# Shell

## 1.shell变量

### 1).变量名的命名须遵循如下规则：

* 首个字符必须为字母（a-z，A-Z）。
* 中间不能有空格，可以使用下划线（\_）。
* 不能使用标点符号。
* 不能使用bash里的关键字（可用help命令查看保留关键字）。

### 2).循环打印

for skill in Ada Coffe Action Java; do

echo "I am good at ${skill}Script"

done

${} 使用已经定义过的变量

### 3).设置只读变量（readonly）

url=”http://www.baudu.com”

readonly url

### 4).删除变量(unset)

unset varilable\_name

### 5).shell变量类型

### 6).变量类型

运行shell时，会同时存在三种变量：

* 1) 局部变量 局部变量在脚本或命令中定义，仅在当前shell实例中有效，其他shell启动的程序不能访问局部变量。
* 2) 环境变量 所有的程序，包括shell启动的程序，都能访问环境变量，有些程序需要环境变量来保证其正常运行。必要的时候shell脚本也可以定义环境变量。
* 3) shell变量 shell变量是由shell程序设置的特殊变量。shell变量中有一部分是环境变量，有一部分是局部变量，这些变量保证了shell的正常运行

### 7).shell的字符串类型

单引号字符串的限制：

单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；

单引号字串中不能出现单引号（对单引号使用转义符后也不行）。

双引号的优点：

双引号里可以有变量

双引号里可以出现转义字符

### 8).字符串拼接

your\_name="qinjx"

greeting="hello, "$your\_name" !"

greeting\_1="hello, ${your\_name} !"

echo $greeting $greeting\_1

### 9).字符串的长度

string="abcd"

echo ${#string} #输出 4

### 10).给写好的shell命令加可执行权限

chomd +X helloword.sh

### 11).set查看系统中的所有环境变量

set | grep PWD 查看当前目录

### 12.export A 使子进程使用父进程定义的变量

#可以把变量提升为当前进程的全局环境变量,供其他子shell程序使用

### 13.代替export 使子进程调用父进程定义好的变量

* . ./b.sh #注意重点关注最前面的那个点
* source ./b.sh

可以是b.sh在他的父进程中运行

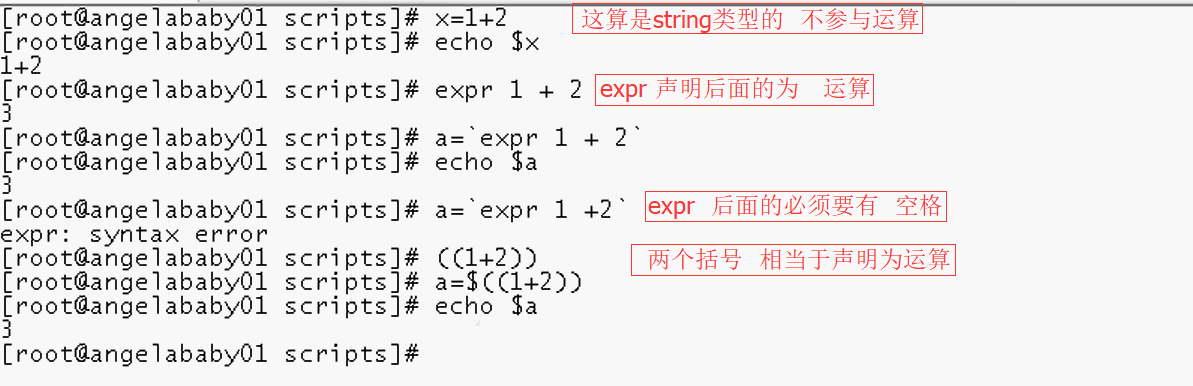
## 2.算数运算

### 1.里面要用反引号

#### 1.用expr

#### 2.用(())

#### 3.用$[]



### 2.流程控制语句

#### 1.语法

if [ condition (必须有空格)] if [ 条件 ]

then then

statements 要执行的

elif [ condition ] elif [ 条件 ]

then then

statement 要执行的

else else

statements 要执行的

fi fi

#### 2.演示

|  |
| --- |
| #!/bin/bash  read -p "请输入用户名:" NAME  if [ $NAME = root ]  then  echo "hello ${NAME} welcome!"  elif [ $NAME = cgb ]  then  echo "hello ${NAME} welcome!"  else  echo "用户名不正确"  fi |

##### 注意:

1）if 语句后面需要跟着then，同时前面要有分号；

2) 空格非常重要，shell 会认为空格前的为一个命令，如果a=3 认为是赋值操作，如果写成a = 3，那么就会认为a为一个命令 this=`ls -l |grep '^-' |  wc -l `

3) 操作符之间要用空格分开 ，如 test ! -d $1，其中的！和-d就要用空格分开

**空格是命令解析中的重要分隔符**

### 3.条件判断组合

注:[] 与 [[]]的区别:[[]]中逻辑组合可以使用 && || 符号

而[]里面逻辑组合可以用-a -o代替

|  |
| --- |
| if [[ a = b && b = c ]];then echo ok ;else echo notok; fi [[]]  if [ a = b -a b = c ];then echo ok ;else echo notok; fi 并且 &&  if [ a = b -o b = b ]; then echo ok ;else echo notok; fi 或者 || |

### 4.判断字符

#### 字符串比较:

* + 相等 =
  + 不等 !=
  + 字符串长度为0返回true -z
  + 字符串长度不为0返回true -n

if [ -n “aa” ];then echo ok;else echo notok;fi

if [ -z “” ];then echo ok;else echo notok;fi

#### 整数比较

* + 小于 -lt
  + 小于等于 -le
  + 等于 -eq
  + 大于 -gt
  + 大于等于 -ge
  + 不等于 -ne

#### 文件判断

* 是否为目录 -d
* 是否为文件 -f
* 是否存在 -e

例子: if [ -d /bin ]; then echo ok;else echo notok;fi

# $? 上一个命令执行的返回码

## 3.函数定义和调用

### 1.函数的定义

|  |
| --- |
| #!/bin/bash  hello() ##定义函数  {  echo "Hello there today's date is `date +%Y-%m-%d`"  return 2 ###返回值其实是状态码,只能在[0~255]范围内  }  #echo "now going to the function hello"  hello ###调用函数  echo $? ###获取函数的返回值 |

#### 注意:

1.必须在调用函数地方之前,先声明函数,shell脚本是逐行运行的,不会像其他语言一样先预编译.

2.函数返回值,只能通过$?系统变量获得,可显示加:return返回,如果不加,将以最后一条命令运行结果,作为返回值.return后跟的数值n(0~255)

### 2.函数的参数

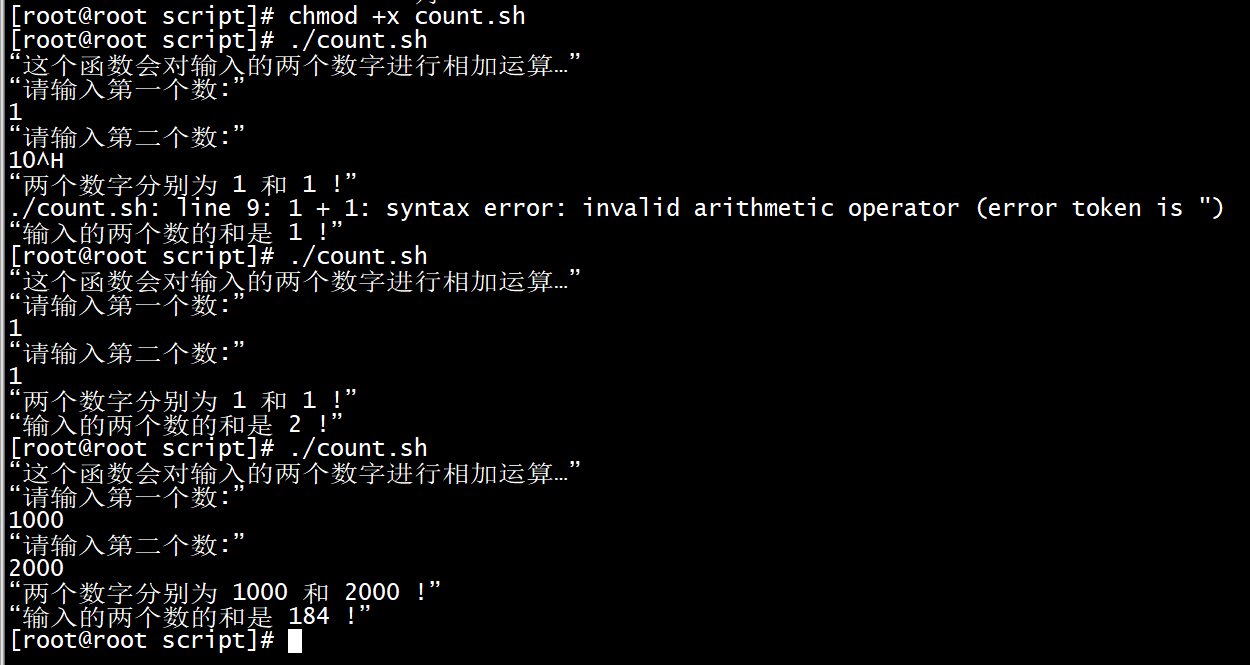
|  |
| --- |
| funTest(){  echo “第一个参数为 $1 !”  echo “第二个参数为 $2 !”  echo “第十个参数为 $10 !”  echo “第十个参数为 ${10} !”  echo “第十一个参数为 ${11} !”  echo “参数总数有 $# 个!”  echo “作为一个字符串输出所有参数 $\* !”  } |

#### 注意:

$10 不能获取第10个参数 ,获取第十个参数时需要${10}. 当n>=10时需要使用${n}来获取参数.

### 3.函数间的调用

|  |
| --- |
| #!/bin/bash  # fun2.sh  funWithReturn(){  echo “这个函数会对输入的两个数字进行相加运算…”  echo “请输入第一个数:”  read Num1  echo “请输入第二个数:”  read Num2  echo “两个数字分别为 $Num1 和 $Num2 !”  return $(( $Num1 + $Num2))  }  funWithReturn  echo “输入的两个数的和是 $? !” |



#### 注意:

经过return返回的值不会超过255

#### 调用演示

|  |
| --- |
| #!/bin/bash  #func  **. /data/script/count.sh ### 注: . 和 /之间有空格**  **#或者 source /data/script/count.sh**  funWithReturn  echo $? |

