**vim的分屏功能**

基本都是ctrl+w然后加上某一个按键字母，触发一个功能。

（1）在vim里打开一个分屏:  
　　创建空白分屏：  
　　:new  
　　打开任意文件：  
　　:vsplit(:vsp) filename  
　　:sp(split) filename  
　　打开当前文件：  
　　ctrl+w 和 s(split)  
　　ctrl+w 和 v(vsplit)

（2）关闭一个分屏:  
　　:only 或者 ctrl+w 和 o取消其它分屏，只保留当前分屏  
　　ctrl+w 和 c(close)  
　　只剩最后一个分屏以后推出：  
　　ctrl+w 和 q(quit)  
（3）移动光标，也就是切换分屏；也可以移动分屏，比如将左分屏移动到右边。  
　　ctrl+w 和 w（各种切换，只有两个分屏的时候还是比较方便的）  
　　ctrl+w 和 h(H) 左  
　　ctrl+w 和 j(J) 下  
　　ctrl+w 和 k(K) 上  
　　ctrl+w 和 l(L) 右  
（4）最后就是改变分屏尺寸的操作了。  
　　ctrl+w 和 < 左  
　　ctrl+w 和 > 右  
　　ctrl+w 和 + 上  
　　ctrl+w 和 - 下  
　　ctrl+w 和 = 恢复均等

**vim的tag功能**

:tabnew新建标签页

:tabs显示已打开标签页的列表

:tabc关闭当前标签页

:tabn或者gt移动到下一个标签页

:tabp移动到上一个标签页

:tabfirst移动到第一个标签页

:tablast移动到最后一个标签页

**vim的删除功能**

**字符删除**

x 删除光标所在处字符或者在Visual模式中选中内容

X 删除光标所在前字符

**单词删除**

dw 删除到下一个单词开头

de 删除到本单词末尾

dE 删除到本单词末尾包括标点在内

db 删除到前一个单词

dB 删除到前一个单词包括标点在内

**行删除**

dd 删除一整行

D d$ 删除光标位置到本行结尾

d0 删除光标位置到本行开头

**tips：**3dd代表删除三行，聪明的你一定早就知道了

# [vim选中字符删除/复制/剪切/粘贴](http://www.cnblogs.com/luosongchao/p/3193153.html)

进入vim中visual模式，visual模式进入，可以有三种方式：

(1)在普通模式（normal）下，直接按键 v  就可以进入默认visual模式，

对于选中的文本进行如下按键：

（1.1）x   ------ 删除操作

（1.2）d   ------ 剪切操作

（1.3）y   -------复制操作

（1.4）p   -------粘贴操作

（1.5）^  --------选中当前行，光标位置到行首（或者使用键盘的HOME键）

（1.6）$  --------选中当前行，光标位置到行尾（或者使用键盘的END键）

（2）Visual Line模式  按键V可以进入

  按键V之后，进入Visual Line模式，使用 j/k键可以选中一行或者多行

（3）Visual Block模式，按键Ctrl + V可以进入

  在块模式下，可以进行多列的同时修改，修改方法是：

   首先进入块模式 Ctrl+ v, windows下用Ctrl+ q(Ctrl+Shirt+v),

   使用按键j/k/h/l进行选中多列

   按键Shift + i 进行 块模式下的插入

   输入字符之后，按键ESC，完成多行的插入

[**vim自动补全功能**](http://www.cnblogs.com/zhangshuli-1989/p/hq_15_12_18_00_10.html)

本章我们学习自动补全功能  
1.自动补全优先从当前的编辑区获得补全列表  
例如：  
我们写下如下内容

aaaaa

aabbb

aaab

当我们再次输入aa，然后我们按下Tab的时候，会弹出一个包含 aaaaa aabbb aaab的列表  
触发补全模式的条件  
1.插入模式下

ctrl+p ctrl+n 或者Tab

同时，ctrl+p ctrl+n还能够上下移动选中补全列表项  
还有其他的补全方法，如下  
这些方法都是以ctrl + x来启动的，然后跟着你想要的补全样式

1.<c-n>-普通关键字

2.<c-x><c-n>-当前缓冲区

3.<c-x><c-i>-包含文件关键字

4.<c-x><c-]>-标签文件关键字

5.<c-x><c-k>-字典查找

6.<c-x><c-l>-整行补全

对于行补全  
例如，我们进行如下操作  
1.自动（这时候我们进行<c-x><c-l>操作，就会自动补全为"本章我们学习自动补全功能")

7.<c-x><c-f>-文件名补全

8.<c-x><c-o>-全能补全

当我们弹出列表以后，接下来我们可以怎么操作这个自动补全列表呢？

[复制代码](javascript:void(0);)

1.<c-n>-匹配下一个

2.<c-p>-匹配上一个

3.<c-y>-确认使用当前匹配项(之前自己都是想要通过enter键来实现选中，一直没成功,虽然选中了，当时会换行)

4.<c-e>-退出补全列表

5.<c-h>-从当前匹配中减少一个字符(比如，当前输入是aa，补全列表光标在aabbb上，你使用<c-h>的话，输入会变成aabb)

6.<c-l>-从当前匹配中增加一个字符(比如，当前输入是aa，补全列表光标在aabbb上，你使用<c-h>的话，输入会变成aab)

7.{char}-插入字符跟当时光标所在的补全列表项，终止补全

[复制代码](javascript:void(0);)

这里面有个小技巧，就是我们在使用补全的时候，尽量通过<c-n>来调出补全列表，然后我们摁下了<c-p>,这时候光标就处在了列表的0行，我们添加或者减少字符>的时候，列表会自动精简或者增加

Vim 高亮选择和取消

1.选中高亮

/character

2.取消高亮

:nohl

# Nerdtree 快捷键

h j k l移动光标定位  
ctrl+w+w 光标在左右窗口切换  
ctrl+w+r 切换当前窗口左右布局  
ctrl+p 模糊搜索文件  
gT 切换到前一个tab  
gt 切换到后一个tab

可以在.vimrc里为标签页进行的配置，通过ctrl h/l切换标签等  
let mapleader = ','  
nnoremap <C-l> gt  
nnoremap <C-h> gT  
nnoremap <leader>t : tabe<CR>

" 设置NerdTree plugin

map <F3> :NERDTreeMirror<CR>

map <F3> :NERDTreeToggle<CR>

注：并非一定需要设F3,可以自行设定其他键

:NERDTree D:\ 更改盘符

o 打开关闭文件或者目录，如果是文件的话，光标出现在打开的文件中  
O 打开结点下的所有目录  
X 合拢当前结点的所有目录  
x 合拢当前结点的父目录

i和s水平分割或纵向分割窗口打开文件  
C 将选中目录或选中文件的父目录设为根结点  
u 将当前根结点的父目录设为根目录，并变成合拢原根结点  
U 将当前根结点的父目录设为根目录，但保持展开原根结点  
t 在标签页中打开  
T 在后台标签页中打开

p 到上层目录  
P 到根目录  
K 到同目录第一个节点  
J 到同目录最后一个节点  
m 显示文件系统菜单（添加、删除、移动操作）  
? 帮助  
:q 关闭

?: 快速帮助文档

o: 打开一个目录或者打开文件，创建的是buffer，也可以用来打开书签

go: 打开一个文件，但是光标仍然留在NERDTree，创建的是buffer

t: 打开一个文件，创建的是Tab，对书签同样生效

T: 打开一个文件，但是光标仍然留在NERDTree，创建的是Tab，对书签同样生效

i: 水平分割创建文件的窗口，创建的是buffer

gi: 水平分割创建文件的窗口，但是光标仍然留在NERDTree

s: 垂直分割创建文件的窗口，创建的是buffer

gs: 和gi，go类似

x: 收起当前打开的目录

X: 收起所有打开的目录

e: 以文件管理的方式打开选中的目录

D: 删除书签

P: 大写，跳转到当前根路径

p: 小写，跳转到光标所在的上一级路径

K: 跳转到第一个子路径

J: 跳转到最后一个子路径

<C-j>和<C-k>: 在同级目录和文件间移动，忽略子目录和子文件

C: 将根路径设置为光标所在的目录

u: 设置上级目录为根路径

U: 设置上级目录为跟路径，但是维持原来目录打开的状态

r: 刷新光标所在的目录

R: 刷新当前根路径

I: 显示或者不显示隐藏文件

f: 打开和关闭文件过滤器

q: 关闭NERDTree

A: 全屏显示NERDTree，或者关闭全屏

# a.vim 快捷键

自定义的快捷键:

ch : =:A 头文件／源文件切换

ih: 切换至光标所在文件\*

ic: 切换至光标所在处(单词所指)文件的配对文件(如光标所在处为foo.h，则切换至foo.c/foo.cpp...)

默认的快捷键:

1. 头/源文件切换命令
2. :A 头文件／源文件切换
3. :AS 分割窗后并切换头/源文件(切割为上下两个窗口)
4. :AV 垂直切割窗口后切换头/源文件(切割为左右两个窗口)
5. :AT 新建Vim标签式窗口后切换
6. :AN 在多个匹配文件间循环切换
7. 将光标所在处单词作为文件名打开
8. :IH 切换至光标所在文件
9. :IHS 分割窗口后切换至光标所在文件(指将光标所在处单词作为文件名打开)
10. :IHV 垂直分割窗口后切换
11. :IHT 新建标签式窗口后切换
12. :IHN 在多个匹配文件间循环切换
13. 快捷键操作
14. <Leader>ih 切换至光标所在文件\*
15. <Leader>is 切换至光标所在处(单词所指)文件的配对文件(如光标所在处为foo.h，则切换至foo.c/foo.cpp...)
16. <Leader>ihn 在多个匹配文件间循环切换

# cscope 快捷键

1. 建立cscope使用的索引文件  
         1. 在你需要浏览源码的根目录下（如你想用cscope看linux源码)使用下面命令：  
                \* #: cscope -Rbkq<回车>  
         2. R 表示把所有子目录里的文件也建立索引  
         3. b 表示cscope不启动自带的用户界面，而仅仅建立符号数据库  
         4. q生成cscope.in.out和cscope.po.out文件，加快cscope的索引速度  
         5. k在生成索引文件时，不搜索/usr/include目录  
   2. 在源码根目录下打开任意.c文件，使用如下命令：  
         1. Ctrl+]将跳到光标所在变量或函数的定义处 Ctrl+T返回  
         2. ：cs find s ---- 查找C语言符号，即查找函数名、宏、枚举值等出现的地方  
            ：cs find g ---- 查找函数、宏、枚举等定义的位置，类似ctags所提供的功能  
            ：cs find d ---- 查找本函数调用的函数  
            ：cs find c ---- 查找调用本函数的函数  
            ：cs find t: ---- 查找指定的字符串  
            ：cs find e ---- 查找egrep模式，相当于egrep功能，但查找速度快多了  
            ：cs find f ---- 查找并打开文件，类似vim的find功能  
            ：cs find i ---- 查找包含本文件的文  
         3. 2的所以命令也可以且按銉来实现：  
               1. \fs 表示：cs find s命令  
               2. 同理实现cs find + g,d,c,t,e,f,i命令