# shell编程-脚本语言

## 执行脚本

脚本文件：已.sh为后缀名，安装 Shell 脚本之前，需要确保文件有执行权限

执行脚本：bash<脚本文件名 // (常用)、不能带参数

执行脚本：bash 脚本文件名 //必须加上路径 /bin/bash 或 脚本加上 #!/bin/bash

执行脚本：./脚本文件名 //需要可执行权限

# 脚本代码

脚本基本代码

#!/bin/bash //作用：执行时自动完成路径

set 实参1 实参2 实参3... //脚本里面设置实参

echo [参数] 内容 // 在echo里执行命令用 `...` 或 $(...)

## echo输出

echo输出参数

-e：启用转义字符。

-E: 不启用转义字符（默认）

-n: 结尾不换行。

echo中-e参数的转义字符

\a 发出警告声；

\b 删除前一个字符；

\c 不产生进一步输出 (\c 后面的字符不会输出)；

\f 换行但光标仍旧停留在原来的位置；

\n 换行且光标移至行首；

\r 光标移至行首，但不换行；

\t 插入tab；

\v 与\f相同；

\\ 插入\字符；

\nnn 插入 nnn（八进制）所代表的ASCII字符；

## 特殊符号

在双引号中 //echo输出时在单引号中全部原样输出，以下符号都没用

$：变量替换 //$HOME家目录、$PATH当前路径、创建变量 a=`date`

`：表示命令替换 //执行命令结果

\：转义字符 //区分$和`是不是特殊符号

执行命令中

<：输入重定向符 //命令 < 文件名、将一个文件作为命令的输入

>：输出重定向符 //命令 > 文件名、将命令的输出重定向到一个文件中

>>：附加重定向符 //命令 >> 文件名、将命令的输出追加到一个文件中

|：管道符 //将一个命令的输出传递给另一个命令作为输入

#：注释行

;:顺序执行 //用;隔开命令，再按顺序执行

||：逻辑或 //如果命令1为假，再执行命令2。命令1为真，不执行命令2

&&：逻辑与 //如果命令1为真，再执行命令2。命令1为假，不执行命令2

=：赋值符

## 变量

### 用户变量

用户变量：在编写shell脚本时，由用户自定义变量，变量名区分大小写

创建变量：变量名=”变量” //变量中有空格时，使用””，创建变量不需要$

注意：算数运算时，需要使用 `expr 变量 + 变量`，如：`expr $a + $b`

### 内部变量

内部变量：用户不能修改的

$#：传递shell程序的位置参数的个数

$\*：传递给shell程序时所传递的所有参数组成的字符串

$0：表示shell名称

$HOME：表示当前用户的主目录。

$PWD：表示当前工作目录（即你所在的目录）的路径。

$USER：表示当前登录用户的用户名。

$SHELL：表示当前使用的 Shell 程序的名称和路径。

$PATH：表示系统查找可执行文件时要搜索的路径列表。

$PS1：表示 Shell 提示符的格式。

$PS2：表示 Shell 在读取多行命令时的提示符的格式。

$IFS：表示用于分隔 Shell 命令中的字段的内部字段分隔符。

$@：表示传递给 Shell 脚本或函数的所有参数（每个参数都是一个独立的字符串）。

### 环境变量

环境变量：由系统管理员或用户设置的，可以在 Shell 中使用 export 命令导出为环境变量

$PATH：表示系统查找可执行文件时要搜索的路径列表。

$HOME：表示当前用户的主目录。

$SHELL：表示当前使用的 Shell 程序的名称和路径。

$USER：表示当前登录用户的用户名。

$PS1：表示 Shell 提示符的格式。

$EDITOR：表示用于编辑文本的程序名称和路径。

$LANG：表示当前语言环境。

$TMPDIR：表示临时文件存储目录的路径。

$LD\_LIBRARY\_PATH：装路径。

$PYTHONPATH：表示 Python 模表示动态链接库的搜索路径列表。

$JAVA\_HOME：表示 Java 的安块的搜索路径列表。

$MANPATH：表示 man 帮助文档的搜索路径列表。

$RANDOM：随机数(0~32767)。

### 位置变量

位置变量：接收实参的变量（形参），名称 $1、$2、$3......$9

使用位置变量：bash 脚本名 实参1 实参2...

## Set设置实参命令

设置实参：set 实参1 实参2 实参3... //脚本里面设置实参

## Shift参数左移命令

实参参数左移：shift [数字] //不加数字左移1位，加数字左移n位

## 正则表达式

查找内容：grep “正则表达式” 文件目录 // 在文件内容中查找有关内容字符串行

正则表达式

^ 表示匹配行首的文本(以谁开始)

$ 表示匹配行尾的文本(以谁结束)

. 任意一个字符

\* 重复零次或更多次

+ 重复一次或更多次

? 重复零次或一次

{n} 重复n次

{n,} 重复n次或更多次

{n,m} 重复n到m次

[] 匹配字符集合// [abc]，匹配a或b或c，[a-z0-9A-Z]，匹配数字和大小字母其中一个

[^...] 取反字符集合 // [^abc]，匹配除了a或b或c

\d 匹配0-9之间的任一数字，相当于[0-9]

\D 匹配所有0-9以外的字符，相当于 [^0-9]

\w 匹配任意的字母、数字和下划线，相当于[A-Za-z0-9\_]

\W 除所有字母、数字和下划线以外的字符，相当于 [^A-Za-z0-9\_]

\s 匹配空格 (包括换行符、制表符、空格符等)，相等于[\t\r\n\v\f]

\S 匹配非空格的字符，相当于 [^\t\r\n\v\f]

## 算数运算

整数算术运算：let “算数表达式” 或 ((算数表达式)) //加号前后无空格

底层算数运算：`expr 算数表达式` //如：a=`expr $a + $b`，加号前后有空格  
括号算术运算：$((算数表达式)) 或 $((算数表达式)) //加号前后无空格

## 控制结构

### 数字条件判断符号

等于：-eq //== //如： if [ $a -eq $b ]  
不等于：-ne //!=  
大于：-gt //>  
大于等于：-ge //>=  
小于：-lt //<  
小于等于：-le // <=

### 字符串条件判断符号

是否为空：-z // 例如： if [ -z $str]

是否非空：-n // 例如： if [ -n $str]

是否相等：= // 例如：if [ $str = “wgs”]，数字判断用==，字符串可以用=

### 文件条件判断符号

“-e”：如果文件存在，则返回 True。

“-f”：如果文件存在并且是一个普通文件，则返回 True。

“-d”：如果文件存在并且是一个目录，则返回 True。

“-s”：如果文件存在并且大小不为零，则返回 True。

“-r”：如果文件存在并且具有读权限，则返回 True。

“-w”：如果文件存在并且具有写权限，则返回 True。

“-x”：如果文件存在并且具有执行权限，则返回 True。

“-z”：如果文件存在并且大小为零，则返回 True。

### 逻辑判断符号

“&&” 或 “-a”：表示逻辑与，前后两个条件都满足返回True 例如：if [ -f file1 ] && [ -f file2 ];

“||” 或 “-o”：表示逻辑或，前后两个条件满足其一返回True 例如：if [ -f file1 ] || [ -f file2 ];

“!”：表示逻辑非，取反 例如：if [ ! -f file1 ];

### 条件控制语句

#### If条件控制语句：

if [条件判断] //if开头

then

真

elif [条件判断]

then

真

else

假

fi //fi结尾

#### case 条件控制语句

case 字符串/变量 in //case开头

匹配字符串/变量)

真

;;

匹配字符串/变量)

真

;;

\*)

假

;;

esac //esac结尾

#### while循环语句

while [条件判断] //while开头。条件为：时，无限循环

do

真

done //done结尾

#### for循环语句

格式一

for 变量名 in 变量值 //变量值赋值给变量名

do

真

done

例：

#!/bin/bash

for fruit in apple banana orange

do

echo "I like eating $fruit."

done

格式二

for 变量名 in 变量值集合 //变量值赋值给变量名

do

真

done

例：

#!/bin/bash

fruits=apple banana orange

for fruit in $fruits

do

echo "I like eating $fruit."

done

格式三

for 变量名 in $\* //$\*为传入的实参值集合

do

真

done

#### until循环语句

until [条件判断] //until开头

do

假

done //done结尾

### break终止循环语句

停止循环：break [数字] //默认数字为1，结束一层循环

### continue跳出本次循环

跳出本次循环：continue //跳出本次循环，进入下一次循环

## exit退出shell脚本

结束shell脚本：exit

## 输入变量

输入变量：read 变量名 //用户自定义存入变量

## Seq创建等差数列命令

seq [参数]... 尾数 //输出：1 2 3 ... 尾数

seq [参数]... 首数 尾数 //输出：首数 ... 尾数

seq [参数]... 首数 增量 尾数 //输出：首数 首数+增量 首数+2\*增量 ... 尾数

seq命令参数

-s：指定输出分隔符，默认为换行符 \n ，如 -s\_ 以下划线分隔

-w：将生成的每个数字输出为固定宽数位的字符串，左侧不足补零，如01 02 03

## Shell函数

创建函数：function 函数名 () {echo $1...} //function可以省略不写

调用函数：函数名 参数1 参数2...