## 函数

|  |
| --- |
| 和其它编程语言一样， Bash 也可以定义函数 |
| 一个函数就是一个子程序，用于完成特定的任务，当有重复代码，或者一个任务只需要很少的修改就被重复几次执行时, 这时你应考虑使用函数。 |
| 函数的一般格式  function function\_name {  commands  }  function\_name() {  commands  }  #! /bin/bash  func1()  {  echo "the parameter's count:$#"  echo "the first parameter:$1"  echo "the second parameter:$2"  }  func1 a b |

### 字符串操作

|  |
| --- |
|  |
| **注：pattern，old 中可以使用通配符。** |

### 随机数和 expr 命令

|  |
| --- |
| echo $RANDOM |

expr：通用的表达式计算命令

|  |
| --- |
| 表达式中参数与操作符必须以空格分开，表达式中的运算可以是算术运算，比较运算，字符串运算和逻辑运算 |
| expr 5 % 3 |
| expr 5 \\* 3 # 乘法符号必须被转义 |
| 注意：目前比较少使用，可用$[ ... ]替换 |

### shitf命令

|  |
| --- |
| 一般用于函数或者脚本程序参数处理，特别是参数多于10以上的时候 |
| 将所有参数变量向下移动一个位置,$2变成$1,$3变成$2，依次递进，但$0保持不变 |
| **while [ "$1" != "" ]**  **do**  **echo $1**  **shift**  **done** |

### eval命令

|  |
| --- |
| 将所有的参数连接成一个表达式，并计算或执行该表达式，参数中的任何变量都将被展开 |
| eval arg1 [arg2] ... [argN] |
| listpage="ls -l | more"  $listpage |
| listpage="ls -l | more"  eval $listpage |

### trap命令

|  |
| --- |
| trap command signal |
| command：  1．一般情况下是Linux命令  2．’ ’表示发生陷阱时为空指令，不做任何动作  3．’-’表示发生陷阱时采用缺省指令 |
| signal：  HUP(1) 挂起；一般因终端掉线或用户退出而引发  INT(2) 中断；一般因按下”Ctrl+C”组合键引发  QUIT(3) 退出；一般因按下”Ctrl+\”组合键引发  ABRT(6) 异常中止；一般因某些严重的执行错误而引发  ALRM(14) 闹钟；一般是超时时钟到来而引发  TREM(15) 中止；一般由系统在关机的时候发出 |
| #!/bin/bash  #安装2、3号信号  #处理代码为"rm -f tmp$$;exit 0"  trap "rm -f tmp$$;exit 0" 2 3  #生成文件，文件名为tmp+当前进程号  touch tmp$$  #睡眠60秒，以便向当前进程发送信号  sleep 60 |

## sed简单用法

|  |
| --- |
| sed 是一个精简的、非交互式的流式编辑器，它在命令行中输入编辑命令和指定文件名，然后在屏幕上查看输出 |
| **逐行**读取文件内容存储在临时**缓冲区**中，称为“模式空间”（pattern space），接着用**sed命令**处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。原文件内容并没有改变。 |

### 用法

|  |
| --- |
|  |
| 1.sed ‘4,$d’ test.in  2.sed ‘3q’ test.in  3.sed ‘s/public/PUBLIC/’ test.in  4.sed -n ‘s/public/PUBLIC/p’ test.in |

### 指定命令的方法

|  |
| --- |
| 1、用分号分隔命令  sed 's/public/PUBLIC/;s/north/NORTH/' test.in |
| 2、每个命令前放置-e  sed -e 's/public/PUBLIC/' -e 's/north/NORTH/' test.in |
| 3、使用分行命令功能，在输入单引号后按return键就会出现多行提示符（>）  sed '  > s/public/PUBLIC/  > s/north/NORTH/' test.in |

### 脚本中使用命令

|  |
| --- |
| test.sed文件内容如下：  s/public/PUBLIC/  s/north/NORTH/  sed -f test.sed test.in |

### 命令语法

|  |
| --- |
| **[address]sed\_edit\_cmd** (通常用单引号括起来)，其中 **address** 为 **sed** 的行定位模式，用于指定将要被  **sed** 编辑的行。如果省略，**sed** 将编辑所有的行。**sed\_edit\_cmd** 为 **sed** 对被编辑行将要进行的**编辑操作**。 |

#### 定义语法

|  |
| --- |
|  |

#### 编辑命令

|  |
| --- |
| p ：打印匹配行  sed -n '3,5p' test.in // ('3,5!p')  sed -n '$p' test.in  sed -n '/north/p' test.in |
| **=** ：显示匹配行的行号  sed -n '/north/=' test.in |
| d ：删除匹配的行  sed '/north/d' test.in |
| 在指定行后面追加一行或多行文本，并显示添加的新内容，该命令主要用于 **sed** 脚本中。  sed '/north/a\AAA\  >BBB\  >CCC' test |
| i\ ：在指定行前插入一行或多行，并显示添加的新内容，使用格式同 a\ |
| c\ ：用新文本替换指定的行，使用格式同 a\ |
| l ：显示指定行中所有字符，包括控制字符(非打印字符)  sed -n '/north/l' test.in |
| s ：替换命令，使用格式为：  **[address]s/old/new/[gpw]**  **address** ：如果省略，表示编辑所有的行。  **g** ：全局替换  **p** ：打印被修改后的行  **w fname** ：将被替换后的行内容写到指定的文件中  **sed -n 's/north/NORTH/gp' test.in**  **sed -n 's/north/NORTH/w data' test.in**  **sed 's/[0-9][0-9]$/&.5/' datafile**  **& 符号用在替换字符串中时，代表 被替换的字符串** |
| r ：读文件，将另外一个文件中的内容附加到指定行后。  **sed '$r data' test.in** |
| **w** ：写文件，将指定行写入到另外一个文件中。  **sed -n '/public/w data2' test.in** |
| n ：将指定行的下面一行读入编辑缓冲区。  **sed -n '/public/{n;s/north/NORTH/p}' test.in**  对指定行同时使用多个 sed 编辑命令时，需用大括号 “ {} ” 括起来，命令之间用分号 “ ; ” 格开。注意与 -e 选项的区别 |
| **q** ：退出，读取到指定行后退出 **sed**  **sed '3q' test.in** |

### 命令小结

|  |
| --- |
|  |

### sed命令示例

|  |
| --- |
| 以下两个命令都能显式vsftpd.conf中的配置行  grep -v '^#' /etc/vsftpd/vsftpd.conf  cat /etc/vsftpd/vsftpd.conf | sed 's/^#.\*$//g'| sed '/^$/d' |