



## 第十章、vim 程序编辑器

最近更新日期：2008/01/13

系统管理员的重要工作就是要修改与设定某些重要软件的配置文件，因此至少得要学会一种以上的文字接口的文书编辑器。在所有版本的 Linux 上头都会有的一套文书编辑器就是 vi，而且很多软件默认也是使用 vi 做为他们编辑的接口，因此鸟哥建议您务必要学会使用 vi 这个正规的文书编辑器。此外，vim 是进阶版的 vi，vim 不但可以用不同颜色显示文字内容，还能够进行诸如 shell script, C program 等程序编辑功能，你可以将 vim 视为一种程序编辑器！鸟哥也是用 vim 编辑鸟站的网页文章呢！ ^\_^

1. vi 与 vim: 为何要学 vim ?
2. vi 的使用
  - 2.1 简易执行范例
  - 2.2 按键说明
  - 2.3 一个案例的练习
  - 2.4 vim 的暂存档、救援回复与开启时的警告讯息
3. vim 的额外功能
  - 3.1 区块选择(Visual Block)
  - 3.2 多档案编辑
  - 3.3 多窗口功能
  - 3.4 vim 环境设定与记录: `~/.vimrc`, `~/.viminfo`
4. 其他 vim 使用注意事项
  - 4.1 中文编码的问题
  - 4.2 DOS 与 Linux 的断行字符: `dos2unix`, `unix2dos`
  - 4.3 语系编码转换: `iconv`
5. 重点回顾
6. 本章习题
7. 参考数据与延伸阅读
8. 针对本文的建议: <http://phorum.vbird.org/viewtopic.php?t=23883>



### vi 与 vim: 为何要学 vim ?

由前面一路走来，我们一直建议使用文本模式来处理 Linux 的系统设定问题，因为不但可以让你比较容易了解到 Linux 的运作状况，也比较容易了解整个设定的基本精神，更能『保证』你的修改可以顺利的被运作。所以，在 Linux 的系统中使用文本编辑器来编辑你的 Linux 参数配置文件，可是一件很重要的事情啦！也因此呢，系统管理员至少应该要熟悉一种字处理器的！

### Tips:

这里要再次的强调，不同的 Linux distribution 各有其不同的附加软件，例如 Red Hat Enterprise Linux 与 Fedora 的 `ntsysv` 与 `setup` 等，而 SuSE 则有 `YOU` 管理工具等等，因此，如果你只会使用此种类型的软件来控制你的 Linux 系统时，当接管不同的 Linux distributions 时，呵呵！那可就苦恼了！



在 Linux 的世界中，绝大部分的配置文件都是以 ASCII 的纯文本形态存在，因此利用简单的文字编辑软件就能够修改设定了！与微软的 Windows 系统不同的是，如果你用惯了 Microsoft Word 或 Corel Wordperfect 的话，那么除了 X window 里面的图形接口编辑程序(如 `xemacs`)用起来尚可应付外，在 Linux 的文本模式下，会觉得文书编辑程序都没有窗口接口来的直观与方便。

### Tips:

什么是纯文本档？其实档案记录的就是 0 与 1，而我们透过编码系统来将这些 0 与 1 转成我们认识的文字就是了。在第零章里面的数据表示方式有较多说明，请自行查阅。ASCII 就是其中一种广为使用的文字编码系统，在 ASCII 系统中的图标与代码可以参考 <http://zh.wikipedia.org/wiki/ASCII> 呢！



那么 Linux 在文字接口下的文书编辑器有哪些呢？其实有非常多喔！常常听到的就有：`emacs`，`pico`，`nano`，`joe`，与 `vim` 等等(注 1)。既然有这么文字接口的文书编辑器，那么我们为什么一定要学 `vi` 啊？还有那个 `vim` 是做啥用的？底下就来谈一谈先！

- 
- 为何要学 `vim`？

文书编辑器那么多，我们之前在第五章也曾经介绍过那简单好用的 `nano`，既然已经学会了 `nano`，干嘛鸟哥还一直要你学这不是很友善的 `vi` 呢？其实是有原因的啦！因为：

- 所有的 Unix Like 系统都会内建 `vi` 文书编辑器，其他的文书编辑器则不一定会存在；
- 很多个别软件的编辑接口都会主动呼叫 `vi`（例如未来会谈到的 `crontab`，`visudo`，`edquota` 等指令）；
- `vim` 具有程序编辑的能力，可以主动的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。

其实重点是上述的第二点，因为有太多 Linux 上面的指令都默认使用 `vi` 作为数据编辑的接口，所以你必须、一定要学会 `vi`，否则很多指令你根本就无法

操作呢！这样说，有刺激到你务必要学会 vi 的热情了吗？ ^\_^

那么什么是 vim 呢？其实你可以将 vim 视作 vi 的进阶版本，vim 可以用颜色或底线等方式来显示一些特殊的信息。举例来说，当你使用 vim 去编辑一个 C 程序语言的档案，或者是我们后续会谈到的 `shell script` 程序时，vim 会依据档案的扩展名或者是档案内的开头信息，判断该档案的内容而自动的呼叫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。也就是说，这个 vim 是个『程序编辑器』啦！甚至一些 Linux 基础配置文件内的语法，都能够用 vim 来检查呢！例如我们在第八章谈到的 `/etc/fstab` 这个档案的内容。

简单的来说，vi 是老式的字处理器，不过功能已经很齐全了，但是还是有可以进步的地方。vim 则可以说是程序开发者的一项很好用的工具，就连 vim 的官方网站 (<http://www.vim.org>) 自己也说 vim 是一个『程序开发工具』而不是文字处理软件~^\_^。因为 vim 里面加入了很多额外的功能，例如支持正规表示法的搜寻架构、多档案编辑、区块复制等等。这对于我们在 Linux 上面进行一些配置文件的修订工作时，是很棒的一项功能呢！

#### Tips:

什么时候会使用到 vim 呢？其实鸟哥的整个网站都是在 vim 的环境下一字一字的建立起来的喔！早期鸟哥使用网页制作软件在编写网页，但是老是发现网页编辑软件都不怎么友善，尤其是写到 PHP 方面的程序代码时。后来就干脆不使用所见即所得的编辑软件，直接使用 vim，然后标签 (tag) 也都自行用键盘输入！这样整个档案也比较干净！所以说，鸟哥我是很喜欢 vim 的啦！ ^\_^



底下鸟哥会先就简单的 vi 做个介绍，然后再跟大家报告一下 vim 的额外功能与用法呢！



### vi 的使用

基本上 vi 共分为三种模式，分别是『一般模式』、『编辑模式』与『指令列命令模式』。这三种模式的作用分别是：

般模式：

vi 打开一个档案就直接进入一般模式了(这是默认的模式)。在这个模式中，你可以使用『上下左右』按键来移动光标，你可以使用『删除字符』或『删除整行』处理档案内容，也可以使用『复制、贴上』来处理你的文件数据。

辑模式：

一般模式中可以进行删除、复制、贴上等等的动作，但是却无法编辑文件内容的！

等到你按下『i, I, o, O, a, A, r, R』等任何一个字母之后才会进入编辑模式。

注意了！通常在 Linux 中，按下这些按键时，在画面的左下方会出现『INSERT 或 PLACE』的字样，此时才可以进行编辑。而如果要回到一般模式时，则必须要按下『Esc』这个按键即可退出编辑模式。

令列命令模式：

一般模式当中，输入『: / ?』三个中的任何一个按钮，就可以将光标移动到底下那一行。在这个模式当中，可以提供你『搜寻资料』的动作，而读取、存、大量取代字符、离开 vi、显示行号等等的动作则是在此模式中达成的！

简单的说，我们可以将这三个模式想成底下的图标来表示：

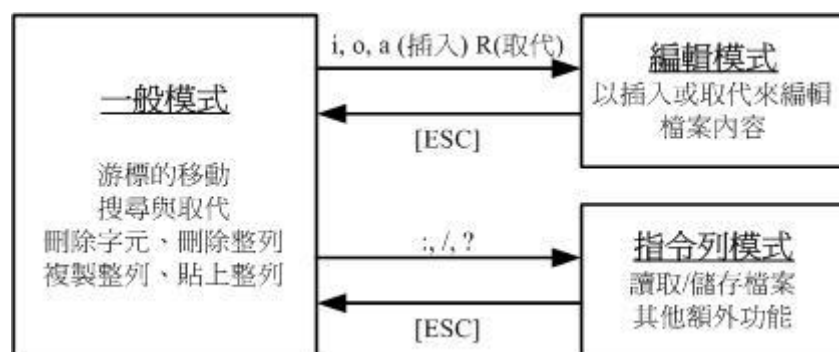


图 2.1、vi 三种模式的相互关系

注意到上面的图标，你会发现一般模式可与编辑模式及指令列模式切换，但编辑模式与指令列模式之间不可互相切换喔！这非常重要啦！闲话不多说，我们底下以一个简单的例子来进行说明吧！

---

### 🐦 简易执行范例

如果你想要使用 vi 来建立一个名为 test.txt 的档案时，你可以这样做：

用 vi 进入一般模式；

```
[root@www ~]# vi test.txt
```

接输入『vi 档名』就能够进入 vi 的一般模式了。请注意，记得 vi 后面一定加档名，不管该档名存在与否！整个画面主要分为两部份，上半部与最底下一两者可以视为独立的。如下图 2.1.1 所示，图中那个虚线是不存在的，鸟哥用说明而已啦！上半部显示的是档案的实际内容，最底下一行则是状态显示列(如图的[New File]信息)，或者是命令下达列喔！

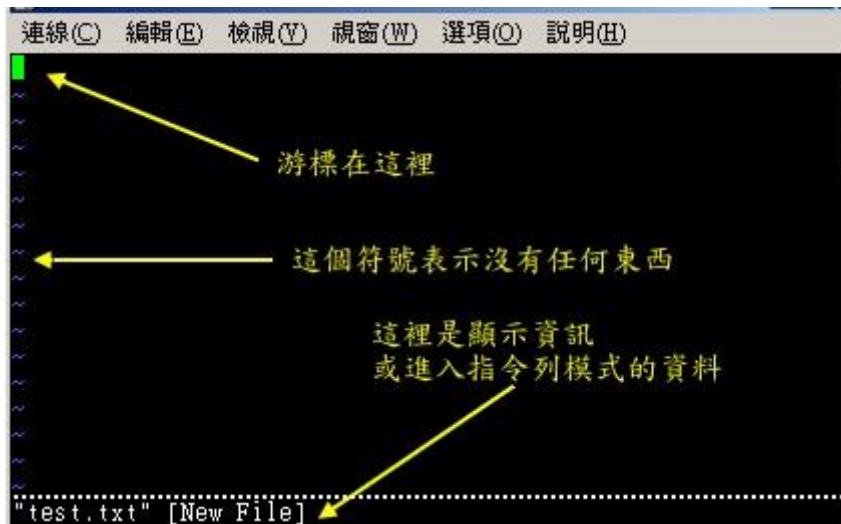


图 2.1.1、用 vi 开启一个新档案

果你开启的档案是旧档(已经存在的档案)，则可能会出现如下的信息：

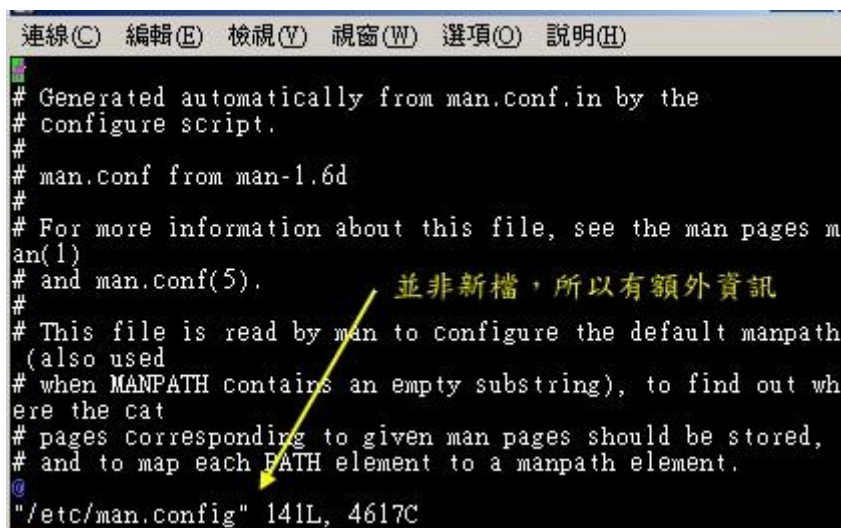


图 2.1.2、用 vi 开启一个旧档案

上图 2.1.2 所示，箭头所指的那个『"/etc/man.config" 141L, 4617C』代表的『档名为 /etc/man.conf，档案内有 141 行 以及具有 4617 个字符』的意思！一行的内容并不是在档案内，而是 vi 显示一些信息的地方喔！此时是在一般模的环境下啦。接下来开始来输入吧！

下 i 进入编辑模式，开始编辑文字

一般模式之中，只要按下 i, o, a 等字符就可以进入编辑模式了！在编辑模式中，你可以发现在左下角状态栏中会出现 - INSERT- 的字样，那就是可以输入意字符的提示啰！这个时候，键盘上除了 [Esc] 这个按键之外，其他的按键都以视作为一般的输入按钮了，所以你可以进行任何的编辑啰！

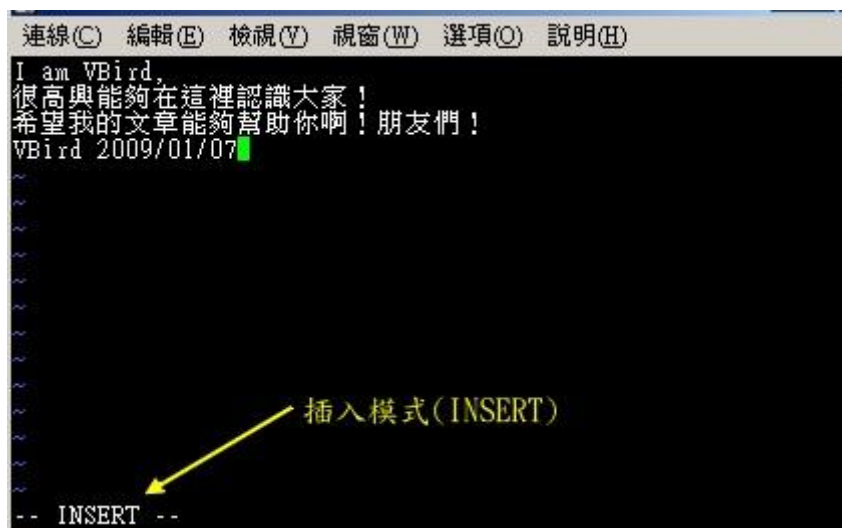


图 2.1.3、开始用 vi 来进行编辑

**Tips:**

在 vi 里面，[tab] 这个按钮所得到的结果与空格符所得到的结果是不一样的，特别强调一下！



下 [ESC] 按钮回到一般模式

了，假设我已经按照上面的样式给他编辑完毕了，那么应该要如何退出呢？是的！错！就是给他按下 [Esc] 这个按钮即可！马上你就会发现画面左下角的 -- INSERT -- 不见了！

一般模式中按下 :wq 储存后离开 vi

，我们要存档了，存盘并离开的指令很简单，输入『:wq』即可存档离开！（注了，按下：该光标就会移动到底下一行去！）这时你在提示字符后面输入『ls』即可看到我们刚刚建立的 test.txt 档案啦！整个图示有点像底下这样：



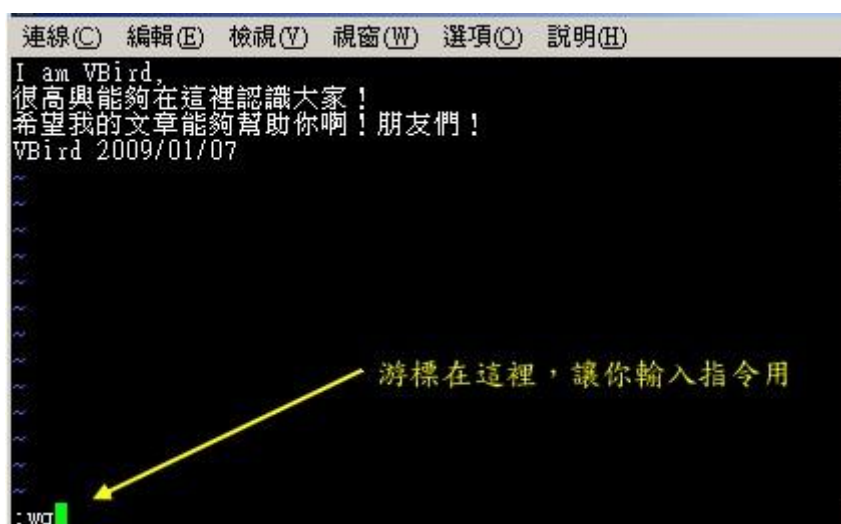


图 2.1.4、储存并离开 vi 环境

如此一来，你的档案 test.txt 就已经建立起来啰！需要注意的是，如果你的档案权限不对，例如为 -r--r--r-- 时，那么可能会无法写入，此时可以使用『强制写入』的方式吗？可以！使用『 :wq! 』多加一个惊叹号即可！不过，需要特别注意呦！那个是在『你的权限可以改变』的情况下才能成立的！关于权限的概念，请自行回去翻一下[第六章](#)的内容吧！

---

### 按键说明

除了上面简易范例的 i, [Esc], :wq 之外，其实 vim 还有非常多的按键可以使用喔！在介绍之前还是要再次强调，vim 的三种模式只有一般模式可以与编辑、指令列模式切换，编辑模式与指令列模式之间并不能切换的！这点在[图 2.1](#)里面有介绍到，注意去看看喔！底下就来谈谈 vim 软件中会用到的按键功能吧！

- 
- 第一部份：一般模式可用的按钮说明，光标移动、复制贴上、搜寻取代等

移动光标的方法	
h 或 向左箭头键(←)	光标向左移动一个字符
j 或 向下箭头键(↓)	光标向下移动一个字符
k 或 向上箭头键(↑)	光标向上移动一个字符
l 或 向右箭头键(→)	光标向右移动一个字符
如果你将右手放在键盘上的话，你会发现 h i k l 是排列在一起的，因此可	

以使用这四个按钮来移动光标。 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]	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
+	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』，例如 20 。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。
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)
H	光标移动到这个屏幕的最上方那一行的第一个字符
M	光标移动到这个屏幕的中央那一行的第一个字符
L	光标移动到这个屏幕的最下方那一行的第一个字符
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。光标向下移动 n 行(常用)
搜寻与取代	
/word	向光标之下寻找一个名称为 word 的字符串。例如要在档案内搜寻 vbird 这个字符串，就输入 /vbird 即可！（常用）
?word	向光标之上寻找一个字符串名称为 word 的字符



	串。
n	这个 n 是英文按键。代表『 <u>重复前一个搜寻的动作</u> 』。举例来说， 如果刚刚我们执行 /vbird 去向下搜寻 vbird 这个字符串，则按下 n 后，会向下继续搜寻下一个名称为 vbird 的字符串。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 的字符串！
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词！	
:n1,n2s/word1/word2/g	n1 与 n2 为数字。在第 n1 与 n2 行之间寻找 word1 这个字符串，并将该字符串取代为 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 字符串，并将该字符串取代为 word2 ！（常用）
:1,\$s/word1/word2/gc	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2 ！且在取代前显示提示字符给用户确认（confirm）是否需要取代！（常用）
删除、复制与贴上	
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	n 为数字，连续向后删除 n 个字符。举例来说，我要连续删除 10 个字符，『10x』。
dd	删除游标所在的那一整列（常用）
ndd	n 为数字。删除光标所在的向下 n 列，例如 20dd 则是删除 20 列（常用）
d1G	删除光标所在到第一行的所有数据
dG	删除光标所在到最后一行的所有数据
d\$	删除游标所在处，到该行的最后一个字符
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符
yy	复制游标所在的那一行（常用）

nyy	n 为数字。复制光标所在的向下 n 列,例如 20yy 则是复制 20 列(常用)
y1G	复制光标所在列到第一列的所有数据
yG	复制光标所在列到最后一列的所有数据
y0	复制光标所在的那个字符到该行行首的所有数据
y\$	复制光标所在的那个字符到该行行尾的所有数据
p, P	p 为将已复制的数据在光标下一行贴上, P 则为贴在光标上一行! 举例来说,我目前光标在第 20 行,且已经复制了 10 行数据。则按下 p 后, 那 10 行数据会贴在原本的 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	将光标所在列与下一列的数据结合成同一列
c	重复删除多个数据, 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	
i, I	进入插入模式(Insert mode): i 为『从目前光标所在处插入』, I 为『在目前所在行的第一个非空格符处开始插入』。(常用)
a, A	进入插入模式(Insert mode): a 为『从目前光标所在的下一个字符处开始插入』, A 为『从光标所在行的最后一个字符处开始插入』。(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『在目前光标所在的下一行处插入新的一行』；O 为在目前光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 只会取代光标所在的那一个字符一次；R 会一直取代光标所在的文字，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符时，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，回到一般模式中(常用)

- 第三部份：一般模式切换到指令列模式的可用的按钮说明

指令列的储存、离开等指令	
:w	将编辑的数据写入硬盘档案中(常用)
:w!	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	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]	将编辑的数据储存成另一个档案（类似另存新档）
:r [filename]	在编辑的数据中，读入另一个档案的数据。亦即将『filename』这个档案内容加到游标所在行后面
:n1,n2 w [filename]	将 n1 到 n2 的内容储存成 filename 这个档案。
:! command	暂时离开 vi 到指令列模式下执行 command 的显

	示结果！例如 『:! ls /home』即可在 vi 当中察看 /home 底下以 ls 输出的档案信息！
vim 环境的变更	
: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
:set nonu	与 set nu 相反，为取消行号！

特别注意，在 vi 中，『数字』是很有意义的！数字通常代表重复做几次的意思！也有可能是代表去到第几个什么什么的意思。举例来说，要删除 50 行，则是用『50dd』对吧！数字加在动作之前～那我要向下移动 20 行呢？那就是『20j』或者是『20↓』即可。

OK!会这些指令就已经很厉害了，因为常用到的指令也只有不到一半！通常 vi 的指令除了上面鸟哥注明的常用的几个外，其他是不用背的，你可以做一张简单的指令表在你的屏幕墙上，一有疑问可以马上查询啦！这也是当初鸟哥使用 vim 的方法啦！

---

### 一个案例练习

来来来！赶紧测试一下你是否已经熟悉 vi 这个指令呢？请依照底下的需求进行指令动作。（底下的操作为使用 CentOS 5.2 中的 man.config 来做练习的，该档案你可以在这里下载：

[http://linux.vbird.org/linux\\_basic/0310vi/man.config](http://linux.vbird.org/linux_basic/0310vi/man.config)。）看看你的显示结果与鸟哥的结果是否相同啊？

1. 请在 /tmp 这个目录下建立一个名为 vitest 的目录；
2. 进入 vitest 这个目录当中；
3. 将 /etc/man.config 复制到本目录下(或由上述的连结下载 [man.config](#) 档案)；
4. 使用 vi 开启本目录下的 man.config 这个档案；
5. 在 vi 中设定一下行号；
6. 移动到第 58 行，向右移动 40 个字符，请问你看到的双引号内是什么目录？
7. 移动到第一行，并且向下搜寻一下『 bzip2 』这个字符串，请问他在第几行？
8. 接着下来，我要将 50 到 100 行之间的『小写 man 字符串』改为『大写 MAN 字符串』，并且一个一个挑选是否需要修改，如何下达指令？如果在挑选过程中一直按『y』，结果会在最后一行出现改变了几个 man 呢？

9. 修改完之后，突然反悔了，要全部复原，有哪些方法？
10. 我要复制 65 到 73 这九行的内容(含有 MANPATH\_MAP)，并且贴到最后一行之后；
11. 21 到 42 行之间的开头为 # 符号的批注数据我不要了，要如何删除？
12. 将这个档案另存成一个 man.test.config 的檔名；
13. 去到第 27 行，并且删除 15 个字符，结果出现的第一个单字是什么？
14. 在第一行新增一行，该行内容输入『I am a student...』；
15. 储存后离开吧！

整个步骤可以如下显示：

1. 『mkdir /tmp/vitest』
2. 『cd /tmp/vitest』
3. 『cp /etc/man.config .』
4. 『vi man.config』
5. 『:set nu』然后你会在画面中看到左侧出现数字即为行号。
6. 先按下『58G』再按下『40→』会看到『/dir/bin/foo』这个字样在双引号内；
7. 先执行『1G』或『gg』后，直接输入『/bzip2』，则会去到第 118 行才对！
8. 直接下达『:50,100s/man/MAN/gc』即可！若一直按『y』最终会出现『在 23 行内置换 25 个字符串』的说明。
9. (1)简单的方法可以一直按『u』回复到原始状态，(2)使用不储存离开『:q!』之后，再重新读取一次该档案；
10. 『65G』然后再『9yy』之后最后一行会出现『复制九行』之类的说明字样。按下『G』到最后一行，再给他『p』贴上九行！
11. 因为 21~42 22 行，因此『21G』→『22dd』就能删除 22 行，此时你会发现游标所在 21 行的地方变成 MANPATH 开头啰，批注的 # 符号那几行都被删除了。

12. 『 :wman.test.config 』，你会发现最后一行出现 "man.test.config" [New].. 的字样。
13. 『27G』 之后，再给他 『 15x 』即可删除 15 个字符，出现 『 you 』的字样；
14. 先 『 1G 』 去到第一行，然后按下大写的 『 O 』 便新增一行且在插入模式；开始输入 『I am a student...』 后， 按下[Esc]回到一般模式等待后续工作；
15. 『 :wq 』

如果你的结果都可以查的到，那么 vi 的使用上面应该没有太大的问题啦！剩下的问题会是在…打字练习…。

---

### vim 的暂存档、救援回复与开启时的警告讯息

在目前主要的编辑软件都会有『回复』的功能，亦即当你的系统因为某些原因而导致类似当机的情况时， 还可以透过某些特别的机制来让你将之前未储存的数据『救』回来！这就是鸟哥这里所谓的『回复』功能啦！ 那么 vim 有没有回复功能呢？有的！ vim 就是透过『暂存档』来救援的啦！

当我们在使用 vim 编辑时， vim 会在与被编辑的档案的目录下，再建立一个名为 .filename.swp 的档案。 比如说我们在上一个小节谈到的编辑 /tmp/vitest/man.config 这个档案时， vim 会主动的建立 /tmp/vitest/.man.config.swp 的暂存档，你对 man.config 做的动作就会被记录到这个 .man.config.swp 当中喔！如果你的系统因为某些原因断线了， 导致你编辑的档案还没有储存，这个时候 .man.config.swp 就能够发会救援的功能了！我们来测试一下吧！ 底下的练习有些部分的指令我们尚未谈到，没关系，你先照着做，后续再回来了解啰！

```
[root@www ~]# cd /tmp/vitest
[root@www vitest]# vim man.config
# 此时会进入到 vim 的画面，请在 vim 的一般模式下按下
『 [ctrl]-z 』的组合键

[1]+  Stopped                  vim man.config  <==按下
[ctrl]-z 会告诉你这个讯息
```



当我们在 vim 的一般模式下按下 [ctrl]-z 的组合按键时,你的 vim 会被丢到背景去执行! 这部份的功能我们会在[第五篇的程序管理](#)当中谈到, 你这里先知道一下即可。回到命令提示字符后, 接下来我们来模拟将 vim 的工作不正常的中断吧!

```
[root@www vitest]# ls -al
total 48
drwxr-xr-x 2 root root 4096 Jan 12 14:48 .
drwxrwxrwt 7 root root 4096 Jan 12 13:26 ..
-rw-r--r-- 1 root root 4101 Jan 12 13:55 man.config
-rw-r--r-- 1 root root 4096 Jan 12
14:48 .man.config.swp <==就是他, 暂存档
-rw-r--r-- 1 root root 4101 Jan 12 13:43
man.test.config

[root@www vitest]# kill -9 %1 <==这里仿真断线停止 vim
工作
[root@www vitest]# ls -al .man.config.swp
-rw-r--r-- 1 root root 4096 Jan 12
14:48 .man.config.swp <==暂存档还是会存在!
```

那个 kill 可以仿真将系统的 vim 工作删除的情况,你可以假装当机了啦! 由于 vim 的工作被不正常的中断,导致暂存盘无法藉由正常流程来结束, 所以暂存档就不会消失,而继续保留下来。此时如果你继续编辑那个 man.config, 会出现什么情况呢? 会出现如下所示的状态喔:

```
[root@www vitest]# vim man.config
E325: ATTENTION <==错误代码
Found a swap file by the name ".man.config.swp" <==
底下数行说明有暂存档的存在
    owned by: root    dated: Mon Jan 12 14:48:24
2009
    file name: /tmp/vitest/man.config <==这个
暂存盘属于哪个实际的档案?
    modified: no
    user name: root   host name: www.vbird.tsai
    process ID: 11539
While opening file "man.config"
    dated: Mon Jan 12 13:55:07 2009
底下说明可能发生这个错误的两个主要原因与解决方案!
(1) Another program may be editing the same file.
    If this is the case, be careful not to end up with
```

```

two
    different instances of the same file when making
changes.
    Quit, or continue with caution.

(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r
man.config"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file
".man.config.swp"
    to avoid this message.

Swap file ".man.config.swp" already exists!底下说明
你可进行的动作
[O]pen Read-Only, (E)dit anyway, (R)ecover, (D)ele
it, (Q)uit, (A)bort:

```

由于暂存盘存在的关系，因此 vim 会主动的判断你的这个档案可能有些问题，在上面的图示中 vim 提示两点主要的问题与解决方案，分别是这样的：

- 问题一：可能有其他人或程序同时在编辑这个档案：

由于 Linux 是多人多任务的环境，因此很可能有很多人同时在编辑同一个档案。如果在多人共同编辑的情况下，万一大家同时储存，那么这个档案的内容将会变的乱七八糟！为了避免这个问题，因此 vim 会出现这个警告窗口！解决的方法则是：

- 找到另外那个程序或人员，请他将该 vim 的工作结束，然后你再继续处理。
- 如果你只是要看该档案的内容并不会有任何修改编辑的行为，那么可以选择开启成为只读(O)档案，亦即上述画面反白部分输入英文『o』即可，其实就是 [O]pen Read-Only 的选项啦！

- 问题二：在前一个 vim 的环境中，可能因为某些不知名原因导致 vim 中断 (crashed)：

这就是常见的不正常结束 vim 产生的后果。解决方案依据不同的情况而不同喔！常见的处理方法为：

- 如果你之前的 vim 处理动作尚未储存，此时你应该要按下『R』，

亦即使用 (R)ecover 的项目，此时 vim 会载入 .man.config.swp 的内容，让你自己来决定要不要储存！这样就能够救回来你之前未储存的工作。不过那个 .man.config.swp 并不会在你结束 vim 后自动删除，所以你离开 vim 后还得要自行删除 .man.config.swp 才能避免每次打开这个档案都会出现这样的警告！

- 如果你确定这个暂存盘是没有用的，那么你可以直接按下『D』删除掉这个暂存盘，亦即 (D)elele it 这个项目即可。此时 vim 会载入 man.config，并且将旧的 .man.config.swp 删除后，建立这次会使用的新的 .man.config.swp 喔！

至于这个发现暂存盘警告讯息的画面中，有出现六个可用按钮，各按钮的说明如下：

- [O]pen Read-Only：打开此档案成为只读档，可以用在你只是想要查阅该档案内容并不想要进行编辑行为时。一般来说，在上课时，如果你是登入到同学的计算机去看他的配置文件，结果发现其实同学他自己也在编辑时，可以使用这个模式；
- (E)dit anyway：还是用正常的方式打开你要编辑的那个档案，并不会载入暂存盘的内容。不过很容易出现两个使用者互相改变对方的档案等问题！好不好！
- (R)ecover：就是加载暂存盘的内容，用在你要救回之前未储存的工作。不过当你救回来并且储存离开 vim 后，还是要手动自行删除那个暂存档喔！
- (D)elele it：你确定那个暂存档是无用的！那么开启档案前会先将这个暂存盘删除！这个动作其实是比较常做的！因为你可能不确定这个暂存档是怎么来的，所以就删除掉他吧！哈哈！
- (Q)uit：按下 q 就离开 vim，不会进行任何动作回到命令提示字符。
- (A)bort：忽略这个编辑行为，感觉上与 quit 非常类似！也会送你回到命令提示字符就是啰！



## vim 的额外功能

其实，目前大部分的 distributions 都以 vim 取代 vi 的功能了！如果你使用 vi 后，却看到画面的右下角有显示目前光标所在的行列号码，那么你的 vi 已经被 vim 所取代啰～为什么要用 vim 呢？因为 vim 具有颜色显示的功能，并且还支持许多的程序语法 (syntax)，因此，当你使用 vim 编辑程序时（不论

是 C 语言，还是 shell script)，我们的 vim 将可帮你直接进行『程序除错 (debug)』的功能！真的很不赖吧！^\_^

如果你在文本模式下，输入 alias 时，出现这样的画面：

```
[root@www ~]# alias
....其他省略....
alias vi='vim' <==重点在这行啊！
```

这表示当你使用 vi 这个指令时，其实就是执行 vim 啦！如果你没有这一行，那么你就必须要使用 vimfilename 来启动 vim 啰！基本上，vim 的一般用法与 vi 完全一模一样～没有不同啦！那么我们就来看看 vim 的画面是怎样啰！假设我想要编辑 /etc/man.config，则输入『vim /etc/man.config』

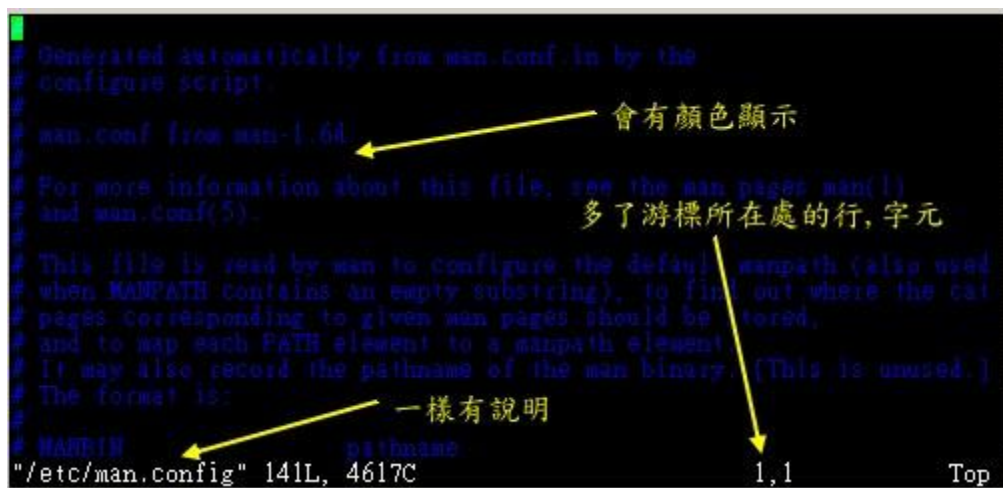


图 3.1.1、vim 的图示示意

上面是 vim 的画面示意图，在这个画面中有几点特色要说明喔：

1. 由于 man.config 是系统规划的配置文件，因此 vim 会进行语法检验，所以你会看到画面中内部主要为深蓝色，且深蓝色那一行是以批注符号 (#) 为开头；
2. 最底下一行的左边显示该档案的属性，包括 141 行与 4617 字符；
3. 最底下一行的右边出现的 1,1 表示光标所在为第一行，第一个字符位置之意(请看一下上图中的游标所在)；

所以，如果你向下移动到其他位置时，出现的非批注的数据就会有点像这样：



图 3.1.2、 vim 的图示示意

看到了喔！除了批注之外，其他的行就会有特别的颜色显示呢！可以避免你打错字啊！而且，最右下角的 30% 代表目前这个画面占整体档案的 30% 之意！这样瞭乎？

### 🔑 区块选择 (Visual Block)

刚刚我们提到的简单的 vi 操作过程中，几乎提到的都是以行为单位的操作。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](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，并且加到每一行的后面，亦即每一行的结果要是『192.168.1.2 host2.class.net host2』这样的情况时，在传统或现代的窗口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，这个时候光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

来实际进行我们需要的动作吧！就是将 host 再加到每一行的最后面，你可以这样做：

1. 使用 `vim hosts` 来开启该档案，记得该档案请由[上述的连结](#)下载先！
2. 将光标移动到第一行的 host 那个 h 上头，然后按下 `[ctrl]-v`，左下角出现区块示意字样：



图 3.1.1、进入区块功能的示意图

3. 将光标移动到最底部，此时光标移动过的区域会反白！如下图所示：

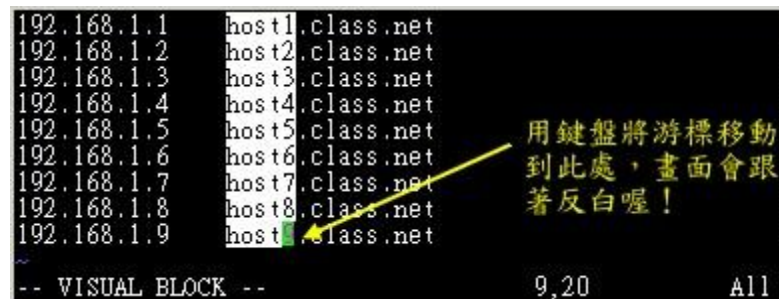


图 3.1.2、区块选择的结果示意图

4. 此时你可以按下 `『y』` 来进行复制，当你按下 `y` 之后，反白的区块就会消失不见啰！
5. 最后，将光标移动到第一行的最右边，并且再用编辑模式向右按两个空格键，回到一般模式后，再按下 `『p』` 后，你会发现很有趣！如下图所示：



```
192.168.1.1    host1.class.net  host1
192.168.1.2    host2.class.net  host2
192.168.1.3    host3.class.net  host3
192.168.1.4    host4.class.net  host4
192.168.1.5    host5.class.net  host5
192.168.1.6    host6.class.net  host6
192.168.1.7    host7.class.net  host7
192.168.1.8    host8.class.net  host8
192.168.1.9    host9.class.net  host9

1,33      All
```

图 3.1.3、将区块的资料贴上后的结果

透过上述的功能，你可以复制一个区块，并且是贴在某个『区块的范围』内，而不是以行为单位来处理你的整份文件喔！鸟哥个人是觉得这玩意儿非常的有帮助啦！至少在进行排列整齐的文本文件中复制/删除区块时，会是一个非常棒的功能！

🐦 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	
<code>:n</code>	编辑下一个档案
<code>:N</code>	编辑上一个档案
<code>:files</code>	列出目前这个 <code>vim</code> 的开启的所有档案

在过去，鸟哥想要将 A 档案内的十条消息『移动』到 B 档案去，通常要开两个 `vim` 窗口来复制，偏偏每个 `vim` 都是独立的，因此并没有办法在 A 档案下达『`nny`』再跑到 B 档案去『`p`』啦！在这种情况下最常用的方法就是透过鼠标圈选，复制后贴上。不过这样一来还是有问题，因为鸟哥超级喜欢使用 `[Tab]` 按键进行编排对齐动作，透过鼠标却会将 `[Tab]` 转成空格键，这样内容就不一样了！此时这个多档案编辑就派上用场了！

现在你可以做一下练习看看说！假设你要将刚刚鸟哥提供的 `hosts` 内的前四行 IP 资料复制到你的 `/etc/hosts` 档案内，那可以怎么进行呢？可以这样啊：

1. 透过 `vim hosts /etc/hosts` 来使用一个 `vim` 开启两个档案;
2. 在 `vim` 中先使用『`:files`』察看一下编辑的档案数据有啥? 结果如下所示。至于下图的最后一行显示的是『按下任意键』就会回到 `vim` 的一般模式中!



```
192.168.1.4    host4.class.net host4
192.168.1.5    host5.class.net host5
192.168.1.6    host6.class.net host6
192.168.1.7    host7.class.net host7
192.168.1.8    host8.class.net host8
192.168.1.9    host9.class.net host9
~
:files
 1 %a    "hosts"                                line 1
 2      "/etc/hosts"                             line 0
Press ENTER or type Command to Continue
```

我們輸入的指令

vim 告知我們有兩個檔案在編輯

图 3.2.1、多档案编辑示意图”

3. 在第一行输入『`4yy`』复制四行;
4. 在 `vim` 的环境下输入『`:n`』会来到第二个编辑的档案,亦即 `/etc/hosts` 内;
5. 在 `/etc/hosts` 下按『`G`』到最后一行,在输入『`p`』贴上;
6. 按下多次的『`u`』来还原原本的档案数据;
7. 最终按下『`:q`』来离开 `vim` 的多档案编辑吧!

看到了吧? 利用多档案编辑的功能,可以让你很快速的就将需要的资料复制到正确的档案内。当然啰,这个功能也可以利用窗口接口来达到,那就是底下要提到的多窗口功能。

---

## 💡多窗口功能

在开始这个小节前, 先来想象两个情况:

- 当我有一个档案非常的大,我查阅到后面的数据时,想要『对照』前面的数据, 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;

在一般窗口接口下的编辑软件大多有『分割窗口』或者是『冻结窗口』的功能来将一个档案分割成多个窗口的展现, 那么 `vim` 能不能达到这个功能啊? 可以啊! 但是如何分割窗口并放入档案呢? 很简单啊! 在指令列模式输入『`:sp {filename}`』即可! 那个 `filename` 可有可无, 如果想要在新窗口启动另一个

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窗口间！

让我们来测试一下，你先使用『 `vim /etc/man.config` 』打开这个档案，然后『 `1G` 』去到第一行，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：

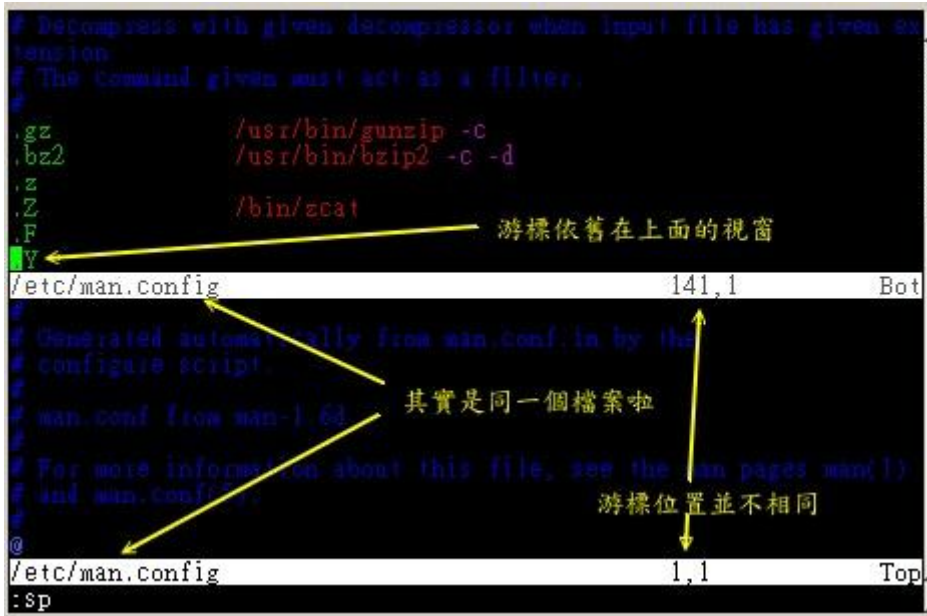


图 3.3.1、窗口分割的示意图

万一你再输入『 `:sp /etc/hosts` 』时，就会变成下图这样喔：




图 3.3.2、窗口分割的示意图

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

好了：

多窗口情况下的按键功能	
:sp [filename]	开启一个新窗口，如果有加 filename， 表示在新窗口开启一个新档案，否则表示两个窗口为同一个档案内容(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 (或向下箭头键)，则光标可移动到下方的窗口。
[ctrl]+w+ k [ctrl]+w+ ↑	同上，不过光标移动到上面的窗口。
[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窗口，那么利用 [ctrl]+w+ ↓ 移动到下方窗口后，按下 :q 即可离开， 也可以按下 [ctrl]+w+q 啊！

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

 vim 环境设定与记录： ~/.vimrc, ~/.viminfo

有没有发现，如果我们以 vim 软件来搜寻一个档案内部的某个字符串时，这个字符串会被反白，而下次我们再次以 vim 编辑这个档案时，该搜寻的字符串反白情况还是存在呢！甚至于在编辑其他档案时， 如果其他档案内也存在这个字符串，哇！竟然还是主动反白耶！真神奇！ 另外，当我们重复编辑同一个档案时，当第二次进入该档案时， 光标竟然就在上次离开的那一行上头呢！真是好方便啊～但是，怎么会这样呢？

这是因为我们的 vim 会主动的将你曾经做过的行为登录下来，好让你下次可以轻松的作业啊！ 那个记录动作的档案就是： ~/.viminfo ！如果你曾经使用过 vim， 那你的家目录应该会存在这个档案才对。这个档案是自动产生的， 你不必自行建立。而你在 vim 里头所做过的动作，就可以在这个档案内部查询到啰～ ^\_^

此外，每个 distributions 对 vim 的预设环境都不太相同，举例来说，某些版本在搜寻到关键词时并不会高亮度反白， 有些版本则会主动的帮你进行缩排的行为。但这些其实都可以自行设定的，那就是 vim 的环境设定啰～ vim 的环境设定参数有很多，如果你要知道目前的设定值，可以在一般模式时输入『 :set all 』来查阅，不过..... 设定项目实在太多了～所以，鸟哥在这里仅列出一些平时比较常用的一些简单的设定值， 提供给你参考啊。

### Tips:

所谓的缩排，就是当你按下 Enter 编辑新的一行时，光标不会在行首，而是在与上一行的第一个非空格符处对齐！



vim 的环境设定参数	
<code>:set nu</code> <code>:set nonu</code>	就是设定与取消行号啊！
<code>:set hlsearch</code> <code>:set nohlsearch</code>	hlsearch 就是 high light search(高亮度搜寻)。这个就是设定是否将搜寻的字符串反白的设定值。默认值是 hlsearch
<code>:set autoindent</code> <code>:set noautoindent</code>	是否自动缩排？autoindent 就是自动缩排。
<code>:set backup</code>	是否自动储存备份档？一般是 nobackup 的，如果设定 backup 的话，那么当你更动任何一个档案时，则源文件会被另存成一个档名为 filename~ 的档案。举例来说，我们编辑 hosts，设定 :set backup，那么当更动 hosts 时，在同目录下，就会产生 hosts~ 文件名的档案，记录原始的 hosts 档案内容
<code>:set ruler</code>	还记得我们提到的右下角的一些状态栏说明吗？这个 ruler 就是在显示或不显示该设定值的啦！
<code>:set showmode</code>	这个则是，是否要显示 --INSERT-- 之类的字眼在左下角的状态栏。
<code>:set backspace=(012)</code>	一般来说，如果我们按下 i 进入编辑模式后，可以利用退格键 (backspace) 来删除任意字符的。但是，某些 distribution 则不许如此。此时，我们就可以透过 backspace 来设定啰~ 当 backspace 为 2 时，就是可以删除任意值；0 或 1 时，仅可删除刚刚输入的字符，而无法删除原本就已经存在的文字了！
<code>:set all</code>	显示目前所有的环境参数设定值。
<code>:set</code>	显示与系统默认值不同的设定参数，一般来说就是你有自行变动过的设定参数啦！
<code>:syntax on</code> <code>:syntax off</code>	是否依据程序相关语法显示不同颜色？举例来说，在编辑一个纯文本档时，如果开头是以 # 开始，那么该行就会变成蓝色。如果你懂得写程序，那么这个 :syntax on 还会主动的帮你除错呢！但是，如果你仅是编写纯文本档案，要避免颜色对



	你的屏幕产生的干扰，则可以取消这个设定。
<pre>:set bg=dark :set bg=light</pre>	可用以显示不同的颜色色调，预设是『 light 』。如果你常常发现批注的字体深蓝色实在很不容易看，那么这里可以设定为 dark 喔！试看看，会有不同的样式呢！

总之，这些设定值很有用处的啦！但是.....我是否每次使用 vim 都要重新设定一次各个参数值？这不太合理吧？没错啊！所以，我们可以透过配置文件来直接规定我们习惯的 vim 操作环境呢！整体 vim 的设定值一般是放置在 /etc/vimrc 这个档案，不过，不建议你修改他！你可以修改 ~/.vimrc 这个档案（预设不存在，请你自行手动建立！），将你所希望的设定值写入！举例来说，可以是这样的一个档案：

```
[root@www ~]# vim ~/.vimrc
"这个档案的双引号（"）是批注
set hlsearch           "高亮度反白
set backspace=2        "可随时用退格键删除
set autoindent         "自动缩排
set ruler              "可显示最后一行的状态
set showmode           "左下角那一行的状态
set nu                 "可以在每一行的最前面显示行号啦！
set bg=dark            "显示不同的底色色调
syntax on              "进行语法检验，颜色显示。
```

在这个档案中，使用『 set hlsearch 』或『 :set hlsearch 』，亦即最前面有没有冒号『 : 』效果都是一样的！至于双引号则是批注符号！不要用错批注符号，否则每次使用 vim 时都会发生警告讯息喔！建立好这个档案后，当你下次重新以 vim 编辑某个档案时，该档案的预设环境设定就是上头写的啰～ 这样，是否很方便你的操作啊！多多利用 vim 的环境设定功能呢！^\_^



#### 其他 vim 使用注意事项

vim 其实不是那么好学，虽然他的功能确实非常强大！所以底下我们还有一些需要注意的地方要来跟大家分享喔！

---

#### 中文编码的问题

很多朋友常常哀嚎，说他们的 vim 里面怎么无法显示正常的中文啊？其实这很



有可能是因为编码的问题！因为中文编码有 big5 与 utf8 两种，如果你的档案是使用 big5 编码制作的，但在 vim 的终端接口中你使用的是万国码(utf8)，由于编码的不同，你的中文档案内容当然就是一堆乱码了！怎么办？这时你得要考虑许多东西啦！有这些：

1. 你的 Linux 系统默认支持的语系数据：这与 /etc/sysconfig/i18n 有关；
2. 你的终端界面 (bash) 的语系：这与 LANG 这个变数有关；
3. 你的档案原本的编码；
4. 开启终端机的软件，例如在 GNOME 底下的窗口接口。

事实上最重要的是上头的第三与第四点，只要这两点的编码一致，你就能够正确的看到与编辑你的中文档案。否则就会看到一堆乱码啦！

一般来说，中文编码使用 big5 时，在写入某些数据库系统中，在『许、盖、功』这些字体上面会发生错误！所以近期以来大多希望大家能够使用万国码 utf8 来进行中文编码！但是在 Windows XP 上的软件常常默认使用 big5 的编码，包括鸟哥由于沿用以前的文件数据文件，也大多使用 big5 的编码。此时就得要注意上述的这些咚咚啰。

在 Linux 本机前的 tty1~tty6 原本默认就不支持中文编码，所以不用考虑这个问题！因为你一定会看到乱码！呵呵！现在鸟哥假设俺的文件档案内编码为 big5 时，而且我的环境是使用 Linux 的 GNOME，启动的终端接口为 GNOME-terminal 软件，那鸟哥通常是这样来修正语系编码的行为：

```
[root@www ~]# LANG=zh_TW.big5
```

然后在终端接口工具栏的『终端机』-->『设定字符编码』-->『中文（正体）(BIG5)』项目点选一下，如果一切都没有问题了，再用 vim 去开启那个 big5 编码的档案，就没有问题了！以上！报告完毕！

---

## DOS 与 Linux 的断行字符

我们在[第七章](#)里面谈到 cat 这个指令时，曾经提到过 DOS 与 Linux 断行字符的不同。而我们可以利用 cat -A 来观察以 DOS (Windows 系统) 建立的档案的特殊格式，也可以发现在 DOS 使用的断行字符为 ^M\$，我们称为 CR 与 LF 两个符号。而在 Linux 底下，则是仅有 LF (\$) 这个断行符号。这个断行符号对于 Linux 的影响很大喔！为什么呢？

我们说过，在 Linux 底下的指令在开始执行时，他的判断依据是『Enter』，而 Linux 的 Enter 为 LF 符号，不过，由于 DOS 的断行符号是 CRLF，也就是多了一个 ^M 的符号出来，在这样的情况下，如果是一个 shell script 的

程序档案，呵呵～将可能造成『程序无法执行』的状态～ 因为他会误判程序所下达的指令内容啊！这很伤脑筋吧！

那怎么办啊？很简单啊，将格式转换成为 Linux 即可啊！『废话』，这当然大家都知道，但是，要以 vi 进入该档案，然后一个一个删除每一行的 CR 吗？当然没有这么没人性啦！我们可以透过简单的指令来进行格式的转换啊！

```
[root@www ~]# dos2unix [-kn] file [newfile]
[root@www ~]# unix2dos [-kn] file [newfile]
选项与参数：
-k  : 保留该档案原本的 mtime 时间格式（不更新档案上次内容经过修订的时间）
-n  : 保留原本的旧档，将转换后的内容输出到新档案，如：
dos2unix -n old new

范例一：将刚刚上述练习的 /tmp/vitest/man.config 修改成为 dos 断行
[root@www ~]# cd /tmp/vitest
[root@www vitest]# cp -a /etc/man.config .
[root@www vitest]# ll man.config
-rw-r--r-- 1 root root 4617 Jan  6  2007 man.config
[root@www vitest]# unix2dos -k man.config
unix2dos: converting file man.config to DOS format ...
# 屏幕会显示上述的讯息，说明断行转为 DOS 格式了！
[root@www vitest]# ll man.config
-rw-r--r-- 1 root root 4758 Jan  6  2007 man.config
# 断行字符多了 ^M ，所以容量增加了！

范例二：将上述的 man.config 转成 man.config.linux 的 Linux 断行字符
[root@www vitest]# dos2unix -k -n man.config
man.config.linux
dos2unix: converting file man.config to file
man.config.linux in UNIX format ...
[root@www vitest]# ll man.config*
-rw-r--r-- 1 root root 4758 Jan  6  2007 man.config
-rw----- 1 root root 4617 Jan  6  2007
man.config.linux
```

因为断行字符以及 DOS 与 Linux 操作系统底下一些字符的定义不同，因此，不建议你在 Windows 系统当中将档案编辑好之后，才上传到 Linux 系统，会发生错误问题。而且，如果你在不同的系统之间复制一些纯文本档案时，千万记得要使用 unix2dos 或 dos2unix 来转换一下断行格式啊！

---

## 🐦 语系编码转换

很多朋友都会有的问题，就是想要将语系编码进行转换啦！举例来说，想要将 big5 编码转成 utf8 。这个时候怎么办？难不成要每个档案打开会转存成 utf8 吗？不需要这样做啦！使用 iconv 这个指令即可！鸟哥将之前的 vi 章节做成 big5 编码的档案，你可以照底下的连结来下载先：

- [http://linux.vbird.org/linux\\_basic/0310vi/vi.big5](http://linux.vbird.org/linux_basic/0310vi/vi.big5)

在终端机的环境下你可以使用『wget 网址』来下载上述的档案喔！鸟哥将他下载到 /tmp/vitest 目录下。接下来让我们来使用 iconv 这个指令来玩一玩编码转换吧！

```
[root@www ~]# iconv --list
[root@www ~]# iconv -f 原本编码 -t 新编码 filename
[-o newfile]
选项与参数:
--list : 列出 iconv 支持的语系数据
-f      : from , 亦即来源之意, 后接原本的编码格式;
-t      : to , 亦即后来的新编码要是什么格式;
-o file: 如果要保留原本的档案, 那么使用 -o 新档名,
可以建立新编码档案。

范例一: 将 /tmp/vitest/vi.big5 转成 utf8 编码吧!
[root@www ~]# cd /tmp/vitest
[root@www vitest]# iconv -f big5 -t utf8 vi.big5 -o
vi.utf8
[root@www vitest]# file vi*
vi.big5: ISO-8859 text, with CRLF line terminators
vi.utf8: UTF-8 Unicode text, with CRLF line terminators
# 是吧! 有明显的不同吧! ^_^
```

这指令支持的语系非常之多，除了正体中文的 big5, utf8 编码之外，也支持简体中文的 gb2312 ，所以对岸的朋友可以简单的将鸟站的网页数据下载后，利用这个指令来转成简体，就能够轻松的读取文件数据啰！不过，不要将转成简体的档案又上传成为您自己的网页啊！这明明是鸟哥写的不是吗？ ^\_^

不过如果是要将正体中文的 utf8 转成简体中文的 utf8 编码时，那就得费些功夫了！举例来说，如果要将刚刚那个 vi.utf8 转成简体的 utf8 时，可以这样做：

```
[root@www vitest]# iconv -f utf8 -t big5 vi.utf8 | \  
> iconv -f big5 -t gb2312 | iconv -f gb2312 -t utf8 -o  
vi.gb.utf8
```



## 重点回顾

- Linux 底下的配置文件多为文本文件，故使用 vim 即可进行设定编辑；
- vim 可视为程序编辑器，可用以编辑 shell script，配置文件等，避免打错字；
- vi 为所有 unix like 的操作系统都会存在的编辑器，且执行速度快速；
- vi 有三种模式，一般模式可变换到编辑与指令列模式，但编辑模式与指令列模式不能互换；
- 常用的按键有 i, [Esc], :wq 等；
- vi 的画面大略可分为两部份，(1)上半部的本文与(2)最后一行的状态+指令列模式；
- 数字是有意义的，用来说明重复进行几次动作的意思，如 5yy 为复制 5 行之意；
- 光标的移动中，大写的 G 经常使用，尤其是 1G, G 移动到文章的头/尾功能！
- vi 的取代功能也很棒！ :n1,n2s/old/new/g 要特别注意学习起来；
- 小数点『.』为重复进行前一次动作，也是经常使用的按键功能！
- 进入编辑模式几乎只要记住： i, o, R 三个按钮即可！尤其是新增一行的 o 与取代的 R
- vim 会主动的建立 swap 暂存档，所以不要随意断线！
- 如果在文章内有对其的区块，可以使用 [ctrl]-v 进行复制/贴上/删除的行为
- 使用 :sp 功能可以分割窗口
- vim 的环境设定可以写入在 ~/.vimrc 档案中；
- 可以使用 iconv 进行档案语系编码的转换
- 使用 dos2unix 及 unix2dos 可以变更档案每一行的行尾断行字符。



## 本章练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 我要在某个档案的第 34 行向右移动 15 个字符，应该在一般模式下达什么指令？

(1)先按下 34G 到第 34 行；(2)再按下 [ 15 + 向右键 ]，或 [ 15l ] 亦可！

- 在 vi 里面， PageDown 按钮可以使用什么组合键来取代？

[Ctrl] + f 可以向后翻一页

- 如何去到 vi 该档案里面的页首或页尾？

去页首按下 1G ； 去页尾按下 G 即可

- 如何在一行中，移动到行头及行尾？

移动到行头，按 0 ， 移动到行尾按 \$ 即可！

- vi 里面， r 有什么功能？

取代光标所在的那个字符

- 如何将目前的页面另存新档？

:w newfilename

- 在 linux 底下最常使用的文书编辑器为 vi ， 请问如何进入编辑模式？

在一般模式底下输入： i, I, a, A 为在本行当中输入新字符；（出现 - Insert- ）

在一般模式当中输入： o, O 为在一个新的一行输入新字符；

在一般模式当中输入： r, R 为取代字符！（左下角出现 - Replace-）

- 如何由编辑模式跳回一般模式？

可以按下[Esc]

- 若上下左右键无法使用时，请问如何在一般模式移动光标？

[h, j, k, l]分别代表[左、下、上、右]

- 若 [pagedown] [pageup] 在一般模式无法使用时，如何往前或往后翻一页？

(1)向下翻 [Ctrl] + [f] (2)向前翻 [Ctrl] + [b]

- 如何到本档案的最后一行、第一行；本行的第一个字符、最后一个字符？

分别为： G, 1G, 0, \$

- 如何删除一行、n 行；如何删除一个字符？

分别为 dd, ndd, x 或 X (dG 及 d1G 分别表示删除到页首及页尾)

- 如何复制一行、n 行并加以贴上?

分别为 yy, nyy, p 或 P

- 如何搜寻 string 这个字符串?

?string (往前搜寻)

/string (往后搜寻)

- 如何取代 word1 成为 word2, 而若需要使用者确认机制, 又该如何?

:1,\$s/word1/word2/g 或

:1,\$s/word1/word2/gc (需要使用者确认)

- 如何读取一个档案 filename 进来目前这个档案?

:r filename

- 如何另存新档成为 newfilename?

:w newfilename

- 如何存档、离开、存档后离开、强制存档后离开?

:w; :q; :wq; :wq!

- 在 vi 底下作了很多的编辑动作之后, 却想还原成原来的档案内容, 应该怎么进行?

直接按下 :e! 即可恢复成档案的原始状态!

- 我在 vi 这个程序当中, 不想离开 vi, 但是想执行 ls /home 这个指令, vi 有什么额外的功能可以达到这个目的:

事实上, 可以使用 [ :! ls /home ] 不过, 如果你学过后面的章节之后, 你会发现, 执行 [ ctrl + z ] 亦可暂时退出 vi 让你在指令列模式当中执行指令喔!

- 如何设定与取消行号?

:set nu

:set nonu

---





## 参考数据与延伸阅读

- 维基百科: ASCII 的代码与图示对应表:  
<http://zh.wikipedia.org/wiki/ASCII>
- 注 1: 常见文书编辑器项目计划连结:
  - emacs: <http://www.gnu.org/software/emacs/>
  - pico:  
<http://www.ece.uwaterloo.ca/~ece250/Online/Unix/pico/>
  - nano: <http://sourceforge.net/projects/nano/>
  - joe: <http://sourceforge.net/projects/joe-editor/>
  - vim: <http://www.vim.org>
  - 常见文书编辑器比较:  
<http://encyclopedia.thefreedictionary.com/List+of+text+editors>
  - 维基百科的文书编辑器比较:  
[http://en.wikipedia.org/wiki/Comparison\\_of\\_text\\_editors](http://en.wikipedia.org/wiki/Comparison_of_text_editors)
- 关于 vim 是什么的『中文』说明:  
<http://www.vim.org/6k/features.zh.txt>。
- 李果正兄的: 大家来学 vim (<http://info.sayya.org/~edt1023/vim/>)
- 麦克星球 Linux Fedora 心得笔记:  
正体/简体中文的转换方法:  
<http://blog.xuite.net/michaelr/linux/15650102>

---

2002/04/05: 第一次完成

2003/02/07: 重新编排与加入 FAQ

2003/02/25: 新加入本章节与 LPI 的相关性说明!

2005/07/28: 将旧文章移动到 [这里](#)。

2005/08/01: 加入果正兄文章的参考, 还有查阅 vim 官方网站的数据!

2008/12/18: 将原本针对 FC4 版本的文章移动到 [此处](#)

2009/01/13: 这么简单的一篇改写, 竟改了一个月! 原因只是期末考将近太忙了~

---

2002/01/21 以来统计人数

586920



本网页主要以 [firefox](#) 配合分辨率 1024x768 作为设计依据

<http://linux.vbird.org> is designed by [VBird](#) during 2001-2009. [Aerosol Lab](#).