# 第一部份：一般指令模式可用的按钮说明，游标移动、复制贴上、搜寻取代等

## 一、移动游标的方法

|  |  |
| --- | --- |
| 移动游标的方法 | |
| h 或向左方向键(←) | 游标向左移动一个字元 |
| j 或向下方向键(↓) | 游标向下移动一个字元 |
| k 或向上方向键(↑) | 游标向上移动一个字元 |
| l 或向右方向键(→) | 游标向右移动一个字元 |
| 如果你将右手放在键盘上的话，你会发现hjkl 是排列在一起的，因此可以使用这四个按钮来移动游标。如果想要进行多次移动的话，例如向下移动30 行，可以使用"30j" 或"30↓" 的组合按键， 亦即加上想要进行的次数(数字)后，按下动作即可！ | |
| [Ctrl] + [f] | 萤幕『向下』移动一页，相当于[Page Down]按键( 常用 ) |
| [Ctrl] + [b] | 萤幕『向上』移动一页，相当于[Page Up]按键( 常用 ) |
| [Ctrl] + [d] | 萤幕『向下』移动半页 |
| [Ctrl] + [u] | 萤幕『向上』移动半页 |
| + | 游标移动到非空白字元的下一行 |
| - | 游标移动到非空白字元的上一行 |
| n<space> | 那个n 表示『数字』，例如20 。按下数字后再按空白键，游标会向右移动这一行的n 个字元。例如20<space> 则游标会向后面移动20 个字元距离。 |
| 0 或功能键[Home] | 这是数字『 0 』：移动到这一行的最前面字元处( 常用 ) |
| $ 或功能键[End] | 移动到这一行的最后面字元处( 常用 ) |
| H | 游标移动到这个萤幕的最上方那一行的第一个字元 |
| M | 游标移动到这个萤幕的中央那一的第一个字元 |
| L | 游标移动到这个萤幕的最下方那一行的第一个字元 |
| 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(可配合:set nu) |
| gg | 移动到这个档案的第一行，相当于1G啊！( 常用 ) |
| n<Enter> | n为数字。游标向下移动n行( 常用 ) |

## 二、搜寻与取代

|  |  |
| --- | --- |
| 搜寻与取代 | |
| /word | 向游标之下寻找一个名称为word的字串。例如要在档案内搜寻vbird这个字串，就输入/vbird即可！( 常用 ) |
| ?word | 向游标之上寻找一个字串名称为word 的字串。 |
| n | 这个n是英文按键。代表『重复前一个搜寻的动作』。举例来说，如果刚刚我们执行/vbird去向下搜寻vbird这个字串，则按下n后，会向下继续搜寻下一个名称为vbird的字串。如果是执行?vbird的话，那么按下n则会向上继续搜寻名称为vbird的字串！ |
| N | 这个N 是英文按键。与n 刚好相反，为『反向』进行前一个搜寻动作。例如/vbird 后，按下N 则表示『向上』搜寻vbird 。 |
| 使用/word 配合n 及N 是非常有帮助的！可以让你重复的找到一些你搜寻的关键字！ | |
| :n1,n2s/word1/word2/g | n1与n2为数字。在第n1与n2行之间寻找word1这个字串，并将该字串取代为word2 ！举例来说，在100到200行之间搜寻vbird并取代为VBIRD则： 『:100,200s/vbird/VBIRD/g』。( 常用 ) |
| :1,$s/word1/word2/g | 从第一行到最后一行寻找word1字串，并将该字串取代为word2 ！( 常用 ) |
| :1,$s/word1/word2/gc | 从第一行到最后一行寻找word1字串，并将该字串取代为word2 ！且在取代前显示提示字元给使用者确认(confirm)是否需要取代！( 常用 ) |

## 三、删除、复制与贴上

|  |  |
| --- | --- |
| 删除、复制与贴上 | |
| x, X | 在一行字当中，x为向后删除一个字元(相当于[del]按键)， X为向前删除一个字元(相当于[backspace]亦即是倒退键) ( 常用 ) |
| nx | n 为数字，连续向后删除n 个字元。举例来说，我要连续删除10 个字元， 『10x』。 |
| dd | 删除游标所在的那一整行( 常用 ) |
| ndd | n为数字。删除游标所在的向下n行，例如20dd则是删除20行( 常用 ) |
| d1G | 删除游标所在行到第一行的所有资料 |
| dG | 删除游标所在到最后一行的所有资料 |
| d$ | 删除游标所在处，到该行的最后一个字元 |
| d0 | 那个是数字的0 ，删除游标所在处，到该行的最前面一个字元 |
| yy | 复制游标所在的那一行( 常用 ) |
| nyy | n为数字。复制游标所在的向下n行，例如20yy则是复制20行( 常用 ) |
| y1G | 复制游标所在行到第一行的所有资料 |
| yG | 复制游标所在行到最后一行的所有资料 |
| y0 | 复制游标所在的那个字元到该行行首的所有资料 |
| y$ | 复制游标所在的那个字元到该行行尾的所有资料 |
| p, P | p为将已复制的资料在游标下一行贴上，P则为贴在游标上一行！举例来说，我目前游标在第20行，且已经复制了10行资料。则按下p后，那10行资料会贴在原本的20行之后，亦即由21行开始贴。但如果是按下P呢？那么原本的第20行会被推到变成30行。( 常用 ) |
| J | 将游标所在行与下一行的资料结合成同一行 |
| c | 重复删除多个资料，例如向下删除10 行，[ 10cj ] |
| u | 复原前一个动作。( 常用 ) |
| [Ctrl]+r | 重做上一个动作。( 常用 ) |
| 这个u 与[Ctrl]+r 是很常用的指令！一个是复原，另一个则是重做一次～ 利用这两个功能按键，你的编辑，嘿嘿！很快乐的啦！ | |
| . | 不要怀疑！这就是小数点！意思是重复前一个动作的意思。如果你想要重复删除、重复贴上等等动作，按下小数点『.』就好了！( 常用 ) |

# 第二部份：一般指令模式切换到编辑模式的可用的按钮说明

## 进入插入或取代的编辑模式

|  |  |
| --- | --- |
| 进入插入或取代的编辑模式 | |
| i, I | 进入插入模式(Insert mode)： i为『从目前游标所在处插入』， I为『在目前所在行的第一个非空白字元处开始插入』。( 常用 ) |
| a, A | 进入插入模式(Insert mode)： a为『从目前游标所在的下一个字元处开始插入』， A为『从游标所在行的最后一个字元处开始插入』。( 常用 ) |
| o, O | 进入插入模式(Insert mode)： 这是英文字母o的大小写。o为『在目前游标所在的下一行处插入新的一行』； O为在目前游标所在处的上一行插入新的一行！( 常用 ) |
| r, R | 进入取代模式(Replace mode)： r只会取代游标所在的那一个字元一次；R会一直取代游标所在的文字，直到按下ESC为止；( 常用 ) |
| 上面这些按键中，在vi 画面的左下角处会出现『--INSERT--』或『--REPLACE--』的字样。由名称就知道该动作了吧！！特别注意的是，我们上面也提过了，你想要在档案里面输入字元时， 一定要在左下角处看到INSERT 或REPLACE 才能输入喔！ | |
| [Esc] | 退出编辑模式，回到一般指令模式中( 常用 ) |

# 第三部份：一般指令模式切换到指令行模式的可用按钮说明

## 一、指令行模式的储存、离开等指令

|  |  |
| --- | --- |
| 指令行模式的储存、离开等指令 | |
| :w | 将编辑的资料写入硬碟档案中( 常用 ) |
| :w! | 若档案属性为『唯读』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 | 离开vi ( 常用 ) |
| :q! | 若曾修改过档案，又不想储存，使用! 为强制离开不储存档案。 |
| 注意一下啊，那个惊叹号(!) 在vi 当中，常常具有『强制』的意思～ | |
| :wq | 储存后离开，若为:wq!则为强制储存后离开( 常用 ) |
| 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] | 将编辑的资料储存成另一个档案（类似另存新档） |
| :r [filename] | 在编辑的资料中，读入另一个档案的资料。亦即将『filename』 这个档案内容加到游标所在行后面 |
| :n1,n2 w [filename] | 将n1 到n2行的内容储存成filename 这个档案。 |
| :! command | 暂时离开vi到指令行模式下执行command的显示结果！例如 『:! ls /home』即可在vi当中察看/home底下以ls输出的档案资讯！ |

## vim 环境的变更

|  |  |
| --- | --- |
| vim 环境的变更 | |
| :set nu | 显示行号，设定之后，会在每一行的字首显示该行的行号 |
| :set nonu | 与set nu 相反，为取消行号！ |

# 第四部分：区块选择

刚刚我们提到的简单的vi 操作过程中，几乎提到的都是以列为单位的操作。那么如果我想要搞定的是一个区块范围呢？举例来说，像底下这种格式的档案：

|  |
| --- |
| 192.168.1.1 host1.class.net  192.168.1.2 host2.class.net  192.168.1.3 host3.class.net  192.168.1.4 host4.class.net .....中间省略...... |

这个档案我将他放置到<http://linux.vbird.org/linux_basic/0310vi/hosts> ，你可以自行下载来看一看这个档案啊！现在我们来玩一玩这个档案吧！假设我想要将host1, host2...等等复制起来，并且加到每一列的后面，亦即每一列的结果要是『 192.168.1.2 host2.class.net host2 』这样的情况时，在传统或现代的视窗型编辑器似乎不容易达到这个需求，但是咱们的vim是办的到的喔！那就使用区块选择(Visual Block)吧！当我们按下v或者V或者[Ctrl]+v时，这个时候游标移动过的地方就会开始反白，这三个按键的意义分别是：

|  |  |
| --- | --- |
| 区块选择的按键意义 | |
| v | 字元选择，会将游标经过的地方反白选择！ |
| V | 列选择，会将游标经过的列反白选择！ |
| [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 |
| p | 将刚刚复制的区块，在游标所在处贴上！ |

# 第五部分：多档案编辑

假设一个例子，你想要将刚刚我们的hosts 内的IP 复制到你的/etc/hosts 这个档案去， 那么该如何编辑？我们知道在vi 内可以使用:r filename 来读入某个档案的内容， 不过，这样毕竟是将整个档案读入啊！如果我只是想要部分内容呢？呵呵！这个时候多档案同时编辑就很有用了。我们可以使用vim 后面同时接好几个档案来同时开启喔！相关的按键有：

|  |  |
| --- | --- |
| 多档案编辑的按键 | |
| :n | 编辑下一个档案 |
| :N | 编辑上一个档案 |
| :files | 列出目前这个vim 的开启的所有档案 |

# 第六部分：多视窗功能

在开始这个小节前，先来想像两个情况：

* 当我有一个档案非常的大，我查阅到后面的资料时，想要『对照』前面的资料， 是否需要使用[ctrl]+f 与[ctrl]+b (或pageup, pagedown 功能键) 来跑前跑后查阅？
* 我有两个需要对照着看的档案，不想使用前一小节提到的多档案编辑功能；

在一般视窗介面下的编辑软体大多有『分割视窗』或者是『冻结视窗』的功能来将一个档案分割成多个视窗的展现，那么vim能不能达到这个功能啊？可以啊！但是如何分割视窗并放入档案呢？很简单啊！在指令列模式输入『:sp {filename}』即可！那个filename可有可无， 如果想要在新视窗启动另一个档案，就加入档名，否则仅输入:sp时，出现的则是同一个档案在两个视窗间！

让我们来测试一下，你先使用『 vim /etc/man\_db.conf 』打开这个档案，然后『 1G 』去到第一列，之后输入『 :sp 』 再次的打开这个档案一次，然后再输入『 G 』，结果会变成底下这样喔：

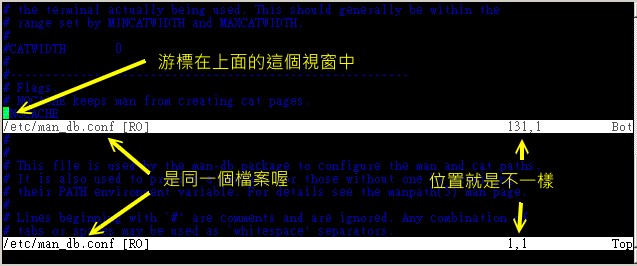


图9.3.7、vim 的视窗分割示意图

万一你再输入『 :sp /etc/hosts 』时，就会变成下图这样喔：



图9.3.8、vim 的视窗分割示意图

怎样？帅吧！两个档案同时在一个萤幕上面显示，你还可以利用『[ctrl]+w+↑』及『[ctrl]+w+↓』 在两个视窗之间移动呢！这样的话，复制啊、查阅啊等等的，就变的很简单啰～ 分割视窗的相关指令功能有很多，不过你只要记得这几个就好了：

|  |  |
| --- | --- |
| 多视窗情况下的按键功能 | |
| :sp [filename] | 开启一个新视窗，如果有加filename， 表示在新视窗开启一个新档案，否则表示两个视窗为同一个档案内容(同步显示)。 |
| [ctrl]+w+ j  [ctrl]+w+↓ | 按键的按法是：先按下[ctrl] 不放， 再按下w 后放开所有的按键，然后再按下j (或向下方向键)，则游标可移动到下方的视窗。 |
| [ctrl]+w+ k  [ctrl]+w+↑ | 同上，不过游标移动到上面的视窗。 |
| [ctrl]+w+ q | 其实就是:q 结束离开啦！举例来说，如果我想要结束下方的视窗，那么利用[ctrl]+w+↓ 移动到下方视窗后，按下:q 即可离开， 也可以按下[ctrl]+w+q 啊！ |

鸟哥第一次玩vim 的分割视窗时，真是很高兴啊！竟然有这种功能！太棒了！^\_^

# 第七部分：的挑字补全功能

我们知道bash的环境底下可以按下[tab]按钮来达成指令/参数/档名的补全功能，而我们也知道很多的程式编辑器，例如鸟哥用来在windows系统上面教网页设计、java script等很好用的notepad++ ( [https://notepad-plus-plus.org/](https://notepad-plus-plus.org/" \t "http://linux.vbird.org/linux_basic/_blank) )这种类的程式编辑器，都会有(1)可以进行语法检验及(2)可以根据副档名来挑字的功能！这两个功能对于程式设计者来说，是很有帮助的！毕竟偶尔某些特定的关键字老是背不起来...

在语法检验方面，vim 已经使用颜色来达成了！这部份不用伤脑筋的！比较伤脑筋的应该是在挑字补全上面！就是上面谈到的可以根据语法来挑选可能的关键字， 包括程式语言的语法以及特定的语法关键字等等。既然notepad ++ 都有支援了，没道理vim 不支援吧？呵呵！没错！是有支援的～只是你可能要多背两个组合按钮就是了！

鸟哥建议可以记忆的主要vim 补齐功能，大致有底下几个：

|  |  |
| --- | --- |
| 组合按钮 | 补齐的内容 |
| [ctrl]+x -> [ctrl]+n | 透过目前正在编辑的这个『档案的内容文字』作为关键字，予以补齐 |
| [ctrl]+x -> [ctrl]+f | 以当前目录内的『档名』作为关键字，予以补齐 |
| [ctrl]+x -> [ctrl]+o | 以副档名作为语法补充，以vim 内建的关键字，予以补齐 |

在鸟哥的认知中，比较有用的是第1, 3 这两个组合键，第一个组合按键中，你可能会在同一个档案里面重复出现许多相同的关键字， 那么就能够透过这个补全的功能来处理。如果你是想要使用vim 内建的语法检验功能来处理取得关键字的补全，那么第三个项目就很有用了。不过要注意，如果你想要使用第三个功能，就得要注意你编辑的档案的副档名。我们底下来做个简单测试好了。

假设你想要编写网页，正要使用到CSS 的美化功能时，突然想到有个背景的东西要处理，但是突然忘记掉背景的CSS 关键语法，那可以使用如下的模样来处置！请注意，一定要使用.html 或.php 的副档名，否则vim 不会呼叫正确的语法检验功能喔！因此底下我们建立的档名为html.html 啰！

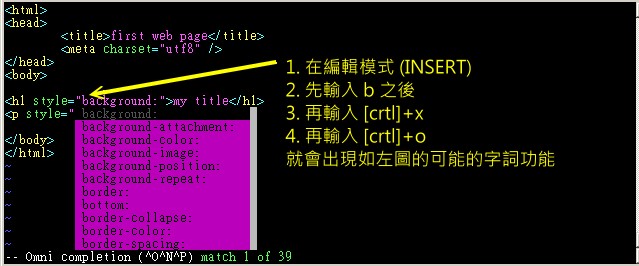


图9.3.9、vim 的挑字补全功能

由于网页通常会支援CSS 的语法，而CSS 的美化语法使用的是style 这个关键字，这个关键字后面接的就是CSS 的元素与元素值。若想要取得可能的元素有哪些，例如背景(background) 的语法中，想要了解有哪些跟它有关的内建元素，如上图， 直接输入b 然后按下[ctrl]+x 再按下[ ctrl]+o 就会出现如上的相关字词可以选择，此时你就能够使用上下按钮来挑选所需要的关键元素！这样使用上当然方便很多啊！只是要注意，一定要使用正确的副档名，否则会无法出现任何关键字词喔！