Linux 基础

第 14 讲　shell 脚本基础

shell 脚本介绍

- shell 可以从一个文件读取命令并逐条执行，这个文件一般被称为 shell 脚本。
- shell 脚本最常用来做系统管理之类的工作，它通过组合现有的程序完成特定的任务。

shell 脚本和其他脚本语言

- shell 脚本看起来像是一种语言，并且也提供了逻辑和循环等关键字，也支持函数功能。
- 但是严格来说，它只是逐条执行语句，和常见的脚本语言如 Python、 JS、 Perl 等还是有很大区别的。

为何要使用 shell 脚本

- 没有任何理由要求必须要强制使用。
- 但是以下几点是需要考虑的因素：
  - 不需要安装其他软件即可完成自动化或批量任务处理
  - 可以通过少量的代码完成复杂的功能
  - 系统服务管理都使用了 shell 脚本
  - 公司要求 . . .

认识 shell 脚本

- 以下是一段 shell 脚本代码，它显示系统详细信息，并以树形结构显示设备总线。

```
1 #!/bin/bash
2
3 uname -a
4 lspci -vt
5
```

位于第一行的 #!/bin/bash

- 这不是 shell 所独有的，因为经常会见到：

```
#!/usr/bin/python
#!/usr/bin/node
```

- 这仅仅是告诉系统，如果遇到这类文件，应该使用哪个程序去执行。 #! 后面可以是任何可执行的程序路径。

第一行是 #!/bin/ls 就会显示当前目录的内容。

系统如何对待脚本类的可执行文件

- Linux 使用标志位来识别可执行文件。
- 但是当系统去执行程序时，发现不是支持的可执行文件格式[1]，那么就会认为是脚本。
- 此时会去扫描第一行 #！的标记，根据路径查找是否存在此程序。存在则交给此程序去执行。

[1] Linux/Unix 采用 ELF 格式的文件，而 Windows 使用了 PE 格式，并使用扩展名 exe 表示可执行文件。

一个简单的脚本所具备的条件

- 创建一个文本文件，扩展名可以有 `.sh` 也可以没有。
- 把要执行的命令按顺序编写。
- 最开头的 `#!` 如果没有则会使用默认的 `shell` 执行。
- 给文件加上可执行权限（但这不是必需的）。
- `#` 开头表示注释。

执行 shell 脚本的方式

- 添加可执行权限，并使用 #! 声明 ：

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

- 指定执行的 shell，这时候 #! 的声明会忽略：

```
bash a.sh
```

#!/bin/sh

- 脚本中经常出现 #!/bin/sh，而不是：

  #!/bin/bash

- sh 是一个符号链接指向 /bin/dash。
- dash 被设计用来快速执行脚本，功能不如 bash 强，但是执行速度快。

测试脚本： bash 和 dash 的不同

```
1 #!/bin/sh
2
3 for ((i=0; i<5; i++)) ; do
4     echo $i
5 done
```

指定使用 bash 运行： bash b.sh

```
0
1
2
3
4
```

使用默认的 sh 执行，不支持 for 循环扩展计算。

```
./b.sh: 3: ./b.sh: Syntax error: Bad for loop variable
```

变量

- 变量对于正规的程序很重要，它可以保存有用的数据，并且可以用于管理程序状态。
- 在 shell 中，变量很简单，就是保存字符串。
- 由于算数运算也是十分必要的，所以 bash 也有用运算的机制。

设置变量

- 在 shell 中设置变量使用以下方式：

  a=123 // 变量名称可以是字母数字下划线，但不能是数字开头

  b=123+234 # 这仅仅是保存了文本，不会进行计算

  c=“go php python” # 空格使用双引号或单引号

  d=’Linux Unix’

- = 左右不能有空格，否则会认为是命令去执行。

获取变量的值

- 获取变量的值使用 `$` 加变量的名称。
- 输出变量：

```
echo $a
```

计算

- 类似 `a=12+23` 这样的操作，并不会进行计算。
- 要使用算术运算，需要一些特殊的语法：

```
a=$((12+23))
a=234 ; b=345 ; c=$((a+b))
```

- `((···))` 会扩展其中的表达式进行计算， `$` 是取值操作。

使用内建命令 `let`

- `let` 可以对变量进行运算操作。
- 在编程语言中常见的算术操作都被支持。

```
let a++ ; let b=b+a ; let c=b*a+b
```

保存程序的执行结果

- 使用 ` 包含命令并赋值给变量，可以保存程序的执行结果。
- 例： `a=\`ls /usr/share\``
- 输出： `echo $a`
- 变量保存的是一个空格分割的列表。

清除变量

- 使用内建命令 `unset` 可以清除不需要的变量：

```
unset a b
```

脚本示例

以下脚本用于计算 /usr/share 目录所有 .json 文件的行数。

```
1 #!/bin/bash
2
3 JSON_LIST=`find /usr/share -iname *.json 2> /dev/null`
4
5 cat $JSON_LIST | wc -l
6
```