



1-2vi 文書編輯器

vi 文書編輯器有 FreeBSD 文書編輯器所有的特徵，它可以編輯處理我們的文字。Vi 雖然比 pico 文書編輯器較不易學習，但它強大的功能和高容納度，與作業系統的相容性，確是最好，而且是最多人使用的 FreeBSD 文書編輯器。我們可以先使用 vi 編輯器。我們在 shell 指令提示器上輸入 vi first.sh

```
[root@flash 1-1]# vi first.sh
```

我們進入 vi 文書編輯器後，我們可以按下 a 鍵或 I 鍵，來進入輸入模式。我們可以輸入 ls -la，我們再按下<enter>鍵。再輸入 who 再按下<enter>鍵。最後輸入 pwd，輸入完後我們按下<Esc>鍵到命令模式。

```
ls -la
who
pwd
~
~
~
```

我們在命令模式，也就是最底層，輸入 :wq。這樣就可以儲存檔案，並且離開 vi 文書編輯器。

```
~
~
~
:wq
```

我們可以輸入/bin/bash first.sh 這樣就可以執行我們剛才輸入的指令了。

```
[root@flash 1-1]# /bin/bash first.sh
total 24
drwxr-xr-x  2 root    root      4096  8月 14 00:43 .
drwxrwxrwx 22 chaiyen chaiyen   4096  8月 13 23:53 ..
-rwxrwxrwx  1 root    root        15  8月 14 00:43 first.sh
-rw-r--r--  1 root    root         40  8月 14 00:04 lib.h
-rw-r--r--  1 root    root        369  8月 13 23:17 power.c
-rw-----  1 root    root        439  8月 13 23:25 power.c.save
chaiyen pts/1    Aug 13 23:59 (61-218-29-5.HINET-IP.hinet.net)
root    pts/0    Aug 13 23:49
/home/chaiyen/1-1
```

1-2-1 如何開始 vi 文書編輯器、儲存檔案、離開 vi 文書編輯器

vi 指令讓我們編輯新的檔案或已存在的檔案。

語法：

指令：vi 參數 檔案

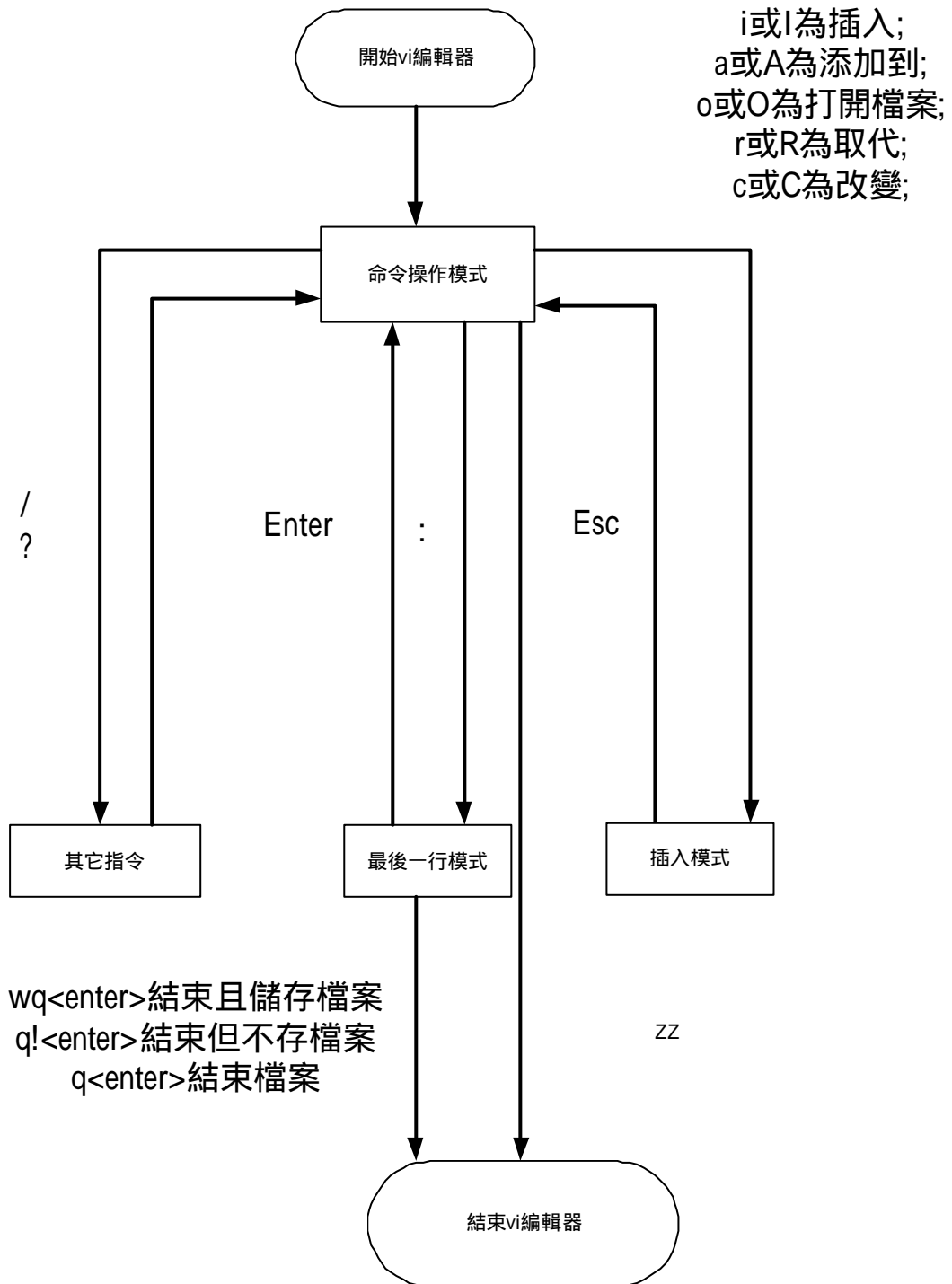
參數：

+n：在第 n 行開始編輯檔案。

+/exp：在字串 exp 開始的那一行開始編輯。

當我們進入 vi 文書編輯器後，會有兩種操作模式，一個為命令操作模式，另外一個為插入操作模式。命令操作模式是由一些指令參數所組成的命令。插入操作模式允許我們輸入文字。

我們一開始會進入到命令操作模式，我們按下<A>鍵就可以插入文字到游標那一行的最後。當我們按下<Esc>時，就可以從插入模式到命令操作模式。當我們在命令模式輸入：冒號，在後面再輸入指令 wq，就可以將資料輸入檔案，並且離開 vi。我們也可以開以在命令模式輸入：冒號，在後面再輸入命令 w 檔名，這樣就可以將資料寫入指定的檔案。



vi 文書編輯器預設的是以命令模式開始。當我們要從命令模式轉變成插入模式時有三個鍵，分別是 a 鍵，i 鍵和 o 鍵。

a：從目前游標下一個位置開始插入。

i：從目前游標所在位置開始插入。

o：從目前游標下一行位置開始插入。

我們使用 vi 文書編輯器編輯 power1.c 檔案。

```
[root@flash chaiyen]# vi power1.c
```

當我們進入 vi 時，如果按下 I 就會進入插入模式。

```
double compute(double x,double y);
main ()
{
    float x, y;
    printf ("請輸入x的y次方，以便求x的y次方的值\n");
    printf ("請輸入x:");
    scanf ("%f", &x);
    printf ("請輸入y:");
    scanf ("%f", &y);
    printf ("x的y次方: %f\n",compute(x,y));
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

我們按下 I 進入插入模式，從目前游標所在位置開始插入。

```
double  
main ()  
{
```

當我們按下 a 進入插入模式，從目前游標下一個位置開始插入

```
double compute  
main ()  
{
```

當我們按下 o 進入插入模式，從目前游標下一行位置開始插入。

```
double compute(double x,double y);  
main ()  
{  
    float x, y;  
    printf ("請輸入x的y次方，以便求x的y次方的值\n");  
    printf ("請輸入x:");  
    scanf ("%f", &x);  
    printf ("請輸入y:");  
    scanf ("%f", &y);  
    printf ("x的y次方: %f\n",compute(x,y));  
}
```

下面是 vi 編輯器命令的語法的範例

| 語法 | 說明 |
|------------|-------------------------------|
| 5dw | 開始在目前游標所在的地方，刪除 5 個字元。 |
| 7dd | 在目前所在位置刪除七行。 |
| 7o | 在目前所在位置後空七行 |
| 7O | 在目前所在位置前空七行 |
| 1G | 將游標移到第一行 |
| 10yy | 拷背 yanks 下十行到緩衝記憶體 |
| d | 刪除字元或行數 |
| u | 回復到上一個動作 |
| P | 貼上所複製的文字在目前所在的位置之後 |
| p | 貼上所複製的文字在目前所在的位置之前 |
| : r 檔名 | 從指令檔名讀取資料，再將資料插入目前所在位置 |
| : q! | 離開 vi 編輯器而沒有儲存任緩充區的何資料 |
| : wq | 儲存緩衝區的資料然後離開 |
| : w 檔名 | 儲存緩衝區的資料到指定的檔名 |
| : w! 檔名 | 覆寫目前的資料到指定的檔名中 |
| ZZ | 離開 vi 編輯器，並且儲存檔案 |
| : e 檔名 | 建立新檔案 |
| : n 檔名 | 載入新的檔案 |
| : set nu | 顯示每一行的編號 |
| : set nonu | 不顯示每一行的編號 |
| / 字串 | 尋找所輸入的字串。找到字串時，再輸入 n 就會再找下一個。 |
| ? 字串 | 尋找所輸入的字串。找到字串時，再輸入 n 就會再找下一個。 |
| ^b | 將游標向前捲動一頁 |
| ^f | 將游標向後捲動一頁 |

在進入檔案後

命令模式：

用 `esc` 鍵切換至此模式，開始編輯時，`vi` 在此模式。

輸入模式：

`vi` 會將鍵入的字當作本文而顯示在螢幕上。

最後一行模式：在命令模式下鍵入：`:` 或 `/` 可使 `vi` 進入此模式

`:set nu` 可以顯示行數。

`:/` 字 可以搜尋所要搜尋的字。

`:q` 離開 `vi` 編輯器

`:q!` 離開 `vi` 編輯器而不儲存

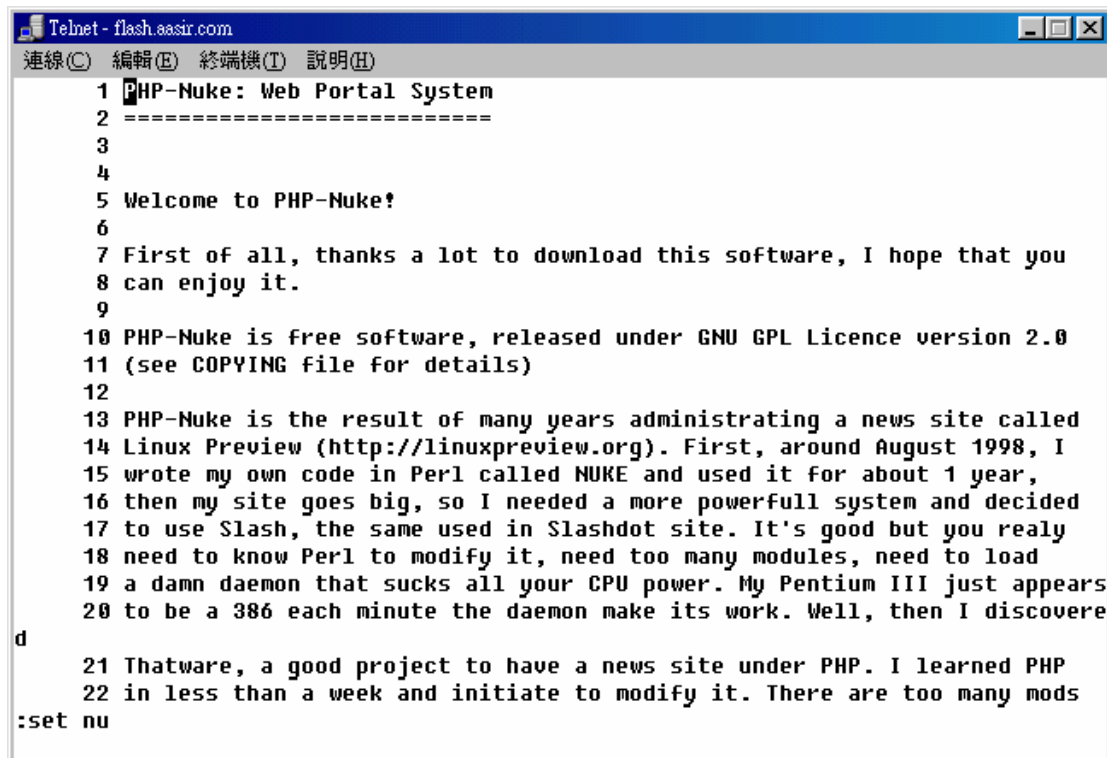
`:qw` 離開 `vi` 編輯器且儲存

我們使用 `vi test.php` 來編輯 `test.php` 的檔案。



```
Telnet - flash.aasir.com
連線(C) 編輯(E) 終端機(T) 說明(H)
[root@flash chaiyen]# vi test.php
```

我們使用 `set nu` 來顯示資料的行號。



```
Telnet - flash.aasir.com
連線(C) 編輯(E) 終端機(T) 說明(H)
 1 PHP-Nuke: Web Portal System
 2 =====
 3
 4
 5 Welcome to PHP-Nuke!
 6
 7 First of all, thanks a lot to download this software, I hope that you
 8 can enjoy it.
 9
10 PHP-Nuke is free software, released under GNU GPL Licence version 2.0
11 (see COPYING file for details)
12
13 PHP-Nuke is the result of many years administrating a news site called
14 Linux Preview (http://linuxpreview.org). First, around August 1998, I
15 wrote my own code in Perl called NUKE and used it for about 1 year,
16 then my site goes big, so I needed a more powerfull system and decided
17 to use Slash, the same used in Slashdot site. It's good but you really
18 need to know Perl to modify it, need too many modules, need to load
19 a damn daemon that sucks all your CPU power. My Pentium III just appears
20 to be a 386 each minute the daemon make its work. Well, then I discovere
d
21 Thatware, a good project to have a news site under PHP. I learned PHP
22 in less than a week and initiate to modify it. There are too many mods
:set nu
```

我們使用 `:/Linux` 來搜尋 `Linux` 這個字串。

`:/Linux`

我們使用 `qW` 來儲存所編輯的資料。

我們使用 `q!` 不除存資料而離開 `vi` 編輯器。

| 字串替代命令 | 說明 |
|-----------------------------------|---|
| <code>: s/scanf/good</code> | 只有將這一行的 <code>scanf</code> 字串替換成 <code>good</code> 字串，一次。 |
| <code>: s/scanf/good/g</code> | 在游標所在位置的那一行字串 <code>scanf</code> 都替換成 <code>good</code> |
| <code>: 1,10s/scanf/good/g</code> | 從第一行到第十行的 <code>scanf</code> 字串都替換成 <code>good</code> 字串 |
| <code>: 1,\$s/scanf/good/g</code> | 將整個檔案的 <code>scanf</code> 字串都替換成 <code>good</code> 字串 |

替代字串語法為：`[替代範圍]s/舊字串/新字串`。

1-2-2 複製與剪下與貼上文字

我們使用 vi 編輯 power1.c。

當我們進入 vi 時，我們使用 yy 命令來複製那一行，並且使用 5p，來將所複製的那一行貼上去五遍。

```
double compute(double x,double y);
main ()
{
    float x, y;
    printf ("請輸入x的y次方，以便求x的y次方的值\n");
    printf ("請輸入x:");
    scanf ("%f", &x);
    printf ("請輸入y:");
    scanf ("%f", &y);
    printf ("x的y次方: %f\n",compute(x,y));
}
```

我們使用 5p 來將所複製的那一行貼上去五遍。

```
double compute(double x,double y);
main ()
{
    float x, y;
    printf ("請輸入x的y次方，以便求x的y次方的值\n");
    printf ("請輸入x:");
    printf ("請輸入x:");
    printf ("請輸入x:");
    printf ("請輸入x:");
    printf ("請輸入x:");
    printf ("請輸入x:");
    scanf ("%f", &x);
    printf ("請輸入y:");
    scanf ("%f", &y);
    printf ("x的y次方: %f\n",compute(x,y));
}
```


這是替代的情況。

```
double compute(double x,double y);
main ()
{
    float x, y;
    printf ("請輸入x的y次方，以便求x的y次方的值\n");
    printf ("請輸入x:");
    printf ("請輸入x:");
void good(char *);
void lucky(char *);

    printf ("請輸入x:");
    掃描 ("%f", &x);
    printf ("請輸入y:");
    掃描 ("%f", &y);
    printf ("x的y次方: %f\n",compute(x,y));
}
~
~
~
~
~
~
~
~
:1,$s/scanf/掃描/g
```

我們在命令模式中使用 : set nu , 來顯示每一行的編號。

```
double compute(double x,double y);
main ()
{
    float x, y;
    printf ("請輸入x的y次方，以便求x的y次方的值\n");
    printf ("請輸入x:");
    printf ("請輸入x:");
void good(char *);
void lucky(char *);

    printf ("請輸入x:");
    掃描 ("%f", &x);
    printf ("請輸入y:");
    掃描 ("%f", &y);
    printf ("x的y次方: %f\n",compute(x,y));
}
~
~
~
~
~
~
~
~
:set nu
```

我們要把第五行到第七行的 x 字元替換成 z 字元，我們在命令模式中使用

:5,7s/xz/g，將第五行到第七行的 x 字元替換成 z 字元。

```
1 double compute(double x,double y);
2 main ()
3 {
4   float x, y;
5   printf ("請輸入x的y次方，以便求x的y次方的值\n");
6   printf ("請輸入x:");
7   printf ("請輸入x:");
8 void good(char *);
9 void lucky(char *);
10
11  printf ("請輸入x:");
12  掃描 ("%f", &x);
13  printf ("請輸入y:");
14  掃描 ("%f", &y);
15  printf ("x的y次方: %f\n",compute(x,y));
16 }
```

~
~
~
~
~
~
~
~

:5,7s/x/z/g

這是替換的情況。

```
5   printf ("請輸入z的y次方，以便求z的y次方的值\n");
6   printf ("請輸入z:");
7   printf ("請輸入z:");
```

我們使用<Esc>回到命令模式中，我們按下 u 就可以回到還沒替換的步驟前。u 就是 undo 的意義。u 就是回到前一步驟。這是還未替換 x 成 z 前的情況。

```
1 double compute(double x,double y);
2 main ()
3 {
4   float x, y;
5   printf ("請輸入x的y次方，以便求x的y次方的值\n");
6   printf ("請輸入x:");
7   printf ("請輸入x:");
8   void good(char *);
9   void lucky(char *);
10
11   printf ("請輸入x:");
12   掃描 ("%f", &x);
13   printf ("請輸入y:");
14   掃描 ("%f", &y);
15   printf ("x的y次方: %f\n",compute(x,y));
16 }
```

我們使用 w power.c 來將資料儲存到 power.c 上面。

```
1 double compute(double x,double y);
2 main ()
3 {
4   float x, y;
5   printf ("請輸入x的y次方，以便求x的y次方的值\n");
6   printf ("請輸入x:");
7   printf ("請輸入x:");
8   void good(char *);
9   void lucky(char *);
10
11   printf ("請輸入x:");
12   掃描 ("%f", &x);
13   printf ("請輸入y:");
14   掃描 ("%f", &y);
15   printf ("x的y次方: %f\n",compute(x,y));
16 }
~
~
~
~
~
~
~
:w power.c
```

我們按下 ZZ 就可以離開 vi 編輯器了。

我們在 vi 文書編輯器上的命令模式下也可以執行 shell。

語法：

:!shell 指令

我們在命令模式輸入 :!ls 就可以顯示目錄的內容。

```
:!ls
[No write since last change]
\      1-7      good.c      love.txt  power.c,v  test.TXT.bak
001.tif compute.c  good.txt   lucky.c   powert.c   total
002.tif cvsqrt    good.txt.save macrol    powerv     total2
003.tif fol.txt   heapsort   makefile  -print    total.c
1-1    fo2.txt  heapsortl  oo.txt    rect       total.c,v
1-3    fool.txt heapsort.c power     rectangle.c uu.tt
1-4    foo.paths lib.a     powerl    rectangle.c~ win.c
1-5    foo.txt  lib.h     powerl.c  test.c
1-6    foo.txt,v love_pico.txt power.c   test.TXT
```

我們在命令模式輸入 :!pwd 就可以顯示目前的目錄。

```
:!pwd
[No write since last change]
/home/chaiyen
```