Linux 下的 shell 编程入门

通常情况下，我们从命令行输入命令每输入一次就能够得到系统的一次响应。一旦需要我们一个接着一个的输入命令而最后才得到结果的时候，这样的做法显然就没有效率。要达到这样的目的，通常我们利用 **shell** 程序或者 **shell** 脚本来实现。

一、简介

Shell 编程有很多类似 C 语言和其他程序语言的特征，但是又没有编程语言那样复杂。Shell 程序就是放在一个文件中的一系列 Linux 命令和实用程序，在执行的时候，通过 Linux 一个接着一个地解释和执行每个命令。

下面我们来看一个简单的 shell 程序：

1、首先建立一个内容如下的文件，名字为 date,将其存放在目录下的 bin 子目录中。

#Program date

#usageto ::show the date in this way (注释)

echo “Mr.$USER,Today is:”

echo date “+%B%d%A”

echo “Whish you a lucky day !”

2、编辑完该文件之后它还不能执行，我们需要给它设置可执行权限。使用如下命令：

chmod +x date

通过以上过程之后，我们就可以像使用 ls 命令一样执行这个 shell 程序.

[beichen@localhost bin]$ date

Mr.beichen,Today is:

January 13 Friday

Whish you a lucky day !

为了在任何目录里都可以执行这个程序，可以将 bin 的这个目录添加到路径中去。

[beichen@localhost bin]$ PATH=$PATH:$HOME/bin

(注：这里的$HOME 代替的是/home/beichen,而 bin 目录是自己建的一个目录)

另外一种执行 date 的方法就是把它作为一个参数传给 shell 命令:

[beichen@localhost /]$ bash date

Mr.beichen,Today is:

January 13 Friday

Whish you a lucky day !

尽管在前面我们使用 chmod +x date 将 date 设置为可执行，其实不设置也没有关系，但在 Linux 里执行它，需要先告诉系统它是一个可执行的脚本。

[beichen@localhost /]$ .date

Mr.beichen,Today is:

January 13 Friday

Whish you a lucky day !

即在 date 前面加上一个点”.”，并且用空格与后面的 shell 脚本的文件名隔开。当然，不推荐这样做。

二、shell 参数

如同 ls 命令可以接受目录等作为它的参数一样，在 shell 编程时同样可以使用参数。Shell 有位置参数和内部参数。

1、 位置参数

由系统提供的参数称为位置参数。位置参数的值可以用$N 得到，N 是一个数字，如果为 1，即$1.类似 C 语言中的数组，Linux 会把输入的命令字符串分段并给每段进行标号，标号从 0 开始。第 0 号为程序名字，从 1 开始就表示传递给程序的参数。如$0 表示程序的名字，$1 表示传递给程序的第一个参数，以此类推。

2、 内部参数

上述过程中的$0 是一个内部变量，它是必须的，而$1 则可有可无。和$0 一样的内部变

量还有以下几个。

$# ----传递给程序的总的参数数目

$? ----上一个代码或者 shell 程序在 shell 中退出的情况，如果正常退出则返回 0，反之为非 0 值。

$* ----传递给程序的所有参数组成的字符串。

下面举例进行说明：

建立一个内容为如下的程序 P1:

echo “Program name is $0”

echo “There are totally $# parameters passed to this program”

echo “The last is $?”

echo “The parameters are $*”

执行后的结果如下：

[beichen@localhost bin]$ P1 this is a test program //传递 5 个参数

Program name is /home/beichen/bin/P1 //给出程序的完整路径和名字

There are totally 5 parameters passed to this program //参数的总数

The last is 0 //程序执行结果

The parameters are this is a test program //返回有参数组成的字符串

下面我们利用内部变量和位置参数编写一个名为 del 的简单删除程序：

```
#name: del

#author: liangnian

#this program to compress a file to the dustbin

if test $# -eq 0

then
```

echo “Please specify a file!”

else

gzip $1 //先对文件进行压缩

mv $1.gz $HOME/dustbin //移动到回收站

echo “File $1 is deleted !”

fi

三、变量表达式

在上面我们编写的小程序中我们用到了一个关键字 test，其实它是 shell 程序中的一个表达式?D?D 比较(test)。通过和 shell 提供的 if 等条件语句(后面我们会介绍)相结合我们可以方便的完判断。

其用法如下：

test 表达式

表达式所代表的操作符有字符串操作符、数字操作符、逻辑操作符以及文件操作符。其中文件操作符是一种 shell 独特的操作符，因为 shell 里的变量都是字符串，为了达到对文件进行操作的目的，于是才提供了这样的一种操作符。

1、 字符串比较

作用：测试字符串是否相等、长度是否为零，字符串是否为 NULL(注：bash 区分零长度字符串和空字符串)

常用的字符穿操作符有：

= 比较两个字符串是否相同，同则为“是”!= 比较两个字符串是否相同，不同则为“是”

-n 比较字符串长度是否大于零，如果大于零则为“是”

-z 比较字符串的穿度是否等于零，如果等于则为“是”

2、 数字比较

这里区别于其他编程语言，test 语句不使用>?类似的符号来表达大小的比较，而是

用整数式来表示这些。

-eq 相等

-ge 大于等于

-le 小于等于

-ne 不等于

-gt 大于

-lt 小于

3、 逻辑操作! 反：与一个逻辑值相反的逻辑值

-a 与(and)：两个逻辑值为“是”返回值才为“是”，反之为“否”

-o 或(or)：两个逻辑值有一个为“是”，返回值就为“是”

4、 文件操作

文件测试表达式通常是为了测试文件的信息，一般由脚本来决定文件是否应该备份、复制或删除。由于 test 关于文件的操作符有很多，我们只列举一些常用的。

-d 对象存在且为目录返回值为“是”

-f 对象存在且为文件返回值为“是”

-L 对象存在且为符号连接返回值为“是”

-r 对象存在且可读则返回值为“是”

-s 对象存在且长度非零则返回值为“是”

-w 对象存在且可写则返回值为“是”

-x 对象存在且可执行则返回值为“是”

file1 ?Cnt(-ot) file2 文件 1 比文件 2 新(旧)

四、循环结构语句

shell 常见的循环语句有 for 循环、while 循环、until 循环

for 循环

语法：for 变量 in 列表

do

操作

done

注：变量是要在循环内部用来指代当前所指代的列表中的那个对象的。

列表是在 for 循环的内部要操作的对象，可以是字符串也可以是文件，如果是文件则为文件名。

例：删除垃圾箱中的所有.gz 文件

#delete all file with extension of “gz” in the dustbin

for I in $HOME/dustbin/*.gz

do

rm ?Cf $i

echo “$i has been deleted!”

done

执行结果如下：

[beichen@localhost bin]$.f_rmgz

/home/beichen/dustbin/nessus-4.0.0.2.tar.gz has been deleted!

/home/beichen/dustbin/gftp-2.2.1.tar.gz has been deleted!

While 循环

语法：while 表达式

do

操作

done

只要 while 表达式成立，do 和 done 之间的操作就一直会进行。

until 循环

语法：until 表达式

do

操作

done

重复 do 和 done 之间的操作直到表达式成立为止。

例:

#test until

#add from 1 to 100

total=0

num=0

until test num ?Ceq 100

do

total=`expr $total + $num` //注意，这里的引号是反引号,下同

num=`expr $num+1`

done

echo “The result is $total”

执行结果如下：

[beichen@localhost bin]$until

The result is 5050!

五、条件语句

Shell 程序中的条件语句主要有 if 语句、case 语句；

If 语句

语法：if 表达式 1 then

操作

elif 表达式 2 then

操作

elif 表达式 3 then

操作

…..

else

操作

fi

Linux 里的 if 的结束标志是将 if 反过来写成 fi；而 elif 其实是 else if 的缩写

其中 elif 理论上可以有无限多个。

Case 语句

语法：case 字符串 in

值 1|值 2)

操作::

值 3|值 4)

操作::

值 5|值 6)

操作::

*}

操作::

esac

case 的作用就是当字符串与某个值相同是就执行那个值后面的操作。如果同一个操作对于多个值，则使用”|”将各个值分开。在 case 的每一个操作的最后面都有两个”::”，分号是必须的。

例：

case $USER in

beichen)

Echo “You are beichen!”;;

liangnian)

echo “You are liangnian”; //注意这里只有一个分号

echo “Welcome!”;; //这里才是两个分号

root)

echo “You are root!:echo Welcome!”;; //将两命令写在一行，用一个分号作为分隔符

*)

echo “Who are you?$USER?”;;

esac

执行结果：

[liangnian@localhost bin]$ test

You are liangnian

Welcome!

关于 shell 编程基础的东西就介绍这么多，如果你想更进一步了解 shell 编程的知识，请查阅相关书籍。