# 1 Shell 编程

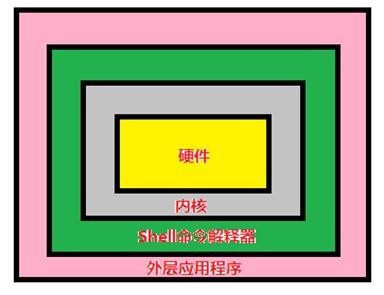
## 1.1 为什么要学习 Shell 编程

1)Linux 运维工程师在进行服务器集群管理时，需要编写 Shell 程序来进行服务器管理。

2)对于 JavaEE 和 Python 程序员来说，工作的需要，你的老大会要求你编写一些 Shell 脚本进行程序或者是服务器的维护，比如编写一个定时备份数据库的脚本。 3)对于大数据程序员来说，需要编写 Shell 程序来管理集群。

## 1.2 Shell 是什么

画一个示意图：



Shell 是一个命令行解释器，它为用户提供了一个向 Linux 内核发送请求以便运行程序的界面系统级程序，用户可以用 Shell 来启动、挂起、停止甚至是编写一些程序.

## 1.3 shell 编程快速入门-Shell 脚本的执行方式

### 1.3.1 脚本格式要求

1. 脚本以#!/bin/bash 开头
2. 脚本需要有可执行权限

### 1.3.2 编写第一个 Shell 脚本

•需求说明

创建一个 Shell 脚本，输出 hello world!

看案例：



### 1.3.3 脚本的常用执行方式

•方式 1(输入脚本的绝对路径或相对路径)

1)首先要赋予 helloworld.sh 脚本的+x 权限

2)执行脚本



•方式 2(sh+脚本)，不推荐

说明：不用赋予脚本+x 权限，直接执行即可



## 1.4 shell 的变量

### 17.4.1 Shell 的变量的介绍

1. Linux Shell 中的变量分为，系统变量和用户自定义变量。
2. 系统变量：$HOME、$PWD、$SHELL、$USER 等等比如： echo $HOME 等等..



1. 显示当前 shell 中所有变量：set

### 1.4.2 shell 变量的定义

•基本语法

1)定义变量：变量=值

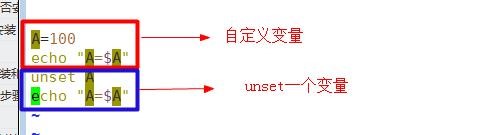
2)撤销变量：unset 变量

3) 声明静态变量：readonly 变量，注意：不能 unset

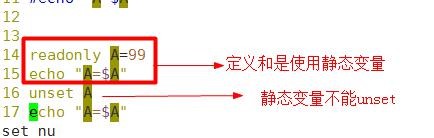
•快速入门

案例 1：定义变量 A

案例2：撤销变量A



案例 3：声明静态的变量 B=2，不能 unset



案例 4：可把变量提升为全局环境变量，可供其他 shell 程序使用【一会举例。】

### 1.4.3 •定义变量的规则

1. 变量名称可以由字母、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. 变量名称一般习惯为大写

### 1.4.4 •将命令的返回值赋给变量（重点）

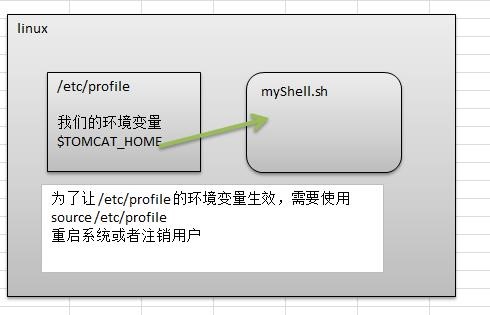
1. A=`ls -la` 反引号，运行里面的命令，并把结果返回给变量 A
2. A=$(ls -la) 等价于反引号



## 1.5 设置环境变量

### 1.5.1 基本语法

1. export 变量名=变量值 （功能描述：将 shell 变量输出为环境变量）
2. source 配置文件 （功能描述：让修改后的配置信息立即生效）
3. echo $变量名 （功能描述：查询环境变量的值）



### 1.5.2 快速入门

1. 在/etc/profile 文件中定义 TOMCAT\_HOME 环境变量



1. 查看环境变量 TOMCAT\_HOME 的值

echo $TOMCAT\_HOME

1. 在另外一个 shell 程序中使用 TOMCAT\_HOME



注意：在输出 TOMCAT\_HOME 环境变量前，需要让其生效

source /etc/profile

## 1.6 位置参数变量

### 1.6.1 介绍

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

### 1.6.2 基本语法

$n （功能描述：n 为数字，$0 代表命令本身，$1-$9 代表第一到第九个参数，十以上的参数，十以上的参数需要用大括号包含，如${10}）

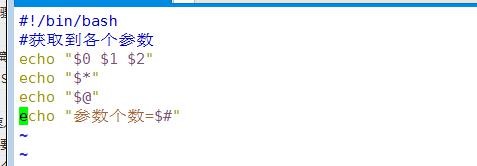
$\* （功能描述：这个变量代表命令行中所有的参数，$\*把所有的参数看成一个整体）

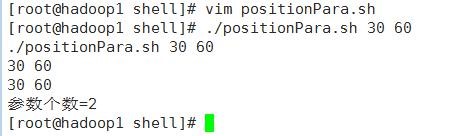
$@（功能描述：这个变量也代表命令行中所有的参数，不过$@把每个参数区分对待）

$#（功能描述：这个变量代表命令行中所有参数的个数）

### 1.6.3 位置参数变量应用实例

案例：编写一个 shell 脚本 positionPara.sh ， 在脚本中获取到命令行的各个参数信息





## 1.7 预定义变量

### 1.7.1 基本介绍

就是 shell 设计者事先已经定义好的变量，可以直接在 shell 脚本中使用

### 1.7.2 基本语法

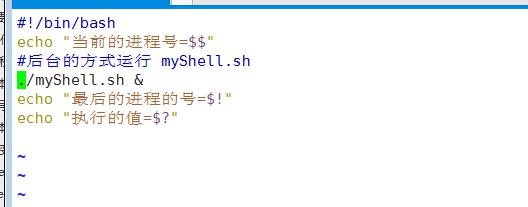
$$ （功能描述：当前进程的进程号（PID））

$! （功能描述：后台运行的最后一个进程的进程号（PID））

$？（功能描述：最后一次执行的命令的返回状态。如果这个变量的值为 0，证明上一个命令正确执行；如果这个变量的值为非 0（具体是哪个数，由命令自己来决定），则证明上一个命令执行不正确了。）

### 1.7.3 应用实例

在一个 shell 脚本中简单使用一下预定义变量



## 1.8 运算符

### 1.8.1 基本介绍

学习如何在 shell 中进行各种运算操作。

### 1.8.2 基本语法

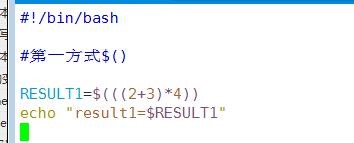
1. “$((运算式))”或“$[运算式]”
2. expr m + n

注意 expr 运算符间要有空格

1. expr m - n
2. expr \\*, /, % 乘，除，取余

•应用实例案例 1：计算（2+3）X4 的值

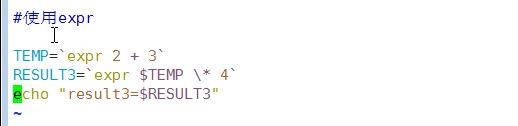
1. $((运算式))



1. $[运算式]



1. expr



案例 2：请求出命令行的两个参数[整数]的和



## 1.9 条件判断判断语句

### 1.9.1 •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）

### 1.9.2 •应用实例

[ atguigu ] 返回 true

[] 返回 false

[condition] && echo OK || echo notok 条件满足，执行后面的语句

### 1.9.3 •常用判断条件

1)两个整数的比较

= 字符串比较

-lt 小于

-le 小于等于

-eq 等于

-gt 大于

-ge 大于等于

-ne 不等于

2) 按照文件权限进行判断

-r 有读的权限 [ -r 文件 ]

1. 有写的权限
2. 有执行的权限

3)按照文件类型进行判断

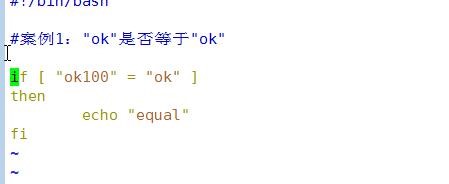
-f 文件存在并且是一个常规的文件

-e 文件存在

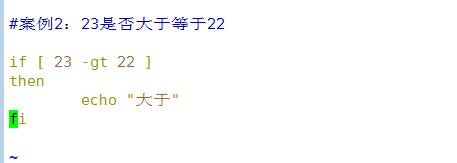
-d 文件存在并是一个目录

### 1.9.4 应用实例

案例 1："ok"是否等于"ok" 判断语句：



案例 2：23 是否大于等于 22 判断语句：



案例 3：/root/install.log 目录中的文件是否存在

判断语句：



## 1.10流程控制

### 1.10.1 if 判断

•基本语法

if [ 条件判断式 ];then

程序

fi

或者 if [ 条件判断式 ]

then

程序

elif [条件判断式]

then

程序

fi

注意事项：（1）[ 条件判断式 ]，中括号和条件判断式之间必须有空格 (2) 推荐使用第二种方式

•应用实例

案例：请编写一个 shell 程序，如果输入的参数，大于等于 60，则输出 "及格了"，如果小于 60, 则输出 "不及格"



### 1.10.2 case 语句

•基本语法

case $变量名 in

"值 1"）

如果变量的值等于值 1，则执行程序 1

;;

"值 2"）

如果变量的值等于值 2，则执行程序 2

;;

…省略其他分支…

\*）

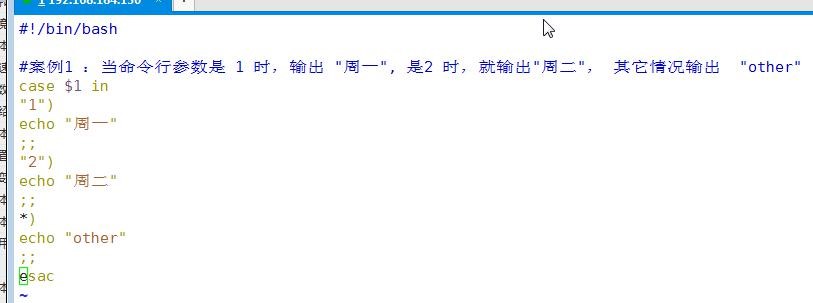
如果变量的值都不是以上的值，则执行此程序

;;

esac

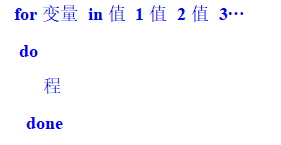
•应用实例

案例 1 ：当命令行参数是 1 时，输出 "周一", 是 2 时，就输出"周二"， 其它情况输出 "other"



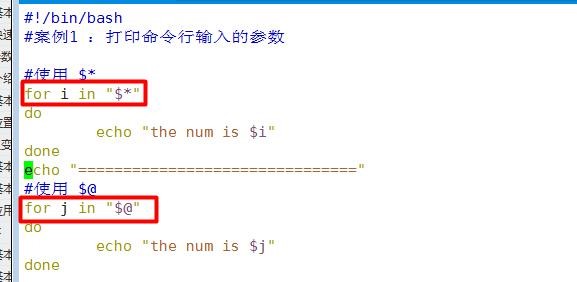
### 1.10.3 for 循环

基本语法1

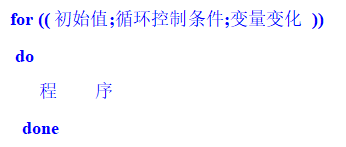


•应用实例

案例 1 ：打印命令行输入的参数 【会使用到$\* $@】



•基本语法 2

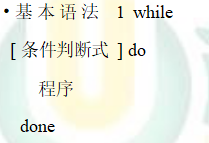


•应用实例

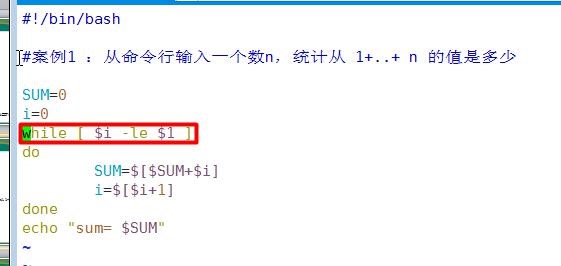
案例 1 ：从 1 加到 100 的值输出显示



### 1.10.4 while 循环



•应用实例案例 1 ：从命令行输入一个数 n，统计从 1+..+ n 的值是多少？



## 1.11read 读取控制台输入

### 1.11.1 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，就不再等待了。。

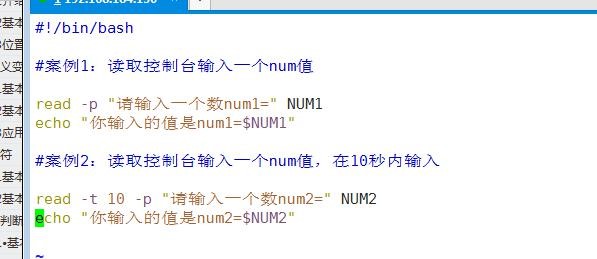
参数

变量：指定读取值的变量名

### 1.11.2 应用实例

案例 1：读取控制台输入一个 num 值

案例 2：读取控制台输入一个 num 值，在 10 秒内输入。



## 1.12函数

### 1.12.1 函数介绍

shell 编程和其它编程语言一样，有系统函数，也可以自定义函数。系统函数中，我们这里就介绍两个。

### 1.12.2 系统函数

•basename 基本语法

功能：返回完整路径最后 / 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。

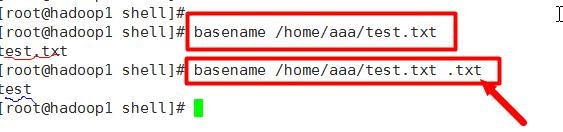
选项： suffix 为后缀，如果 suffix 被指定了，basename 会将 pathname 或 string 中的 suffix 去掉。

•dirname 基本语法

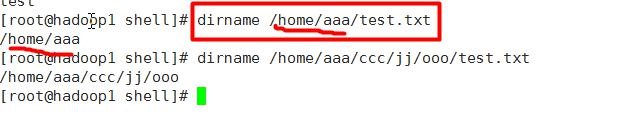
功能：返回完整路径最后 / 的前面的部分，常用于返回路径部分 dirname 文件绝对路径 （功能描述：从给定的包含绝对路径的文件名中去除文件名（非目录的部分），然后返回剩下的路径（目录的部分））

### 1.12.3 •应用实例

案例 1：请返回 /home/aaa/test.txt 的 "test.txt" 部分



案例 2：请返回 /home/aaa/test.txt 的 /home/aaa



### 1.12.4 自定义函数

•基本语法

[ function ] funname[()]

{

Action;

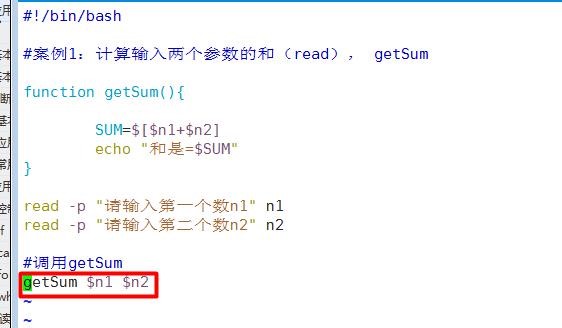
[return int;]

}

调用直接写函数名：funname [值]

•应用实例

案例 1：计算输入两个参数的和（read）， getSum



## 1.13Shell 编程综合案例

需求分析

1)每天凌晨 2:10 备份 数据库 atguiguDB 到 /data/backup/db

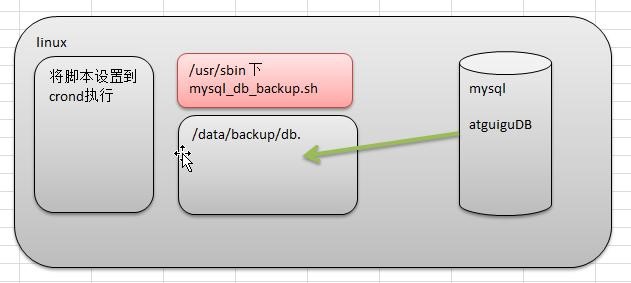
2)备份开始和备份结束能够给出相应的提示信息

3)备份后的文件要求以备份时间为文件名，并打包成 .tar.gz 的形式，比如：

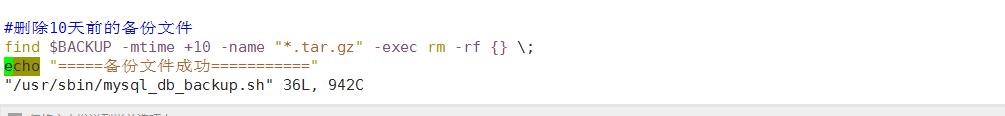
2018-03-12\_230201.tar.gz

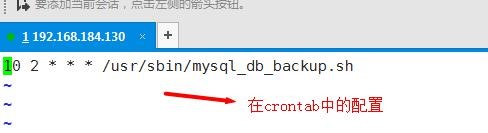
4) 在备份的同时，检查是否有 10 天前备份的数据库文件，如果有就将其删除。编写一个 shell 脚本。

思路分析：



代码实现：





#!/bin/bash

#完成数据库的定时备份。

#备份的路径

BACKUP=/data/backup/db

#当前的时间作为文件名

DATETIME=$(date +%Y\_%m\_%d\_%H%M%S)

#可以输出变量调试

#echo ${DATETIME}

echo "=======开始备份========"

echo "=======备份的路径是 $BACKUP/$DATETIME.tar.gz"

#主机

HOST=localhost

#用户名

DB\_USER=root

#密码

DB\_PWD=root

#备份数据库名

DATABASE=atguiguDB

#创建备份的路径

#如果备份的路径文件夹存在，就使用，否则就创建

[ ! -d "$BACKUP/$DATETIME" ] && mkdir -p "$BACKUP/$DATETIME"

#执行mysql的备份数据库的指令

mysqldump -u${DB\_USER} -p${DB\_PWD} --host=$HOST $DATABASE | gzip > $BACKUP/$DATETIME/$DATETIME.sql.gz

#打包备份文件

cd $BACKUP

tar -zcvf $DATETIME.tar.gz $DATETIME

#删除临时目录

rm -rf $BACKUP/$DATETIME

#删除10天前的备份文件

find $BACKUP -mtime +10 -name "\*.tar.gz" -exec rm -rf {} \;

echo "=====备份文件成功==========="