第15章 引　　号

上一章介绍了变量和替换操作，在脚本中执行变量替换时最容易犯的一个错误就是由于引用错误。在命令行中引用是很重要的。

本章内容有：

- 引用的必要性。
- 双引、单引和反引号。
- 使用反斜线实现屏蔽。

15.1 引用必要性

这里只讲述引用的基本规则。因为使用引用的例子很多。本书接下来的两个部分将一一予以讲述。脚本中执行行操作时， shell将对脚本设置予以解释。要采取一种方法防止 shell这样做，即使用引用号，包括各式引用或使用反斜线。

一些用户在对文本字符串进行反馈操作时觉得使用引用很麻烦。有时不注意，只引用了一半，这时问题出现了。最好在反馈文本字符串时使用双引号。下面是各种引用的例子。

```
$ echo Hit the star button to exit *

Hit the star button to exit DIR_COLORS HOSTNAME Muttrc X11 adjtime
aliases alias
...
```

文本返回了，但由于未使用双引号， *被shell误解， shell认为用户要做目录列表。用双引号得结果如下：

```
$ echo "Hit the star button to exit *"
Hit the star button to exit *
```

这样就不会有误解产生。表15-1列出各种引用类型。

表15-1　shell引用类型

""	双引号	`	反引号
'	单引号	\	反斜线

15.2 双引号

使用双引号可引用除字符$、`、\外的任意字符或字符串。这些特殊字符分别为美元符号，反引号和反斜线，对 shell来说，它们有特殊意义。如果使用双引号将字符串赋给变量并反馈它，实际上与直接反馈变量并无差别。

```
$ STRING="MAY DAY, MAY DAY, GOING DOWN"
$ echo "$STRING"
MAY DAY, MAY DAY, GOING DOWN

$ echo $STRING
MAY DAY, MAY DAY, GOING DOWN
```

现在假定要设置系统时间输出到变量 mydate。

```
$ MYDATE="date"
$ echo $MYDATE
date
```

因为shell将""符号里的字符串赋予变量 mydate，date已没有特定意义，故此变量只保存单词date。

如果要查询包含空格的字符串，经常会用到双引号。以下使用grep抽取名字“ Davey Wire ”，因为没有加双引号， grep将“ Davey ”认作字符串，而把“ Wire ”当作文件名。

```
$ grep "Davey Wire" /etc/passwd
grep: Wire: No such file or directory
```

要解决这个问题，可将字符串加双引号。这样 shell会忽略空格，当使用字符时，应总是使用双引号，无论它是单个字符串或是多个单词。

```
$ grep "Davey Wire" /etc/passwd
davyboy:9sdJUK2s:106:Davey Wire:/home/ap
```

在一个反馈的文本行里可以使用双引号将变量引起来。下面的例子中， shell反馈文本行，遇到符号$，知道这是一个变量，然后用变量值 boy替换变量$BOY。

```
$ BOY="boy"
$ echo " The $BOY did well"
 The boy did well

$ echo " The "$BOY" did well"
 The boy did well
```

15.3 单引号

单引号与双引号类似，不同的是 shell会忽略任何引用值。换句话说，如果屏蔽了其特殊含义，会将引号里的所有字符，包括引号都作为一个字符串。使用上一个例子，结果如下：

```
$ GIRL='girl'
$ echo "The '$GIRL' did well"
The 'girl' did well
```

15.4 反引号

反引号用于设置系统命令的输出到变量。 shell将反引号中的内容作为一个系统命令，并执行其内容。使用这种方法可以替换输出为一个变量。反引号可以与引号结合使用。下面将举例说明。

下面的例子中， shell试图替代单词hello为系统命令并执行它，因为 hello脚本或命令不存在，返回错误信息。

```
$ echo `hello`
sh: hello: command not found
```

现在用date命令再试一次。

```
$ echo `date`
Sun May 16 16:40:19 GMT 1999
```

这次命令有效， shell正确执行。

下面将命令输出设置为变量mydate，时间格式如下：

```
$ date +%A" the "%e" of "%B" "%Y
Sunday the 16 of May 1999
```

设置到mydate，并显示其值：

```
$ mydate=`date +%A" the "%e" of "%B" "%Y`
$ echo $mydate
Sunday the 16 of May 1999
```

当然也可以将date命令输出至mydate：

```
$ mydate=`date`
$ echo $mydate
Sun May 16 16:48:16 GMT 1999
```

另一个例子中，将反引号嵌在双引号里：

```
$ echo "The date today is `date`"
The date today is Sun May 16 16:56:53 GMT 1999
```

打印当前系统上用户数目：

```
$ echo "There are `who | wc -l` users on the system"
There are  13   users on the system
```

上面的例子中，打印字符串后， shell遇到反引号，将其看作一条命令执行它。

15.5 反斜线

如果下一个字符有特殊含义，反斜线防止 shell误解其含义，即屏蔽其特殊含义。下述字符包含有特殊意义：& * + ^ $ ` " | ?。

假定echo命令加*，意即以串行顺序打印当前整个目录列表，而不是一个星号 *。

```
$ echo *

conf.linuxconf conf.modules cron.daily cron.hourly cron.monthly
cron.weekly crontab csh.cshrc default dosemu.conf dosemu.users exports
fdprm fstab gettydefs gpm-root.c
onf group group- host.conf hosts hosts.allow hosts.deny httpd inetd
...
```

为屏蔽星号特定含义，可使用反斜线。

```
$ echo \*
*
```

上述语句同样可用于 $$命令， shell解释其为现在进程 ID号，使用反斜线屏蔽此意，仅打印$$。

```
$ echo $$
284
$ echo \$$
$$
```

在打印字符串时要加入八进制字符（ ASCII相应字符），必须在前面加反斜线，否则 shell将其当作普通数字处理。

```
$ echo " This is a copyright 251 sign"
 This is a copyright \251 sign
$ echo  " This is a copyright \251 sign"
 This is a copyright © sign
```

如果是LINUX，则……

记住使用-e选项来显示控制字符。

```
$ echo -e "This is a copyright \251 sign"
 This is a copyright © sign
```

使用命令expr时，用*表示乘法会出现错误，在*前加上反斜线才会正确。

```
$ expr 12 * 12
expr: syntax error


$ expr 12 \* 12
144
```

在echo命令中加入元字符，必须用反斜线起屏蔽作用。下面的例子要显示价格 $19.99。其中$屏蔽与不屏蔽将产生不同的结果。

```
$ echo "That video looks a good price for $19.99"
That video looks a good price for 9.99
```

使用反斜线屏蔽$，可得更好的结果。

```
$ echo "That video looks a good price for \$19.99"
That video looks a good price for $19.99
```

15.6 小结

在引用时会遇到一些问题且经常出错。我在使用引用时遵循两条规则：

1) 反馈字符串用双引号；但不要引用反馈本身。

2) 如果使用引用得到的结果不理想，再试另一种，毕竟只有三种引用方式，可以充分尝试。