# 第17章 条件测试

写脚本时，有时要判断字符串是否相等，可能还要检查文件状态或是数字测试。基于这些测试才能做进一步动作。Test命令用于测试字符串，文件状态和数字，也很适合于下一章将提到的if、then、else条件结构。

本章内容有：

- 对文件、字符串和数字使用test命令。
- 对数字和字符串使用expr命令。

expr命令测试和执行数值输出。使用最后退出状态命令 $?可测知test和expr，二者均以0表示正确，1表示返回错误。

## 17.1 测试文件状态

test一般有两种格式，即：

```
test condition
```

或

```
[condition]
```

使用方括号时，要注意在条件两边加上空格。

测试文件状态的条件表达式很多，但是最常用的可在表 17-1中查到。

**表17-1 文件状态测试**

| | | | |
|---|---|---|---|
| -d | 目录 | -s | 文件长度大于0、非空 |
| -f | 正规文件 | -w | 可写 |
| -L | 符号连接 | -u | 文件有suid位设置 |
| -r | 可读 | -x | 可执行 |

使用两种方法测试文件scores.txt是否可写并用最后退出状态测试是否成功。记住， 0表示成功，其他为失败。

```
$ ls -l scores.txt
-rw-r--r--   1 dave   admin  0 May 15 11:29 scores.txt
$ [ -w scores.txt ]
$ echo $?
0

$ test -w scores.txt
$ echo $?
0
```

两种状态均返回0，可知文件scores.txt可写，现在测试其是否可执行：

```
$ [ -x scores.txt ]
$ echo $?
1
```

查看文件scores.txt权限列表，可知结果正如所料。

下面的例子测试是否存在appsbin目录

```
drwxr-xr-x  2 dave  admin    1024 May 15 15:53 appsbin
$ [ -d appsbin ]
$ echo $?
0
```

目录appsbin果然存在。

测试文件权限是否设置了suid位

```
-rwsr-xr--  1 root  root    28 Apr 30 13:12 xab
$ [ -u xab ]
$ echo $?
0
```

从结果知道suid位已设置。

## 17.2 测试时使用逻辑操作符

测试文件状态是否为OK，但是有时要比较两个文件状态。 shell提供三种逻辑操作完成此功能。

-a 逻辑与，操作符两边均为真，结果为真，否则为假。

-o 逻辑或，操作符两边一边为真，结果为真，否则为假。

！ 逻辑否，条件为假，结果为真。

下面比较两个文件：

```
-rw-r--r--    1 root   root      0 May 15 11:29 scores.txt
-rwxr-xr--    1 root   root      0 May 15 11:49 results.txt
```

下面的例子测试两个文件是否均可读。

```
$ [ -w results.txt -a -w scores.txt ]
$ echo $?
0
```

结果为真。

要测试其中一个是否可执行，使用逻辑或操作。

```
$ [ -x results.txt -o -x scores.txt ]
$ echo $?
0
```

scores.txt不可执行，但results.txt可执行。

要测试文件results.txt是否可写、可执行：

```
$ [ -w results.txt -a -x results.txt ]
$ echo $?
0
```

结果为真。

## 17.3 字符串测试

字符串测试是错误捕获很重要的一部分，特别在测试用户输入或比较变量时尤为重要。字符串测试有5种格式。

```
test "string"
test string_operator "string"
test "string" string_operator "string"
[ string_operator string ]
```

**[ string string_operator string ]**

这里，string_operator可为：

= 两个字符串相等。

！= 两个字符串不等。

-z 空串。

-n 非空串。

要测试环境变量EDITOR是否为空：

```
$ [ -z $EDITOR ]
$ echo $?
1
```

非空，取值是否是vi？

```
$ [ $EDITOR = "vi" ]
$ echo $?
0
```

是的，用echo命令反馈其值：

```
$ echo $EDITOR
vi
```

测试变量tape与变量tape2是否相等：

```
$ TAPE="/dev/rmt0"
$ TAPE2="/dev/rmt1"
$ [ "$TAPE" = "$TAPE2" ]
$ echo $?
1
```

不相等。没有规定在设置变量时一定要用双引号，但在进行字符串比较时必须这样做。

测试变量tape与tape2是否不相等。

```
$ [ "$TAPE" != "$TAPE2" ]
$ echo $?
0
```

是的，它们不相等。

## 17.4 测试数值

测试数值可以使用许多操作符，一般格式如下：

```
"number"numeric_operator"number"
```

或者

```
["number"numeric_operator"number"]
```

numeric_operator可为：

-eq 数值相等。

-ne 数值不相等。

-gt 第一个数大于第二个数。

-lt 第一个数小于第二个数。

-le 第一个数小于等于第二个数。

-ge 第一个数大于等于第二个数。

下面的例子返回结果都一样。均为测试两个数是否相等（130是否等于130）。

```
$ NUMBER=130
$ [ "$NUMBER" -eq "130" ]
$ echo $?
0
```

结果果然正确。

改变第二个数，结果返回失败，状态1（130不等于200）

```
$ [ "$NUMBER" -eq "100" ]
$ echo $?
1
```

测试130是否大于100：

```
$ [ "$NUMBER" -gt "100" ]
$ echo $?
0
```

当然。

也可以测试两个整数变量。下面测试变量 source_count是否小于dest_count:

```
$ SOURCE_COUNT=13
$ DEST_COUNT=15
$ [ "$DEST_COUNT" -gt "$SOURCE_COUNT" ]
$ echo $?
0
```

可以不必将整数值放入变量，直接用数字比较即可，但要加引号。

```
$ [ "990" -le "995" ]
$ echo $?
0
```

可以用逻辑操作符将两个测试表达式结合起来。仅需要用到一对方括号，而不能用两个，否则将返回错误信息“ too many arguments ”。

```
$ [ "990" -le "995" ] -a [ "123" -gt "33" ]
sh:[: too many arguments
```

下面例子测试两个表达式，如果都为真，结果为真，正确使用方式应为：

```
$ [ "990" -le "995"  -a  "123" -gt "33" ]
$ echo $?
0
```

## 17.5 expr用法

expr命令一般用于整数值，但也可用于字符串。一般格式为：

```
expr argument operator argument
```

expr也是一个手工命令行计数器。

```
$ expr 10 + 10
20
$ expr 900 + 600
1500

$ expr 30 / 3
10

$ expr 30 / 3 / 2
5
```

使用乘号时，必须用反斜线屏蔽其特定含义。因为 shell可能会误解显示星号的意义。

```
$ expr 30 \* 3
90
```

### 17.5.1 增量计数

expr在循环中用于增量计算。首先，循环初始化为 0，然后循环值加 1，反引号的用法意即替代命令。最基本的一种是从（expr）命令接受输出并将之放入循环变量。

```
$ LOOP=0
$ LOOP=`expr $LOOP + 1`
```

### 17.5.2 数值测试

可以用expr测试一个数。如果试图计算非整数，将返回错误。

```
$ expr rr + 1
expr: non-numeric argument
```

这里需要将一个值赋予变量（不管其内容如何），进行数值运算，并将输出导入 dev/null，然后测试最后命令状态，如果为 0，证明这是一个数，其他则表明为非数值。

```
$ VALUE=12
$ expr $VALUE + 10 > /dev/null 2>&1
$ echo $?
0
```

这是一个数。

```
$ VALUE=hello
$ expr $VALUE + 10 > /dev/null 2>&1
$ echo $?
2
```

这是一个非数值字符。

expr也可以返回其本身的退出状态，不幸的是返回值与系统最后退出命令刚好相反，成功返回1，任何其他值为无效或错误。下面的例子测试两个字符串是否相等，这里字符串为“ hello ”和“ hello ”。

```
$ VALUE=hello
$ expr $VALUE = "hello"
1
$ echo $?
0
```

expr返回1。不要混淆了，这表明成功。现在检验其最后退出状态，返回 0表示测试成功，“ hello ”确实等于“ hello ”。

### 17.5.3 模式匹配

expr也有模式匹配功能。可以使用 expr通过指定冒号选项计算字符串中字符数。 .*意即任何字符重复 0次或多次。

```
$ VALUE=accounts.doc
$ expr $VALUE : October 8, '.*'
12
```

在expr中可以使用字符串匹配操作，这里使用模式 .doc抽取文件附属名。

```
$ expr $VALUE : '\(.*\).doc'
accounts
```

## 17.6 小结

本章涉及expr和test基本功能，讲到了怎样进行文件状态测试和字符串赋值，使用其他的条件表达式如if then else和case可以进行更广范围的测试及对测试结果采取一些动作。