

第6章 命令执行顺序

在执行某个命令的时候，有时需要依赖于前一个命令是否执行成功。例如，假设你希望将一个目录中的文件全部拷贝到另外一个目录中后，然后删除源目录中的全部文件。在删除之前，你希望能够确信拷贝成功，否则就有可能丢失所有的文件。

在本章中，我们将讨论：

- 命令执行控制。
- 命令组合。

如果希望在成功地执行一个命令之后再执行另一个命令，或者在一个命令失败后再执行另一个命令，`&&`和`||`可以完成这样的功能。相应的命令可以是系统命令或 shell脚本。

Shell还提供了在当前shell或子shell中执行一组命令的方法，即使用`()`和`{ }`。

6.1 使用&&

使用`&&`的一般形式为：

```
命令1 && 命令2
```

这种命令执行方式相当地直接。`&&`左边的命令（命令1）返回真（即返回0，成功被执行）后，`&&`右边的命令（命令2）才能够被执行；换句话说，“如果这个命令执行成功 `&&` 那么执行这个命令”。

这里有一个使用`&&`的简单例子：

```
$ cp justice.doc justice.bak && echo "if you are seeing this then cp was OK"
```

```
if you are seeing this then cp was OK
```

在上面的例子中，`&&`前面的拷贝命令执行成功，所以 `&&`后面的命令（`echo`命令）被执行。

再看一个更为实用的例子：

```
$ mv /apps/bin /apps/dev/bin && rm -r /apps/bin
```

在上面的例子中，`/apps/bin`目录将会被移到`/apps/dev/bin`目录下，如果它没有被成功执行，就不会删除`/apps/bin`目录。

在下面的例子中，文件`quarter_end.txt`首先将被排序并输出到文件`quarter.sorted`中，只有这一命令执行成功之后，文件`quarter.sorted`才会被打印出来：

```
$ sort quarter_end.txt > quarter.sorted && lp quarter.sorted
```

6.2 使用||

使用`||`的一般形式为：

```
命令1 || 命令2
```

||的作用有一些不同。如果||左边的命令(命令1)未执行成功,那么就执行||右边的命令(命令2);或者换句话说,“如果这个命令执行失败了||那么就执行这个命令”。

这里有一个使用||的简单例子:

```
$ cp wopper.txt oops.txt || echo "if you are seeing this cp failed"
```

```
cp: wopper.txt: No such file or directory
if you are seeing this cp failed
```

在上面的例子中,拷贝命令没有能够被成功执行,因此||后面的命令被执行。

这里有一个更为实用的例子。我希望从一个审计文件中抽取第1个和第5个域,并将其输出到一个临时文件中,如果这一操作未成功,我希望能够收到一个相应邮件:

```
$ awk '{print$1,$5}' acc.qtr >qtr.tmp || echo "Sorry the payroll
extraction didn't work" | mail dave
```

在这里不只可以使用系统命令;这里我们首先对 month_end.txt文件执行了一个名为 comet 的 shell 脚本,如果该脚本未执行成功,该 shell 将结束。

```
$ comet month_end.txt || exit
```

6.3 用 () 和 { } 将命令结合在一起

如果希望把几个命令合在一起执行, shell 提供了两种方法。既可以在当前 shell 也可以在子 shell 中执行一组命令。

为了在当前 shell 中执行一组命令,可以用命令分隔符隔开每一个命令,并把所有的命令用圆括号 () 括起来。

它的一般形式为:

```
(命令1;命令2;...)
```

如果使用 { } 来代替 (), 那么相应的命令将在子 shell 而不是当前 shell 中作为一个整体被执行,只有在 { } 中所有命令的输出作为一个整体被重定向时,其中的命令才被放到子 shell 中执行,否则在当前 shell 执行。它的一般形式为:

```
{命令1;命令2;...}
```

我很少单独使用这两种方法。我一般只和 && 或 || 一起使用这两种方法。

再回到前面那个 comet 脚本的例子,如果这个脚本执行失败了,我很可能会希望执行两个以上的命令,而不只是一个命令。我可以使用这两种方法。这是原先那个例子:

```
$ comet month_end.txt || exit
```

现在如果该脚本执行失败了,我希望先给自己发个邮件,然后再退出,可以用下面的方法来实现:

```
$ comet month_end || (echo "Hello, guess what! Comet did not work"|mail
dave; exit)
```

在上面的例子中,如果只使用了命令分隔符而没有把它们组合在一起, shell 将直接执行最后一个命令 (exit)。

我们再回头来看看前面那个使用 && 排序的例子,下面是原来的那个例子:

```
$ sort quarter_end.txt > quarter.sorted && lp quarter.sorted
```

使用命令组合的方法,如果 sort 命令执行成功了,可以先将输出文件拷贝到一个日志区,

然后再打印。

```
$ sort quarter_end.txt > quarter.sorted && ( cp quarter.sorted /logs/  
quarter.sorted; lp quarter.sorted )
```

6.4 小结

在编写 shell 脚本时，使用 `&&` 和 `||` 对构造判断语句非常有用。如果希望在前一个命令执行失败的情况不执行后面的命令，那么本章所讲述的方法非常简单有效。使用这样的方法，可以根据 `&&` 或 `||` 前面命令的返回值来控制其后面命令的执行。