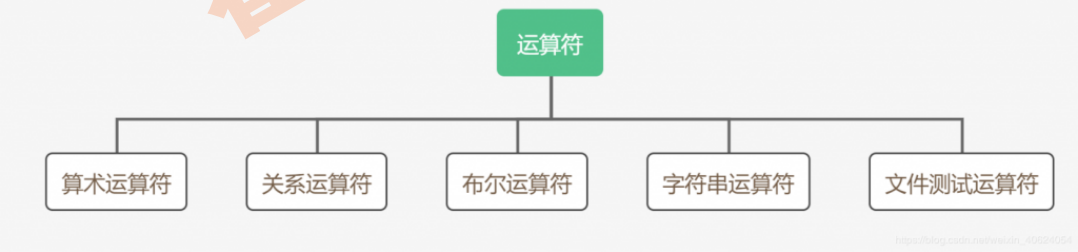

```
:<<EOF
注释内容...
注释内容...
注释内容...
EOF
```

EOF可以换成别的符号:

```
:<<'
注释内容...
注释内容...
注释内容...
'

:<<!
注释内容...
注释内容...
注释内容...
!
```

霍格沃兹测试学院

基本运算符



Shell 和其他编程语言一样，支持多种运算符，包括：

- 算数运算符
- 关系运算符
- 布尔运算符
- 字符串运算符
- 文件测试运算符

软件测试开发职位内推Q群：485353510

需要注意的是，原生bash**不支持**简单的数学运算，但是可以通过其他命令来实现，例如 awk和expr。所谓的expr是一款**表达式**计算工具，使用它能完成表达式的求值操作.例如两个数相加：

```
val=`expr 2 + 2`
echo "$val"
---
4
```

1. 表达式和运算符之间要有**空格**，2+2是不对的，必须写成2 + 2。
2. 完整的表达式要被**反引号**`包含，与上面查找字符串一样，不要看错了。

算术运算符

下表列出了常用的算术运算符，假定变量a为10，变量 b为20：

运算符	说明	举例
+	加法	`expr $a + $b` 结果为 30。
–	减法	`expr $a – $b` 结果为 -10。
*	乘法	`expr $a * $b` 结果为 200
/	除法	`expr $b / $a` 结果为 2
%	取余	`expr $b % $a` 结果为 0
=	赋值	a=$b 将把变量b的值赋给 a
==	相等	[$a == $b] 返回 false。
!=	不相等	[$a != $b] 返回 true。

> 注意：条件表达式要放在方括号之间，并且要有空格，例如: [$a==$b] 是错误的，必须写成 [$a == $b]。

算术运算符实例如下：

```
a=10
b=20

val=`expr $a + $b`
echo "a + b : $val"
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
val=`expr $a - $b`
echo "a - b : $val"

val=`expr $a \* $b`
echo "a * b : $val"

val=`expr $b / $a`
echo "b / a : $val"

val=`expr $b % $a`
echo "b % a : $val"

if [ $a == $b ]
then
   echo "a 等于 b"
fi
if [ $a != $b ]
then
   echo "a 不等于 b"
fi
---

a + b : 30
a - b : -10
a * b : 200
b / a : 2
b % a : 0
a 不等于 b
```

霍格沃兹测试学院

注意：

- 乘号*前边必须加反斜杠\才能实现乘法运算；
- if...then...fi 是条件语句，后续将会讲解。
- 在 MAC 中 shell 的 expr 语法是：$((表达式))，此处表达式中的 * 不需要转义符号 \ 。

关系运算符

关系运算符只支持**数字**，不支持字符串，除非字符串的值是数字。
下表列出了常用的关系运算符，假定变量 a为10，变量b为 20：

软件测试开发职位内推Q群：485353510

运算符	说明	举例
-eq	检测两个数是否相等，相等返回 true	[$a -eq $b] 返回 false
-ne	检测两个数是否不相等，不相等返回 true	[$a -ne $b] 返回 true
-gt	检测左边的数是否大于右边的，如果是，则返回 true	[$a -gt $b] 返回 false
-lt	检测左边的数是否小于右边的，如果是，则返回 true	[$a -lt $b] 返回 true
-ge	检测左边的数是否大于等于右边的，如果是，则返回 true	[$a -ge $b] 返回false
-le	检测左边的数是否小于等于右边的，如果是，则返回 true	[$a -le $b] 返回true

运算符例子如下：

```
a=10
b=20

if [ $a -eq $b ]
then
   echo "$a -eq $b : a 等于 b"
else
   echo "$a -eq $b: a 不等于 b"
fi

if [ $a -ne $b ]
then
   echo "$a -ne $b: a 不等于 b"
else
   echo "$a -ne $b : a 等于 b"
fi

if [ $a -gt $b ]
then
   echo "$a -gt $b: a 大于 b"
else
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
    echo "$a -gt $b: a 不大于 b"
fi

if [ $a -lt $b ]
then
    echo "$a -lt $b: a 小于 b"
else
    echo "$a -lt $b: a 不小于 b"
fi

if [ $a -ge $b ]
then
    echo "$a -ge $b: a 大于或等于 b"
else
    echo "$a -ge $b: a 小于 b"
fi

if [ $a -le $b ]
then
    echo "$a -le $b: a 小于或等于 b"
else
    echo "$a -le $b: a 大于 b"
fi
---

10 -eq 20: a 不等于 b
10 -ne 20: a 不等于 b
10 -gt 20: a 不大于 b
10 -lt 20: a 小于 b
10 -ge 20: a 小于 b
10 -le 20: a 小于或等于 b
```

霍格沃兹测试学院

布尔运算符

下表列出了常用的布尔运算符，假定变量 a 为 10，变量 b 为 20：

运算符	说明	举例
非运算，表达式为 true 则返回 false，否则返回 true	[! false] 返回 true	

软件测试开发职位内推Q群：485353510

运算符	说明	举例
或运算，有一个表达式为 true 则返回 true	[$a -lt 20 -o $b -gt 100] 返回 true	
与运算，两个表达式都为 true 才返回 true	[$a -lt 20 -a $b -gt 100] 返回 false	

```
a=10
b=20

if [ $a != $b ]
then
   echo "$a != $b : a 不等于 b"
else
   echo "$a != $b: a 等于 b"
fi

if [ $a -lt 100 -a $b -gt 15 ]
then
   echo "$a 小于 100 且 $b 大于 15 : 返回 true"
else
   echo "$a 小于 100 且 $b 大于 15 : 返回 false"
fi

if [ $a -lt 100 -o $b -gt 100 ]
then
   echo "$a 小于 100 或 $b 大于 100 : 返回 true"
else
   echo "$a 小于 100 或 $b 大于 100 : 返回 false"
fi

if [ $a -lt 5 -o $b -gt 100 ]
then
   echo "$a 小于 5 或 $b 大于 100 : 返回 true"
else
   echo "$a 小于 5 或 $b 大于 100 : 返回 false"
fi
---
10 != 20 : a 不等于 b
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
10 小于 100 且 20 大于 15 : 返回 true
10 小于 100 或 20 大于 100 : 返回 true
10 小于 5 或 20 大于 100 : 返回 false
```

逻辑运算符

以下介绍 Shell 的逻辑运算符，假定变量a为10，变量b为20:

运算符	说明	举例
逻辑的 AND	[[$a -lt 100 && $b -gt 100]] 返回 false	
逻辑的 OR	[[$a -lt 100 || $b -gt 100]] 返回 true	

逻辑运算符实例如下:

```
a=10
b=20

if [[ $a -lt 100 && $b -gt 100 ]]
then
   echo "返回 true"
else
   echo "返回 false"
fi

if [[ $a -lt 100 || $b -gt 100 ]]
then
   echo "返回 true"
else
   echo "返回 false"
fi
---
返回 false
返回 true
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510

有人可能有疑问，布尔与，布尔或和这里的&&和||有什么区别呀?
首先，语法上很明显的看出逻辑运算符需要[[两个中括号，而布尔运算符用[一个中括号。
其次，&&为短路与，&&之前为true时，则执行&&之后的命令，&&之前的命令为false时，**不会**执行&&后的命令。
同理，||为短路或，||之前为false时，则执行||后的命令，||之前为true时，**不会**执行||后的命令。
而布尔运算符无论真假都会执行后面的命令。

字符串运算符

下表列出了常用的字符串运算符，假定变量a为"abc"，变量b为 "efg"：

运算符	说明	举例
=	检测两个字符串是否相等，相等返回 true	[$a = $b] 返回 false
!=	检测两个字符串是否相等，不相等返回 true	[$a != $b] 返回 true
-z	检测字符串长度是否为0，为0返回 true	[-z $a] 返回 false
-n	检测字符串长度是否为0，不为0返回 true	[-n "$a"] 返回 true
$	检测字符串是否为空，不为空返回 true	[$a] 返回 true

字符串运算符实例如下：

```
a="abc"
b="efg"

if [ $a = $b ]
then
   echo "$a = $b : a 等于 b"
else
   echo "$a = $b: a 不等于 b"
fi

if [ $a != $b ]
then
   echo "$a != $b : a 不等于 b"
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
else
   echo "$a != $b: a 等于 b"
fi

if [ -z $a ]
then
   echo "-z $a : 字符串长度为 0"
else
   echo "-z $a : 字符串长度不为 0"
fi

if [ -n "$a" ]
then
   echo "-n $a : 字符串长度不为 0"
else
   echo "-n $a : 字符串长度为 0"
fi

if [ $a ]
then
   echo "$a : 字符串不为空"
else
   echo "$a : 字符串为空"
fi
---
abc = efg: a 不等于 b
abc != efg : a 不等于 b
-z abc : 字符串长度不为 0
-n abc : 字符串长度不为 0
abc : 字符串不为空
```

文件测试运算符

文件测试运算符用于检测 Unix文件的各种属性。

属性检测描述如下：

操作符	说明	举例
-b file	检测文件是否是块设备文件，如果是，则返回 true	[-b $file] 返回 false

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510

操作符	说明	举例
-c file	检测文件是否是字符设备文件，如果是，则返回 true	[-c $file] 返回 false
-d file	检测文件是否是目录，如果是，则返回 true	[-d $file] 返回 false
-f file	检测文件是否是普通文件（既不是目录，也不是设备文件），如果是，则返回 true	[-f $file] 返回 true
-g file	检测文件是否设置了 SGID 位，如果是，则返回 true	[-g $file] 返回 false
-k file	检测文件是否设置了粘着位(Sticky Bit)，如果是，则返回 true	[-k $file] 返回 false
-p file	检测文件是否是有名管道，如果是，则返回 true	[-p $file] 返回 false
-u file	检测文件是否设置了 SUID 位，如果是，则返回 true	[-u $file] 返回 false
-r file	检测文件是否可读，如果是，则返回 true	[-r $file] 返回 true
-w file	检测文件是否可写，如果是，则返回 true	[-w $file] 返回 true
-x file	检测文件是否可执行，如果是，则返回 true	[-x $file] 返回 true
-s file	检测文件是否为空（文件大小是否大于0），不为空返回 true	[-s $file] 返回 true
-e file	检测文件（包括目录）是否存在，如果是，则返回 true	[-e $file] 返回 true

霍格沃兹测试学院

实例：

变量file表示文件"/var/www/testerhome/test.sh"，它的大小为100字节，具有 rwx 权限。下面的代码，将检测该文件的各种属性：

```
file="/var/www/testerhome/test.sh"
if [ -r $file ]
```

软件测试开发职位内推Q群：485353510

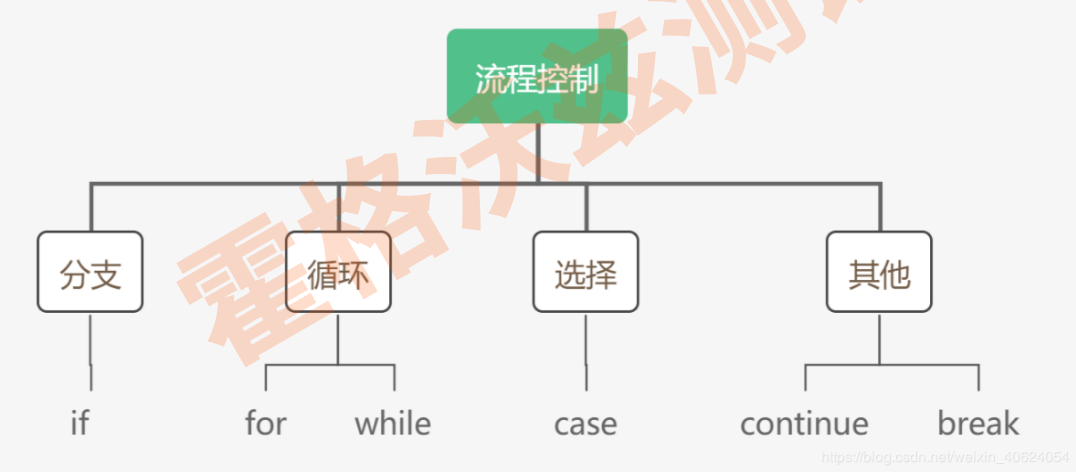

```
then
   echo "文件可读"
else
   echo "文件不可读"
fi

if [ -w $file ]
then
   echo "文件可写"
else
   echo "文件不可写"
fi

if [ -x $file ]
then
   echo "文件可执行"
else
   echo "文件不可执行"
fi

if [ -f $file ]
then
   echo "文件为普通文件"
else
   echo "文件为特殊文件"
fi

if [ -d $file ]
then
   echo "文件是个目录"
else
   echo "文件不是个目录"
fi

if [ -s $file ]
then
   echo "文件不为空"
else
   echo "文件为空"
fi
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
if [ -e $file ]
then
   echo "文件存在"
else
   echo "文件不存在"
fi
---
文件可读
文件可写
文件可执行
文件为普通文件
文件不是个目录
文件不为空
文件存在
```

流程控制



流程控制与其他语言一样，非常非常重要，他们不多但是我们会使用，因此在每个命令下，我都有很多例子。

if else

if 语句语法格式：

```
if condition
then
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
    command1
    command2
    ...
    commandN
fi
```

上面的命令可以也可以写成一行（适用于终端命令提示符）：

```
if [ 2 -gt 1 ]; then echo "true"; fi
```

末尾的fi就是if倒过来拼写，后面还会遇到类似的。
if else 语法格式：

```
if condition
then
    command1
    command2
    ...
    commandN
else
    command
fi
```

霍格沃兹测试学院

其中的else指：当condition为false时，执行command。
下面是elif的使用，elif的意思其实就是else if：

```
a=10
b=20
if [ $a == $b ]
then
   echo "a 等于 b"
elif [ $a -gt $b ]
then
   echo "a 大于 b"
elif [ $a -lt $b ]
then
```

软件测试开发职位内推Q群：485353510


```
    echo "a 小于 b"
else
    echo "没有符合的条件"
fi
---
a 小于 b
```

for

for循环一般格式为：

```
for var in item1 item2 ... itemN
do
    command1
    command2
    ...
    commandN
done
```

也可以写成一行：

```
for var in item1 item2 ... itemN; do command1; command2… done;
```

in后面的item是**列表**，它可以是列表，字符串，变量，甚至是一条命令，举例如下：
列表

```
for loop in 1 2 3 4 5
do
    echo "The value is: $loop"
done
---
The value is: 1
The value is: 2
The value is: 3
The value is: 4
The value is: 5
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510

字符串

```
for str in 'This is a testerhome'
do
    echo $str
done
---
This is a testerhome
```

变量

```
list="the student of testerhome"
#向已有列表中添加或拼接一个值
list=$list" success"
for state in $list
do
  echo "this word  is $state"
done
---
this word  is the
this word  is student
this word  is of
this word  is testerhome
this word  is success
```

命令

> states文件的内容：
> Alabama BOb
> Tom Console

```
file="states"
#for state in `ps -ef | grep  'tomcat.8701' | awk '{print $2}'`
for state in $(cat $file)
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
do
   echo "visit beautiful $state"
done
---
visit beautiful Alabama
visit beautiful BOb
visit beautiful Tom
visit beautiful Console
```

while 语句

```
while condition
do
    command
done
```

以下是一个基本的while循环：如果int小于等于5，int从0开始，每次循环处理时，int加1。

```
int=1
while(( $int<=5 ))
do
    echo $int
    let "int++"
done
---
1
2
3
4
5
```

Bash let 命令，它用于执行一个或多个表达式，变量计算中不需要加上$ 来表示变量，具体可查阅：Bash let 命令

while循环可用于读取键盘信息。下面的例子中，输入信息被设置为变量FILM，按结束循环。

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
echo '按下 <CTRL-D> 退出'
echo -n '输入你最喜欢的网站名: '
while read FILM
do
    echo "是的! $FILM 是一个好学院"
done
```

运行脚本，输出类似下面：

```
按下 <CTRL-D> 退出
输入你最喜欢的网站名:霍格沃兹
是的! 霍格沃兹 是一个好学院
```

case

case语句为**多选择**语句。可以用case语句匹配一个值与一个模式，如果匹配成功，执行相匹配的命令。case语句格式如下：

```
case 值 in
模式1)
    command1
    command2
    ...
    commandN
    ;;
模式2）
    command1
    command2
    ...
    commandN
    ;;
esac
```

1. 取值后面必须为单词in，每一模式必须以)结束。
2. 取值可以为**变量或常数**。
3. 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;;。
4. 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。
5. 如果无一匹配模式，使用*捕获该值，再执行后面的命令。

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510

下面的脚本提示输入1到4，与每一种模式进行匹配：

```
echo '输入 1 到 4 之间的数字:'
echo '你输入的数字为:'
read aNum
case $aNum in
    1)  echo '你选择了 1'
    ;;
    2)  echo '你选择了 2'
    ;;
    3)  echo '你选择了 3'
    ;;
    4)  echo '你选择了 4'
    ;;
    *)  echo '你没有输入 1 到 4 之间的数字'
    ;;
esac
---

输入 1 到 4 之间的数字:
你输入的数字为:
3
你选择了 3
```

霍格沃兹测试学院

break和continue

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，Shell使用两个命令来实现该功能：break和continue。

break命令

break命令允许跳出所有循环（终止执行后面的所有循环）。

下面的例子中，脚本进入死循环直至用户输入数字大于5。要跳出这个循环，返回到shell提示符下，需要使用break命令。

```
while :
do
    echo -n "输入 1 到 5 之间的数字:"
    read aNum
```

软件测试开发职位内推Q群：485353510


```
    case $aNum in
        1|2|3|4|5) echo "你输入的数字为 $aNum!"
        ;;
        *) echo "你输入的数字不是 1 到 5 之间的! 游戏结束"
            break
        ;;
    esac
done
---
输入 1 到 5 之间的数字:3
你输入的数字为 3!
输入 1 到 5 之间的数字:7
你输入的数字不是 1 到 5 之间的! 游戏结束
```

continue

continue命令与break命令类似，它表示**跳过本次**，开始下一次循环迭代。

对上面的例子进行修改：

```
#!/bin/bash
while :
do
    echo -n "输入 1 到 5 之间的数字: "
    read aNum
    case $aNum in
        1|2|3|4|5) echo "你输入的数字为 $aNum!"
        ;;
        *) echo "你输入的数字不是 1 到 5 之间的!"
            continue
            echo "游戏结束"
        ;;
    esac
done
esac
```

当输入大于5的数字时，执行到continue，会跳到新一轮循环，不会执行echo "游戏结束"。

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510

函数

shell可以用户定义**函数**，然后在shell脚本中可以随便调用。
shell中函数的定义格式如下：

```
[ function ] funname [()]

{

    action;
   [return int;]
}
```

1. 可以带function fun() 定义，也可以直接fun() 定义,不带任何参数。
2. 参数返回，可以return返回，如果不加，将以**最后一条命令**作为返回值。
   下面的例子定义了一个函数并进行调用：

```
demoFun(){
    echo "这是我的第一个 shell 函数!"
}
echo "-----函数开始执行-----"
demoFun
echo "-----函数执行完毕-----"
---

-----函数开始执行-----
这是我的第一个 shell 函数!
-----函数执行完毕-----
```

下面定义一个带有return语句的函数：

```
funWithReturn(){
    echo "这个函数会对输入的两个数字进行相加运算..."
    echo "输入第一个数字: "
    read aNum
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
    echo "输入第二个数字: "
    read anotherNum
    return $(($aNum+$anotherNum))
}
funWithReturn
echo "输入的两个数字之和为 $? !"
---
这个函数会对输入的两个数字进行相加运算...
输入第一个数字:
1
输入第二个数字:
2
两个数字分别为 1 和 2 !
输入的两个数字之和为 3 !
```

1. 函数返回值在调用该函数后通过$?来获得。
2. 所有函数在**使用前**必须定义。

函数参数

在Shell中，调用函数时可以向其**传递参数**。在函数体内部，通过$n的形式来获取参数的值，例如，$1表示第一个参数，$2表示第二个参数…

带参数的函数示例：

```
funWithParam(){
    echo "第一个参数为 $1 !"
    echo "第二个参数为 $2 !"
    echo "第十个参数为 $10 !"
    echo "第十个参数为 ${10} !"
    echo "第十一个参数为 ${11} !"
    echo "参数总数有 $# 个!"
    echo "作为一个字符串输出所有参数 $* !"
}
funWithParam 1 2 3 4 5 6 7 8 9 34 73
---
第一个参数为 1 !
第二个参数为 2 !
第十个参数为 10 !
第十个参数为 34 !
```

霍格沃兹测试学院

软件测试开发职位内推Q群：485353510


```
第十一个参数为 73 !
参数总数有 11 个!
作为一个字符串输出所有参数 1 2 3 4 5 6 7 8 9 34 73 !
```

1. $10不能获取第十个参数，获取第十个参数需要${10}。
2. 当**n>=10**时，需要使用${n}来获取参数。

另外，还有几个特殊字符用来处理参数：

参数处理	说明
$#	传递到脚本的参数个数
$*	以一个单字符串显示所有向脚本传递的参数
$$	脚本运行的当前进程ID号
$!	后台运行的最后一个进程的ID号
$@	与$*相同，但是使用时加引号，并在引号中返回每个参数
$-	显示Shell使用的当前选项，与set命令功能相同
$?	显示最后命令的退出状态，0表示没有错误，其他任何值表明有错误

相关链接

霍格沃兹测试学院官网首页:https://testing-studio.com

霍格沃兹测试学院