Shell 编程

李 明

无兄弟 不编程

E-mail/QQ: samlee@lampbrother.net

课程大纲

- Shell编程语法
- Shell脚本调试
- Shell应用实例

一个简单的shell程序

```
$ cat  example
#!/bin/sh
#This is to show what a example looks like.
echo "Our first example"
echo  # This inserts an empty line in output.
echo "We are currently in the following directory."
/bin/pwd
echo
echo "This directory contains the following files"
/bin/ls
```

一个简单的shell程序

shell结构：

1. #!指定执行脚本的shell
2. #注释行
3. 命令和控制结构

创建shell程序的步骤：

第一步：创建一个包含命令和控制结构的文件。

第二步：修改这个文件的权限使它可以执行。

使用chmod　u+x

第三步：执行　./example

（也可以使用“sh example”执行）

Shell变量

变量：是shell传递数据的一种方法，用来代表每个取值的符号名。

Shell有两类变量：临时变量和永久变量。

临时变量是shell程序内部定义的，其使用范围仅限于定义它的程序，对其它程序不可见。包括：用户自定义变量、位置变量。永久变量是环境变量，其值不随shell脚本的执行结束而消失。

用户自定义变量

用户定义的变量由字母或下划线开头，由字母、数字或下划线序列组成，并且大小写字母意义不同。变量名长度没有限制。

在使用变量值时，要在变量名前加上前缀“$”。

设置和使用变量

设置变量：习惯上用大写字母来命名变量。变量名只能以字母表中的字符开头，不能用数字。

变量赋值：赋值号“=”两边应没有空格。

定义时赋值，如NUM=1

将一个命令的执行结果赋给变量，如：TIME=`date`

将一个变量赋给另一个变量，如：A =$B

使用echo命令查看变量值。例如：echo $A

设置和使用变量

列出所有的变量：

```
# set
```

包含多个字的变量：

```
$NAME=Mike  Ron
```

运行时出错，应改为：

```
$NAME=“Mike Ron” 或 $NAME=‘Mike  Ron’
```

设置和使用变量

单引号和双引号的区别：

```
# $ABC=‘$NAME Junior’
# echo   $ABC
  $NAME  Junior
```

单引号之间的内容原封不动地指定给了变量。

删除变量：

```
# unset NAME
```

位置变量和特殊变量

Shell解释执行用户命令时，将命令行的第一个部分作为命令名，其它部分作为参数。由出现在命令行上的位置确定的参数称为位置参数。

例如：

ls -l file1 file2 file3

$0 这个程序的文件名 ls -l
$n 这个程序的第n个参数值，n=1-9

特殊变量

$* 这个程序的所有参数
$# 这个程序的参数个数
$$ 这个程序的PID
$! 执行上一个后台命令的PID
$? 执行上一个命令的返回值

Shell命令

read命令：从键盘读入数据，赋给变量

如：read USERNAME

read 命令

read 的例子：

```
#! /bin/sh
read first second third
echo "the first parameter is $first"
echo "the second parameter is $second”
echo "the third parameter is $third"
```

expr 命令

Shell变量的算术运算：

expr命令：对整数型变量进行算术运算

例如：expr 3 + 5

expr $var1 - 5

expr $var1 / $var2

expr $var3 * 10

复杂的expr命令

复杂的运算：

```
expr `expr 5 + 7`/$var4
```

将运算结果赋予变量：

```
var4=` expr  $var1 / $var2 `
```

expr 命令

```
#!/bin/sh
a=10
b=20
c=30
value1=`expr $a + $b + $c`
echo "The value of value1 is $value1"
value2=`expr $c / $b`
echo "The value of value2 is $value2"
value3=`expr $c \* $b`
echo "The value of value3 is $value3"
value4=`expr $a + $c / $b`
echo "The value of value4 is $value4"
```

变量测试语句

变量测试语句：用于测试变量是否相等、 是否为空、文件类型等。

格式：

test　测试条件

测试范围：整数、字符串、文件

变量测试语句

字符串测试：

test str1=str2	测试字符串是否相等
test str1!=str2	测试字符串是否不相等
test str1	测试字符串是否不为空
test -n str1	测试字符串是否不为空
test -z str1	测试字符串是否为空

变量测试语句

整数测试：

test int1 -eq int2	测试整数是否相等
test int1 -ge int2	测试int1是否>=int2
test int1 -gt int2	测试int1是否>int2
test int1 -le int2	测试int1是否<=int2
test int1 -lt int2	测试int1是否<int2
test int1 -ne int2	测试整数是否不相等

变量测试语句

文件测试：

test -d file	指定文件是否目录
test -f file	指定文件是否常规文件
test -x file	指定文件是否可执行
test -r file	指定文件是否可读
test -w file	指定文件是否可写
test -a file	指定文件是否存在
test -s file	文件的大小是否非0

变量测试语句

变量测试语句一般不单独使用，一般做为if语句的测试条件，如：

```
if   test  -d  $1  then
     …
fi
```

变量测试语句可用[]进行简化，如

test -d $1 等价于 [-d $1]

变量测试语句

```
#!/bin/sh
if [ $# -ne 2 ]; then
   echo "Not enough parameters"
   exit 0
fi
if [ $1 -eq $2 ]; then
   echo "$1 equals $2"
elif [ $1 -lt $2 ]; then
   echo "$1 littler than  $2"
elif [ $1 -gt $2 ]; then
   echo "$1 greater than  $2"
fi
```

流控制语句

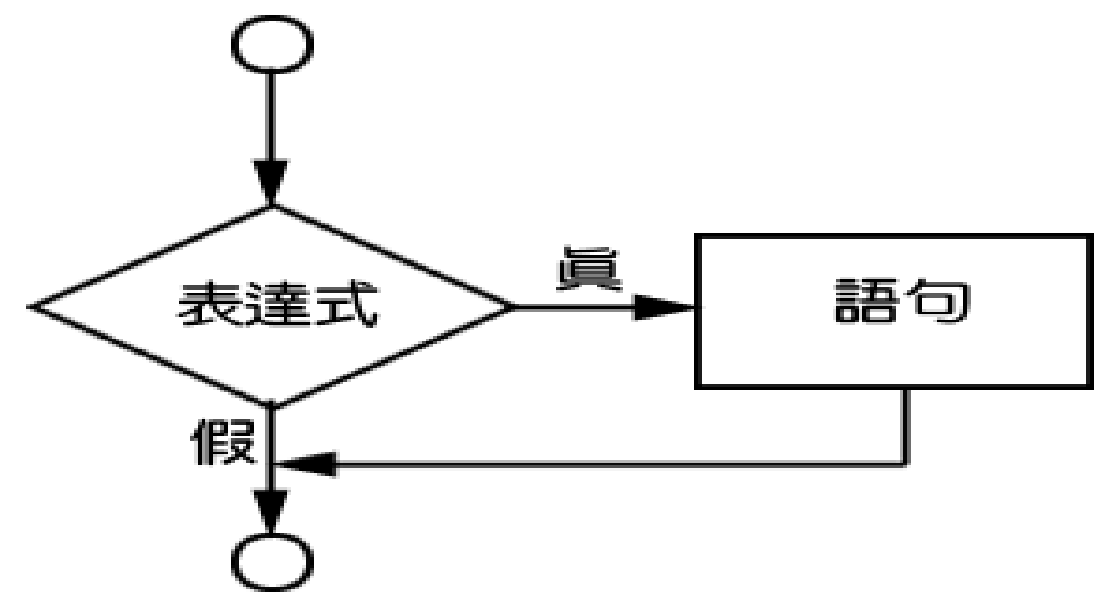
流控制语句：用于控制shell程序的流程

exit语句：退出程序执行，并返回一个返回码，返回码为0表示正常退出，非0表示非正常退出。

例如：exit 0

流控制语句

if语句的流程图



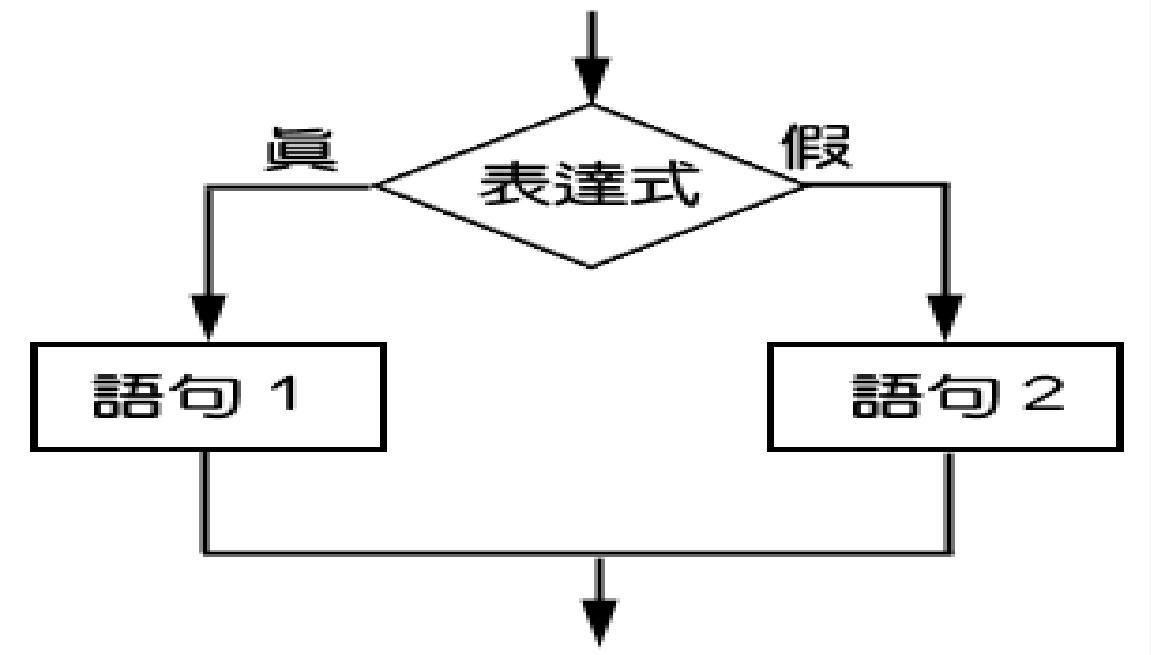
流控制语句

if …then …fi语句，例如：

```
#!/bin/sh
if  [ -x  /etc/rc.d/init.d/httpd ]
  then
  /etc/rc.d/init.d/httpd restart
fi
```

流控制语句

if/else语句的流程图



流控制语句

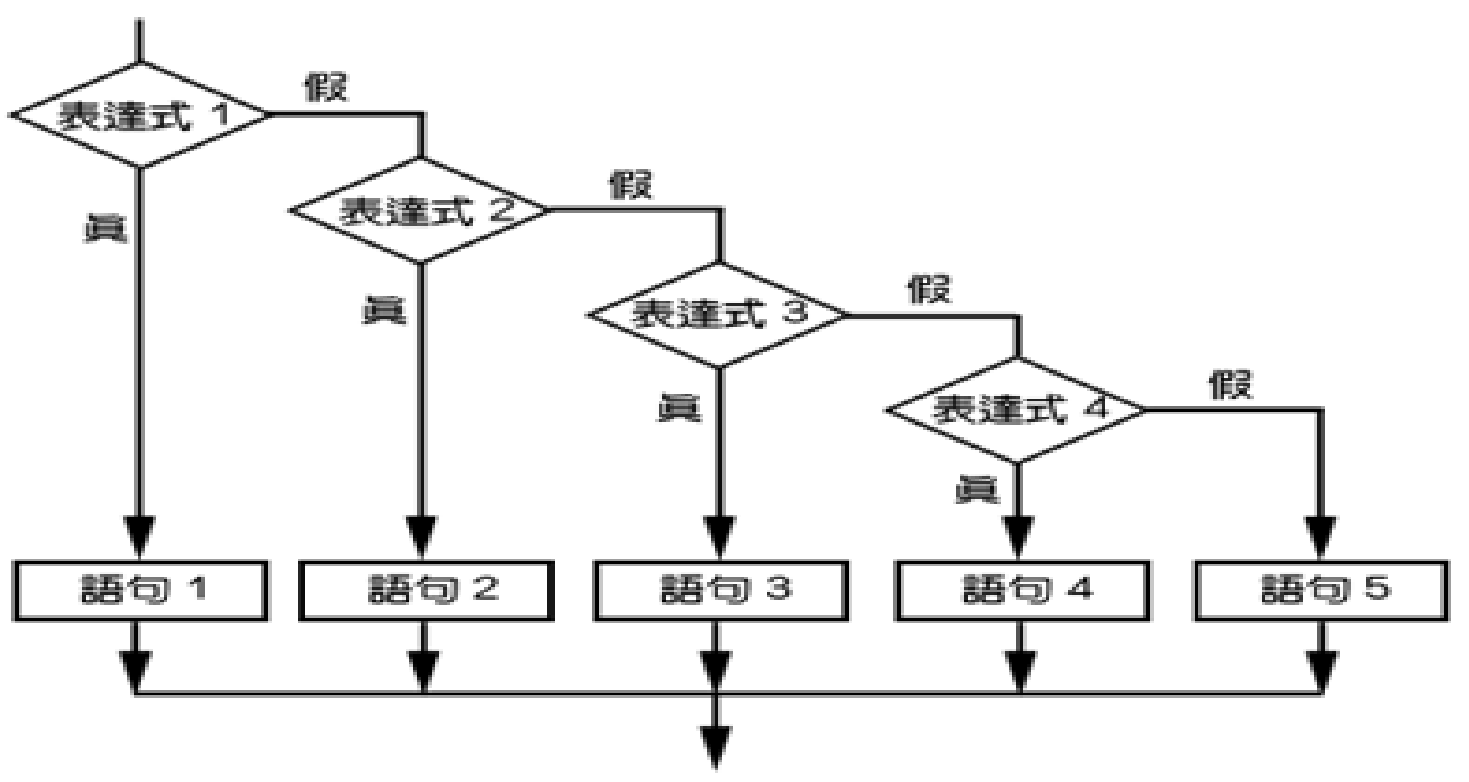
更复杂的if语句：

```
if   条件1   then
    命令1
elif   条件2   then
     命令2
else
    命令3
fi
```

流控制语句

:

if/else嵌套的流程图



流控制语句

多个条件的联合：

-a：逻辑与，仅当两个条件都成立时，结果为真。

-o：逻辑或，两个条件只要有一个成立，结果为真。

流控制语句

```
echo "please input a file name:"
read file_name
if [ -d $file_name ]
  then
  echo "$file_name is a directory"
elif [ -f $file_name ]
  then
  echo "$file_name is a common file"
elif [ -c $file_name -o -b $file_name ]
  then
  echo "$file_name is a device file“
else
  echo "$file_name is an unknown file"
fi
```

流控制语句

for…done语句

格式：for 变量 in 名字表

do

命令列表

done

流控制语句

例子：

```
#!/bin/sh
for DAY in Sunday Monday Tuesday Wednesday
    Thursday Friday Saturday
do
  echo "The day is : $DAY"
done
```

流控制语句

select 变量 in 关键字
do
- command 1
  - ✓... ...
- command n

done

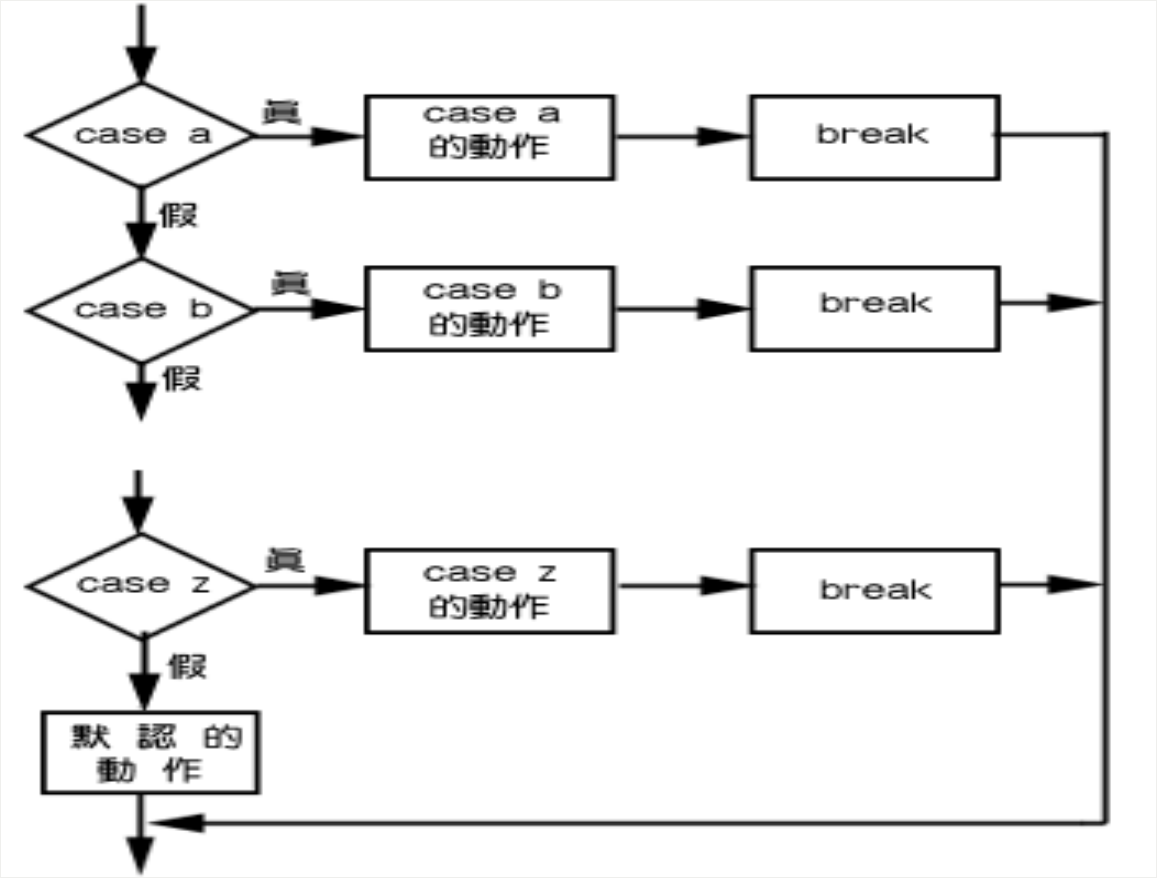
select把关键字中的每一项做成类似表单，以交互的方式执行do和done之间的命令。

流控制语句

case…esac语句，格式：

```
case 变量 in
        字符串1)    命令列表1
                    ;;
            ...
        字符串n)    命令列表n
                    ;;
esac
```

流控制语句


case a
真
case a
的動作
break
假
case b
真
case b
的動作
break
假
case z
真
case z
的動作
break
假
默 認 的
動 作

流控制语句

while语句，格式：

```
while  条件
do
    命令
done
```

流控制语句

```
#! /bin/sh
num=1
while [ $num -le 10 ]
do
    SUM=`expr $num \* $num`
    echo $SUM
    num=`expr $num + 1`
done
```

流控制语句

until语句，格式：

```
until  条件
do
    命令
done
```

until类似while循环，不同的是until是条件返回值为假时才继续执行。

流控制语句

跳出循环：break和continue

break：跳出整个循环

continue：跳过本次循环，进行下次循环

流控制语句

shift指令：参数左移，每执行一次，参数序列顺次左移一个位置，$#的值减1，

用于分别处理每个参数，移出去的参数不再可用

流控制语句

```
#! /bin/sh
if  [ $# -le 0 ]
then
 echo "Not enough parameters"
 exit 0
fi
sum=0
while [ $# -gt 0 ]
do
  sum=`expr $sum + $1`
  shift
done
echo $sum
```

函数应用

函数的定义：

```
函数名 ()
{
  命令序列
}
```

函数的调用：不带()

函数名　参数1　参数2 …

函数应用

函数中的变量：

变量均为全局变量，没有局部变量

函数中的参数：调用函数时，可以传递参数，在函数中用$1、$2…来引用

Shell 脚本调试

sh -x script

这将执行该脚本并显示所有变量的值。

sh -n script

不执行脚本只是检查语法的模式，将返回所有语法错误。

awk命令应用

awk -F 域分隔符 ‘命令’

示例：

1、检测系统中UID为0的用户

awk -F: '$3==0 {print $1}' /etc/passwd

2、检测系统中密码为空的用户

awk -F: 'length($2)==0 {print $1}' /etc/shadow

知识点总结

- 掌握Shell编程的基本语法
- 掌握结合系统命令编写应用脚本
- 掌握Shell编程调试命令


谢谢
不抛弃
不放弃……
无兄弟
不编程
LAMP兄弟连
www.lampbrother.net