**shell：**

**是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核服务。是一个用c语言编写的程序，他是用户使用linux的桥梁。既是一种命令语言，有是一种程序设计语言。**

**shell脚本(shell script),是一种为shell编写的脚本程序。**

**shell脚本编程跟java,php编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。**

**linux的shell种类众多，常见的有：**

**Bourne shell(/usr/bin/sh或者/bin/sh)**

**Bourne Again shell(/bin/bash) [大多数linux默认的shell]**

**C shell(/usr/bin/csh)**

**K shell(/usr/bin/ksh)**

**shell for root (/sbin/sh)**

**一般情况下#！/bin/sh 和 #!/bin/bash没有区别，可以互换。**

**#!告诉系统 其后路径所指定的程序 就是解释此脚本文件的shell程序**

**第一个shell脚本：新建一个test.sh文件，扩展名sh(sh代表shell),[扩展名并不影响脚本的执行，见名知其意就好，如果用php写的shell脚本，扩展名就用php好了]**

**#!/bin/sh**

**echo "hello world!"**

**#!是一个约定的标记，他告诉系统这个脚本需要什么解释器来执行，即使有哪一种shell.**

**echo 命令用于向窗口输出文本。**

**运行shell脚本有两种方法：**

**1.作为可执行程序**

**将上面的代码保存为test.sh,并cd到相应目录。**

**chmod +x ./test.sh #使脚本具有执行权限。**

**./test.sh #执行脚本**

**注意一定要写成./test.sh,而不是test.sh，运行其他二进制的程序也一样，直接写test.sh,linux系统会去PATH里面寻找有没有叫test.sh的，而只有/bin,/sbin,/usr/bin,/usr/sbin等在PATH里，你的当前目录通常不再PATH里，所以写成test.sh是会找不到命令的，要用./test.sh告诉系统说，就在当前目录找。**

**2.作为解释器参数**

**这种运行方式是，直接运行解释器，其参数就是shell脚本的文件名，如：**

**/bin/sh test.sh**

**/bin/php test.php**

**这种方式运行的脚本，不需要再第一行指定解释器信息，写了也没用。shell是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核服务。**

**shell是一个用c语言编写的程序，他是用户使用linux的桥梁。**

**shell既是一种命令语言，有是一种程序设计语言。**

**shell脚本(shell script),是一种为shell编写的脚本程序。**

**shell脚本编程跟java,php编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。**

**linux的shell种类众多，常见的有：**

**Bourne shell(/usr/bin/sh或者/bin/sh)**

**Bourne Again shell(/bin/bash) [大多数linux默认的shell]**

**C shell(/usr/bin/csh)**

**K shell(/usr/bin/ksh)**

**shell for root (/sbin/sh)**

**一般情况下#！/bin/sh 和 #!/bin/bash没有区别，可以互换。**

**#!告诉系统 其后路径所指定的程序 就是解释此脚本文件的shell程序**

**第一个shell脚本：新建一个test.sh文件，扩展名sh(sh代表shell),[扩展名并不影响脚本的执行，见名知其意就好，如果用php写的shell脚本，扩展名就用php好了]**

**#!/bin/sh**

**echo "hello world!"**

**#!是一个约定的标记，他告诉系统这个脚本需要什么解释器来执行，即使有哪一种shell.**

**echo 命令用于向窗口输出文本。**

**运行shell脚本有两种方法：**

**1.作为可执行程序**

**将上面的代码保存为test.sh,并cd到相应目录。**

**chmod +x ./test.sh #使脚本具有执行权限。**

**./test.sh #执行脚本**

**注意一定要写成./test.sh,而不是test.sh，运行其他二进制的程序也一样，直接写test.sh,linux系统会去PATH里面寻找有没有叫test.sh的，而只有/bin,/sbin,/usr/bin,/usr/sbin等在PATH里，你的当前目录通常不再PATH里，所以写成test.sh是会找不到命令的，要用./test.sh告诉系统说，就在当前目录找。**

**2.作为解释器参数**

**这种运行方式是，直接运行解释器，其参数就是shell脚本的文件名，如：**

**/bin/sh test.sh**

**/bin/php test.php**

**这种方式运行的脚本，不需要再第一行指定解释器信息，写了也没用。shell是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核服务。**

**shell是一个用c语言编写的程序，他是用户使用linux的桥梁。**

**shell既是一种命令语言，有是一种程序设计语言。**

**shell脚本(shell script),是一种为shell编写的脚本程序。**

**shell脚本编程跟java,php编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。**

**linux的shell种类众多，常见的有：**

**Bourne shell(/usr/bin/sh或者/bin/sh)**

**Bourne Again shell(/bin/bash) [大多数linux默认的shell]**

**C shell(/usr/bin/csh)**

**K shell(/usr/bin/ksh)**

**shell for root (/sbin/sh)**

**一般情况下#！/bin/sh 和 #!/bin/bash没有区别，可以互换。**

**#!告诉系统 其后路径所指定的程序 就是解释此脚本文件的shell程序**

**第一个shell脚本：新建一个test.sh文件，扩展名sh(sh代表shell),[扩展名并不影响脚本的执行，见名知其意就好，如果用php写的shell脚本，扩展名就用php好了]**

**#!/bin/sh**

**echo "hello world!"**

**#!是一个约定的标记，他告诉系统这个脚本需要什么解释器来执行，即使有哪一种shell.**

**echo 命令用于向窗口输出文本。**

**运行shell脚本有两种方法：**

**1.作为可执行程序**

**将上面的代码保存为test.sh,并cd到相应目录。**

**chmod +x ./test.sh #使脚本具有执行权限。**

**./test.sh #执行脚本**

**注意一定要写成./test.sh,而不是test.sh，运行其他二进制的程序也一样，直接写test.sh,linux系统会去PATH里面寻找有没有叫test.sh的，而只有/bin,/sbin,/usr/bin,/usr/sbin等在PATH里，你的当前目录通常不再PATH里，所以写成test.sh是会找不到命令的，要用./test.sh告诉系统说，就在当前目录找。**

**2.作为解释器参数**

**这种运行方式是，直接运行解释器，其参数就是shell脚本的文件名，如：**

**/bin/sh test.sh**

**/bin/php test.php**

**这种方式运行的脚本，不需要再第一行指定解释器信息，写了也没用。shell是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核服务。**

**shell是一个用c语言编写的程序，他是用户使用linux的桥梁。**

**shell既是一种命令语言，有是一种程序设计语言。**

**shell脚本(shell script),是一种为shell编写的脚本程序。**

**shell脚本编程跟java,php编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。**

**linux的shell种类众多，常见的有：**

**Bourne shell(/usr/bin/sh或者/bin/sh)**

**Bourne Again shell(/bin/bash) [大多数linux默认的shell]**

**C shell(/usr/bin/csh)**

**K shell(/usr/bin/ksh)**

**shell for root (/sbin/sh)**

**一般情况下#！/bin/sh 和 #!/bin/bash没有区别，可以互换。**

**#!告诉系统 其后路径所指定的程序 就是解释此脚本文件的shell程序**

**第一个shell脚本：新建一个test.sh文件，扩展名sh(sh代表shell),[扩展名并不影响脚本的执行，见名知其意就好，如果用php写的shell脚本，扩展名就用php好了]**

**#!/bin/sh**

**echo "hello world!"**

**#!是一个约定的标记，他告诉系统这个脚本需要什么解释器来执行，即使有哪一种shell.**

**echo 命令用于向窗口输出文本。**

**运行shell脚本有两种方法：**

**1.作为可执行程序**

**将上面的代码保存为test.sh,并cd到相应目录。**

**chmod +x ./test.sh #使脚本具有执行权限。**

**./test.sh #执行脚本**

**注意一定要写成./test.sh,而不是test.sh，运行其他二进制的程序也一样，直接写test.sh,linux系统会去PATH里面寻找有没有叫test.sh的，而只有/bin,/sbin,/usr/bin,/usr/sbin等在PATH里，你的当前目录通常不再PATH里，所以写成test.sh是会找不到命令的，要用./test.sh告诉系统说，就在当前目录找。**

**2.作为解释器参数**

**这种运行方式是，直接运行解释器，其参数就是shell脚本的文件名，如：**

**/bin/sh test.sh**

**/bin/php test.php**

**这种方式运行的脚本，不需要再第一行指定解释器信息，写了也没用。**

**10小时带你shell从入门到精通 ---学神IT教育**

**1.shell编程基本语法(变量的使用方法)**

**shell基本语法：变量，表达式，判断语句，if表达式。**

**先看一个简单的shell程序：**

**mkdir test #创建目录**

**cd test/ #进入目录**

**vim example01.sh #创建shell程序**

**精讲：**

**#!:跟shell命令的完整路径，显示后期命令以哪种shell来执行。**

**which shell #显示shell的所在路径**

**cat /etc/shells #其他shell的完整路径**

**#：shell中表示注释**

**shell程序一般以.sh结尾。**

**执行的三种方法：**

**./example01.sh （需要权限）**

**写绝对路径 #/home/tony/test/example01.sh （需要权限）**

**使用bash #bash example01.sh (不需要权限，可以省略chmod 这一步）**

**chmod +x example01.sh #修改文件权限**

**./example01.sh #运行shell程序**

**which bash**

**cat /etc/shells #查看其他shell的完整路径**

**echo $PATH #查看环境变量**

**which ls #因为平时所用的命令都在环境变量中，所以直接ls就可以在执行。**

**set #列出所有变量**

**set | grep a #过滤出a变量**

**shell变量:**

**变量是shell传递数据的一种方法。变量是用来代表每个值的符号名。**

**shell 有两种变量：临时变量和永久变量**

Shell

图形界面：Windows explorer; KDE,gnome

命令行shell(command line lnterface shell,即）

交互式shell和非交互式shell

登录式shell和非登录式shell

History:查看命令历史（1000条）

！命令历史中的序号（叹号和数字之间无空格）-----运行此命令。

！命令前两个字符（没有空格）----运行此命令。

！$------运行上一条命令。

通配符：

# ls \*.txt //匹配零个或者多个文件

A.txt b.txt

#ls ?.txt //匹配一个字符

#ls [a,b].txt //中括号里面任选一个

A.txt b.txt

#ls \*txt //通配零个

A.txt b.txt txt

变量：

#a=1

#echo $a

#1

Ctrl+z 暂停

Ctrl+c 终止

正确重定向:

重定向： > >>

反向重定向： <

#echo “111” > file.txt

#echo “111” >> file.txt //追加，添加。

# cat < 1.txt //直接输出1.txt的内容。（cat不能换成文件如2.txt)

#mail -s ”zhuti” zhan2103208467@163.com（名字或者地址） < 1.txt(邮箱内容)

错误重定向：

2> 2>>

# ls listing

#----么有那个文件-----

# ls listing 2> error.log //错误信息不在打印，而存到error.log文件中了。

#cat error.log

#----么有那个文件-----

#ls listing 2>> error.log //追加

#cat error.log

#----么有那个文件-----

----么有那个文件-----

Eg：

#sh sss.sh > 1.log 2> 1.log //运行脚本sss.sh把其中的错误和正确信息把重定向到了1.log这个文件中，并且不会覆盖。

#sh sss.sh >> 1.log 2>> 1.log //追加

#sh sss.sh >> 1.log 2>>&1

管道命令：前面的输出作为后面的输入.

Sleep:

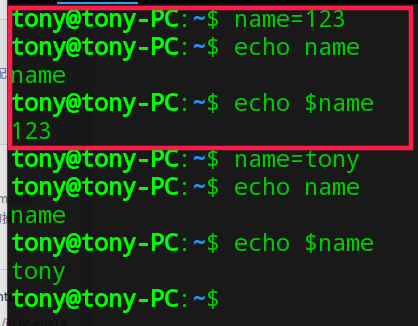
Slepp 100 //睡眠110秒

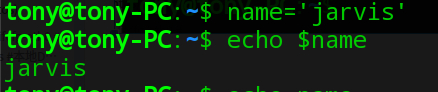
Ctrl+z 暂停

Ctrl+c 终止

Jobs ----查看已经被暂停的任务。

Fg 序号 ---



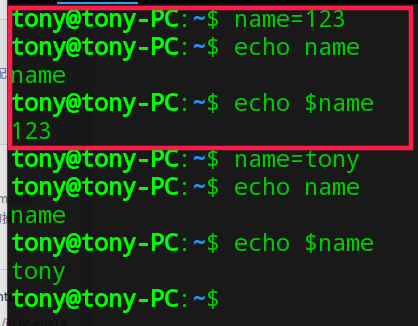


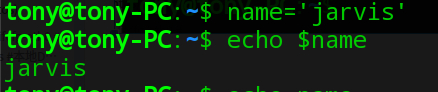
循环：

**关系运算符**

关系运算符只支持数字，不支持字符串，除非字符串的值是数字。

return只支持数字：<https://blog.csdn.net/zycamym/article/details/45191093>





循环：

**关系运算符**

关系运算符只支持数字，不支持字符串，除非字符串的值是数字。

return只支持数字：<https://blog.csdn.net/zycamym/article/details/45191093>