# Shell

定义:shell编程就是保护操作系统的,它也是一门语言

目的:在使用操作系统时候,需要使用shell来进行语法的编辑

基本的使用流程

1. 新建.sh结尾的文件
2. 第一行输入解释器
   1. #!空格/bin/bash
3. 自己的shell:

Echo”写自己想要输出的文字”

1. 在终端中打开:

Sh 想要执行的文件名

注释语法

单行注释语法:在代码前面加一个#

注意:在测试web和app 时候在最后项目上线之前的测试中,一定要求把注释的代码删除掉,

变量的调用语法

"${变量名}" 或者 ${变量名} 或者 $变量名 最后一种比较简洁

删除变量

unset 变量名

echo ${name2}

read –p “提示文字”变量名

例：read –p “请输入”：

当用户输入值之后．就会保存在str中．我们可以随时使用调用变量的语法来获取：

# 命令变量

使用反单引号（数字1左侧按钮）包裹的linux命令就可以执行完毕后在使用

# 计数语法

Ls 空格/home空格|空格wc空格-l

Wc –l就是获取个数的意思

# 比较运算符

#大于

#[ ${num1} -gt $num2 ]

#小于

[ ${num1} -lt ${num2} ]

#小于等于

#[ ${num1} -le ${num2} ]

#大于等于

#[ ${num1} -ge ${num2} ]

#等于

#[ ${num1} -eq ${num2} ]

#不等于

#[ ${num1} -ne ${num2} ]

#[]结果里面是相反的

echo "$?"

# 逻辑运算

格式：[ 表达式 ]

​ -a == 与 and

​ -o == 或 or

​ ! == 非

​ 例：[ $num1 -gt $num2 -a $num2 -lt $num3 ]

​ [ ! $num1 -gt $num2 ] 注意叹号前后的空格！

# 字符串判断

格式：[ 表达式 ]

​ 符号：

​ == :判断字符串内容是否一样

​ != : 判断两个字符串内容不等

​ -z : 判断字符串是不是空串

​ 例：

​ [ $str1 == $str2 ]

​

​ str3=""

​ [ -z $str3 ]

# 判断文件

格式：[ 表达式 ]

符号：

-e === 文件或文件夹是否存在 exists

-d === 是不是文件夹 类型dir

-f === 是不是文件类型file

# 二、

# 函数

1.定义：最小的封装单元，实现了一些功能

好处：代码的复用，程序结构更加清晰

语法：

Function **hello**（）{

函数体例如echo “helloworld”

}

调用hello

注意：函数必须先声明，在调用，function可以不写

# 函数传参

语法：

He（）{

Quhe=$(($1+$2))

Echo “${quhe}”

Echo “$0”

Echo “$1”

Echo “$\*”

Echo “$#”

}

Quhe 6 9

# 三、了解Shell的返回值

定义：在函数体中的结果．可以返回给调用函数的位置，但是在shell中，不用返回值，

一样可以获取函数体中的结果所以返回值语法在shell中不用出现、

# 四、分支语句

语法

if[条件判断]

then

满足条件后的代码

fi

双分支语法：

前面与单分支一致

else：

否则的代码

举例：

#! /bin/bash

read -p "请输入年龄：" age

if [ $age -lt 18 ]

then

echo "未成年"

else

echo “成年”

fi

# 五、多分支语法

语法：

前面都一样，

Elif[条件2]

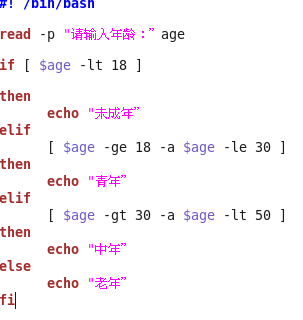
then

条件满足才执行的代码2

Elif：

条件满足才执行的代码段

否则的代码



# Case分支语法

Case和if都可以实现分支，case比较适合在手机端使用，因为运行效率比if高

语法：

#! /bin/bash

read -p "请输入数字：" num

case $num in

1)

echo "简单";;

2)

echo "一般";;

3)

echo "困难";;

\*)

echo 以上两个都不满足，可以执行这里的错误

esac

总结：if可以替代case，因为if能实现的，case能实现，case不能实现的，if也能实现

# For in 循环遍历语法

定义：循环遍历

语法：

for变量名in要遍历的区间

do

循环体

done

特殊用法：遍历区间

`seq`数字1数字2 ；是数字1到数字2的循环

`seq`数字1数字2 数字3 ；是数字1到数字3的循环，数字2是步长

#! /bin/bash

#for num in 1 2 3 4 5 6 7

#for num in `seq 90 100`

#for num in `seq 90 2 100`

do

echo "$num"

done

# 重定向（把日志保存在磁盘里）

语法规则：

一 ls /home 1>>jg.txt

解释：把ls的最终执行结果，放在jg.txt文件中，追加效果

2. ls /home 2>>jg.txt

解释：把ls的最终执行结果，放在jg.txt文件中，追加效果；（语法必须写错，才会把报错的信息放在txt文件中）

二．

ls /home 1>jg.txt

解释：把ls的最终执行结果，放在jg.txt文件中，覆盖效果

ls /home 2>jg.txt

解释：把ls的最终执行结果，放在jg.txt文件中，覆盖效果；（语法必须写错，才会把报错的信息放在txt文件中）

# 数组

数组：就是保存数据的一种方式，可以保存多条数据

1. 数组的新增（声明）

shuzu={值1 值2 值3……}

1. 获取数组中的值
2. 获取某一个值：

${shuzu[0]} 数字代表索引，地址下标

1. 获取所有值

${shuzu[@或者\*]} 里面的@和\*都可以获取所有值

1. 获取数组的长度

${# shuzu[@或者\*]}

1. 数组的内容更新，修改

Shuzu[0]=”改后的值”

1. 清空数组

unset shuzu