Vim基本使用

我接触linux的时候，用的是nano这个文本编辑器，个人觉得他的最大优点就是方便，快速。打开文件，修改文件，到退出都十分的快速，简单。这一点上比vi,vim要强，但是nano不能显示彩色字，不能进行整行删除，编辑各种配置文件的时候，总感觉不是很方便。针对于nano的这些缺点，vim可以很好的解决，并且vim打开文件，里面的字是有颜色的，这一点根editplus,eclipse是一样的，对于做程序的来说，就比较爽了。下面的操作，都是命令状态下进行的，不要进入插入状态了，如果进入了插入状态，[Esc]退出输入状态

# 一，光标移动

大家不要觉得光标移动不重要，其实它是基础，更好的光标移动，复制，粘贴，删除等才能更加的得心应手，进入了编辑器里面后，鼠标就不能用了。

|  |  |
| --- | --- |
| 光标移动 | |
| h 或 向左箭头键(←)  20h或者20(←) | 光标向左移动一个字符  光标向左移动20个字符, |
| j 或 向下箭头键(↓)  20j或者20(↓) | 光标向下移动一行  光标向下移动20行, |
| k 或 向上箭头键(↑)  20k或者20(↑) | 光标向上移动一行  光标向上移动20行, |
| l 或 向右箭头键(→)  20l或者20(→) | 光标向右移动一个字符  光标向右移动20字符, |
| 在这里要注意一点的是,不要用小键盘输入数字, 不然就会直接输入模式了.个人觉得掌握上面的这些,基本上能满足我们对光标移动的要求了,看下图  [vim](http://blog.51yip.com/wp-content/uploads/2010/11/vim.jpg)  vim我们可以根据编辑器右下角的数字来判断,我们要跳转到哪一行，如果想更精确的话，:set nu 设置一下环境变量，让它显示行号是最好的 | |
| Ctrl + f | 屏幕『向下』移动一页，相当于 [Page Down]按键 (常用) |
| Ctrl + b | 屏幕『向上』移动一页，相当于 [Page Up] 按键 (常用) |
| Ctrl + d | 屏幕『向下』移动半页 |
| Ctrl + u | 屏幕『向上』移动半页 |
| Ctrl + e | 屏幕『向下』移动一行 |
| Ctrl + y | 屏幕『向上』移动一行 |
| + | 光标移动到非空格符的下一列 |
| - | 光标移动到非空格符的上一列 |
| n<space> | 那个 n 表示『数字』，按下数字后再按空格键，光标会向右移动这一行的 n 个字符。例如 20<space> 则光标会向后面移动 20 个字符距离。 |
| n<Enter> | n 为数字。光标向下移动 n 行(常用) |
| 0 或功能键[Home] | 这是数字『 0 』：移动到这一行的最前面字符处 (常用) |
| $ 或功能键[End] | 移动到这一行的最后面字符处(常用),这里的$在正则里面表示是结尾的意思,这样理解一下就能记住 |
| H | 光标移动到这个屏幕的最上方那一行的第一个字符,H你就把它记成是header的缩写，这样就好记了 |
| M | 光标移动到这个屏幕的中夬那一行的第一个字符,M你就把它记成middle的缩写 |
| L | 光标移动到这个屏幕的最下方那一行的第一个字符,L你就把它记成last的缩写 |
| G | 移动到这个档案的最后一行(常用) |
| nG | n 为数字。移动到这个档案的第 n 行。例如 20G 则会移动到这个档案的第 20 行 |
| gg | 移动到这个档案的第一行，相当于 1G 啊！ (常用) |

# 二，删除，复制，粘贴，撤销

|  |  |
| --- | --- |
| 删除，复制，粘贴，撤销 | |
| x, X | 在一行字当中，x 为向后删除一个字符 (相当于 [del] 按键)， X 为向前删除一个字符(相当于 [backspace] 亦即是退格键) (常用) |
| nx  nX | n 为数字，连续向后删除 n 个字符。举例来说，我要连续删除光标后 10 个字符， 『10x』。  n 为数字，连续删除光标前面的 n 个字符。举例来说，我要连续删除光标前的 10 个字符， 『10X』。 |
| dd  yy | 删除光标所在的那一整行(常用)，  复制光标所在的那一行(常用) |
| ndd  nyy | n 为数字。删除光标所在行向下 n 行，例如 20dd 则是删除 20 行(常用)  n 为数字。复制光标所在行向下 n 行，例如 20yy 则是复制 20 行(常用) |
| d1G  y1G | 删除光标所在行到第一行的所有数据  复制光标所在行到第一行的所有数据 |
| dG  yG | 删除光标所在行到最后一行的所有数据  复制光标所在行到最后一行的所有数据 |
| d$  y$ | 删除光标所在处，到该行的最后一个字符  复制光标所在的那个字符到该行行尾的所有数据 |
| d0  y0 | 那个是数字的 0 ，删除光标所在处，到该行的最前面一个字符  复制光标所在的那个字符到该行行首的所有数据 |
| p, P | p将复制的数据，粘贴在光标的下一行，P将复制的数据,粘贴到光标的上一行 |
| J | 将光标所在行不下一行的数据结合成同一行 |
| c | 重复删除多个数据，可以通过上下键来决定删除光标上面的，还是下面的 |
| u | 撤销。(常用) |
| Ctrl + r | 撤销的撤销。(常用) |

# 三，区块选择，查找，替换

|  |  |
| --- | --- |
| 区块选择，查找，替换 | |
| v | 字符选择，会将光标经过的地方反白选择！ |
| V | 行选择，会将光标经过的行反白选择！ |
| Ctrl + v | 区块选择，可以用长方形的方式选择资料 |
| y | 将反白的地方复制起来 |
| d | 将反白的地方删除掉 |
| [vim](http://blog.51yip.com/wp-content/uploads/2010/11/vim_v_V.jpg)  v或者V | [ctrl+v](http://blog.51yip.com/wp-content/uploads/2010/11/vim_ctrl_v.jpg)  ctrl+v |
| 从上面可以看出,v和V主要的是进行,行选择,而Ctrl+v是进行的列选择，从灵活性上来说，ctrl+v更加的灵活一点。 | |
| /string  ?string | 向光标之下寻找一个名称为string字符串  向光标之上寻找一个名称为string字符串 |
| n  N | 正向查找，搜索出的string,可以理解成next  反向查找，搜索出的string,可以理解成Not next |
| [find](http://blog.51yip.com/wp-content/uploads/2010/11/vim_find.jpg)  find | 为了说明,n和N的区别,作了个例子,看左图,左图是搜索ddd所产生的结果，黄底黑字就是搜索的结果，这个时候，我们可以按n或者N,在这4个搜索区块内,进行切换.  n的顺序是12341234  N的顺序是43214321  总之一句话,n是从上到下，N是从下到上 |
| :n1,n2s/string1/string2/g | 这里的n1是查找的开始行数，n2是查找结束的行数。【:2,7s/ddd/fff/g】在第2行,第7行之间，将ddd替换成fff |
| :1,$s/string1/string2/g | 从第一行到最后一行寻找 string1 字符串，并将该字符串替换为 string2 ！(常用) |
| :1,$s/string1/string2/gc | 从第一行到最后一行寻找 string1 字符串，并将该字符串替换为 string2 ！且在替换前显示提示字符给用户确认 (confirm) 是否需要替换！(常用) |

# 四，保存，退出，环境变量

|  |  |
| --- | --- |
| 保存，退出，环境变量 | |
| :w  :w! | 将编辑的数据写入硬盘档案中(常用)  若文件属性为『只读』时，强制写入该档案。不过，到底能不能写入， 还是跟你对该档案的档案权限有关啊! |
| :q  :q! | 离开 vi (常用)  若曾修改过档案，又不想储存，使用 ! 为强制离开不储存档案。 |
| :wq  :wq! | 储存后离开  强制储存后离开 |
| ZZ | 这是大写的 Z 喔！若档案没有更动，则不储存离开，若档案已经被更动过，则储存后离开! |
| :w [filename] | 另存为 |
| :r [filename] | 在编辑的数据中，读入另一个档案的数据。亦即将 『filename』 这个档案内容加到光标所在行后面 |
| :! command | 暂时离开 vi 到指令列模式下执行 command 的显示结果！例如 『:! cat ./test』即可在 vi 当中察看当前文件夹中的test文件中的内容 |
| i,I,a,A,o,O,r,R | 进入插入模式，虽然有不同之处，感觉差别不大，就放在一起了 |
| Esc | 退出插入模式，进入命令模式 |
| :set nu | 显示行号，设定之后，会在每一行的前缀显示该行的行号,这一点，非常有用，当我们修改配置文件的时候，能知道多少行，就方便多了。 |
| :set nonu | 与 set nu 相反，取消行号 |

# 五，小结

个人觉得，上面的操作能满足，80%--90%的人的需要，如果对多文件进行操作，我想各种开发工具，要比vim更加的方便.