Ubuntu分辨率设置：

终端输入xrandr，显示虚拟机可以设置的分辨率以及刷新率。

设置分辨率命令行:

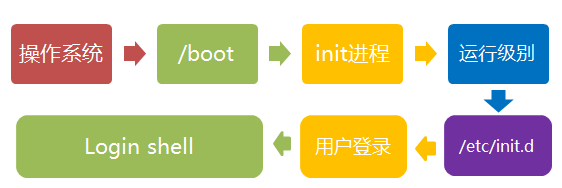
xrandr -s 1024x768

设置->设备->显示进行设置分辨率

# Linux 系统启动过程

过程可以分为5个阶段：

* 内核的引导。
* 运行 init。
* 系统初始化。
* 建立终端 。
* 用户登录系统。



正确的关机流程为：sync > shutdown > reboot > halt

不管是重启系统还是关闭系统，首先要运行 sync 命令，把内存中的数据写到磁盘中。

关机的命令有 shutdown –h now halt poweroff 和 init 0 , 重启系统的命令有 shutdown –r now reboot init 6。

# Linux 系统目录结构

登录系统后，在当前命令窗口下输入命令：

 ls /

你会看到如下图所示:

[IMG_256](http://www.runoob.com/wp-content/uploads/2014/06/4_20.png)

在linux系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。

/etc： 上边也提到了，这个是系统中的配置文件，如果你更改了该目录下的某个文件可能会导致系统不能启动。

/bin, /sbin, /usr/bin, /usr/sbin: 这是系统预设的执行文件的放置目录，比如 ls 就是在/bin/ls 目录下的。

值得提出的是，/bin, /usr/bin 是给系统用户使用的指令（除root外的通用户），而/sbin, /usr/sbin 则是给root使用的指令。

/var： 这是一个非常重要的目录，系统上跑了很多程序，那么每个程序都会有相应的日志产生，而这些日志就被记录到这个目录下，具体在/var/log 目录下，另外mail的预设放置也是在这里。

# Linux 文件基本属性

[root@www /]# ls -l

total 64

dr-xr-xr-x 2 root root 4096 Dec 14 2012 bin



#### 1、chgrp：更改文件属组

语法：

chgrp [-R] 属组名 文件名

#### 2、chown：更改文件属主，也可以同时更改文件属组

语法：

chown [–R] 属主名 文件名

chown [-R] 属主名：属组名 文件名

#### 3、chmod：更改文件9个属性

数字来代表各个权限，各权限的分数对照表如下：

* r:4
* w:2
* x:1

# Linux 文件与目录管理

### **ls (列出目录)**

选项与参数：

* -a ：全部的文件，连同隐藏档( 开头为 . 的文件) 一起列出来(常用)
* -d ：仅列出目录本身，而不是列出目录内的文件数据(常用)
* -l ：长数据串列出，包含文件的属性与权限等等数据；(常用)

### **mkdir (创建新目录)**

如果想要创建新的目录的话，那么就使用mkdir (make directory)吧。

语法：

mkdir [-mp] 目录名称

选项与参数：

* -m ：配置文件的权限喔！直接配置，不需要看默认权限 (umask) 的脸色～
* -p ：帮助你直接将所需要的目录(包含上一级目录)递归创建起来

### **rmdir (删除空的目录)**

语法：

rmdir [-p] 目录名称

选项与参数：

* **-p ：**连同上一级『空的』目录也一起删除

不过要注意的是，这个 rmdir 仅能删除空的目录，你可以使用 rm 命令来删除非空目录。

### **rm (移除文件或目录)**

语法：

rm [-fir] 文件或目录

选项与参数：

* -f ：就是 force 的意思，忽略不存在的文件，不会出现警告信息；
* -i ：互动模式，在删除前会询问使用者是否动作
* -r ：递归删除啊！最常用在目录的删除了！这是非常危险的选项！！

<https://www.shiyanlou.com/courses/1>

基本概念及操作

### 3.2 命令行操作体验

| **按键** | **作用** |
| --- | --- |
| Ctrl+d | 键盘输入结束或退出终端 |
| Ctrl+s | 暂停当前程序，暂停后按下任意键恢复运行 |
| Ctrl+z | 将当前程序放到后台运行，恢复到前台为命令fg |
| Ctrl+a | 将光标移至输入行头，相当于Home键 |
| Ctrl+e | 将光标移至输入行末，相当于End键 |
| Ctrl+k | 删除从光标所在位置到行末 |
| Alt+Backspace | 向前删除一个单词 |
| Shift+PgUp | 将终端显示向上滚动 |
| Shift+PgDn | 将终端显示向下滚动 |

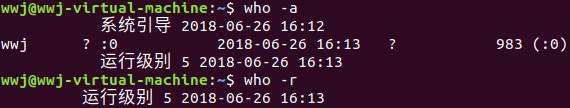
# 用户及文件权限管理

## 二、Linux 用户管理

### 2.1 查看用户

who 命令常用参数

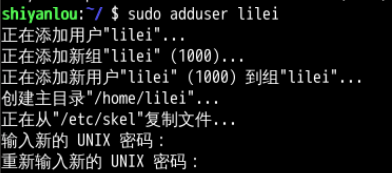
| **参数** | **说明** |
| --- | --- |
| -a | 打印能打印的全部 |
| -d | 打印死掉的进程 |
| -m | 同am i,mom likes |
| -q | 打印当前登录用户数及用户名 |
| -u | 打印当前登录用户登录信息 |
| -r | 打印运行等级 |



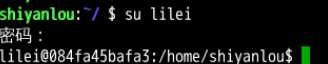
### 2.2 创建用户

sudo <cmd> 可以以特权级别运行 cmd 命令，需要当前用户属于 sudo 组，且需要输入当前用户的密码。

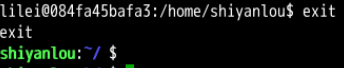
新建一个叫 lilei 的用户：



su <user> 可以切换到用户 user，执行时需要输入目标用户的密码：



退出当前用户跟退出终端一样可以使用 exit 命令或者使用快捷键 Ctrl+d：



#### adduser 和 useradd 的区别是什么？

答：useradd 只创建用户，创建完了用 passwd lilei 去设置新用户的密码。adduser 会创建用户，创建目录，创建密码（提示你设置），做这一系列的操作。

### 2.3 用户组

查看自己属于哪些用户组：

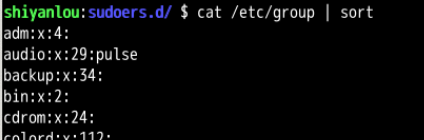
#### 方法一：使用 groups 命令

其中冒号之前表示用户，后面表示该用户所属的用户组。每次新建用户如果不指定用户组的话，默认会自动创建一个与用户名相同的用户组。



#### 方法二：查看 /etc/group 文件

cat 命令用于读取指定文件的内容并打印到终端输出，后面会详细讲它的使用。 | sort 表示将读取的文本进行一个字典排序再输出



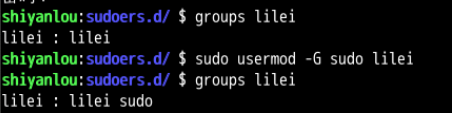


/etc/group 的内容包括用户组（Group）、用户组口令、GID 及该用户组所包含的用户（User），每个用户组一条记录。格式如下：

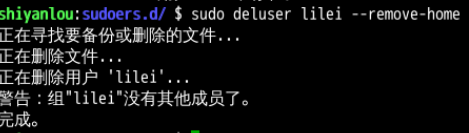
group\_name:password:GID:user\_list

你看到上面的 password 字段为一个 x 并不是说密码就是它，只是表示密码不可见而已。

使用 usermod 命令可以为用户添加用户组，这里用 shiyanlou 用户执行 sudo 命令将 lilei 添加到 sudo 用户组

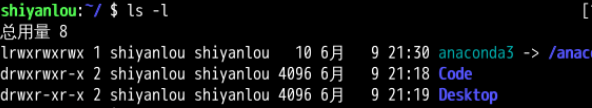


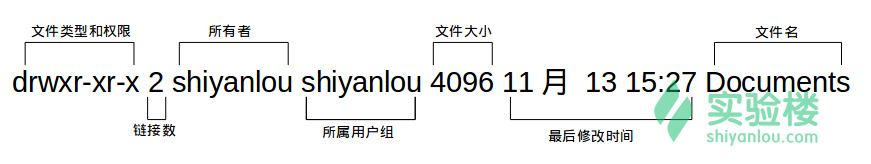
### 2.4 删除用户



## 三、Linux 文件权限

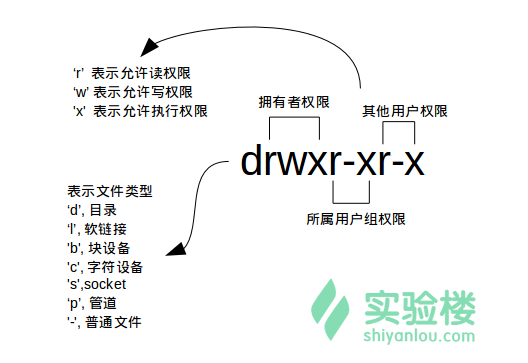
### 3.1 查看文件权限





* 文件类型

**Linux 里面一切皆文件**，软链接等同于 Windows 上的快捷方式



* 文件权限

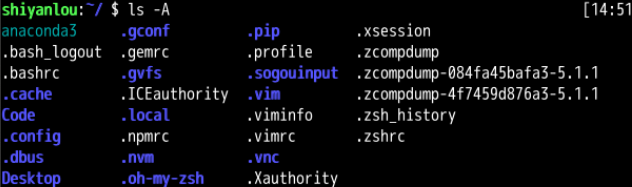
执行权限，通常指可以运行的二进制程序文件或者脚本文件，如同 Windows 上的 exe 后缀的文件，不过 Linux 上不是通过文件后缀名来区分文件的类型。**一个目录同时具有读权限和执行权限才可以打开并查看内部文件，而一个目录要有写权限才允许在其中创建其它文件**

* 链接数

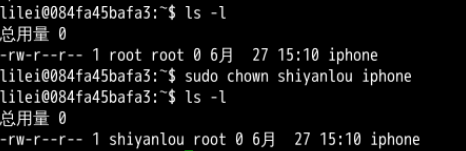
链接到该文件所在的 inode 结点的文件名数目

ls -a显示所有文件，包括隐藏文件（Linux 下以 . 开头的文件为隐藏文件）

ls -A显示除了 .（当前目录）和 ..（上一级目录）之外的所有文件，包括隐藏文件（Linux 下以 . 开头的文件为隐藏文件）。

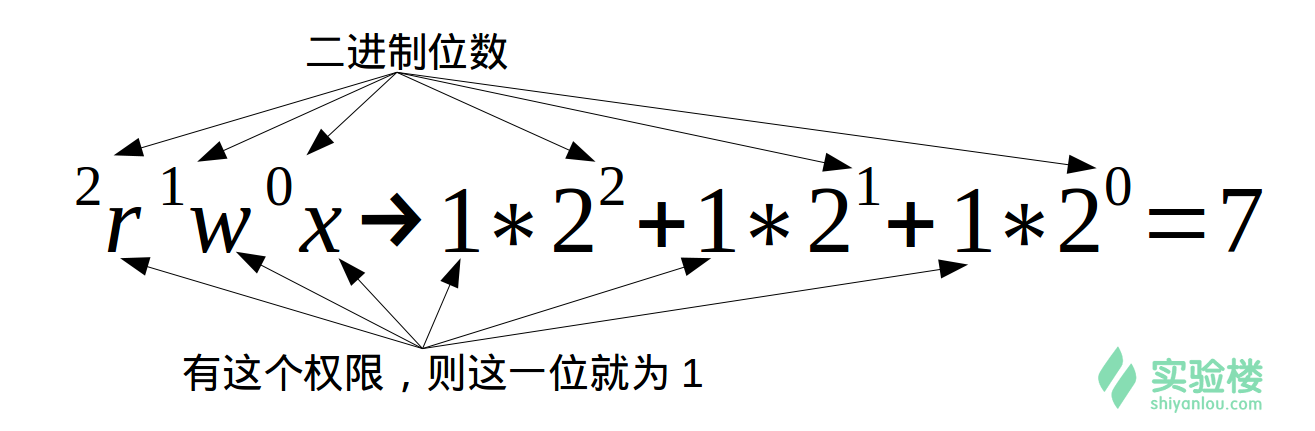


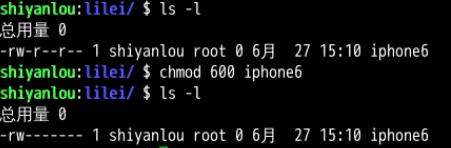
### 3.2 变更文件所有者



### 3.3 修改文件权限

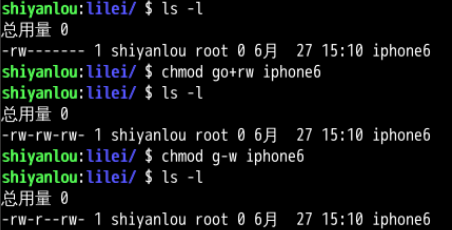
* 方式一：二进制数字表示





* 方式二：加减赋值操作

g、o 还有 u 分别表示 group、others 和 user，+ 和 - 分别表示增加和去掉相应的权限。



# Linux 目录结构及文件基本操作

## 二、Linux 目录结构

### Linux 的目录与 Windows 的目录的区别，一种不同是体现在目录与存储介质（磁盘，内存，DVD 等）的关系上，以往的 Windows 一直是以存储介质为主的，Linux是以目录为主的。可以理解为树形目录是一个用户可操作系统的骨架。

### 1. FHS 标准

FHS（英文：Filesystem Hierarchy Standard 中文：文件系统层次结构标准），多数 Linux 版本采用这种文件组织形式，FHS 定义了系统中每个区域的用途、所需要的最小构成的文件和目录同时还给出了例外处理与矛盾处理。

FHS 定义了两层规范，第一层是， / 下面的各个目录应该要放什么文件数据，例如 /etc 应该放置设置文件，/bin 与 /sbin 则应该放置可执行文件等等。

第二层则是针对 /usr 及 /var 这两个目录的子目录来定义。例如 /var/log 放置系统日志文件，/usr/share 放置共享数据等等。

可用以下命令查看目录树：

$ tree /

### 2. 目录路径

#### 绝对路径

关于绝对路径，简单地说就是以根" / "目录为起点的完整路径，以你所要到的目录为终点，表现形式如： /usr/local/bin

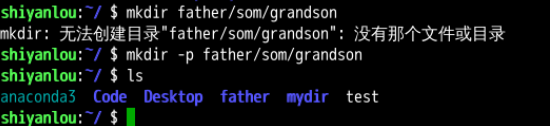
**提示：在进行目录切换的过程中请多使用** Tab **键自动补全，可避免输入错误，连续按两次** Tab **可以显示全部候选结果。**

## 三、Linux 文件的基本操作

cd ~ 切换回用户的 /home/shiyanlou 目录

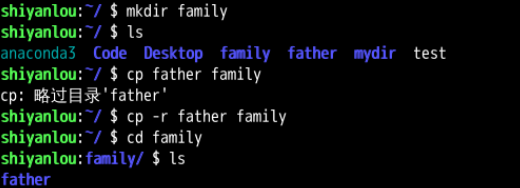
#### 新建目录

使用 -p 参数，同时创建父目录（如果不存在该父目录）



#### 复制目录

要成功复制目录需要加上 -r 或者 -R 参数，表示递归复制



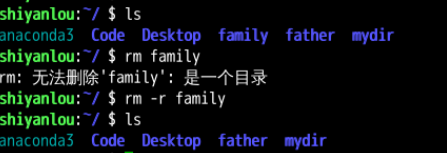
#### 删除文件

遇到想要删除一些为只读权限的文件，直接使用 rm 删除会显示一个提示，可以使用 -f 参数强制删除



#### 删除目录

跟复制目录一样，要删除一个目录，也需要加上 -r 或 -R 参数

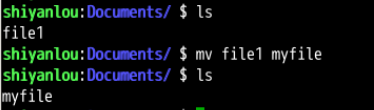


#### 移动文件

mv 源目录文件 目的目录

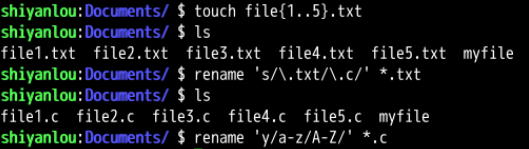
#### 重命名文件

mv 旧的文件名 新的文件名



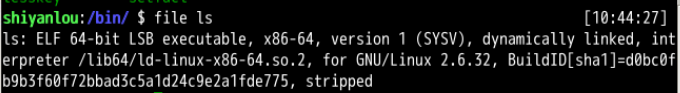
#### 批量重命名

rename 是先使用第二个参数的通配符匹配所有后缀为 .txt 的文件，然后使用第一个参数提供的正则表达式将匹配的这些文件的 .txt 后缀替换为 .c



### 查看文件类型

在 Linux 中文件的类型不是根据文件后缀来判断的，我们通常使用 file 命令查看文件的类型



# 环境变量与文件查找

## 二、环境变量

#### 变量

使用 declare 命令创建一个变量名为 tmp 的变量：

$ declare tmp

其实也可以不用 declare 预声明一个变量，直接即用即创建.

### 1. 环境变量

环境变量的作用域比自定义变量的要大，如 Shell 的环境变量作用于自身和它的子进程。在所有的 UNIX 和类 UNIX 系统中，每个进程都有其各自的环境变量设置，且默认情况下，当一个进程被创建时，除了创建过程中明确指定的话，它将继承其父进程的绝大部分环境设置。Shell 程序也作为一个进程运行在操作系统之上，而我们在 Shell 中运行的大部分命令都将以 Shell 的子进程的方式运行。

通常我们会涉及到的变量类型有三种：

* 当前 Shell 进程私有用户自定义变量，如上面我们创建的 tmp 变量，只在当前 Shell 中有效。
* Shell 本身内建的变量。
* 从自定义变量导出的环境变量。

也有三个与上述三种环境变量相关的命令：set，env，export。这三个命令很相似，都是用于打印环境变量信息，区别在于涉及的变量范围不同。详见下表：

| **命 令** | **说 明** |
| --- | --- |
| set | 显示当前 Shell 所有变量，包括其内建环境变量（与 Shell 外观等相关），用户自定义变量及导出的环境变量。 |
| env | 显示与当前用户相关的环境变量，还可以让命令在指定环境中运行。 |
| export | 显示从 Shell 中导出成环境变量的变量，也能通过它将自定义变量导出为环境变量。 |

你可以更直观的使用 vimdiff 工具比较一下它们之间的差别：

$ temp=shiyanlou

$ export temp\_env=shiyanlou

$ env|sort>env.txt

$ export|sort>export.txt

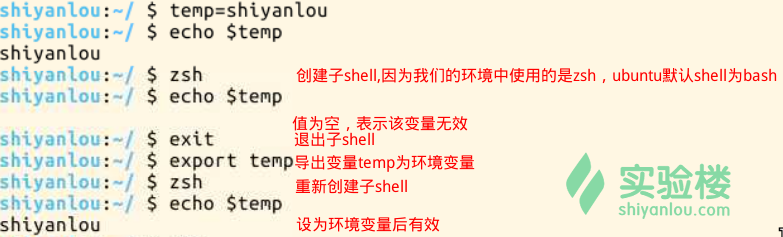
$ set|sort>set.txt

上述操作将命令输出通过管道 | 使用 sort 命令排序，再重定向到对象文本文件中。

$ vimdiff env.txt export.txt set.txt

使用 vimdiff 工具比较导出的几个文件的内容.



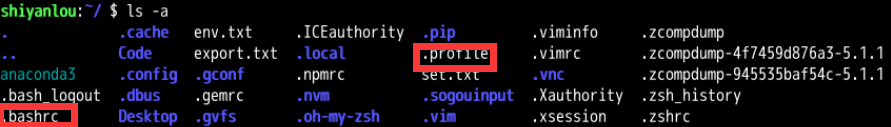
可以简单地理解成在当前进程的子进程有效则为环境变量，否则不是（有些人也将所有变量统称为环境变量，只是以全局环境变量和局部环境变量进行区分，我们只要理解它们的实质区别即可）。我们这里用 export 命令来体会一下，先在 Shell 中设置一个变量 temp=shiyanlou，然后再新创建一个子 Shell 查看 temp 变量的值：

按变量的生存周期来划分，Linux 变量可分为两类：

永久的：需要修改配置文件，变量永久生效；

临时的：使用 export 命令行声明即可，变量在关闭 shell 时失效。

两个重要文件 /etc/bashrc（有的 Linux 没有这个文件） 和 /etc/profile ，它们分别存放的是 shell 变量和环境变量。还有要注意区别的是每个用户目录下的一个隐藏文件：



这个 .profile 只对当前用户永久生效。而写在 /etc/profile 里面的是对所有用户永久生效，所以如果想要添加一个永久生效的环境变量，只需要打开 /etc/profile，在最后加上你想添加的环境变量.

### 2. 命令的查找路径与顺序

环境变量PATH 里面就保存了 Shell 中执行的命令的搜索路径。

查看 PATH 环境变量的内容：

$ echo $PATH

默认情况下你会看到如下输出：

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

创建一个 C 语言“ hello world ”程序：

$ gedit hello\_world.c

#include <stdio.h>

int main(void){

printf("hello world!\n");

return 0;

}

使用 gcc 生成可执行文件：

$ gcc -o hello\_world hello\_world.c

**gcc 生成二进制文件默认具有可执行权限，不需要修改**

### 3. 添加自定义路径到“ PATH ”环境变量

PATH 里面的路径是以 : 作为分割符的，所以我们可以这样添加自定义路径：

$ PATH=$PATH:/home/shiyanlou/mybin

**注意这里一定要使用绝对路径。**

给 PATH 环境变量追加了一个路径，它也只是在当前 Shell 有效，我一旦退出终端，再打开就会发现又失效了。

在每个用户的 home 目录中有一个 Shell 每次启动时会默认执行一个配置脚本，以初始化环境，包括添加一些用户自定义环境变量等等。zsh 的配置文件是 .zshrc，相应 Bash 的配置文件为 .bashrc 。它们在 etc 下还都有一个或多个全局的配置文件，不过我们一般只修改用户目录下的配置文件。

我们可以简单地使用下面命令直接添加内容到 .zshrc 中：

$ echo "PATH=$PATH:/home/shiyanlou/mybin" >> .zshrc

**上述命令中** >> **表示将标准输出以追加的方式重定向到一个文件中，注意前面用到的** > **是以覆盖的方式重定向到一个文件中，使用的时候一定要注意分辨。在指定文件不存在的情况下都会创建新的文件。**

### 4. 修改和删除已有变量

#### 变量删除

可以使用 unset 命令删除一个环境变量：

$ unset temp

### 5. 如何让环境变量立即生效

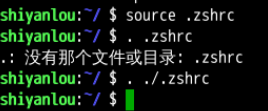
可以使用 source 命令来让其立即生效，如：

$ source .zshrc

source 命令还有一个别名就是 .，注意与表示当前路径的那个点区分开，虽然形式不一样，但作用和使用方式一样，上面的命令如果替换成 . 的方式就该是：

$ . ./.zshrc

注意第一个点后面有一个空格，而且后面的文件必须指定完整的绝对或相对路径名，source 则不需要。



## 三、搜索文件

与搜索相关的命令常用的有 whereis，which，find 和 locate 。

* whereis **简单快速**

这个搜索很快，因为它并没有从硬盘中依次查找，而是直接从数据库中查询。whereis 只能搜索二进制文件(-b)，man 帮助文件(-m)和源代码文件(-s)。

* locate **快而全**

通过“ /var/lib/mlocate/mlocate.db ”数据库查找，不过这个数据库也不是实时更新的，系统会使用定时任务每天自动执行 updatedb 命令更新一次，所以有时候你刚添加的文件，它可能会找不到，需要手动执行一次 updatedb 命令（在我们的环境中必须先执行一次该命令）。它可以用来查找指定目录下的不同文件类型，如查找 /etc 下所有以 sh 开头的文件：

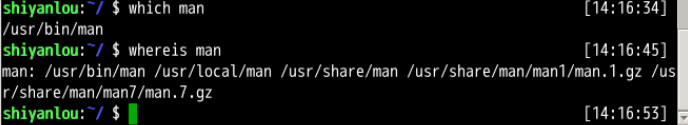
$ sudo apt-get install locate

$ locate /etc/sh

**注意，它不只是在 /etc 目录下查找，还会自动递归子目录进行查找。**

* which **小而精**

which 本身是 Shell 内建的一个命令，我们通常使用 which 来确定是否安装了某个指定的软件，因为它只从 PATH 环境变量指定的路径中去搜索命令：



* find **精而细**

find 应该是这几个命令中最强大的了，它不但可以通过文件类型、文件名进行查找而且可以根据文件的属性（如文件的时间戳，文件的权限等）进行搜索。

find 的第一个参数是要搜索的地方：

$ sudo find /etc/ -name interfaces

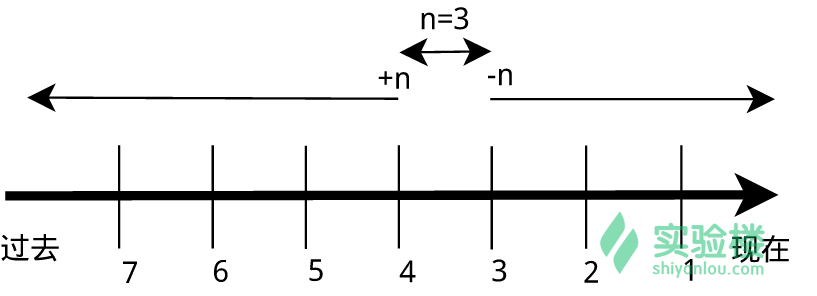
**注意 find 命令的路径是作为第一个参数的， 基本命令格式为 find [path] [option] [action] 。**

与时间相关的命令参数：

| **参数** | **说明** |
| --- | --- |
| -atime | 最后访问时间 |
| -ctime | 最后修改文件内容的时间 |
| -mtime | 最后修改文件属性的时间 |

下面以 -mtime 参数举例：

* -mtime n：n 为数字，表示为在 n 天之前的“一天之内”修改过的文件
* -mtime +n：列出在 n 天之前（不包含 n 天本身）被修改过的文件
* -mtime -n：列出在 n 天之内（包含 n 天本身）被修改过的文件
* -newer file：file 为一个已存在的文件，列出比 file 还要新的文件名



列出 home 目录中，当天（24 小时之内）有改动的文件：

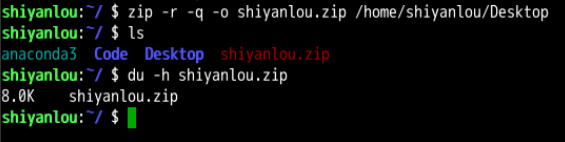
$ find ~ -mtime 0

# 文件打包与压缩

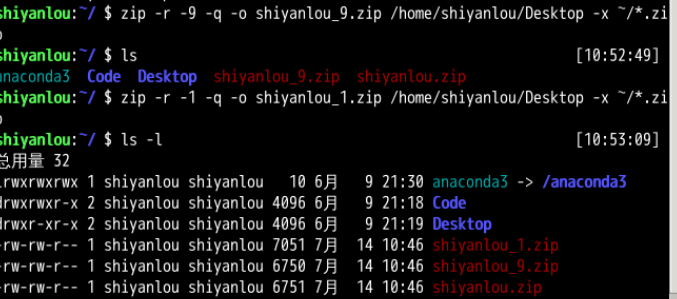
### 3.1 zip 压缩打包程序

* 使用 zip 打包文件夹：

上面命令将目录 /home/shiyanlou/Desktop 打包成一个文件，并查看了打包后文件的大小和类型。第一行命令中，-r 参数表示递归打包包含子目录的全部内容，-q 参数表示为安静模式，即不向屏幕输出信息，-o，表示输出文件，需在其后紧跟打包输出文件名。后面使用 du 命令查看打包后文件的大小.



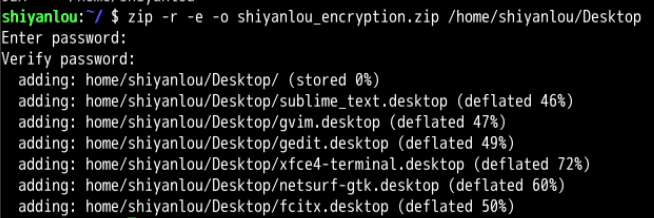
设置压缩级别为 9 和 1（9 最大，1 最小），重新打包：



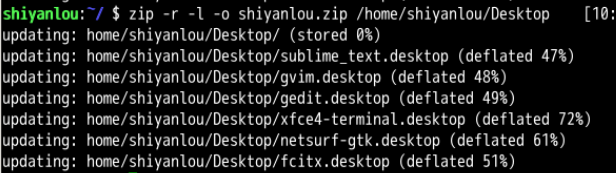
这里添加了一个参数用于设置压缩级别 -[1-9]，1 表示最快压缩但体积大，9 表示体积最小但耗时最久。最后那个 -x 是为了排除我们上一次创建的 zip 文件，否则又会被打包进这一次的压缩文件中，**注意：这里只能使用绝对路径，否则不起作用**。

* 创建加密 zip 包

使用 -e 参数可以创建加密压缩包：



在 Windows 为 CR+LF（Carriage-Return+Line-Feed：回车加换行），而在 Linux/Unix 上为 LF（换行），所以如果在不加处理的情况下，在 Linux 上编辑的文本，在 Windows 系统上打开可能看起来是没有换行的。如果你想让你在 Linux 创建的 zip 压缩文件在 Windows 上解压后没有任何问题，那么你还需要对命令做一些修改：



需要加上 -l 参数将 LF 转换为 CR+LF 来达到以上目的。

### 3.2 使用 unzip 命令解压缩 zip 文件

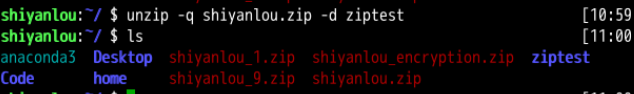
将 shiyanlou.zip 解压到当前目录：

$ unzip shiyanlou.zip

使用安静模式，将文件解压到指定目录：

$ unzip -q shiyanlou.zip -d ziptest

上述指定目录不存在，将会自动创建。



**注意：** 使用 unzip 解压文件时我们同样应该注意兼容问题，不过这里我们关心的不再是上面的问题，而是中文编码的问题，通常 Windows 系统上面创建的压缩文件，如果有有包含中文的文档或以中文作为文件名的文件时默认会采用 GBK 或其它编码，而 Linux 上面默认使用的是 UTF-8 编码，如果不加任何处理，直接解压的话可能会出现中文乱码的问题（有时候它会自动帮你处理），为了解决这个问题，我们可以在解压时指定编码类型。

使用 -O（英文字母，大写 o）参数指定编码类型：

unzip -O GBK 中文压缩文件.zip

### 3.3 tar 打包工具

在 Linux 上面更常用的是 tar 工具，tar 原本只是一个打包工具，只是同时还是实现了对 7z、gzip、xz、bzip2 等工具的支持，这些压缩工具本身只能实现对文件或目录（单独压缩目录中的文件）的压缩，没有实现对文件的打包压缩，所以我们也无需再单独去学习其他几个工具，tar 的解压和压缩都是同一个命令，只需参数不同，使用比较方便。

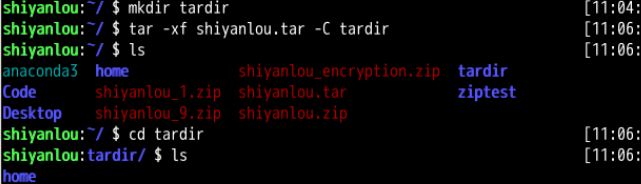
下面先掌握 tar 命令一些基本的使用方式，即不进行压缩只是进行打包（创建归档文件）和解包的操作。

* 创建一个 tar 包：

$ tar -cf shiyanlou.tar /home/shiyanlou/Desktop

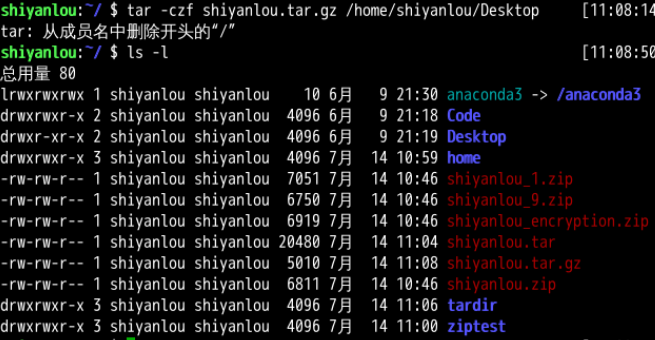
上面命令中，-c 表示创建一个 tar 包文件，-f 用于指定创建的文件名，注意文件名必须紧跟在 -f 参数之后，比如不能写成 tar -fc shiyanlou.tar，可以写成 tar -f shiyanlou.tar -c ~。

解包一个文件（-x 参数）到指定路径的**已存在**目录（-C 参数）：



对于创建不同的压缩格式的文件，对于 tar 来说是相当简单的，需要的只是换一个参数，这里我们就以使用 gzip 工具创建 \*.tar.gz 文件为例来说明。

* 我们只需要在创建 tar 文件的基础上添加 -z 参数，使用 gzip 来压缩文件：



* 解压 \*.tar.gz 文件：

$ tar -xzf shiyanlou.tar.gz

要使用其它的压缩工具创建或解压相应文件只需要更改一个参数即可：

| **压缩文件格式** | **参数** |
| --- | --- |
| \*.tar.gz | -z |
| \*.tar.xz | -J |
| \*tar.bz2 | -j |

其实平常使用的参数并没有那么复杂，只需要记住常用的组合就可以了。 常用命令：

* zip：
  + 打包 ：zip something.zip something （目录请加 -r 参数）
  + 解包：unzip something.zip
  + 指定路径：-d 参数
* tar：
  + 打包：tar -zcvf something.tar something
  + 解包：tar -zxvf something.tar
  + 指定路径：-C 参数

# 文件系统操作与磁盘管理

#### 2.1 查看磁盘和目录的容量

##### 使用 df 命令查看磁盘的容量

实际的物理主机上会更像这样：



物理主机上的 /dev/sda2 是对应着主机硬盘的分区，后面的数字表示分区号，数字前面的字母 a 表示第几块硬盘（也可能是可移动磁盘），你如果主机上有多块硬盘则可能还会出现 /dev/sdb，/dev/sdc 这些磁盘设备都会在 /dev 目录下以文件的存在形式。

##### 使用 du 命令查看目录的容量

这个命令前面其实已经用了很多次了：

# 默认同样以 块 的大小展示$ du # 加上`-h`参数，以更易读的方式展示$ du -h

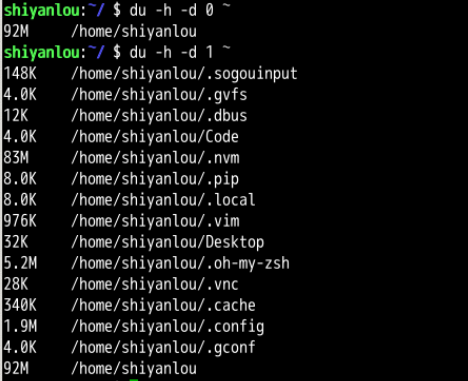
-d参数指定查看目录的深度

# 只查看1级目录的信息

$ du -h -d 0 ~

# 查看2级

$ du -h -d 1 ~



常用参数

du -h #同--human-readable 以K，M，G为单位，提高信息的可读性。

du -a #同--all 显示目录中所有文件的大小。

du -s #同--summarize 仅显示总计，只列出最后加总的值。

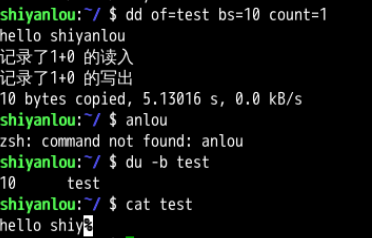
## 三、简单的磁盘管理

### 3.1 创建虚拟磁盘

#### dd 命令简介（部分说明来自[dd (Unix) wiki](http://zh.wikipedia.org/wiki/Dd_(Unix" \t "https://www.shiyanlou.com/courses/1/labs/62/_blank))）

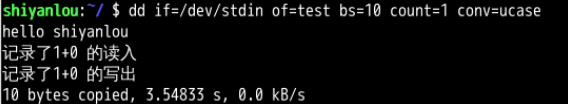
dd命令用于转换和复制文件，不过它的复制不同于cp。

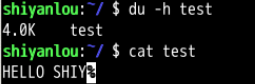
dd的命令行语句与其他的 Linux 程序不同，因为它的命令行选项格式为选项=值，而不是更标准的--选项 值或-选项=值。dd默认从标准输入中读取，并写入到标准输出中，但可以用选项if（input file，输入文件）和of（output file，输出文件）改变。

用dd命令从标准输入读入用户的输入到标准输出或者一个文件中：

上述命令从标准输入设备读入用户输入（缺省值，所以可省略）然后输出到 test 文件，bs（block size）用于指定块大小（缺省单位为 Byte，也可为其指定如'K'，'M'，'G'等单位），count用于指定块数量。如上图所示，我指定只读取总共 10 个字节的数据，当我输入了“hello shiyanlou”之后加上空格回车总共 16 个字节（一个英文字符占一个字节）内容，显然超过了设定大小。使用du和cat命令看到的写入完成文件实际内容确实只有 10 个字节（那个黑底百分号表示这里没有换行符）,而其他的多余输入将被截取并保留在标准输入。

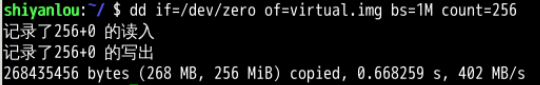
dd在拷贝的同时还可以实现数据转换，那下面就举一个简单的例子：将输出的英文字符转换为大写再写入文件：





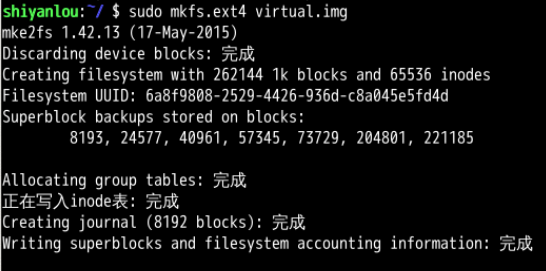
#### 使用 dd 命令创建虚拟镜像文件

从/dev/zero设备创建一个容量为 256M 的空文件：



#### 使用 mkfs 命令格式化磁盘（我们这里是自己创建的虚拟磁盘镜像）

你可以在命令行输入 sudo mkfs 然后按下Tab键，你可以看到很多个以 mkfs 为前缀的命令，这些不同的后缀其实就是表示着不同的文件系统，可以用 mkfs 格式化成的文件系统。

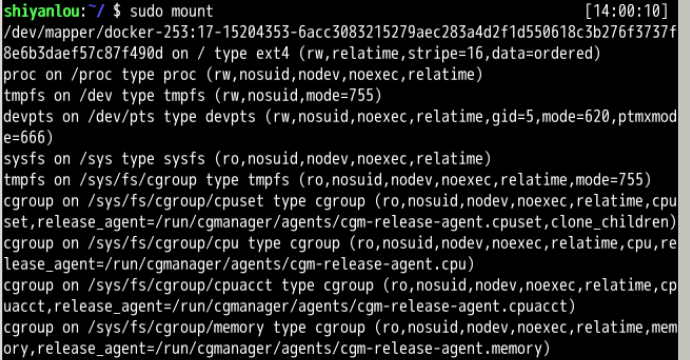
我们可以简单的使用下面的命令来将我们的虚拟磁盘镜像格式化为ext4文件系统：

可以看到实际 mkfs.ext4 是使用 mke2fs 来完成格式化工作的。

#### 使用 mount 命令挂载磁盘到目录树

mount 指令是告诉操作系统，对应的文件系统已经准备好，可以使用了，而该文件系统会对应到一个特定的点（称为挂载点）。挂载好的文件、目录、设备以及特殊文件即可提供用户使用。

我们先来使用mount来查看下主机已经挂载的文件系统：



如何挂载真正的磁盘到目录树呢，mount命令的一般格式如下：

mount [options] [source] [directory]

一些常用操作：

mount [-o [操作选项]] [-t 文件系统类型] [-w|--rw|--ro] [文件系统源] [挂载点]

现在直接来挂载我们创建的虚拟磁盘镜像到/mnt目录：

$ mount -o loop -t ext4 virtual.img /mnt

# 也可以省略挂载类型，很多时候 mount 会自动识别

# 以只读方式挂载

$ mount -o loop --ro virtual.img /mnt

# 或者mount -o loop,ro virtual.img /mnt



#### 使用 umount 命令卸载已挂载磁盘



# 帮助命令

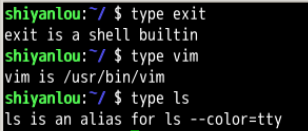
## 二、内建命令与外部命令

有一些查看帮助的工具在内建命令与外建命令上是有区别对待的。

**内建命令**实际上是 shell 程序的一部分，其中包含的是一些比较简单的 Linux 系统命令，这些命令是写在bash源码的builtins里面的，由 shell 程序识别并在 shell 程序内部完成运行，通常在 Linux 系统加载运行时 shell 就被加载并驻留在系统内存中。而且解析内部命令 shell 不需要创建子进程，因此其执行速度比外部命令快。比如：history、cd、exit 等等。

**外部命令**是 Linux 系统中的实用程序部分，因为实用程序的功能通常都比较强大，所以其包含的程序量也会很大，在系统加载时并不随系统一起被加载到内存中，而是在需要时才将其调入内存。虽然其不包含在 shell 中，但是其命令执行过程是由 shell 程序控制的。外部命令是在 Bash 之外额外安装的，通常放在/bin，/usr/bin，/sbin，/usr/sbin等等。比如：ls、vim等。

可以使用　type 命令来区分命令是内建的还是外部的。



### 1. help 命令

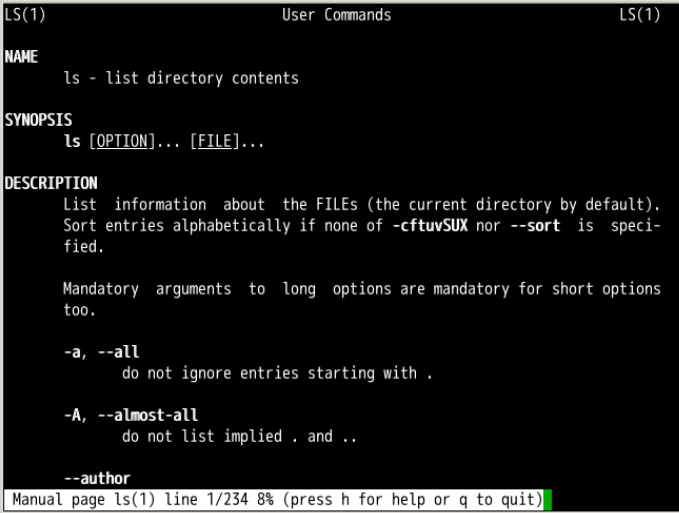
help 命令是用于显示 shell 内建命令的简要帮助信息。外部命令基本上都有一个参数--help,这样就可以得到相应的帮助.



### 2. man 命令

得到的内容比用 help 更多更详细，而且man没有内建与外部命令的区分，因为 man 工具是显示系统手册页中的内容，也就是一本电子版的字典.





最左上角显示“ LS （1）”，在这里，“ LS ”表示手册名称，而“（1）”表示该手册位于第一章节。这个章节又是什么？在 man 手册中一共有这么几个章节

| **章节数** | **说明** |
| --- | --- |
| 1 | Standard commands （标准命令） |
| 2 | System calls （系统调用） |
| 3 | Library functions （库函数） |
| 4 | Special devices （设备说明） |
| 5 | File formats （文件格式） |
| 6 | Games and toys （游戏和娱乐） |
| 7 | Miscellaneous （杂项） |
| 8 | Administrative Commands （管理员命令） |
| 9 | 其他（Linux特定的）， 用来存放内核例行程序的文档。 |

打开手册之后我们可以通过 pgup 与 pgdn 或者上下键来上下翻看，可以按 q 退出当前页面

### 3. info 命令



得到的信息比 man 还要多了，info 来自自由软件基金会的 GNU 项目，是 GNU 的超文本帮助系统，能够更完整的显示出 GNU 信息。man 和 info 就像两个集合，它们有一个交集部分，但与 man 相比，info 工具可显示更完整的　GNU　工具信息。若 man 页包含的某个工具的概要信息在 info 中也有介绍，那么 man 页中会有“请参考 info 页更详细内容”的字样。

# 任务计划crontab

## 二、crontab 的使用

crontab 命令常见于 Unix 和类 Unix 的操作系统之中（Linux 就属于类 Unix 操作系统），用于设置周期性被执行的指令。

### 2.1 crontab 简介

crontab 命令从输入设备读取指令，并将其存放于 crontab 文件中，以供之后读取和执行。通常，crontab 储存的指令被守护进程激活，crond 为其守护进程，crond 常常在后台运行，每一分钟会检查一次是否有预定的作业需要执行。

通过 crontab 命令，我们可以在固定的间隔时间执行指定的系统指令或 shell　script 脚本。时间间隔的单位可以是分钟、小时、日、月、周的任意组合。

# Example of job definition:# .---------------- minute (0 - 59)# | .------------- hour (0 - 23)# | | .---------- day of month (1 - 31)# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat# | | | | |# \* \* \* \* \* user-name command to be executed

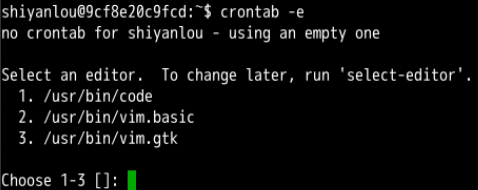
### 2.2 crontab 准备

首先我们会启动 rsyslog，以便我们可以通过日志中的信息来了解我们的任务是否真正的被执行了.



### 2.3 crontab 使用

通过下面一个命令来添加一个计划任务

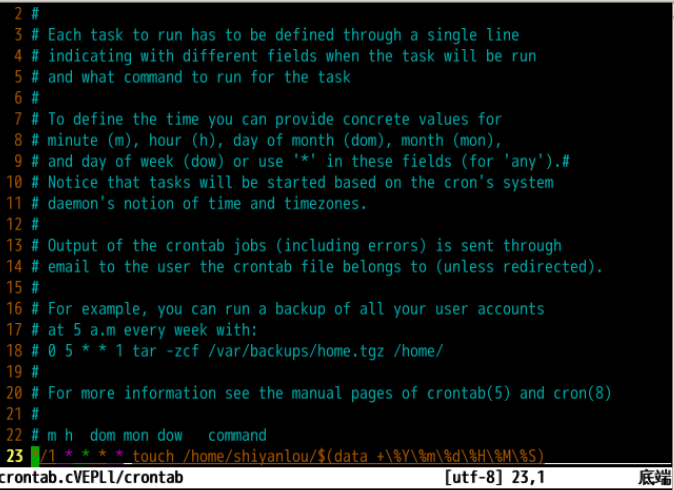


这是让我们选择编辑的工具，选择第二个基本的 vim 就可以了.

在文档的最后一排加上这样一排命令,该任务是每分钟我们会在/home/shiyanlou目录下创建一个以当前的年月日时分秒为名字的空白文件.

\*/1 \* \* \* \* touch /home/shiyanlou/$(date +\%Y\%m\%d\%H\%M\%S)

**注意** “ % ” 在 crontab 文件中，有结束命令行、换行、重定向的作用，前面加 ” \ ” 符号转义，否则，“ % ” 符号将执行其结束命令行或者换行的作用，并且其后的内容会被做为标准输入发送给前面的命令。



添加成功后退出，会有installing new crontab 的一个提示

可以通过这样的一个指令来查看我们添加了哪些任务



# vim

### 2.1 vim模式介绍

* 普通模式(Normal mode)

在普通模式中，用的编辑器命令，比如移动光标，删除文本等等。这也是Vim启动后的默认模式。这正好和许多新用户期待的操作方式相反（大多数编辑器默认模式为插入模式）。

* 插入模式(Insert mode)

在这个模式中，大多数按键都会向文本缓冲中插入文本。大多数新用户希望文本编辑器编辑过程中一直保持这个模式。

* 命令行模式(Command line mode)

在命令行模式中可以输入会被解释成并执行的文本。例如执行命令（:键），搜索（/和?键）或者过滤命令（!键）。在命令执行之后，Vim返回到命令行模式之前的模式，通常是普通模式。

### 2.2 三种常用模式的切换

vim启动进入普通模式，处于插入模式或命令行模式时只需要按Esc或者Ctrl+[(这在vim课程环境中不管用)即可进入普通模式。普通模式中按i（插入）或a（附加）键都可以进入插入模式，普通模式中按:进入命令行模式。命令行模式中输入wq回车后保存并退出vim。

### 2.3 进入vim

按Esc进入普通模式，在该模式下使用方向键或者h,j,k,l键可以移动游标。

| **按键** | **说明** |
| --- | --- |
| h | 左 |
| l | 右（小写L） |
| j | 下 |
| k | 上 |
| w | 移动到下一个单词 |
| b | 移动到上一个单词 |

### 2.4 进入插入模式

在普通模式下使用下面的键将进入插入模式，并可以从相应的位置开始输入

| **命令** | **说明** |
| --- | --- |
| i | 在当前光标处进行编辑 |
| I | 在行首插入 |
| A | 在行末插入 |
| a | 在光标后插入编辑 |
| o | 在当前行后插入一个新行 |
| O | 在当前行前插入一个新行 |
| cw | 替换从光标所在位置后到一个单词结尾的字符 |

### 2.5 保存文档

从普通模式输入:进入命令行模式，输入w回车，保存文档。输入:w 文件名可以将文档另存为其他文件名或存到其它路径下

### 2.6 退出vim

#### 2.6.1 命令行模式下退出vim

从普通模式输入:进入命令行模式，输入wq回车，保存并退出编辑

以下为其它几种退出方式：

| **命令** | **说明** |
| --- | --- |
| :q! | 强制退出，不保存 |
| :q | 退出 |
| :wq! | 强制保存并退出 |
| :w <文件路径> | 另存为 |
| :saveas 文件路径 | 另存为 |
| :x | 保存并退出 |
| :wq | 保存并退出 |

#### 2.6.2 普通模式下退出vim

普通模式下输入Shift+zz即可保存退出vim

### 2.7 删除文本

#### 2.7.1 普通模式下删除vim文本信息

进入普通模式，使用下列命令可以进行文本快速删除：

| **命令** | **说明** |
| --- | --- |
| x | 删除游标所在的字符 |
| X | 删除游标所在前一个字符 |
| Delete | 同x |
| dd | 删除整行 |
| dw | 删除一个单词（不适用中文） |
| d$或D | 删除至行尾 |
| d^ | 删除至行首 |
| dG | 删除到文档结尾处 |
| d1G | 删至文档首部 |

除此之外，你还可以在命令之前加上数字，表示一次删除多行，如：

2dd表示一次删除2行

# Vim文档编辑

### 2.1 vim重复命令

#### 2.1.1 重复执行上次命令

在普通模式下.(小数点)表示重复上一次的命令操作

普通模式下输入x，删除第一个字符，输入.(小数点)会再次删除一个字符

#### 2.1.2 执行指定次数相同的命令

进入普通模式输入N<command>，N 表示重复后面的次数

* 输入10x，删除10个连续字符
* 输入3dd，将会删除3行文本

### 2.2 游标的快速跳转

#### 2.2.1 行间跳转

| **命令** | **说明** |
| --- | --- |
| nG(n Shift+g) | 游标移动到第 n 行**(如果默认没有显示行号，请先进入命令模式，输入**:set nu**以显示行号)** |
| gg | 游标移动到到第一行 |
| G(Shift+g) | 到最后一行 |

#### 2.2.2 行内跳转

普通模式下使用下列命令在行内按照单词为单位进行跳转

| **命令** | **说明** |
| --- | --- |
| w | 到下一个单词的开头 |
| e | 到当前单词的结尾 |
| b | 到前一个单词的开头 |
| ge | 到前一个单词的结尾 |
| 0或^ | 到行头 |
| $ | 到行尾 |
| f<字母> | 向后搜索<字母>并跳转到第一个匹配的位置(非常实用) |
| F<字母> | 向前搜索<字母>并跳转到第一个匹配的位置 |
| t<字母> | 向后搜索<字母>并跳转到第一个匹配位置之前的一个字母(不常用) |
| T<字母> | 向前搜索<字母>并跳转到第一个匹配位置之后的一个字母(不常用) |

### 2.3 复制粘贴和剪切

#### 2.3.1 复制及粘贴文本

普通模式中使用y复制

* + 普通模式中，yy复制游标所在的整行（3yy表示复制3行）
  + 普通模式中，y^ 复制至行首，或y0。不含光标所在处字符。
  + 普通模式中，y$ 复制至行尾。含光标所在处字符。
  + 普通模式中，yw 复制一个单词。
  + 普通模式中，y2w 复制两个单词。
  + 普通模式中，yG 复制至文本末。
  + 普通模式中，y1G 复制至文本开头。

普通模式中使用 p 粘贴

* + 普通模式中，p(小写)代表粘贴至光标后（下）
  + 普通模式中，P(大写)代表粘贴至光标前（上）

#### 2.3.2 剪切及粘贴

其实前面讲得 dd 删除命令就是剪切,每次 dd 删除文档内容后，便可以使用 p 来粘贴，即交换上下行：ddp

# 查找替换

### 2.1 字符的替换及撤销(Undo操作)

#### 2.1.1 替换和撤销(Undo)命令

替换和Undo命令都是针对普通模式下的操作

| **命令** | **说明** |
| --- | --- |
| r+<待替换字母> | 将游标所在字母替换为指定字母 |
| R | 连续替换，直到按下Esc |
| cc | 替换整行，即删除游标所在行，并进入插入模式 |
| cw | 替换一个单词，即删除一个单词，并进入插入模式 |
| C(大写) | 替换游标以后至行末 |
| ~ | 反转游标所在字母大小写 |
| u{n} | 撤销一次或n次操作 |
| U(大写) | 撤销当前行的所有修改 |
| Ctrl+r | redo，即撤销undo的操作 |

### 2.2 快速缩进

#### 2.2.1 使用命令进行快速调整缩进操作

* 普通模式下输入15G，跳转到15行
* 普通模式下输入>> 整行将向右缩进（使用，用于格式化代码超爽）
* 普通模式下输入<< 整行向左回退
* 普通模式下输入:进入命令行模式下对shiftwidth值进行设置可以控制缩进和回退的字符数

#### 2.2.2 shiftwidth命令

shiftwidth命令是指上一节>>命令产生的缩进（可以简写成sw） 普通模式下输入:进入命令行模式下对shiftwidth值进行设置可以控制缩进和回退的字符数 获取目前的设定值

:set shiftwidth?

设置缩进为10个字符

:set shiftwidth=10

#### 2.2.3 调整文本位置

命令行模式下输入:ce(center)命令使本行内容居中

:ce

命令行模式下输入:ri(right)命令使本行文本靠右

:ri

命令行模式下输入:le(left)命令使本行内容靠左

:le

### 2.3 查找

#### 2.3.1 快速查找

普通模式下输入 / 然后键入需要查找的字符串 按回车后就会进行查找。 ？ 与/ 功能相同，只不过 ？ 是向上而 / 是向下查找。 进入查找之后，输入n 和 N 可以继续查找 n表示继续查找，N 反向查找。

· 普通模式下输入/icmp然后回车即可查找字符串 icmp

· 普通模式下输入n查找下一个 icmp

· 普通模式下输入？tcp向上查找字符串 tcp

· 普通模式下输入N查找上一个出现的 tcp

# 高级功能

### 2.1 多文件编辑

#### 2.1.1 使用vim编辑多个文件



默认进入1.txt文件的编辑界面

* 命令行模式下输入 :n 编辑 2.txt 文件，可以加 ! 即 :n! 强制切换，之前一个文件的输入没有保存，仅仅切换到另一个文件
* 命令行模式下输入 :N 编辑 1.txt 文件，可以加 ! 即 :N! 强制切换，之前文件内的输入没有保存，仅仅是切换到另一个文件

#### 2.1.2 进入vim后打开新文件

* 命令行模式下输入:e 3.txt 打开新文件3.txt
* 命令行模式下输入:e# 回到前一个文件
* 命令行模式下输入:ls可以列出以前编辑过的文档
* 命令行模式下输入:b 2.txt（或者编号）可以直接进入文件2.txt编辑
* 命令行模式下输入:bd 2.txt（或者编号）可以删除以前编辑过的列表中的文件项目
* 命令行模式下输入:e! 4.txt，新打开文件4.txt，放弃正在编辑的文件
* 命令行模式下输入:f 显示正在编辑的文件名
* 命令行模式下输入:f new.txt，改变正在编辑的文件名字为new.txt

### 2.2 可视模式

#### 2.2.1 可视模式命令简介

* 在普通模式下输入 v（小写），进入字符选择模式，就可以移动光标，光标走过的地方就会选取。再次按下v会后就会取消选取。
* 在普通模式下输入 Shift+v（小写），进入行选择模式，按下V之后就会把整行选取，您可以上下移动光标选更多的行，同样，再按一次 Shift+v 就可以取消选取。
* 在普通模式下输入 Ctrl+v（小写），这是区域选择模式，可以进行矩形区域选择，再按一次 Ctrl+v 取消选取。
* 在可视模式下输入 d 删除选取区域内容
* 在可视模式下输入y复制选取区域内容

### 2.3 视窗操作

#### 2.3.1 视窗操作简介

vim 可以在一个界面里打开多个窗口进行编辑，这些编辑窗口称为 vim 的视窗。 打开方法有很多种，例如可以使用在命令行模式下输入 :new 打开一个新的 vim 视窗

· 命令行模式下输入:sp 1.txt 打开新的水平分屏视窗来编辑1.txt

· 命令行模式下输入:vsp 2.txt 打开新的垂直分屏视窗来编辑2.txt

· 普通模式下Ctrl+w s 将当前窗口分割成两个水平的窗口

· 普通模式下Ctrl+w v 将当前窗口分割成两个垂直的窗口

### 2.4 文档加密

#### 2.4.1 创建加密文档

$ vim -x file1

输入您的密码 确认密码 这样在下一次打开时，vim就会要求你输入密码

# Bash

Bash（GNU Bourne-Again Shell）是一个为GNU计划编写的Unix shell，它是许多Linux平台默认使用的shell。

shell是一个命令解释器，是介于操作系统内核与用户之间的一个绝缘层。

#### 1.Hello World

使用vim编辑hello.sh ，输入如下代码并保存：

#!/bin/bash

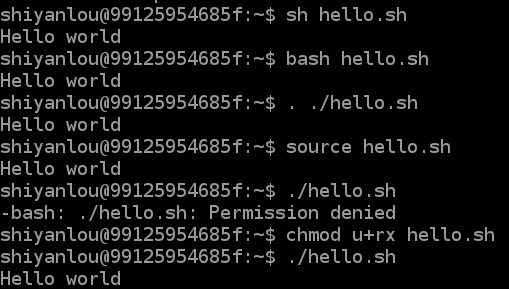
# This is a comment

echo Hello World

Linux 系统根据 "#!" 及该字符串后面的信息确定该文件的类型。

在 BASH 中 第一行的 "#!" 及后面的 /bin/bash 就表明该文件是一个 BASH 程序，需要由 /bin 目录下的 bash 程序来解释执行。

**运行Bash脚本的方式**



#### 2.使用重定向

比如我们想要**保存**刚刚的hello world为一个文本，那么该怎么办呢？

> 这个符号是重定向,执行以下代码，就会在当前目录下生成一个my.txt。打开看看有没有hello world

#!/bin/bash

echo "Hello World" > my.txt

#### 3.使用脚本清除/var/log下的log文件

首先我们看一看/var/log/wtmp

cat /var/log/wtmp

这个文件中记录了系统的一些信息，现在我们需要写一个脚本把里面的东西清空，但是保留文件

$ vim cleanlogs.sh

说明：

/dev/null这个东西可以理解为一个黑洞，里面是空的（可以用cat命令看一看），什么东西都可以往里面扔，扔了就没了

#!/bin/bash# 初始化一个变量

LOG\_DIR=/var/log

cd $LOG\_DIR

cat /dev/null > wtmp

echo "Logs cleaned up."

exit

**执行**

由于脚本中含有对系统日志文件内容的清除操作，这要求要有管理员权限.不然会报permission denide错误

使用sudo命令调用管理员权限才能执行成功：

$ sudo ./cleanlogs.sh

#### 遇到权限不够的提示，如何解决？

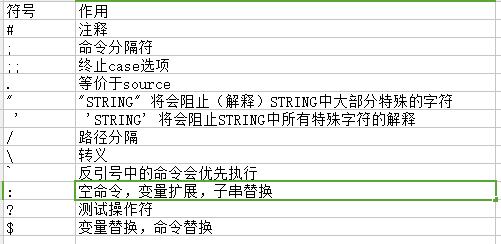
权限不够加sudo啊，可是你会发现

sudo cat /dev/null > /var/log/wtmp 一样会提示权限不够，为什么呢？因为sudo只能让cat命令以sudo的权限执行，而对于>这个符号并没有sudo的权限，我们可以使用

sudo sh -c "cat /dev/null > /var/log/wtmp " 让整个命令都具有sudo的权限执行

# bash中的特殊字符（上）

全部特殊字符的作用已经列表如下：



## 二、分号（;）

#### 1.命令分隔符

使用分号（;）可以在同一行上写两个或两个以上的命令。

#### 2.终止case选项（双分号）

使用双分号（;;）可以终止case选项。

## 三、点号（.）

#### 等价于 source 命令

bash 中的 source 命令用于在当前 bash 环境下读取并执行 FileName.sh 中的命令。

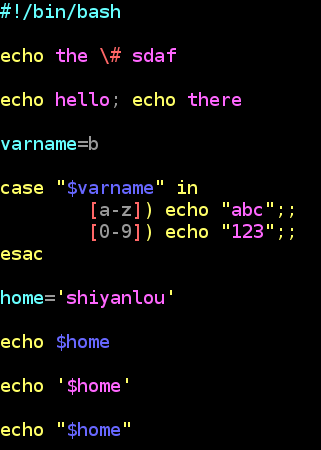
## 四、引号

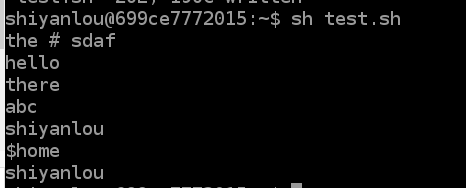
#### 1.双引号（")

"STRING" 将会阻止（解释）STRING中大部分特殊的字符。

#### 2.单引号（'）

'STRING' 将会阻止STRING中所有特殊字符的解释，这是一种比使用"更强烈的形式。





## 五、斜线和反斜线

#### 1.斜线（/）

文件名路径分隔符。在linux中表示路径的时候，许多个/跟一个/是一样的。/home/shiyanlou等同于////home///shiyanlou

## 六、反引号（`）

#### 命令替换

反引号中的命令会优先执行，如：

$ cp `mkdir back` test.sh back$ ls

先创建了 back 目录，然后复制 test.sh 到 back 目录

## 七、冒号（:）

#### 1.空命令

等价于“NOP”（no op，一个什么也不干的命令）。也可以被认为与shell的内建命令true作用相同。“:”命令是一个bash的内建命令，它的退出码（exit status）是（0）。

如：

#!/bin/bashwhile :do

echo "endless loop"done

等价于

#!/bin/bashwhile truedo

echo "endless loop"done

可以在 if/then 中作占位符：

#!/bin/bash

condition=5

if [ $condition -gt 0 ] #gt表示greater than，也就是大于，同样有-lt（小于），-eq（等于） then : # 什么都不做，退出分支else

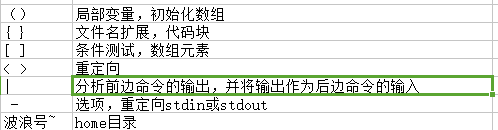
echo "$condition"fi

#### 2.变量扩展/子串替换

在与>重定向操作符结合使用时，将会把一个文件清空，但是并不会修改这个文件的权限。如果之前这个文件并不存在，那么就创建这个文件。

$ : > test.sh # 文件“test.sh”现在被清空了# 与 cat /dev/null > test.sh 的作用相同# 然而,这并不会产生一个新的进程, 因为“:”是一个内建命令

# bash特殊字符（下）



#### 1.命令组

在括号中的命令列表，将会作为一个子 shell 来运行。

在括号中的变量，由于是在子shell中，所以对于脚本剩下的部分是不可用的。父进程，也就是脚本本身，将不能够读取在子进程中创建的变量，也就是在子shell 中创建的变量。如：

$ vim test20.sh

输入代码：

#!/bin/bash

a=123

( a=321; )

echo "$a" #a的值为123而不是321，因为括号将判断为局部变量

运行代码：

$ bash test20.sha = 123

在圆括号中 a 变量，更像是一个局部变量。

## 二、大括号（{ }）

#### 1.文件名扩展

复制 t.txt 的内容到 t.back 中

$ vim test22.sh

输入代码：

#!/bin/bashif [ ! -w 't.txt' ];then

touch t.txtfiecho 'test text' >> t.txt

cp t.{txt,back}

在大括号中，不允许有空白

## 三、中括号（[ ]）

#### 1.条件测试

条件测试表达式放在[ ]中。下列练习中的-lt (less than)表示小于号。

$ vim test24.sh

输入代码：

#!/bin/bash

a=5if [ $a -lt 10 ]then

echo "a: $a"else

echo 'a>10'fi

#### 2.数组元素

在一个array结构的上下文中，中括号用来引用数组中每个元素的编号。

## 四、尖括号（< 和 >）

#### 重定向

test.sh > filename：重定向test.sh的输出到文件 filename 中。如果 filename 存在的话，那么将会被覆盖。

test.sh &> filename：重定向 test.sh 的 stdout（标准输出）和 stderr（标准错误）到 filename 中。

test.sh >&2：重定向 test.sh 的 stdout 到 stderr 中。

test.sh >> filename：把 test.sh 的输出追加到文件 filename 中。如果filename 不存在的话，将会被创建。

## 五、竖线（|）

#### 管道

分析前边命令的输出，并将输出作为后边命令的输入。这是一种产生命令链的好方法。

$ vim test26.sh

输入代码：

#!/bin/bash

tr 'a-z' 'A-Z'exit 0

现在让我们输送ls -l的输出到一个脚本中：

$ chmod 755 test26.sh$ ls -l | ./test26.sh

输出的内容均变为了大写字母。

# 变量和参数

## 一、变量定义

1.概念

如果 variable 是一个变量的名字，那么 $variable 就是引用这个变量的值，即这变量所包含的数据。

$variable 事实上只是 ${variable} 的简写形式。

#### 2.定义变量

定义变量时，变量名不加美元符号（$，PHP语言中变量需要），如： myname="shiyanlou"

**注意**

**变量名和等号之间不能有空格**。同时，变量名的命名须遵循如下规则：

* 首个字符必须为字母（a-z，A-Z）。
* 中间不能有空格，可以使用下划线（\_）。
* 不能使用标点符号。
* 不能使用bash里的关键字（可用help命令查看保留关键字）。

## 二、使用变量

变量名前加**美元符号**

## 三、只读变量

使用 readonly 命令可以将变量定义为只读变量，只读变量的值不能被改变。 下面的例子尝试更改只读变量，结果报错：

#!/bin/bash

myUrl="http://www.shiyanlou.com"readonly myUrl

myUrl="http://www.shiyanlou.com"

## 四、特殊变量

#### 位置参数

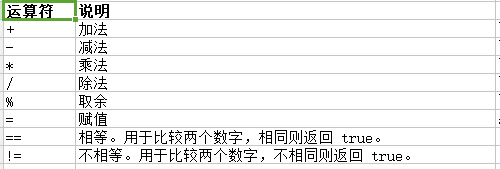
从命令行传递到脚本的参数：$0，$1，$2，$3...

$0就是脚本文件自身的名字，$1 是第一个参数，$2 是第二个参数，$3 是第三个参数，然后是第四个。$9 之后的位置参数就必须用大括号括起来了，比如，${10}，${11}，${12}。

* $# ： 传递到脚本的参数个数
* $\* ： 以一个单字符串显示所有向脚本传递的参数。与位置变量不同,此选项参数可超过 9个
* $$ ： 脚本运行的当前进程 ID号
* $! ： 后台运行的最后一个进程的进程 ID号
* $@ ： 与$#相同,但是使用时加引号,并在引号中返回每个参数
* $： 显示shell使用的当前选项,与 set命令功能相同
* $? ： 显示最后命令的退出状态。 0表示没有错误,其他任何值表明有错误。

# 基本运算符

## 一、算数运算符



$vim test.sh

#!/bin/bash

a=10

b=20

val=`expr $a + $b`

echo "a + b : $val"

val=`expr $a - $b`

echo "a - b : $val"

val=`expr $a \\* $b`

echo "a \* b : $val"

val=`expr $b / $a`

echo "b / a : $val"

val=`expr $b % $a`

echo "b % a : $val"

if [ $a == $b ]

then

echo "a == b"

Fi

if [ $a != $b ]

then

echo "a != b"

fi

运行

$bash test.sha + b : 30a - b : -10a \* b : 200

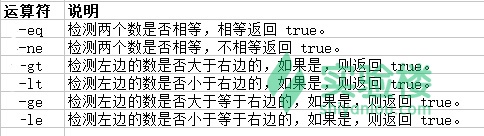
b / a : 2

b % a : 0a != b

* 原生bash不支持简单的数学运算，但是可以通过其他命令来实现，例如 awk 和 expr，expr 最常用。
* expr 是一款表达式计算工具，使用它能完成表达式的求值操作。
* 注意使用的反引号（esc键下边）
* 表达式和运算符之间要有空格$a + $b写成$a+$b不行
* 条件表达式要放在方括号之间，并且要有空格[ $a == $b ]写成[$a==$b]不行
* 乘号（\*）前边必须加反斜杠（\)才能实现乘法运算

## 二、关系运算符

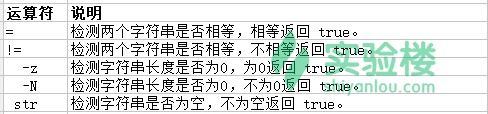
关系运算符只支持数字，不支持字符串，除非字符串的值是数字。



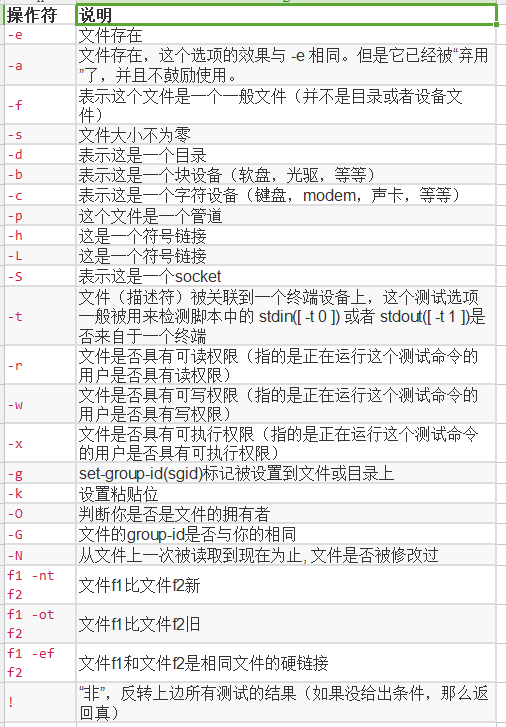
## 三、逻辑运算符



## 四、字符串运算符



## 五、文件测试运算符



# 流程控制

## 一、if else

以下实例判断两个变量是否相等：

a=10

b=20

if [ $a == $b ]

then

echo "a == b"elif [ $a -gt $b ]then

echo "a > b"elif [ $a -lt $b ]then

echo "a < b"else

echo "Ineligible"

fi

输出结果：

a < b

## 二、for 循环

例如，顺序输出当前列表中的数字：

for loop in 1 2 3 4 5

do

echo "The value is: $loop"

done

输出结果：

The value is: 1

The value is: 2

The value is: 3

The value is: 4

The value is: 5

顺序输出字符串中的字符：

for str in This is a string

do

echo $str

done

输出结果：

This

is

a

string

## 六、case

Shell case语句为多选择语句。可以用case语句匹配一个值与一个模式，如果匹配成功，执行相匹配的命令。case语句格式如下：

case 值 in

模式1)

command1

command2

...

commandN

;;

模式2）

command1

command2

...

commandN

;;esac

* 取值后面必须为单词in，每一模式必须以右括号结束。取值可以为变量或常数。匹配发现取值符合某一模式后，其间所有命令开始执行直至 ;;。
* 取值将检测匹配的每一个模式。一旦模式匹配，则执行完匹配模式相应命令后不再继续其他模式。如果无一匹配模式，使用星号 \* 捕获该值，再执行后面的命令。

下面的脚本提示输入1到4，与每一种模式进行匹配：

echo 'Enter a number between 1 and 4:'

echo 'The number you entered is:'

read aNum

case $aNum in

1) echo 'You have chosen 1'

;;

2) echo 'You have chosen 2'

;;

3) echo 'You have chosen 3'

;;

4) echo 'You have chosen 4'

;;

\*) echo 'You did not enter a number between 1 and 4'

;;esac

输入不同的内容，会有不同的结果，例如：

Enter a number between 1 and 4:

The number you entered is:3

You have chosen 3

## 七、跳出循环

在循环过程中，有时候需要在未达到循环结束条件时强制跳出循环，Shell使用两个命令来实现该功能：break和continue。

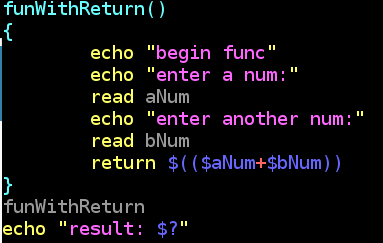
## 九、esac

case的语法和C family语言差别很大，它需要一个esac（就是case反过来）作为结束标记，每个case分支用右圆括号，用两个分号表示break。

# 函数

## 一、函数定义

* 函数返回值在调用该函数后通过 $? 来获得
* 所有函数在使用前必须定义。



## 二、函数参数

在Shell中，调用函数时可以向其传递参数。在函数体内部，通过 $n 的形式来获取参数的值，例如，$1表示第一个参数，$2表示第二个参数... 带参数的函数示例：

#!/bin/bashfunWithParam(){

echo "The first parameter is $1 !"

echo "The second parameter is $2 !"

echo "The tenth parameter is $10 !"

echo "The tenth parameter is ${10} !"

echo "The eleventh parameter is ${11} !"

echo "The total number of parameters is $# !"

echo "Outputs all parameters as a string $\* !"

}

funWithParam 1 2 3 4 5 6 7 8 9 34 73

输出结果：

The first parameter is 1 !

The second parameter is 2 !

The tenth parameter is 10 !

The tenth parameter is 34 !

The eleventh parameter is 73 !

The total number of parameters is 11 !

Outputs all parameters as a string 1 2 3 4 5 6 7 8 9 34 73 !