如何编写和使用自定义的 Shell 函数和函数库

内容

在 Linux 系统下，Shell 脚本可以在各种不同的情形下帮到我们，例如展示信息，甚至 自动执行特定的系统管理任务，创建简单的命令行工具等等。

我们将向 Linux 新手展示如何可靠地存储自定义的 shell 脚本，解释如何编写 shell 函数和函数库，以及如何在其它的脚本中使用函数库中的函数。

Shell 脚本要存储在何处

为了在执行你自己的脚本时不必输入脚本所在位置的完整或绝对路径，脚本必须被存储在 `$PATH` 环境变量所定义的路径里的其中一个。

使用下面的命令可以查看你系统中的 `$PATH` 环境变量：

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

通常来说，如果在用户的家目录下存在名为 `bin` 的目录，你就可以将 shell 脚本存储在那个目录下，因为那个目录会自动地被包含在用户的 `$PATH` 环境变量中（LCTT 译注：在 Centos 6/7 下是这样的，在 Debian 8 下不是这样的，在 Ubuntu 16.04 下又是这样的)。

因此，在你的主目录下创建 `bin` 目录吧（当然这里也可以用来存储 Perl、Awk 或 Python 的脚本，或者其它程序）：

```
$ mkdir ~/bin
```

接着，建立一个名为 `lib`（libraries 的简写）的目录来存放你自己的函数库。你也可以在其中存放其它编程语言的函数库，如 C ，Python 等语言。在 `lib` 目录下建立另一个名为 `sh` 的目录，这个目录将被用来存放你的 shell 函数库：

```
$ mkdir -p ~/lib/sh
```

创建你自己的 Shell 函数和函数库

一个 shell 函数 就是在脚本中能够完成特定任务的一组命令。它们的工作原理与其他编程语言中的过程（LCTT 译注：可能指的是类似 SQL 中的存储过程之类的吧）、子例程、函数类似。

编写一个函数的语法如下：

函数名() { 一系列的命令 }

(LCTT 校注：在函数名前可以加上 function 关键字，但也可省略不写）

例如，你可以像下面那样在一个脚本中写一个用来显示日期的函数：

showDATE() {date;}

每当你需要显示日期时，只需调用该函数的函数名即可：

$ showDATE

简单来说 shell 函数库也是一个 shell 脚本，不过你可以在一个函数库中仅存储其它 shell 脚本中需要调用的函数。

下面展示的是在我的 ~/lib/sh 目录下一个名为 libMYFUNCS.sh 的库函数：

```
#!/bin/bash
### Function to clearly list directories in PATH
showPATH() {
oldifs="$IFS" ### store old internal field separator
IFS=: ### specify a new internal field separator
for DIR in $PATH<br> do<br> echo $DIR<br> done
IFS="$oldifs" ### restore old internal field separator
}
### Function to show logged user
showUSERS() {
echo -e “Below are the user logged on the system:\n”
w
}
### Print a user’s details
printUSERDETS() {
oldifs="$IFS" ### store old internal field separator
```

```
IFS=: ### specify a new internal field separator
read -p "Enter user name to be searched:" uname ### read username
echo ""
### read and store from a here string values into variables
### using : as a field delimiter
read -r username pass uid gid comments homedir shell <<< "$(cat /etc/passwd | grep "^$uname")"
### print out captured values
echo -e "Username is : $username\n"
echo -e "User's ID : $uid\n"
echo -e "User's GID : $gid\n"
echo -e "User's Comments : $comments\n"
echo -e "User's Home Dir : $homedir\n"
echo -e "User's Shell : $shell\n"
IFS="$oldifs" ### store old internal field separator
}
```

保存文件并且给脚本添加执行权限。

如何从函数库中调用函数

要使用某个 `lib` 目录下的函数，首先你需要按照下面的形式 将包含该函数的函数库导入到需要执行的 shell 脚本中：

```
$ . /path/to/lib
或
$ source /path/to/lib
```

（LCTT 译注：第一行的 . 和路径间**必须**是有空格的）

这样你就可以像下面演示的那样，在其它的脚本中使用来自 `~/lib/sh/libMYFUNCS.sh` 的 `printUSERDETS` 函数了。

在下面的脚本中，如果要打印出某个特定用户的详细信息，你不必再一一编写代码，而只需要简单地调用已存在的函数即可。

创建一个名为 `test.sh` 的新文件：

```
#!/bin/bash
### include lib
. ~/lib/sh/libMYFUNCS.sh
```

```
### use function from lib
printUSERDETS
### exit script
exit 0
```

保存这个文件，并使得这个脚本可被执行，然后运行它：

```
$ chmod 755 test.sh
$ ./test.sh
```

编写 shell 函数

在本文中，我们介绍了在哪里可靠地存储 shell 脚本，如何编写自己的 shell 函数和函数库，以及如何在一个普通的 shell 脚本中从函数库中调用库中的某些函数。

在之后，我们还会介绍一种相当简单直接的方式来将 Vim 配置为一个编写 Bash 脚本的 IDE（集成开发环境）。在那之前，记得要经常关注我们 ，如果能和我们分享你对这份指南的想法就更好了。