CentOS 丛书目录 — 系统管理 — 网络服务 — 应用部署

Shell 脚本简介

内容提要

1. 理解 Shell 脚本的作用
2. 熟悉 Shell 脚本中的成分
3. 学会创建和执行 Shell 脚本
4. 熟悉 Shell 脚本的编码规范养成良好的编程习惯

什么是 Shell 脚本

Shell 除了是命令解释器之外还是一种编程语言，用 Shell 编写的程序类似于 DOS 下的批处理程序。

用户可以在文件中存放一系列的命令，通常将 Shell 编写的程序称为 Shell 脚本或Shell程序。

将命令、变量和流程控制有机地结合起来将会得到一个功能强大的编程工具。Shell脚本语言非常擅长处理文本类型的数据，由于Linux系统中的所有配置文件都是纯文本的，所以Shell脚本语言在管理Linux系统中发挥了巨大作用。

Shell脚本中的成分

Shell脚本是以行为单位的，在执行脚本的时候会分解成一行一行依次执行。脚本中所包含的成分主要有：注释、命令、Shell变量和结构控制语句。

- 注释 -- 注释部分用于对脚本进行解释和说明，在注释行的前面要加上符号#，这样在执行脚本的时候Shell就不会对该行进行解释。
- 命令 -- 在Shell脚本中可以出现任何在交互方式下可以使用的命令。
- 变量 -- Shell 支持字符串变量和整型变量。
- 结构控制语句 -- 用于编写复杂脚本的流程控制语句。

Shell脚本的建立与执行

用户可以使用任何文本编辑器编辑Shell脚本文件，如vi、gedit等。

对Shell脚本文件的调用可以采用三种方式：

一种是将文件名作为Shell命令的参数，其调用格式为：

```
$ bash script-file
```

当要被执行脚本文件没有可执行权限时只能使用这种调用方式。 另一种调用方法是先将脚本文件的访问权限改为可执行，以便该文件可以作为执行文件调用。具体方法是：

```
$ chmod u+x script-file
$ script-file
```

当执行一个脚本文件时，Shell就产生一个子Shell（即一个子进程）去执行命令文件中的命令。因此，脚本文件中的变量值不能传递到当前Shell（即父Shell）。

为了使得脚本文件中的变量值传递到当前Shell，必须在命令文件名前面加“source”或“.”命令。即：

```
$ source script-file
或
$ . script-file
```

“source”和“.”命令的功能是在当前Shell中执行脚本文件中的命令，而不是产生一个子Shell去执行命令文件中的命令。

下面给出一个不同方式执行 shell 脚本的例子。

```
# 首先编写一个名为 myset 的简单脚本文件。
$ cat >myset
mydir=`pwd`
export mydir
^d
# 显示脚本myset的内容
$ cat  myset
mydir=`pwd`
export mydir
$
# 为脚本添加执行权限并执行
$ chmod +x myset
# 在子Shell中执行脚本
$ ./myset
# 显示变量mydir的值
$ echo $mydir

$
# 由于这种执行脚本的方式是在子Shell中执行，
# 所以当脚本执行结束返回主Shell后，变量已没有值
# 用.命令执行脚本
$ . myset
# 显示变量mydir的值
$ echo $mydir
/home/lrj
$
# 由于这种执行脚本的方式是在当前Shell中执行，当脚本执行结束变量依然有值
```

与Windows或DOS环境不同，在Linux下没有将当前目录列入PATH环境变量，所以，当用户执行当前目录下的命令或脚本时应该使用如下形式的命令行：

```
$  ./command
$  ./script
```

Shell 脚本的编码规范

一个bash脚本的正确的起始部分应该以 #! 开头：

```
#!/bin/bash
```

在调用 bash 的脚本时候，以 #! 开头的语句通知系统用何解释器执行此脚本。

如果bash是你的默认shell，那么脚本的开头也不用非得写上#!。 但是如果你使用不同的shell来开启一个脚本的话，比如tcsh，那么你就必须需要#!了。

良好的Shell编码规范还要求以注释形式说明如下的内容：

- 脚本名称
- 脚本功能

- 作者及联系方式
- 版本更新记录
- 版权声明
- 复杂脚本应对算法做简要说明

本章为了节省篇幅，通常省略上述说明内容。

- 显示源文件
- 登录