# 常用快捷键

**ctrl + l** 清屏

**ctrl + c** 终止命令

**ctrl + a** 光标移到行首

**ctrl + e** 光标移到行尾

**!!** 上一条命令

# 什么是Linux命令

我们在命令行下敲命令的时候，其实是在跟一个叫shell的程序打交道，shell会根据用户敲的命令名字，在环境变量PATH指定的路径下寻找和命令名字匹配的一个可执行文件，找到后shell会创建一个子进程，在子进程中加载并执行该可执行文件。所以我们平时敲的大部分命令都是一个已经存在的可执行文件，那么这个可执行文件放在哪里呢，通过whereis就可以看到命令所对应的可执行文件的路径了，例如要看看ls这个命令在哪里，就执行 whereis ls就可以看到ls的路径了。如果你把这个文件删掉，那么shell就会告诉你找不到这条命令。大部分命令都是一个可执行文件，但是我们在用whereis查看cd命令的可执行文件路径的时候，shell的输出是这个样子的:“cd：”，竟然没有路径名，这是什么鬼？ 这是因为cd是shell的一个内置命令，不以可执行文件的形式存在。所以咱们平时敲的命令主要分为两种，一种是以可执行文件存在的外部命令，一种是没有可执行文件的内置命令。这里的外部和内置是从shell的角度说的。那么怎么判断一条命令是shell内置命令还是外部命令呢，用type查看就可以了，比如查看cd，在命令行下敲type cd，shell的输出是cd is a shell builtin。那么cd就是一条shell内置命令。

简单来讲，shell就是一个命令解释器，shell和咱们敲的这些命令都是一样的，都是运行在linux内核之上的应用程序，在内核的眼里都是内核要调度执行的进程。这些进程在运行之前，都是一个一个的可执行文件。只是shell是在linux启动之后就会由init进程自动加载执行的。他也对应着一个可执行文件。可以通过查看/etc/passwd文件中每条记录的最后一个字段查看每个用户在启动后执行的shell可执行文件路径，也可以通过SHELL环境变量查看当前启动的shell。echo $SHELL

# linux环境变量

简单来讲，环境变量是用来设置linux应用程序运行环境的变量。我们经常接触的环境变量有PATH、HOME、HISTSIZE、HOSTNAME、PS1、PS2等。在shell命令行下可通过“echo $环境变量名”打印环境变量的值，也可通过env或者export命令查看系统中已有的环境变量。环境变量配置的是应用层的环境，由应用层程序设置并使用。不同的应用程序会关注不同的环境变量。比如很多时候PATH只会由shell关注，我们经常会配置的java环境变量，会由java虚拟机关注，其他的应用程序是不会关注的。如果你在命令行下只敲了cd，不带路径名，那么cd就会关注HOME这个环境变量，并切换到HOME环境变量指定的路径中，这个就是登录用户的家目录了。不过cd是shell的内置命令，所以HOME也主要是shell关注的。我最开始理解环境变量的时候，总是认为它们是内核的一部分，其实环境变量和内核没有多大的关系。环境变量并不是用来配置linux内核的。

环境变量是用来配置应用程序运行环境的变量，所以环境变量是和应用程序息息相关的，而应用程序在运行阶段是以linux进程的形式存在的，每个进程都有自己的环境变量，那么这些环境变量存放在哪里呢？他们是从哪里来的呢？说的环境变量，为什么会扯到进程虚拟地址空间呢？因为我们的环境变量是以环境变量表（数组）的形式存放在进程虚拟地址空间的用户空间里。这也是我上面说环境变量和内核没什么关系的原因。

我们指出，linux运行一条命令，就会创建出一个进程，而这个进程是由shell创建的。也就是你在linux下执行的任何命令所创建的进程，都是shell的子进程。所以一个进程的环境变量，大部分都是从shell进程继承而来的。

shell的环境变量其实就是用户自己设置的，所以我们经常说环境变量的时候，前面总是加上shell，叫shell环境变量。每次shell启动的时候，都会读取一系列的环境配置文件。将环境配置文件中环境变量的值读到自己内存空间的位置，然后由shell生成的子进程就会继承shell的环境变量。

## 查看环境变量的值

echo $环境变量名

env

export

后面两个命令会列出系统中所有的环境变量

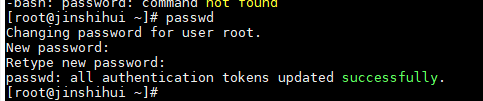
# 查看linux的内核

uname –r



# 修改root用户的密码

直接输入 passwd，输入两遍新密码即可。



# 重启linux系统

输入这个命令：reboot

reboot有一参数，执行的意义就不一样了reboot  -f  就是强制关机不调用shutdown

reboot  -i  在重开机之前，先关闭所有网络界面

reboot -n 重开机之前不检查是否有未结束的程序

# cd命令

## 基本概念

cd命令用于切换当前工作目录至 dirName(目录参数) ，其中 dirName 表示法可为绝对路径或相对路径。若目录名称省略，则变换至使用者的 home 目录 (也就是login 时所在的目录) 。另外，"~" 也表示为 home 目录的意思，"." 表示目前所在的目录，".." 表示目前目录位置的上一层目录。

## 语法

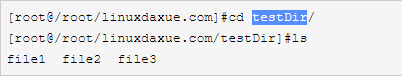
**cd [dirName]**

dirName：要切换的目标目录

## 示例

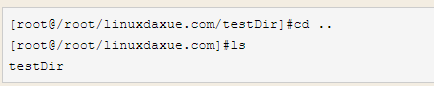
跳入testDir目录

cd testDir/



跳至上层目录

cd ..



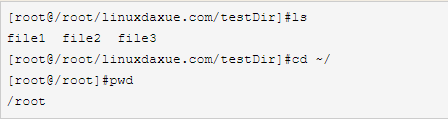
跳至上上层目录

cd ../../



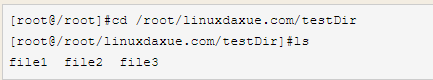
跳入用户主目录

cd ~/



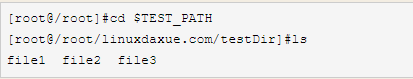
使用绝对路径

cd /root/linuxdaxue/testDir



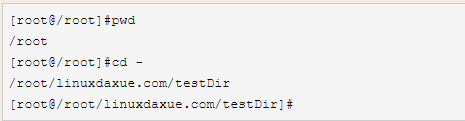
使用环境变量

cd $TEST\_PATH



跳入上次使用目录

cd -



# ls命令

## 基本概念

就是list的缩写，通过ls 命令不仅可以查看linux文件夹包含的文件，而且可以查看文件权限(包括目录、文件夹、文件权限)、查看目录信息等。

## 语法

ls -a 列出目录所有文件，包含以.开始的隐藏文件

ls -A 列出除.及..的其它文件

ls -l 除了文件名之外，还将文件的权限、所有者、文件大小等信息详细列出来

ls -r 反序排列

ls -t 以文件修改时间排序

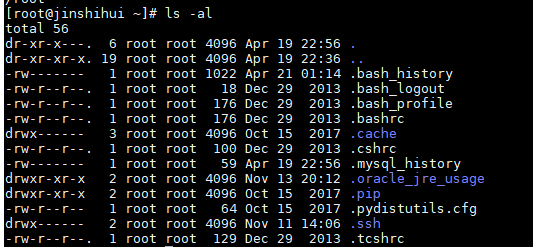
ls -S 以文件大小排序

ls -h 以易读大小显示

## 示例

列出当前目录下面的文件：

ls -al



# pwd命令

## 基本概念

查看当前工作目录路径

## 语法

pwd

## 示例

1、查看当前路径

pwd

2、查看软链接的实际路径

pwd –P

# mkdir命令

## 基本概念

创建文件夹

## 语法

-m: 对新建目录设置存取权限,也可以用chmod命令设置;

-p: 可以是一个路径名称。此时若路径中的某些目录尚不存在,加上此选项后,系统将自动建立好，那些尚不在的目录,即一次可以建立多个目录

## 示例

1、当前工作目录下创建名为t的文件夹

mkdir t

2、在tmp目录下创建路径为test/t1/t的目录，若不存在，则创建

mkdir -p /tmp/test/t1/t

# rm命令

## 基本概念

删除一个目录中的一个或多个文件或目录，如果没有使用- r选项，则rm不会删除目录。如果使用rm 来删除文件，通常仍可以将该文件恢复原状

## 语法

rm [选项] 文件

## 示例

1、删除任何.log文件；删除前逐一询问确认

rm -i \*.log

2、删除test子目录及子目录中所有文件删除,并且不用一一确认

rm -rf test

# rmdir命令

## 基本概念

从一个目录中删除一个或多个子目录项（删除空目录），删除某目录时也必须具有对其父目录的写权限

注意：不能删除非空目录

## 语法

rmdir [-p] dirName

参数：-p 是当子目录被删除后使它也成为空目录的话，则顺便一并删除

## 示例

1、当parent子目录被删除后使它也成为空目录的话，则顺便一并删除

rmdir -p parent/child/child11

# mv命令

## 基本概念

用来为文件或目录改名、或将文件或目录移入其它位置，根据第二参数类型（如目录，则移动文件；如为文件则重命名该文件）

## 语法

mv [options] source dest

mv [options] source... directory

参数说明：

-i: 若指定目录已有同名文件，则先询问是否覆盖旧文件

-f: 在mv操作要覆盖某已有的目标文件时不给任何指示



## 示例

1、将文件 aaa 更名为 bbb

mv aaa bbb

2、将info目录放入logs目录中。注意，如果logs目录不存在，则该命令将info改名为logs。

mv info/ logs

3、将/usr/student下的所有文件和目录移到当前目录下

mv /usr/student/\* .

# cp命令

## 基本概念

主要用于复制文件或目录

## 语法

cp [options] source dest

或者

cp [options] source... directory

参数说明：

-a：此选项通常在复制目录时使用，它保留链接、文件属性，并复制目录下的所有内容。其作用等于dpR参数组合

-d：复制时保留链接。这里所说的链接相当于Windows系统中的快捷方式

-f：覆盖已经存在的目标文件而不给出提示

-i：与-f选项相反，在覆盖目标文件之前给出提示，要求用户确认是否覆盖，回答"y"时目标文件将被覆盖。

-p：除复制文件的内容外，还把修改时间和访问权限也复制到新文件中。

-r：若给出的源文件是一个目录文件，此时将复制该目录下所有的子目录和文件。

-l：不复制文件，只是生成链接文件

## 示例

1、将当前目录"test/"下的所有文件复制到新目录"newtest"下

cp –r test/ newtest

注意：用户使用该指令复制目录时，必须使用参数"-r"或者"-R"。

2、复制a.txt到test目录下，保持原文件时间,如果原文件存在提示是否覆盖

cp -ai a.txt test

# cat命令

## 基本概念

cat主要有三大功能：

1、一次显示整个文件:cat filename

2、从键盘创建一个文件:cat > filename 只能创建新文件,不能编辑已有文件

3、将几个文件合并为一个文件:cat file1 file2 > file

## 语法

cat [-AbeEnstTuv] [--help] [--version] fileName

参数：

**-n 或 --number**：由 1 开始对所有输出的行数编号。

**-b 或 --number-nonblank**：和 -n 相似，只不过对于空白行不编号

**-s 或 --squeeze-blank**：当遇到有连续两行以上的空白行，就代换为一行的空白行。

**-v 或 --show-nonprinting**：使用 ^ 和 M- 符号，除了 LFD 和 TAB 之外。

**-E 或 --show-ends**: 在每行结束处显示 $。

**-T 或 --show-tabs**: 将 TAB 字符显示为 ^I。

**-A, --show-all**：等价于 -vET。

**-e：**等价于"-vE"选项；

**-t：**等价于"-vT"选项；

## 示例

1、把 textfile1 的文档内容加上行号后输入 textfile2 这个文档里：

cat -n textfile1 > textfile2

2、把 textfile1 和 textfile2 的文档内容加上行号（空白行不加）之后将内容附加到 textfile3 文档里

cat -b textfile1 textfile2 >> textfile3

3、清空 /etc/test.txt 文档内容

cat /dev/null > /etc/test.txt

# more命令

## 基本概念

more 命令类似 cat ，不过会以一页一页的形式显示，更方便使用者逐页阅读，而最基本的指令就是按空白键（space）就往下一页显示，按 b 键就会往回（back）一页显示，而且还有搜寻字串的功能（与 vi 相似），使用中的说明文件，请按 h 。

## 语法

more [-dlfpcsu] [-num] [+/pattern] [+linenum] [fileNames..]

参数：

-num 一次显示的行数

-d 提示使用者，在画面下方显示 [Press space to continue, 'q' to quit.] ，如果使用者按错键，则会显示 [Press 'h' for instructions.] 而不是 '哔' 声

-l 取消遇见特殊字元 ^L（送纸字元）时会暂停的功能

-f 计算行数时，以实际上的行数，而非自动换行过后的行数（有些单行字数太长的会被扩展为两行或两行以上）

-p 不以卷动的方式显示每一页，而是先清除萤幕后再显示内容

-c 跟 -p 相似，不同的是先显示内容再清除其他旧资料

-s 当遇到有连续两行以上的空白行，就代换为一行的空白行

-u 不显示下引号 （根据环境变数 TERM 指定的 terminal 而有所不同）

+/pattern 在每个文档显示前搜寻该字串（pattern），然后从该字串之后开始显示

+num 从第 num 行开始显示

fileNames 欲显示内容的文档，可为复数个数

## 示例

1、逐页显示 testfile 文档内容，如有连续两行以上空白行则以一行空白行显示

more -s testfile

2、从第 20 行开始显示 testfile 的文档内容

more +20 testfile

3、在所列出文件目录详细信息，借助管道使每次显示5行

 ls -l | more -5

按空格显示下5行

## 常用操作命令

Enter 向下n行，需要定义。默认为1行

Ctrl+F 向下滚动一屏

空格键 向下滚动一屏

Ctrl+B 返回上一屏

= 输出当前行的行号

：f 输出文件名和当前行的行号

V 调用vi编辑器

!命令 调用Shell，并执行命令

q 退出more

# yum命令

## 基本概念

[yum](http://www.yanghengfei.com/tag/yum/)（全称为 Yellow dog Updater, Modified）是一个在Fedora和RedHat以及SUSE中的Shell前端软件包管理器。基于RPM包管理，能够从指定的服务器自动下载RPM包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软体包，无须繁琐地一次次下载、安装。yum提供了查找、安装、删除某一个、一组甚至全部软件包的命令，而且命令简洁而又好记。

## 语法

yum [options] [command] [package ...]

参数：

options：可选，选项包括-h（帮助），-y（当安装过程提示选择全部为"yes"），-q（不显示安装的过程）等等

command：要进行的操作。

package操作的对象。

## 常用命令

1、列出所有可更新的软件清单命令：yum check-update

2、更新所有软件命令：yum update

3、仅安装指定的软件命令：yum install <package\_name>

4、仅更新指定的软件命令：yum update <package\_name>

# ps命令

## 基本概念

要对进程进行监测和控制,首先必须要了解当前进程的情况,也就是需要查看当前进程,而ps命令就是最基本同时也是非常强大的进程查看命令，使用该命令可以确定有哪些进程正在运行和运行的状态、进程是否结束、进程有没有僵尸、哪些进程占用了过多的资源等等.总之大部分信息都是可以通过执行该命令得到的，**ps**命令最常用的还是用于监控后台进程的工作情况,因为后台进程是不和屏幕键盘这些标准输入/输出设备进行通信的,所以如果需要检测其情况,便可以使用ps命令了。

说明：ps是显示瞬间进程的状态，并不动态连续，如果想对进程进行实时监控应该用top命令。

## 参数

-A：显示所有的进程，与 -e 具有同样的效用。

-a：显示现行终端机下的所有进程，包括其他用户的进程。

-u：以用户为主的进程状态。

-au 显示较详细的进程信息

x：通常与 a 这个参数一起使用，可列出较完整信息。

## 输出格式规划

l: 较长、较详细的将该 PID 的信息列出

j: 工作的格式 (jobs format)

-f: 做一个更为完整的输出

特别说明：由于 ps 能够支持的系统类型相当的多，所以他的参数多的离谱。

而且有没有加上 - 差很多，详细的用法应该要参考 man ps。

示例1：

将目前属于您自己这次登入的 PID 与相关信息列示出来

ps –l

示例2：

列出目前所有的正在内存当中的程序（进程）

ps aux

示例3：

以示例一的显示内容，显示出所有的程序

ps –lA

示例4：

显示jboss所在的进程

ps –aef|grep jboss

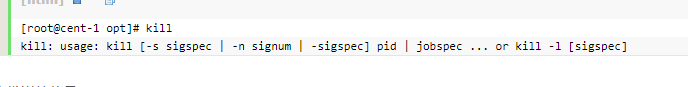
# kill命令

## 基本概念

kill命令是Linux下面的一个终止进程的命令。

## 语法

在Shell下面打kill可以看到kill的语法如下：



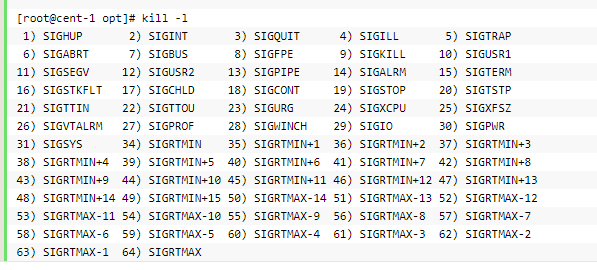
### 参数说明

-s 指定发送的信号

pid 终止的进程编号

sigspec 表示信号

通过kill –l可以看到所有的信号变量



### kill pid

kill pid和kill -s 15 pid含义一样，表示发送一个SIGTERM的信号给对应的程序。程序收到该信号后，将会发生以下事情：

1、程序立刻停止

2、程序释放相应资源后停止

3、程序可能仍然继续运行

大部分程序在接收到SIGTERM信号后，会先释放自己的资源，然后再停止。但也有一些程序在收到信号后，做一些其他事情，并且这些事情是可以配置的，也就是说SIGTERM多半是会被阻塞，忽略的。

### kill -9 pid

kill -9 pid等于kill -s 9 pid，表示强制尽快终止一个进程。多半admin会用这个命令。

说明：kill一个进程需要root权限去执行，一般用户无法执行这个命令。

# top命令

top命令是Linux下常用的。

## 语法

top [-] [d] [p] [q] [c] [C] [S] [s]  [n]

## 参数说明

d: 指定每两次屏幕信息刷新之间的时间间隔。当然用户可以使用s交互命令来改变之。

p: 通过指定监控进程ID来仅仅监控某个进程的状态

常用操作

# find

## 基本概念

find命令用来在指定目录下查找文件。任何位于参数之前的字符串都将被视为欲查找的目录名。如果使用该命令时，不设置任何参数，则find命令将在当前目录下查找子目录与文件。并且将查找到的子目录和文件全部进行显示。

## 语法

find(选项)（参数）

参数：起始目录：查找文件的起始目录

示例：

列出当前目录及子目录下所有文件和文件夹

find .

在/home目录下查找以.txt结尾的文件名

find /home –name "\*.txt"

当前目录及子目录下查找所有以.txt和.pdf结尾的文件

find . \( -name "\*.txt" -o -name "\*.pdf" \)

或

find . -name "\*.txt" -o -name "\*.pdf"

# sudo

简单的说，sudo 是一种权限管理机制，管理员可以授权于一些普通用户去执行一些 root 执行的操作，而不需要知道 root 的密码。严谨些说，sudo 允许一个已授权用户以超级用户或者其它用户的角色运行一个命令。当然，能做什么不能做什么都是通过安全策略来指定的。sudo 支持插件架构的安全策略，并能把输入输出写入日志。第三方可以开发并发布自己的安全策略和输入输出日志插件，并让它们无缝的和 sudo 一起工作。默认的安全策略记录在 /etc/sudoers 文件中。而安全策略可能需要用户通过密码来验证他们自己。也就是在用户执行 sudo 命令时要求用户输入自己账号的密码。如果验证失败，sudo 命令将会退出。(注意，本文介绍的 sudo 命令运行在 ubuntu 14.04中。)

## 命令语法

sudo [-bhHpV][-s ][-u <用户>][指令]

或者

sudo [-klv]

参数：

-b 在后台执行指令

-h 显示帮助

-H 将HOME环境变量设为新身份的HOME环境变量

-k 结束密码的有效期限，也就是下次再执行sudo时便需要输入密码

-l 列出目前用户可执行与无法执行的指令

-p 改变询问密码的提示符号

 -s  执行指定的shell。  
-u  <用户>  以指定的用户作为新的身份。若不加上此参数，则预设以root作为新的身份。  
 -v  延长密码有效期限5分钟。  
-V  显示版本信息。

-S  从标准输入流替代终端来获取密码

## 基本配置

系统默认创建了一个名为 sudo 的组。只要把用户加入这个组，用户就具有了 sudo 的权限。

至于如何把用户加入 sudo 组，您可以直接编辑 /etc/group 文件，当然您得使用一个有 sudo 权限的用户来干这件事，使用如下命令：

sudo vim /ect/group

在sudo 组中加入新的用户，要使用逗号分隔多个用户

或者您可以使用 usermod 命令把用户添加到一个组中，命令如下：

sudo usermod -a -G sudo jack

上面的设置中我们把用户jack添加到了sudo组中，所以当用户 jack 登录后就可以通过 sudo 命令以 root 权限执行命令了。

# vim（重点）

## 基本概念

vi编辑器是所有Unix及Linux系统下标准的编辑器，它的强大不逊色于任何最新的文本编辑器。vim 具有程序编辑的能力，可以以字体颜色辨别语法的正确性，方便程序设计；因为程序简单，编辑速度相当快速。vim可以当作vi的升级版本，他可以用多种颜色的方式来显示一些特殊的信息。vim会依据文件扩展名或者是文件内的开头信息判断该文件的内容而自动的执行该程序的语法判断式，再以颜色来显示程序代码与一般信息。vim里面加入了很多额外的功能，例如支持正则表达式的搜索、多文件编辑、块复制等等。这对于我们在Linux上进行一些配置文件的修改工作时是很棒的功能。

## vi的基本概念

基本上vi可以分为三种状态，分别是命令模式（command mode）、插入模式（Insert mode）和底行模式（last line mode），各模式的功能区分如下：

### （1）命令行模式

控制屏幕光标的移动，字符、字或行的删除，移动复制某区段及进入Insert mode下，或者到 last line mode。

### （2）插入模式

只有在Insert mode下才可以做文字输入，按「ESC」键可回到命令行模式。

### （3）底行模式

将文件保存或退出vi，也可以设置编辑环境，如寻找字符串、列出行号等。不过一般我们在使用时把vi简化成两个模式，就是将底行模式（last line mode）也算入命令行模式command mode）。

## vi基本操作

### （1）进入vi

在系统提示符号中输入vi及文件名称后就进入vi全屏幕编辑画面。不过有一点要特别注意，就是您进入vi之后是处于「命令行模式（command mode）」，您要切换到「插入模式（Insert mode）」才能够输入文字。初次使用vi的人都会想先用上下左右键移动光标，结果电脑一直哔哔叫，把自己气个半死，所以进入vi后先不要乱动，转换到插入模式。

### （2）切换到插入模式编辑文件

在命令行模式（command mode）下**按一下字母「i」**就可以进入插入模式（Insert mode），这时候你就可以开始输入文字了。

### （3） Insert 的切换

您目前处于「插入模式（Insert mode）」，您就只能一直输入文字，如果您发现输错了字，想用光标键往回移动，将该字删除，就要**先按一下「ESC」键**转到「命令行模式（command mode）」再删除文字。

### （4）退出vi及保存文件

在命令行模式下**按一下：冒号**键进入Last line mode，例如：

: w filename （输入 「w filename」将文件以指定的文件名filename保存）

: wq (输入wq，存盘并退出vi)

: q! (输入q!，不存盘强制退出vi)

## 命令行模式功能键

### （1）插入模式

按「i」切换进入插入模式「insert mode」，按“i”进入插入模式后是从光标当前位置开始输入文件

按「a」进入插入模式后，是从目前光标所在位置的下一个位置开始输入文字

按「o」进入插入模式后，是插入新的一行，从行首开始输入文字。

### （2） 从插入模式切换为命令行模式

按「ESC」键

### （3）移动光标

vi可以直接用键盘上的光标来上下左右移动，但正规的vi是用小写英文字母「h」、「j」、「k」、「l」，分别控制光标左、下、上、右移一格。

按「ctrl」+「b」：屏幕往“后”移动一页

按「ctrl」+「f」：屏幕往“前”移动一页

按「ctrl」+「u」：屏幕往“后”移动半页

按「ctrl」+「d」：屏幕往“前”移动半页

按数字「0」：移到行的开头

按「G」：移动到文件的最后一行

按「$」：移动到光标所在行的“行尾”

# vi命令修改文件及保存的使用方法

vi编辑器是所有Unix及Linux系统下标准的编辑器，介绍一下它的用法和一小部分指令。由于对Unix及Linux系统的任何版本，vi编辑器是完全相同的，因此您可以在其他任何介绍vi的地方进一步了解它。

在所有命令执行之前，先按ESC。

## VI的基本概念

基本上vi可以分为三种状态，分别是命令模式（command mode）、插入模式（Insert mode）和底行模式（last line mode），各模式的功能区分如下。

命令行模式：控制屏幕光标的移动，字符、字或行的删除，移动复制某区段及进入Insert mode（插入模式）下，或者到 last line mode（底行模式）。

插入模式：只有在Insert mode下，才可以做文字输入，按「ESC」键可回到命令行模式。

底行模式：将文件保存或退出vi，也可以设置编辑环境，如寻找字符串、列出行号等。

不过一般我们在使用时把vi简化成两个模式，就是将底行模式（last line mode）也算入命令行模式（command mode）

## VI的基本操作

进入VI：在系统提示符号输入vi及文件名后，就进入vi全屏幕编辑画面。

# 怎么执行.sh文件

在Linux系统下运行.sh文件有两种方法，比如我在root目录下有个datelog.sh文件

说明：在编写sh文件的时候不要有空行，否则会报错。

## 第一种方法

这种办法需要用chmod使得文件具备执行条件(x): chmod u+x datelog.sh

1、在任何路径下，输入该文件的绝对路径/root/datelog.sh就可执行该文件（当然要在权限允许情况下）



2、cd到datelog.sh文件的目录下，然后执行./datelog.sh



## 第二种方法（推荐这个方法）

这种办法不需要文件具备可执行的权限也可运行

1、在当前文件路径下输入：sh加上文件名字即可，如下：

sh datelog.sh



2、在任意路径下，sh 加上文件路径及文件名称，如下：

sh /root/ datelog.sh



# linux系统如何清理缓存

## 1、缓存机制介绍

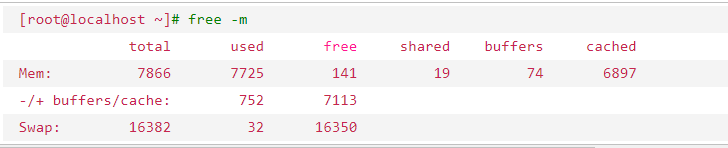
在Linux系统中，为了提高文件系统性能，内核利用一部分物理内存分配出缓冲区，用于缓存系统操作和数据文件，当内核收到读写的请求时，内核先去缓存区找是否有请求的数据，有就直接返回，如果没有则通过驱动程序直接操作磁盘。

缓存机制优点：减少系统调用次数，降低CPU上下文切换和磁盘访问频率。

CPU上下文切换：CPU给每个进程一定的服务时间，当时间片用完后，内核从正在运行的进程中收回处理器，同时把进程当前运行状态保存下来，然后加载下一个任务，这个过程叫做上下文切换。实质上就是被终止运行进程与待运行进程的进程切换。

## 2、查看缓存区及内存使用情况

使用命令：free –m



从上面的命令结果显示中可以看出：内存总共8G，已使用7725M，剩余141M，不少的人都是这么看的，但其实这样并不能作为实际的使用率。因为有了缓存机制，具体算法如下：

空闲内存=free（141）+buffers（74）+cached（6897）  
已用内存=total（7866）-空闲内存  
由此算出空闲内存是7112M，已用内存754M，这才是真正的使用率，也可参考-/+ buffers/cache这行信息也是内存正确使用率。

## 3、缓存区分buffers和cached区别

内核在保证系统能正常使用物理内存和数据量读写情况下来分配缓冲区大小，buffers用来缓存metadata及pages，可以理解为系统缓存，例如，vi打开一个文件。cached是用来给文件做缓存，可以理解为数据块缓存，例如，dd if=/dev/zero of=/tmp/test count=1 bs=1G 测试写入一个文件，就会被缓存到缓冲区中，当下一次再执行这个测试命令时，写入速度会明显很快。

## 4、Swap用途

Swap意思是交换分区，通常我们说的虚拟内存，是从硬盘中划分出的一个分区。当物理内存不够用的时候，内核就会释放缓存区（buffers/cache）里一些长时间不用的程序，然后将这些程序临时放到Swap中，也就是说如果物理内存和缓存区内存不够用的时候，才会用到Swap。

swap清理命令：

swapoff –a && swapon –a

说明：运行这个命令需要管理员的权限，这样清理有个前提条件，空闲的内存必须比已经使用的swap空间大。

## 5、释放缓存区内存的方法

清理pagecache（页面缓存），命令：

/proc/sys/vm/drop\_caches 或者sysctl -w vm.drop\_caches=1

清理dentries（目录缓存）和inodes，命令：

/proc/sys/vm/drop\_caches 或者sysctl -w vm.drop\_caches=2

清理pagecache、dentries和inodes，命令：

/proc/sys/vm/drop\_caches 或者sysctl -w vm.drop\_caches=3

上面三种方式都是临时释放缓存的方法，要想永久释放缓存，需要在/etc/sysctl.conf文件中配置：vm.drop\_caches=1/2/3，然后sysctl -p生效即可。说明：修改这个文件都需要管理员的权限。

另外，可以使用sync命令来清理文件系统缓存，还会清理僵尸(zombie)对象和它们占用的内存，例如：sync

说明：上面操作在大多数情况下都不会对系统造成伤害，只会有助于释放不用的内存。但是如果在执行这些操作时正在写数据，那么实际上在数据到达磁盘之前就将它从文件缓存中清除掉了，这可能会造成很不好的影响。那么如果避免这种事情发生呢？因此，这里不得不提一下/proc/sys/vm/vfs\_cache\_pressure这个文件，告诉内核，当清理inoe/dentry缓存时应该用什么样的优先级，如下图：



释放内存前先使用sync命令做同步，以确保文件系统的完整性，将所有未写的系统缓冲区写到磁盘中，包含已修改的 i-node、已延迟的块 I/O 和读写映射文件。否则在释放缓存的过程中，可能会丢失未保存的文件。

/proc是一个[虚拟文件系统](https://www.baidu.com/s?wd=%E8%99%9A%E6%8B%9F%E6%96%87%E4%BB%B6%E7%B3%BB%E7%BB%9F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLmyRkrAnLmHTvryfYPhDd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH6Yn1DzPWms)，可以通过对它的读写操作作为与kernel实体间进行通信的一种手段。也就是说可以通过修改/proc中的文件，来对当前kernel的行为做出调整。也就是说我们可以通过调整/proc/sys/vm/drop\_caches来释放内存。

