# linux 基础

## shell 编程

shell编程是脚本编程

编译器 或者 解释器 取决与语言

编程语言： 机器语言，汇编语言，高级语言（越接近人类语言，人就能轻松的掌握）

但是要机器能执行，必须由 ‘翻译官’ 翻译称机器语言，计算机才能执行。

语言分类：

静态语言：编译性语言 C , C++，C#,java(两者兼具,先解释为字节码，字节码进行编译)

都是强类型（变量在使用前必须事先声明，甚至还要初始化）

关键字：翻译称机器指令 ，需要事先全部编译称二进制才能执行。

动态语言：解释性语言 PHP, shell, **python** ,perl，lua

通常都是弱类型语言（变量使用时声明，甚至还不区分类型）

不需要编译，边解释边执行 on the fly

动态语言需要一个解释器才能执行。

**bash 就是一个解释器，通常解释器是由动态语言开发的。**

面向过程：linux内核就是使用C面向过程开发的

面向对象：

变量： 内存空间

变量名： 系统内部映射为内存空间的地址值。 16进制

变量类型：字符型（char），数值型（整型 int，浮点型 float,double(双精度)）布尔

**Linux 系统中 0 为真，其他全是假**

## bash变量类型

### 环境变量

作用域为当前shell进程或者子shell进程有效

export varname=value

or:

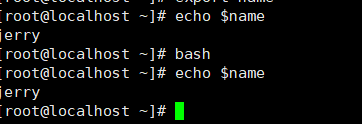
varname=value

export varname

命令行启动的脚本会继承当前shell的环境变量。

因为脚本在启动的时候，会在当前shell进程启动一个子shell进程。

系统自动执行的脚本（跟命令行无关，非命令行启动）就需要自我定义需要的各种环境变量。



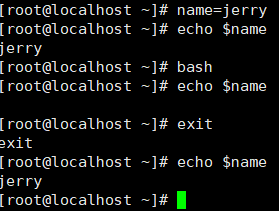
### 本地变量

**或者局部变量**

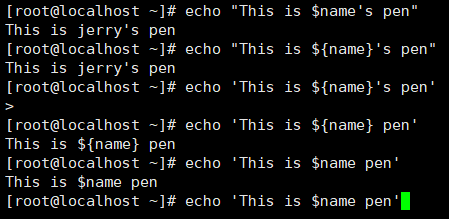
**作用域为当前shell进程**

变量一定是进程的变量，随时声明，随时用。

bash 打开一个子shell，变量name就不能使用了。exit到当前shell，又可以使用了。



引用变量： ${varname} ,大多数的时候{ }可以省略



“” 强引用

‘’ 弱引用

varname = value 作用域整个bash进程有效

local varname=value 作用域当前代码段有效

### 位置变量

**相当于形参**

**$1,$2,$3…$n**

$1 第一个参数

$2 第二个参数

……

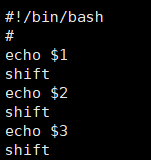
cat /etc/inittab /etc/passwd /etc/fstab

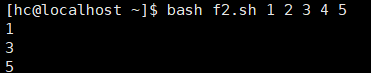
$1 代表 /etc/inittabl

$2 代表 /etc/passwd

$3 代表 /etc/fstab

**shift 轮替**

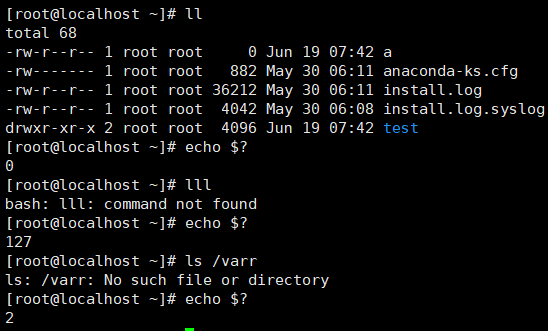




### 特殊变量

（bash内置）或者**系统变量**

**$?** 上一个命令的执行状态返回值，



程序执行可能有两类返回值

程序执行结果

程序状态返回代码（0-255）

0 ： 正确执行

1-255：错误的执行 1，2，127 系统预留，其他可以自定义

输出重定向

>

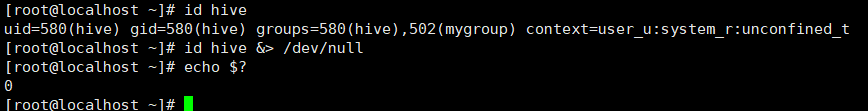
>>

2>

2>>

&>

/dev/null; 设备，软件模拟的设备，软设备，bit bucket, 数据黑洞，进去就消失了



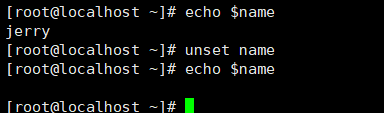
**$# 参数的个数**

**$\* 参数列表**

**$@ 参数列表**

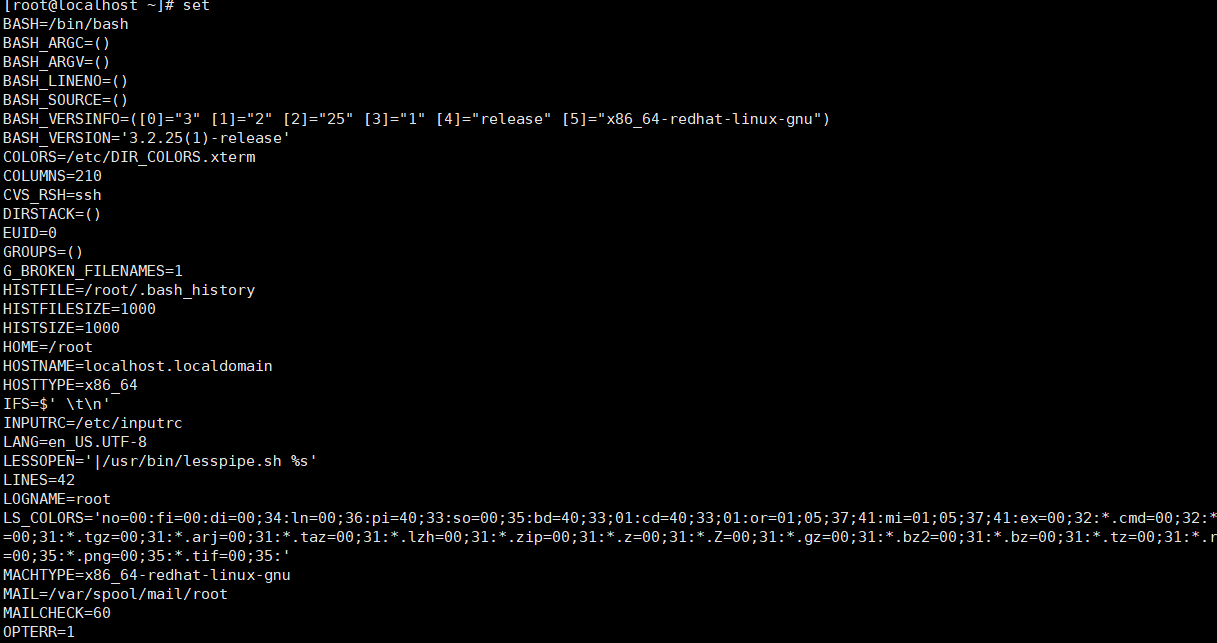
### 撤销变量

**unset varname**



### set

查看当前shell进程中的所有变量

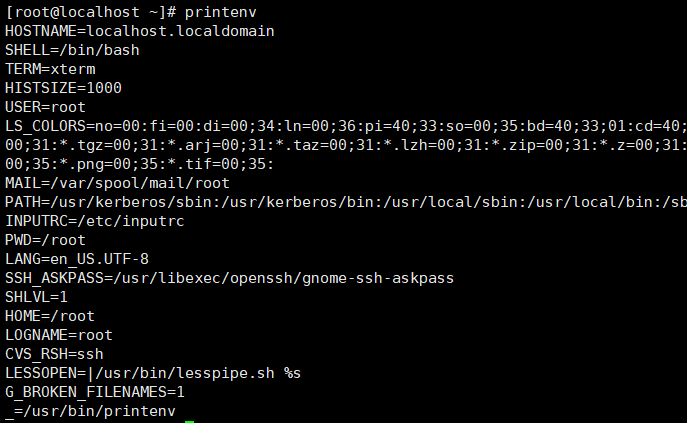
查看所有环境变量

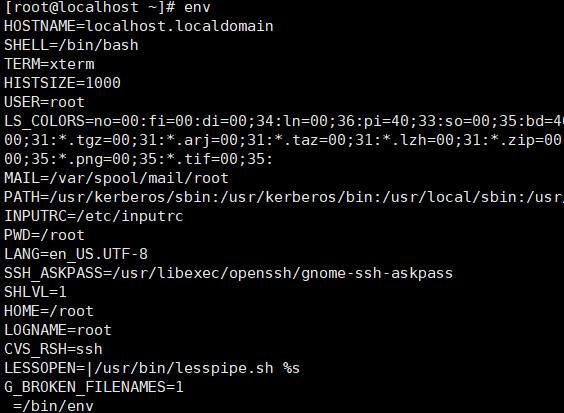
**printenv**

**env**

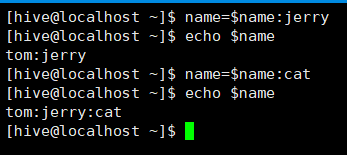
**export**





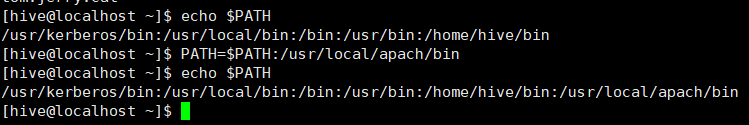


name=$name:jerry



添加路径

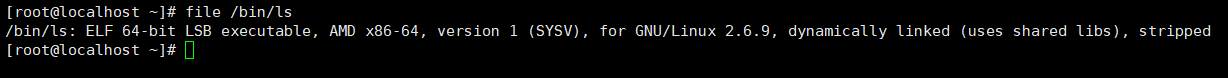
PATH=$PATH:/usr/local/apach/bin



shell 默认的变量是字符串，是不能做算数运算。

## 脚本

**就是命令的堆砌，按实际需要，结合命令流程控制机制实现的源程序。**



linux 内核能够执行的是ELF格式的文件。



ASCII text的文件是不能执行的，所以我们需要启动一个脚本解释器去解释这个文件。

## 条件判断

bash中如何实现条件判断？

**条件测试类型**

**整数测试**

**-eq -lt -le -ge -gt -ne**

**字符（串）测试**

**== ！= > < -n（空为真） -z (不空为真)**

**文件测试**

**-e -f -d -r -w -x**

**组合条件测试**

**-a : 与关系**

**-o : 或关系**

**! : 非关系**

**条件测试的表达式**

**[ expression ]**

**[[ expression ]]**

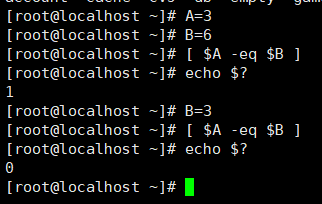
**test expression**

### 整数比较

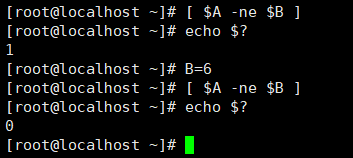
**[ ] 表达式前后必须有空格。**

**-eq** : 比较两个整数是否相等 相等为真，不等为假

[ $A -eq $B ]



**-ne**: 比较两个整数是否相等 相等为假，不等为真



**-gt** : 测试一个数是否大于一个数，大于为真，否则为假

**-lt** : 测试一个数是否小于一个数，小于为真，否则为假

**-ge** : 大于或等于

**-le** : 小于或等于

### 与或非

**命令间的逻辑关系**

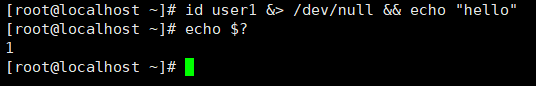
**逻辑与 &&**

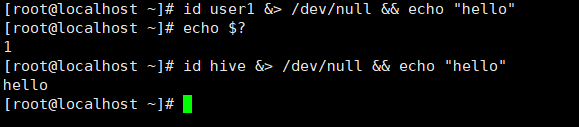
**第一个条件如果为假，结果已有，第二个条件就不用在判断 短路**

**第一个条件如果为真，第二个条件必须得判断。**

**逻辑或 ||**

**逻辑非 ！**





需求： 如果用户存在，不添加，用户不存在，添加。

用户存在为真，取反就为假，后面就不执行了，不添加用户

如果用不存在为假，取反为真，后面执行，添加用户

**! id user1 && useradd user1**

如果用户存在为真，后面就不执行，不添加。

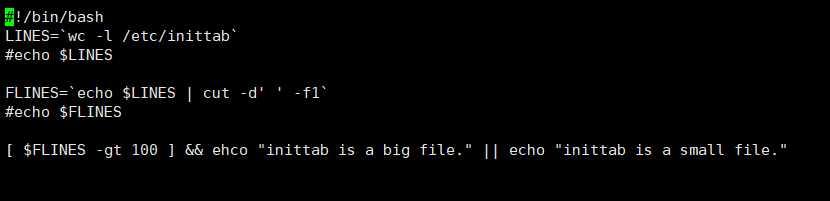
如果用户不存在为假，后面还要执行，添加用户

**id user1 || useradd user1**

**变量名称命名规则**

1. **只能包含字母，数字，下划线**
2. **不能数字开头**
3. **不能与环境变量的重名。**
4. **最好见名知意**

**在脚本中 变量名=值 不能写成 变量名 = 值 不能有空格**



**#!/bin/bash**

**LINES=`wc -l /etc/inittab`**

**#echo $LINES**

**FLINES=`echo $LINES | cut -d' ' -f1`**

**#echo $FLINES**

**[ $FLINES -gt 100 ] && ehco "inittab is a big file." || echo "inittab is a small file."**

如果用户存在提示用户已经存在，否则添加此用户

id user1 && echo “user1 exists” || useradd user1

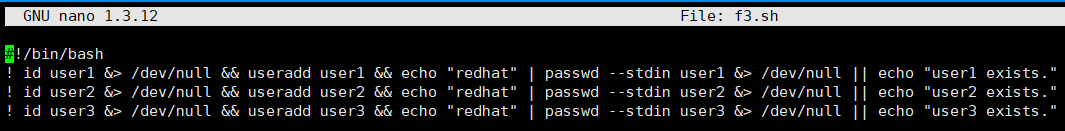
如果用户不存在，就添加，否则显示此用户已经存在

! id user1 && useradd user1 || echo “user1 exists”

如果用户不存在，添加并且设定密码，否则显示用户已经存在。

! id user1 && useradd user1 && echo “user1” | passwd --stdin user1 || echo “user1 exists.”

创建3个用户，并设置密码，如果存在就不用创建，并提示用户已经存在



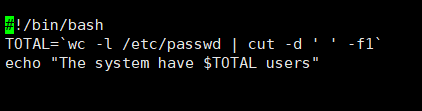
#!/bin/bash

! id user1 &> /dev/null && useradd user1 && echo "redhat" | passwd --stdin user1 &> /dev/null || echo "user1 exists."

! id user2 &> /dev/null && useradd user2 && echo "redhat" | passwd --stdin user2 &> /dev/null || echo "user2 exists."

! id user3 &> /dev/null && useradd user3 && echo "redhat" | passwd --stdin user3 &> /dev/null || echo "user3 exists."

显示系统一共有多少个用户



根据UID判断是普通用户还是管理员

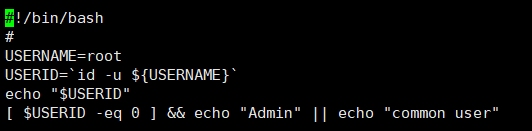
#!/bin/bash

#

USERNAME = user1

USERID = `id -u ${ USERNAME }`

[ $USERID -eq 0 ] && echo "Admin" || echo "普通用户"



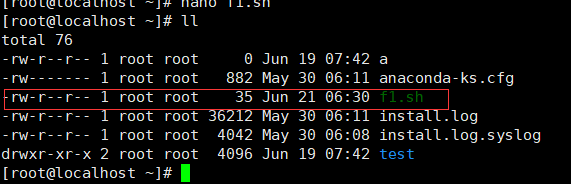
### #!/bin/bash

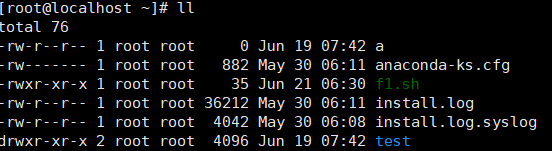
shebang 魔术

（/bin/bash 就是解释器,一看见 #! 就知道是个bash脚本，马上启动/bin/bash进程，来执行脚本）

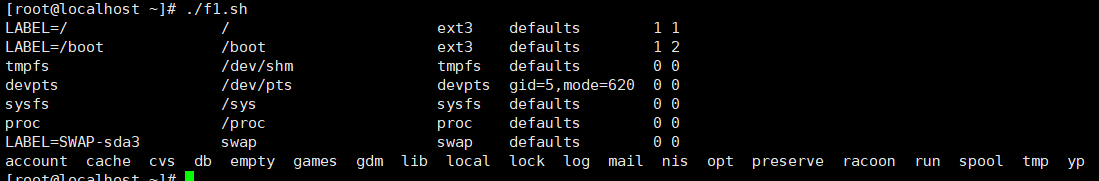


没有执行权限

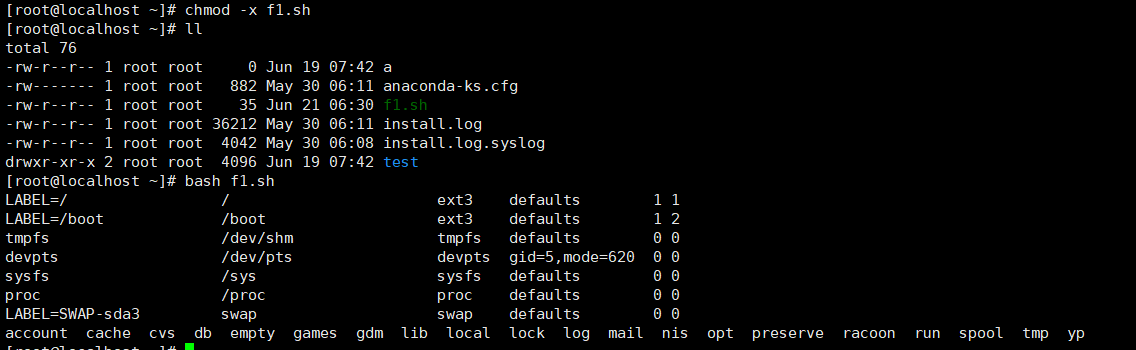


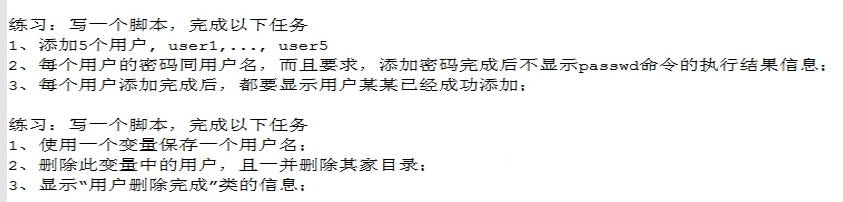


./f1.sh



不加执行权限，使用bash脚本解释器也可以执行。是作为别的命令的参数去执行的，而不是自己作为脚本去执行的。如果要独立执行，不行给执行权限。





useradd user1

echo “user1” |passwd --stdin user1 &> /dev/null

echo “add user1 successfully”

### if fi

if 判断条件；then

………

fi

**#!/bin/bash**

**#**

**NAME=user1f**

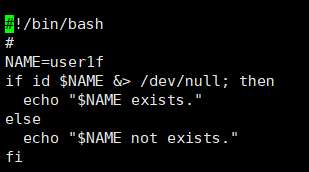
**if id $NAME &> /dev/null; then**

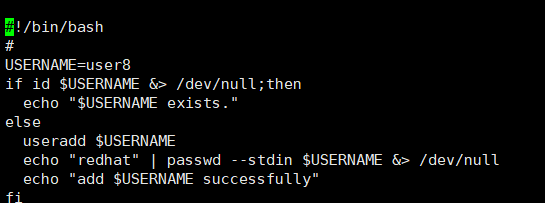
**echo "$NAME exists."**

**else**

**echo "$NAME not exists."**

**fi**





**执行命令的返回结果**

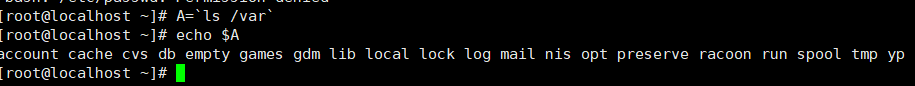
if [ `id -u $USERNAME` -eq 0 ] ;then

echo "admin user."

else

echo "common user."

fi



**执行命令的状态返回结果**

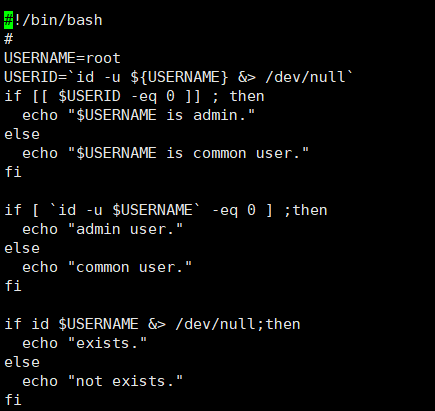
if id $USERNAME &> /dev/null;then

echo "exists."

else

echo "not exists."

fi



统计系统一共有多少个用户使用bash

#!/bin/bash

#

TOTAL=`grep "\<bash$" /etc/passwd | wc -l`

grep "\<bash$" /etc/passwd | wc -l &> /dev/null

RETST=$?

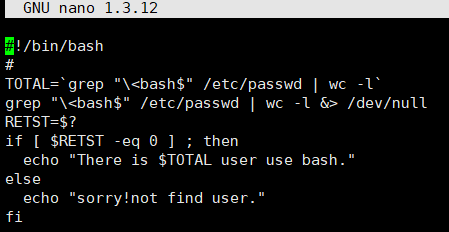
if [ $RETST -eq 0 ] ; then

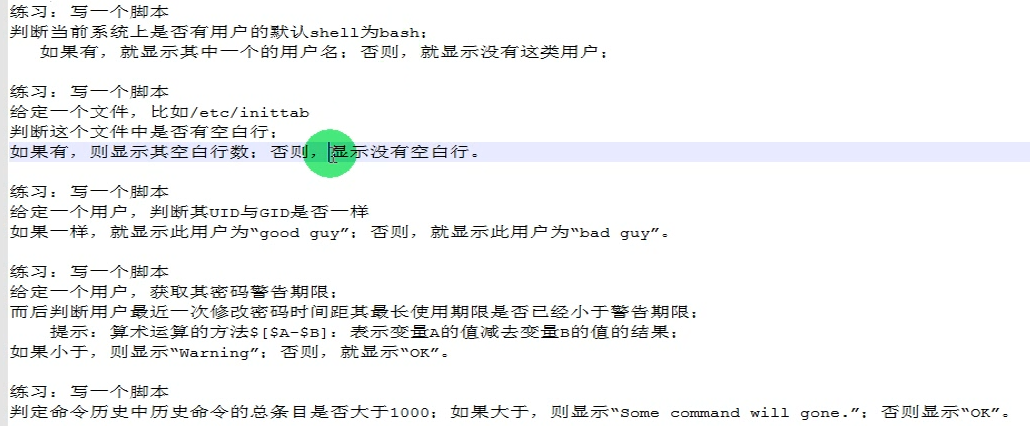
echo "There is $TOTAL user use bash."

else

echo "sorry!not find user."

fi





#!/bin/bash

#

TOTAL=`grep '^$' /etc/inittab | wc -l`

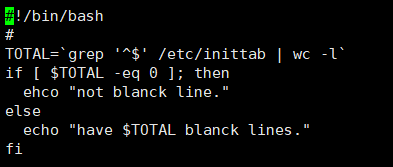
if [ $TOTAL -eq 0 ]; then

ehco "not blanck line."

else

echo "have $TOTAL blanck lines."

fi





#!/bin/bash

#

USERNAME=user99

if ! grep "^$USERNAME" /etc/passwd &> /dev/null ;then

echo "$USERNAME not found in your system!"

exit 1

fi

USERID=`id -u $USERNAME`

GROID=`id -g $USERNAME`

if [ $USERID -eq $GROID ] ;then

echo "good guy."

else

echo "bad guy."

fi

USERID=`grep "^${USERNAME}" /etc/passwd | cut -d : -f 3`

GROID=`grep "^${USERNAME}" /etc/passwd | cut -d : -f 4`

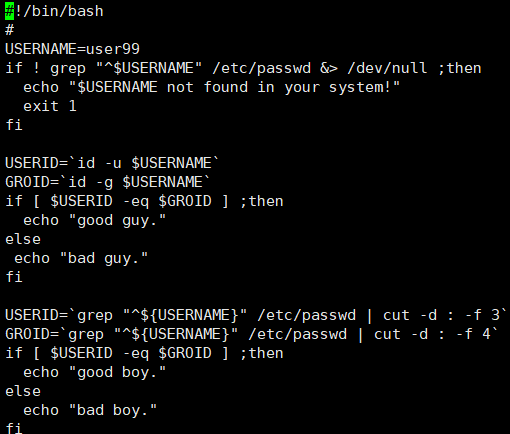
if [ $USERID -eq $GROID ] ;then

echo "good boy."

else

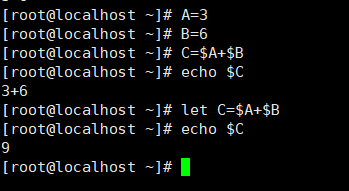
echo "bad boy."

fi

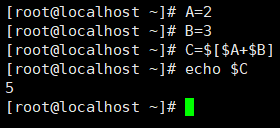


### 算数运算

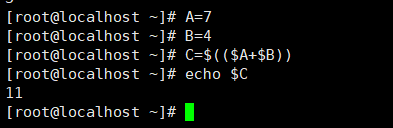
**let**



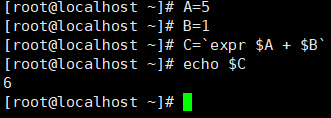
**$[ ]**



**$(( ))**



**expr** 算术运算表达式 ，表达式中的操作数和运算符要有空格，而且要使用命令引用



**exit**

退出当前进程，

脚本中退出当前脚本进程

如果脚本没有明确定义退出状态码，那么，最后执行的一条命令的退出码即为脚本的退出码

### 文件测试

文件类型文件：普通文件，目录，设备文件，链接文件，特殊文件，管道文件，套接字文件。

**-e FILE**  测试文件是否存在

**-f FILE**  测试文件是否是普通文件

**-d FILE** 测试文件是否是目录

**-r FILE** 测试当前用户对当前文件是否有读权限

**-w FILE** 测试当前用户对当前文件是否有写权限

**-x FILE** 测试当前用户对当前文件是否有执行权限

[ -e /etc/passwd ] 测试passwd文件是否存在

[ -f /etc/passwd ] 测试passwd文件是否是普通文件

……

#!/bin/bash

#

FILE=/etc/passwd

if [ ! -e $FILE ] ; then

echo “$FILE not exists.”

exit 1

fi

NAME=`grep “^$” $FILE &> /dev/null`

**多分支判断**

if 条件1；then

statement1

……

elif 条件2；then

statement2

……

elif 条件3 ; then

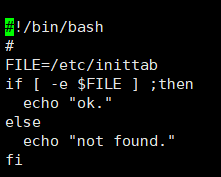
statement3

……

else

statement4

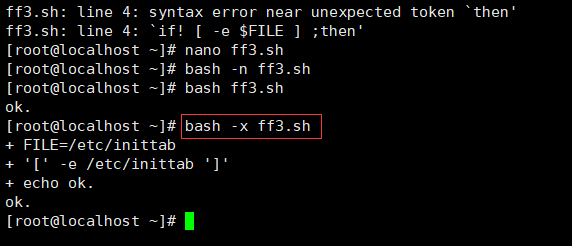
fi



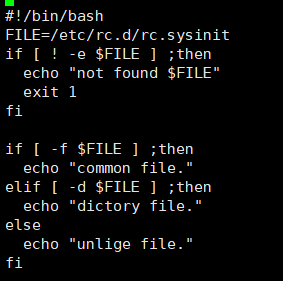
测试脚本是否有语法错误

**bash -n 脚本名称 有时候错误定位不是那么准确**

**bash -x 脚本名称 详细执行的步骤**



给定一个文件，如果文件是一个目录，就显示目录，如果是一个普通文件，就显示这是一个普通文件，否则显示无法识别



### 参数

**$1 $2 …$n**

第一个参数 第二个参数 第三个参数

**$#**

参数个数

**$\***

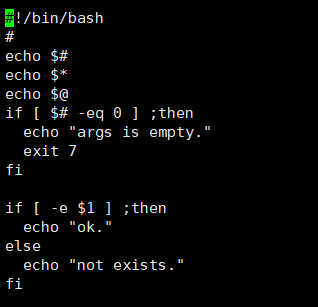
全部的参数

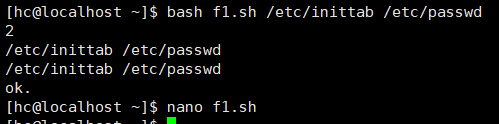
**S@**

全部的参数

**$? 成功返回0 否则返回1**

写一个脚本，能接受一个参数（文件路径），如果这个文件存在，提示 OK，否则提示 文件不存在。





写一个脚本，给两个参数，显示来个参数的和和乘积

