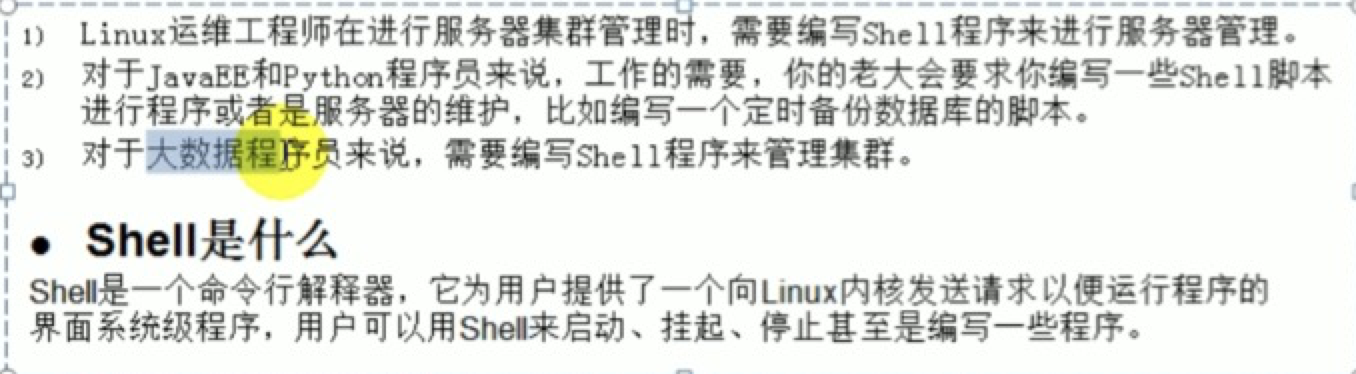
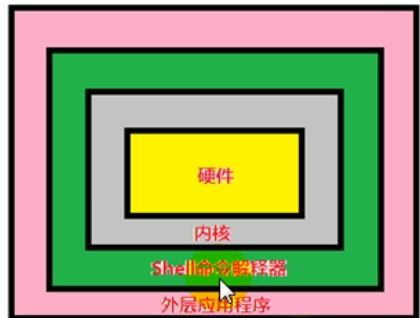
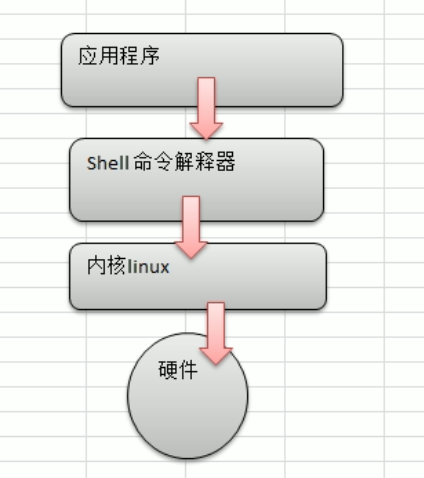
Shell教程

1. **基础知识**





1. **快速入门**

创建一个shell脚本，输出hello

vim hello.sh,内容如下：

#!/bin/bash

echo “hello world”,保存

修改hello.sh的权限为可执行的：chmod 744 hello.sh

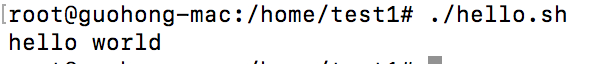
执行hello.sh

（1）./hello.sh （相对路径执行，需要权限）

（2）sh hello.sh或者bash hello.sh（不给可执行权限也能执行，不推荐使用）

（3）/home/test1/hello.sh (绝对路径执行)

（4）source hello.sh (没有权限也能执行)



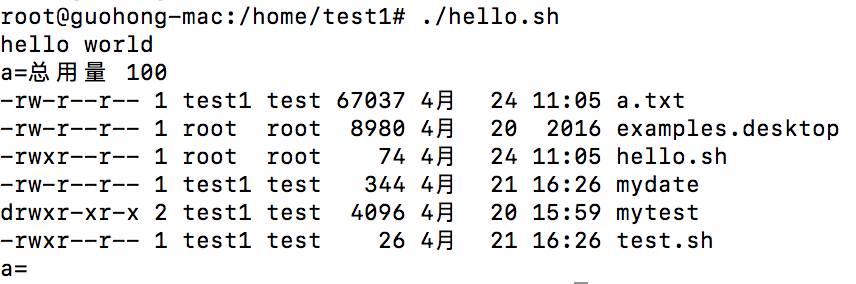
1. **变量**

变量定义的规则：

1. 变量名可以由字符、数字、和下划线组成，但是不能以数字开头
2. 等号两侧不能有空格
3. 变量名称一般习惯大写

将Linux命令的返回值赋给变量：

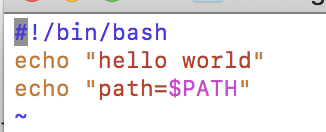
A=`ls –la`（反引号，在按键1旁边）或者A=$(ls -la)



linux shell中的变量分为系统变量和用户自定义变量

系统变量：$HOME、$PWD、$SHELL、$USER等

echo “path=$PATH”:$PATH是系统变量

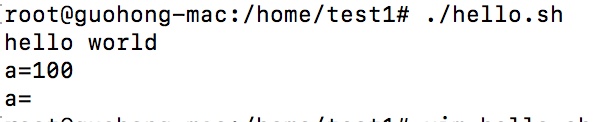


效果：



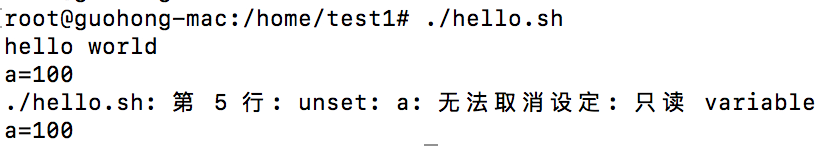
自定义变量：变量=值

撤销变量：unset 变量

声明静态变量：readonly变量，注意：此变量不能unset



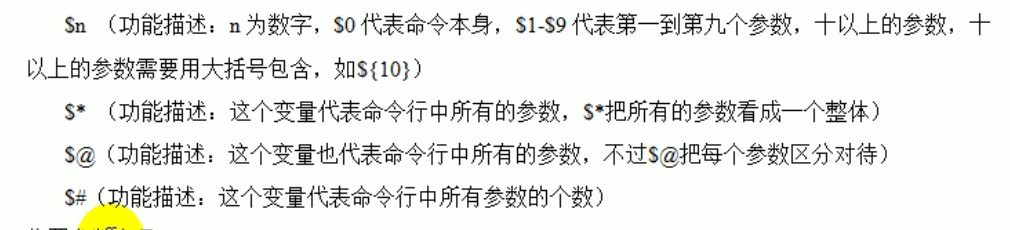


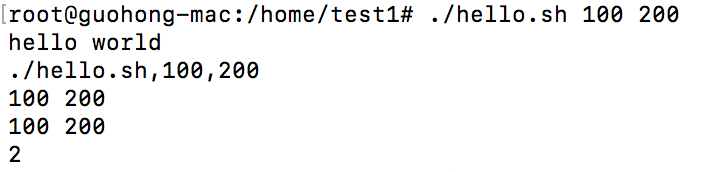
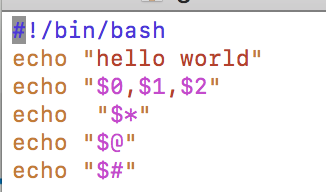
**位置参数变量：**

当我们执行一个shell脚本时，如果希望获取到命令行的参数信息，就可以使用到位置参数变量。

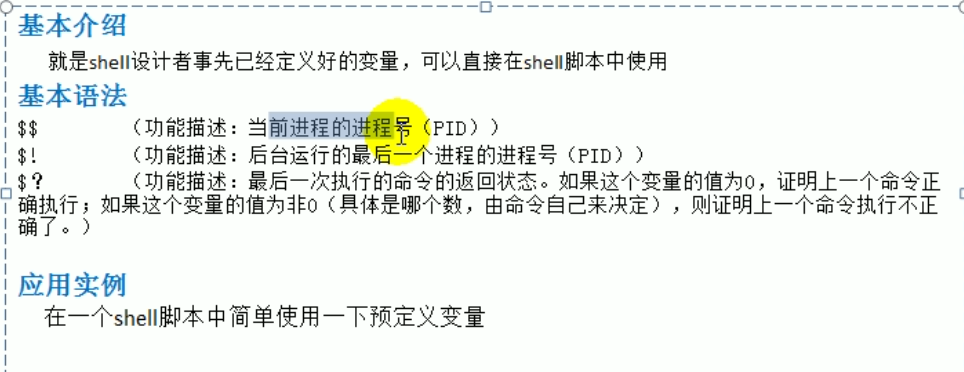
比如：/hello.sh 100 200，这个就是一个执行shell的命令行，可以在hello.sh脚本中获取到参数信息。

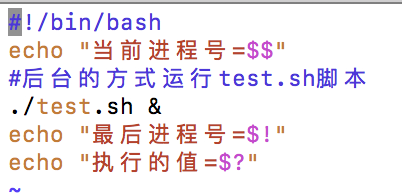
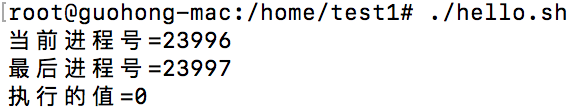
基本语法：





**预定义变量：**



1. **运算符**

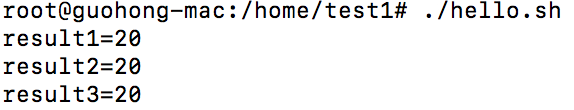
基本语法：

（1）”$((运算式))”或$[运算式]

（2）expr m + n,注意expr运算符间要有空格

（3）expr m-n

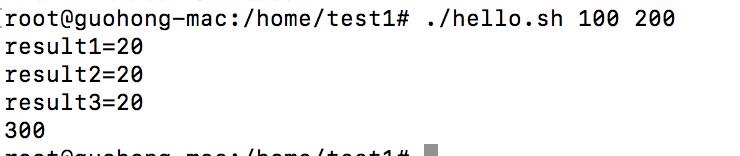
（4）expr \\*,/,% 乘，除，取余

求两个参数之和：

RESULT4=$[$1+$2]

echo “$RESULT4”



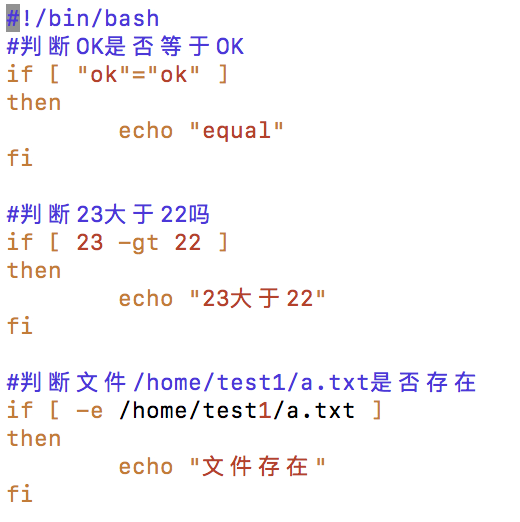
1. **判断语句**

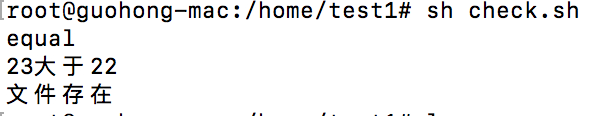
基本语法：[condition]注意condition前后要有空格

#非空返回true，可使用$?验证（0为true，>1为false）



案例：





if 条件

then

---

fi

if 条件

then

--

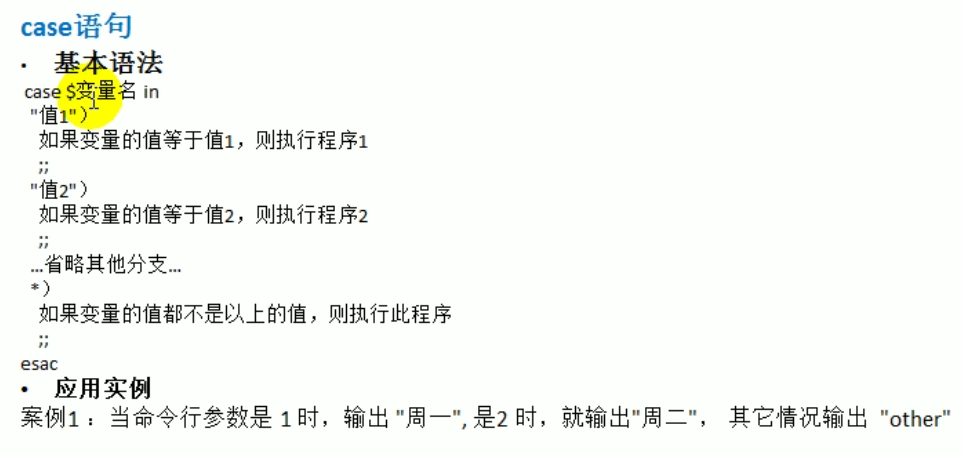
elif 条件

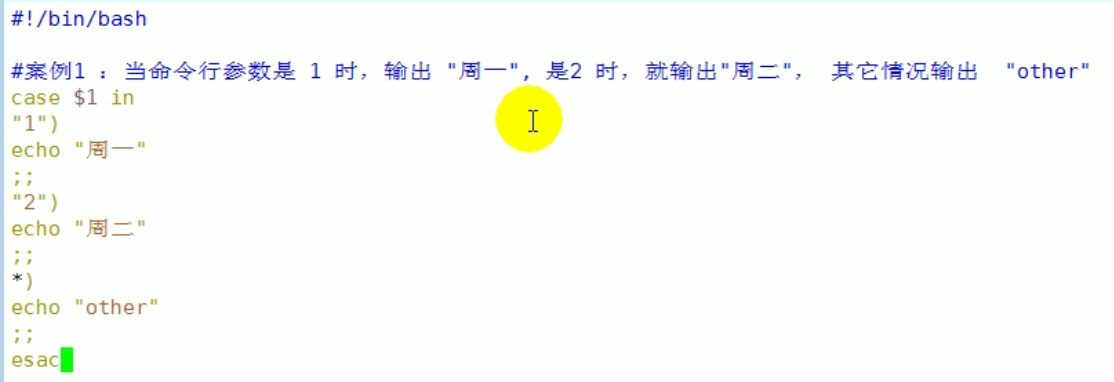
then

--

fi

1. **case语句**





1. **for循环语句**

基本语法：

方式1：

for 变量 in 值1 值2

do

程序

done

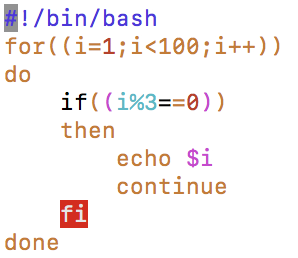
方式2：

for((初始值；循环控制条件；变量变化))

do

程序

done



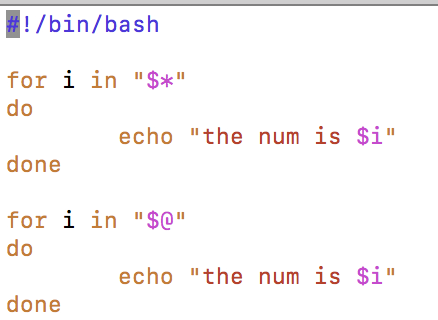
for出错：

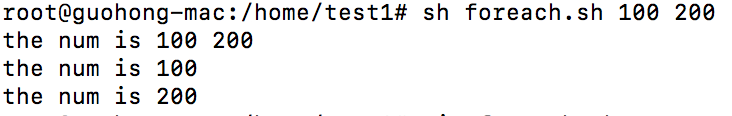
有的时候，执行上面语句，在Ubuntu系统下会出错，是因为因为Ubuntu为了加快开机速度，用dash代替了传统的bash，是dash在捣鬼。

解决方法：取消dash

dpkg-reconfigure dash

出现弹框，选择NO





1. **while循环**

基本语法：

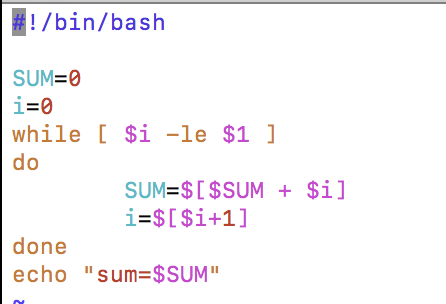
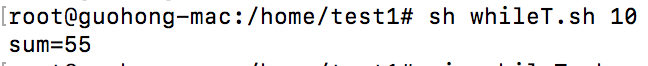
while[条件判断式]

do

程序

done

案例：从命令行输入一个数n，统计从1+…+n的值是多少

1. **read读取控制台输入**

基本语法：

read(选项)（参数）

选项：

-p:指定读取值时的提示符

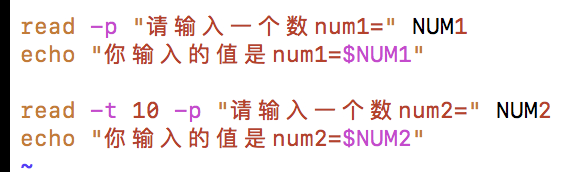
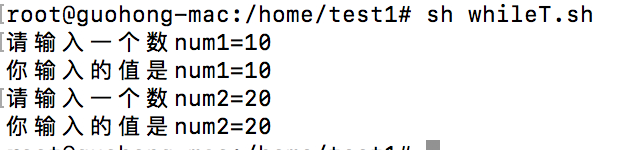
-t:指定读取值时等待的时间（秒），如果没有在指定的时间内输入，就不再等待

参数：

变量：指定读取值的变量名

案例：

1.读取控制台输入一个num值

1. **函数**

shell编程和其他编程语言一样，有系统函数，也可以自定义函数，系统函数中，只介绍两个

**系统函数：**

basename基本语法

功能：返回完整路径最好/的部分，常用于获取文件名

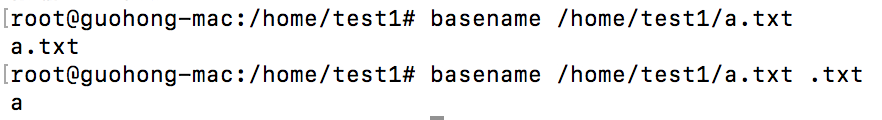
basename[pathname][suffix]

basename[string][suffix](功能描述：basename命令会删掉所有的前缀包括最后一个（‘/’）)字符，然后将字符串显示出来

选项：

suffix为后缀，如果suffix被指定了，basename会将pathname或string中的suffix去掉。

示例：

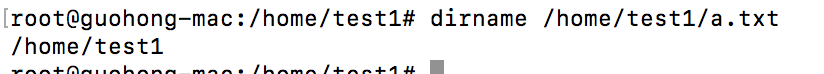


dirname函数：

基本语法：

功能：返回完整路径最后/的前面的部分，常用于返回路径部分

dirname文件绝对路径（功能描述：从给定的包含绝对路径的文件名中去除文件名（非目录部分），然后返回剩下的路径（目录的部分））



自定义函数：

基本语法

[function]funname[()]

{

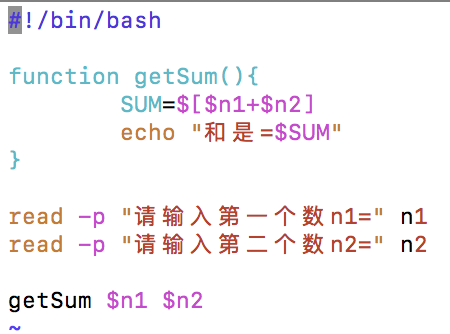
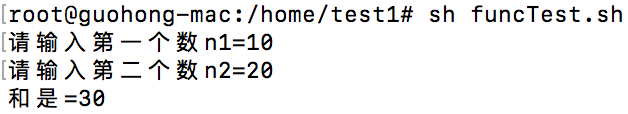
Action;

[return int;]

}

调用直接写函数名：funname

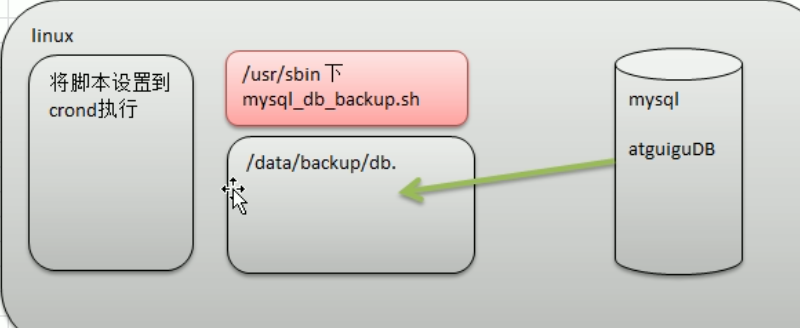
案例：计算两个参数的和

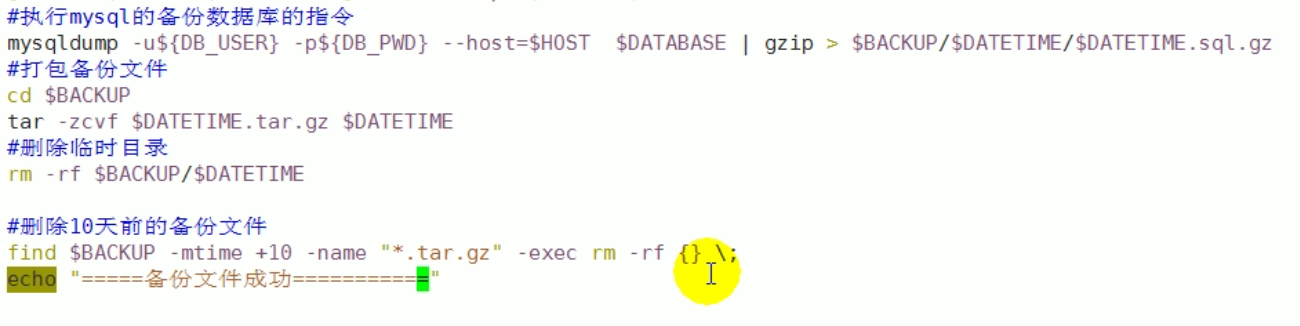
1. **案例**

需求：

1. 每天凌晨2点备份数据库atguiguDB到/data/backup/db
2. 备份开始和备份结束能够给出相应的提示信息
3. 备份后的文件要求以备份的时间为文件名，并打包成tar.gz的形式
4. 在备份的同时，检查是否有10天前备份的数据库文件，如果有就将其删除。







1. **aa**