# 如何從 VMS 轉移到 Linux (From VMS to Linux HOWTO)

By Guido Gonzato, <guido``at''ibogeo.df.unibo.it>, and Mike Miller, <miller5@uiuc.edu>, 譯者：峰舟

v1.1.3, 17 September 1999

這份 HOWTO 主要是針對那些已經用過 VMX 系統，而現在可能因需要而轉型使用 Linux 作業系統的人。希望藉由比較兩者作業系統間指令的差異，而循序漸進的帶領您進入 Linux 的世界。

## 目錄

# 1 簡介

## 1.1 為何要使用 Linux？

您曾經聽過 UNIX 艱深難懂便畏懼丟開 VMS 嗎？別煩惱，Linux，最好的 UNIX 相容機種之一，一點也不會比 VMS 難使用；相反的，我發現 Linux 比 VMS 更容易上手。雖然，VMS 迷可能不以為然，但是，經過許多人的建議是：Linux 確實是比 VMS 更有效力且更是多功。

Linux 和 VMS 兩者都是很好的作業系統，而且本質上都可以非常有效率的完成相同的作業。 但是，Linux 有一些非常吸引人的特色，而使得她成為 VMS 很好的另一個選擇。並且，Linux 可以在個人電腦 (PC) 上使用，而 VMS 卻不行。現在任何一台奔騰(Pentium)的電腦，使用 Linux 為作業系統的效能更可遠遠地勝過 VMS 的機器。更迷人的是，現在以一般的顯示卡，就可以有非常優秀的視窗圖形界面。幾乎總是快過那些專用的電腦。

我想，您應該是大學裡的研究人員或是學生，並且您每天使用 VMS 來完成下列的幾件事：

- 使用 TeX/LaTeX 來寫報告;

- 使用 Fortran 來寫程式;
- 作一些圖形文件;
- 上網使用某些服務;
- 及其他等等.

在下面的章節中 ,我將利用您在 VMS 上的經驗 ,介紹如何在 Linux 上完成上列的每一件事。 首要事物：

- 請確定 Linux 和 X Window 都已經安裝完全;
- 有一位系統管理員可以作為技術指導。（請找他們幫忙 ,請不要來找我 ;-) ）;
- 您系統上的 shell 可以是 bash （請詢問您的系統管理員）;

請注意 ,這份 HOWTO 還不足夠使您全部了解 Linux 的精髓。她只包含了僅有的一些入門的必要知識而已。 您應該學習更多的 Linux 的知識 ,並且充分的利用她。(advanced `bash` features, programming, regular expressions...). 從現在開始 ,RMP 的意思是 'please read the man pages for further details'. 在 Linux 上 ,man pages 其實就是 help 的意思。

The Linux Documentation Project 文件可以在 `<ftp://sunsite.unc.edu:/pub/Linux/docs/LDP>`, 獲得。而這也是重要訊息的來源。 我建議您可以閱讀由 Larry Greenfield 寫的 "Linux User Guide" 書。—這是一本非常適合初學者的入門書。

現在讓我們步入正題吧：

## 1.2 指令的比較和檔案格式

這個表列出 VMS 和 Linux 中常用的指令集。請牢記在心 ,其語法也將大大的不同。 詳細細節請參考相關章節。

```
VMS                             Linux                   注意事項(Linux 部份)
------------------------------------------------------------------------------

@COMMAND                        command                 必須可被執行
COPY file1 file2                cp file1 file2
CREATE/DIR [.dirname]           mkdir dirname           一次只能建一個目錄
CREATE/DIR [.dir1.dir2]         mkdirhier dir/name
DELETE filename                 rm filename
DIFF file1 file2                diff -c file1 file2
DIRECTORY                       ls
DIRECTORY [...]file             find . -name file
DIRECTORY/FULL                  ls -al
EDIT filename                   vi filename,            您可能不會喜歡使用 vi
                                emacs filename,         EDT compatible
                                jed filename            ditto---my favourite
FORTRAN prog.for                g77 prog.f,             不需要作 LINK
                                f77 prog.f,
                                fort77 prog.f
```

| | | |
|---|---|---|
| HELP command | man command | must specify ‘command’ |
| | info command | ditto |
| LATEX file.tex | latex file.tex | |
| LOGIN.COM | .bash_profile, | ‘hidden’ file |
| | .bashrc | ditto |
| LOGOUT.COM | .bash_logout | ditto |
| MAIL | mail, | 最原始的 email 程式 |
| | elm, | 比 mail 好。 |
| | pine | 不錯，但是都只能處理文字。 |
| PRINT file.ps | lpr file.ps | |
| PRINT/QUEUE=laser file.ps | lpr -Plaser file.ps | |
| PHONE user | talk user | |
| RENAME file1 file2 | mv file1 file2 | not for multiple files |
| RUN progname | progname | |
| SEARCH file "pattern" | grep pattern file | |
| SET DEFAULT [-] | cd .. | |
| SET DEFAULT [.dir.name] | cd dir/name | |
| SET HOST hostname | telnet hostname, | 不太相同 |
| | rlogin hostname | |
| SET FILE/OWNER_UIC=joe | chown joe file | 完全不相同 |
| SET NOBROADCAST | mesg | |
| SET PASSWORD | passwd | |
| SET PROT=(perm) file | chmod perm file | 完全不相同 |
| SET TERMINAL | export TERM= | 語法不同 |
| SHOW DEFAULT | pwd | |
| SHOW DEVICE | du, df | |
| SHOW ENTRY | lpq | |
| SHOW PROCESS | ps -ax | |
| SHOW QUEUE | lpq | |
| SHOW SYSTEM | top | |
| SHOW TIME | date | |
| SHOW USERS | w | |
| STOP | kill | |
| STOP/QUEUE | kill, | 針對 processes |
| | lprm | 針對 print queues |
| SUBMIT command | command & | |
| SUBMIT/AFTER=time command | at time command | |
| TEX file.tex | tex file.tex | |
| TYPE/PAGE file | more file | |
| | less file | much better |

當然，這不只是指令的名稱不一樣而已，請繼續讀之。

# 2 牛刀小試

這些絕對是您在第一次登入系統前必須知道的一些事情，放輕鬆，不會太複雜的。

## 2.1 檔案

- 在 VMS 上的檔名的表示法是 filename.extension;version。 但是在 Linux 上 ，檔案的版本號碼將不會再出現（重要的限制 ，請參閱 10.2 (Linux 上的檔案版本)); 檔名最常只能有 255 個字元 ，檔名中可以出現很多個點 “・”。例如：This.is_a_FILEname.txt.
- Linux 對檔名的大小寫是有分別的： FILENAME.txt 和 filename.txt 是兩個完全不一樣的檔名; ls 是一個指令 ，可以被執行 ，但是 LS 卻不是指令 ，會出現錯誤訊息。
- 檔名的第一個字元如果是一點“・” ，則這個檔名是‘隱藏檔’（也就是說 ，如果不加特別指令的話 ，一般的 ls 將不會顯示出這些隱藏檔） ，而檔案的結尾如果是“蚯蚓號”‘~’ ，這表示備份檔案（或是暫存檔）・

現在 ，下表整理出把 VMS 的指令對應到 Linux 上的指令・

```
VMS                                     Linux
---------------------------------------------------------------------

$ COPY file1.txt file2.txt              $ cp file1.txt file2.txt
$ COPY [.dir]file.txt []                $ cp dir/file.txt .
$ COPY [.dir]file.txt [-]               $ cp dir/file.txt ..
$ DELETE *.dat;*                        $ rm *dat
$ DIFF file1 file2                      $ diff -c file1 file2
$ PRINT file                            $ lpr file
$ PRINT/queue=queuename file            $ lpr -Pprintername file
$ SEARCH *.tex;* "geology"              $ grep geology *tex
```

其他對於目錄方面的指令 ，請見一章節；檔案的屬性（protections, ownership...）請參照 8 (進階)。

## 2.2 目錄

- 在同一個節點（node）或是裝置（device） ，在 VMS 上 ，目錄的形式看起來應該像這樣：[top.dir.subdir]; 在 Linux 上是：/top/dir/subdir/. 對上層的目錄我們稱為“根目錄”（/）; 所有子目錄 ，都是由根目錄向下延伸： /bin, /usr, /tmp, /etc, 等等。
- 目錄名稱 /home 是存放使用者目錄 ，例如：/home/guido, /home/warner, 等等。 當使用者登入的時候 ，她們會登入到她們自己擁有的使用者目錄；這和 SYS$LOGIN 是有異曲同工之妙。 切換到只用者目錄有個方便的方法：使用“蚯蚓號”‘~’。 所以 ，cd ~/tmp 和 cd /home/guido/tmp 是一樣的意思。
- 目錄名稱的命名方式和檔案名稱的命名方式相同。 再者 ，每個目錄都會有兩個特別的目錄名稱：一個是：. ，表示指到自己的目錄。 (像 [])； 一個是：.. ，表示指到上一層目錄 (像 [-])。

讓我們來看看其他的例子：

```
VMS                                     Linux
---------------------------------------------------------------------
```

| | |
|---|---|
| `$ CREATE/DIR [.dirname]` | `$ mkdir dirname` |
| `$ CREATE/DIR [.dir1.dir2.dir3]` | `$ mkdirhier dir1/dir2/dir3` |
| `n/a` | `$ rmdir dirname` |
| | `(if dirname is empty)` |
| | `$ rm -R dirname` |
| `$ DIRECTORY` | `$ ls` |
| `$ DIRECTORY [...]file.*;*` | `$ find . -name "file*"` |
| `$ SET DEF SYS$LOGIN` | `$ cd` |
| `$ SET DEF [-]` | `$ cd ..` |
| `$ SET DEF [top.dir.subdir]` | `$ cd /top/dir/subdir` |
| `$ SET DEF [.dir.subdir]` | `$ cd dir/subdir` |
| `$ SHOW DEF` | `$ pwd` |

對於檔案的屬性（protections, ownership），還有一些進階主題，請參照 8 (進階).

## 2.3 程式

- 指令，編譯過的程式（執行檔），和 shell scripts (VMS 上的 'command files') 並沒有強制規定需要有特定的副檔名。像 `.EXE` or `.COM` 等，只要您喜歡，您想要她叫什麼名字都可以。如果您使用 `ls -F` 來察看執行檔的時候，您會發現執行檔會被標示成 '*'。
- 欲執行可執行的程式，只要輸入指令名稱就可以了。（沒有所謂的 `RUN PROGRAM.EXE` 或是 `@COMMAND`）。注意：這個執行檔必須指定某路徑。基本上，預設錄經包含了 `/bin`, `/usr/bin`, `/usr/X11R6/bin`, 等等。如果，您自己寫了一個程式，您可以將這個程式放在您自己的預設路徑中（如何操作，請參照 9 (設定)）。或是您也可以指定一個絕對路徑來執行這隻程式，例如：`/home/guido/data/myprog`; 或是 使用相對路徑 `./myprog`。
- 指令的參數在 VMS 上是使用 `/OPTION=` 來完成，而在 Linux 上，我們使用 `-switch` 或是 `-switch`（`switch` 參數可能是一個字元，許多字的組合，或是一個字）。比較特別的，參數 `-R` (recursive) 在 Linux 上和 在 VMS 上 `[...]` 扮演著相同的功能;
- 在命令列中，您可以一次下很多個指令：

```
$ command1 ; command2 ; ... ; commandn
```

- Linux 大部分的彈性來自於兩個特性，而這兩個特性在 VMS 上卻很難被表示出來，或是根本就沒有。她們分別是：I/O 重導(redirection) 和 管道 (piping)。（曾經有人告訴我最近的 DCL 版本有支援重導(redirection)和管道(piping)，可是我沒有那個版本）。重導(redirection) 在 VMS 上只是從屬的特徵（像指令裡的參數 `/OUTPUT=`），或是一個難討好的行程(process)。例如：

```
$ DEFINE /USER SYS$OUTPUT OUT
$ DEFINE /USER SYS$INPUT IN
$ RUN PROG
```

在 Linux 就只要這樣就可以達成：

```
$ prog < in > out
```

Piping 實際上在 VMS 上並沒有提供，但是在 Linux 卻是一個重要的角色。典型的例子如下：

```
$ myprog < datafile | filter_1 | filter_2 >> result.dat 2> errors.log &
```

> 意思是說：程式 myprog 導入一個檔案為 input （由 “<”讀入檔案 datafile）。 其執行後的結果被轉為(pipe)另一個程式(filter_1) 的 input ，處理後得到的結果再轉為另外一隻程式(filter_2)的 input ，再作進一步的處理後 ，最後的結果被附加到 result.dat 檔案上 ，將錯誤訊息覆蓋到 errors.log 的檔案上。 而這一切的動作 ，都在背景執行。 想要看更精彩的例子 ，請參照 11 (實例).

對於多工 ，‘queues’, 或其他類似主題 ，請參閱 8 (進階).

## 2.4 快速瀏覽

現在您可以來試試 Linux 的威力了。輸入您的 login name 和 password。假設您的 login name 是 john ，Password 是 My_PassWd, 不要 輸入 John 或是 my_passwd。 記住 ，大小寫對 Unix 來說 ，是不同的意思。

一當您登入後 ，您就會看到系統提示符號(prompt)；可能看起來像 machinename:$。 如果您想更改您的系統提示符號 ，或是在您一 Login 後就自動執行一些程式 ，您可能需要去編輯一個隱藏檔（ .profile 或是 .bash_profile）。 （實例請參照“組態設定”一章節 9 (設定)）。 這和 VMS 上的LOGIN.COM 有異曲同工之妙。

您可以使用 ALT–F1, ALT–F2, ... ALT–F6 來切換六個 ‘virtual consoles’。 當您其中的一個 virtual consoles 正在忙於處理某一隻程式的時候 ，您可以任意的切換到任何一個 virtual consoles 繼續您其他的工作。 試著切換登入到其他的 virtual consoles 吧！

現在您可能想要開始執行 X Windows 了吧（以後我們簡稱為 X）。X 是圖形界面環境 ，有點像 DECWindows — 實際上 ，DECWindows 是衍生自 X Window system. 請輸入 startx 並且等待幾秒您將會看到 xterm 或是其他類似的終端機模擬程式被開啓 ，可能還會有一個工作列。（視您系統管理員如何設定 ，不一樣的設定 ，所看到的結果將會不同）。 試著在桌面上同時按下滑鼠的左右鍵 ，您可以看到一個彈跳視選單出現（滑鼠按鍵的功能也因設定不同而有不同的功能）。

進入 X 後 ，如果想要執行文字模式的終端機 (‘console’) 同時按住下列任一組按鍵： CTRL–ALT–F1 ... CTRL–ALT–F6 （需要同時按）。是看看！如果想要跳回視窗界面 ，可以 同時按下 ALT–F7 就可以了。欲結束 X ，這著選單上的 exit 選項結束 ，或是同時按下 CTRL–ALT–BS.

輸入下列指令 ，可以列出您現在目錄下的檔案（包括隱藏檔）：

```
$ ls -al
```

按 SHIFT–PAG UP 可以往回捲到以前的螢幕。

現在我們來看看指令 ls 的說明檔(help) ，輸入：

```
$ man ls
```

按下 ’q’ 可以跳出。如果要登出 ，可以輸入 exit 登出目前終端機。 如果想要關機 ，可以同時按下 CTRL–ALT–DEL 並且等帶數秒就可以了。 (注意： 絕對不要直接就關掉您電腦上的電源 ，請按照一定的關機程序才可以關掉電源 ，不然可能會傷害到檔案系統）。

如果您想您已經可以上路了 ，那就放手去作吧！但是 ，如果我是您的話 ，可會先跳到進階主題這一章節： 8 (進階).

# 3 檔案編輯

Linux 沒有 EDT 編輯器 ,但是有很多的編輯器選擇可供使用。 在所有UNIX 系統中 ,唯一保證一定有 vi 這個編輯器—算了吧 ,您的系統管理員一定已經安裝比 vi 更好用的編輯器了。可能比較受歡迎的編輯器是 emacs ,這可以用來模擬 VMS 上的 EDT ; jed 是另一種類似 EDT 的編輯器。

這兩種編輯器用來編寫程式碼相當的實用 ,因為她們有兩個 EDT 沒有的功能: 高亮度相關字(syntax hilighting)和自動縮排(automatic indentation)。 而且 ,您可以不需要跳出編輯器 ,就可以直接在編輯器裡編譯(Compile)您的程式。 (command ESC-X compile); 如果有語法錯誤 ,游標將會自動跳到有錯誤的地方。 我敢打賭您將不會想再看到那只有藍色的 EDT 了。

如果您有 emacs: 啓動她 ,然後輸入 ESC-X edt-emulation-on。 鍵入 ALT–X or ESC-X 可以跳出 emacs ,像 EDT 的 CTRL–Z。 從現在開始 ,emacs 就像 EDT 一樣 ,除了一些指令不太一樣。 不同處:

- 不行 使用 CTRL–Z 來執行命令列指令。 (如果您做了 ,您的 emacs 將會停止工作 ,不過您可以試著輸入 fg 來繼續您的 emacs);
- Emacs 有大量的線上說明(on-line help)。按 CTRL-H ?, 或是 CTRL-H T 啓動。;
- 存檔: CTRL-X CTRL-S;
- 離開: CTRL-X CTRL-C;
- 新增一個檔案: CTRL-X CTRL-F, 然後 CTRL-X B 可以在個檔案中作切換。

如果您有 jed: 可以請您的系統管理員恰當地設定 jed。 當您啓動她後 ,模擬器就已經自動載入。可以使用 CTRL–H CTRL–H or CTRL-? 啓動線上說明。指令部份 ,大致上和 emacs 相同。 此外 ,還有一些易於操作的操作鍵沒有被裝訂在原來的 EDT 中; 這些操作鍵可以依您的個人喜好自行設定 ,請詢問您的系統管理員。

或許您可能會使用另一個完全不一樣操作界面的編輯器。但是 emacs 直覺上會是一個不錯的選擇。 另一個十分受歡迎的編輯器是:joe, 她可以模擬其他編輯器 ,像 emacs。 (甚至更容易上手) 或是 Dos 的編輯器。 啓動這個編輯器就像 jmacs 或是 jstar 一樣 ,可以使用 CTRL-X H 或是 CTRL-J 得到豐富的線上說明。 emacs 和 jed 比 EDT 更具有威力。

# 4 TeXing

TeX 和 LaTeX 在 Linux 上完全和 VMS 一樣—不同的地方是 ,Linux的執行速度更快 :-), 這個工具用來處理 .dvi 和 .ps 的檔案更俱優勢:

- 由 TeX 執行 TeX 檔 ,就像平常一樣: tex file.tex;
- 轉換 .dvi 擋到 .ps 檔 ,輸入 dvips -o filename.ps filename.dvi;
- 檢視 .dvi 檔 ,在 X Window 下輸入: xdvi filename.dvi &. 在顯示頁按下滑鼠可以放大。 這是一個智慧型的程式:如果您編輯且執行 TeX 產生一個較新版本的 .dvi 檔案 ,xdvi 會自動更新;
- 檢視 .ps 檔 ,在 X Window 下輸入 ghostview filename.ps &. 按滑鼠可以縮放。 全部的檔案或是部份檔案可以被列印出來。 然而還有一個比 TeX 更好的程式:gv.
- 列印 .ps: 通常指令是 lpr mypaper.ps ,但是如果 PostScript 列表機被呼叫(假設使用 ps 指令)您將會是執行: lpr -Pps mypaper.ps。對於更多有關 print queues 的訊息 ,請參照 8.4 (列印埠).

# 5 程式

在 Linux 上寫程式更是舒服：有許多好用得工具可以讓寫程式更加快速更加簡單。 例如：寫程式無非是單調沈悶的工作：編輯–存檔–離開–編譯–重新編輯 一直反覆的執行同樣的動作。 但是 ，如果使用像 emacs 或是 jed 等工具 ，上述那些動作 ，將統一整合在編輯器裡 ，一次完成。

## 5.1 Fortran

大體上 ，沒有多大的差別 ，但是請注意 ，這個免費的編譯器可不是百分之百的和 VMS 相容的。 可能會有些奇怪的結果發生。（實際上 ，VMS 的 Fortran 編譯器並不是一個可以標準延伸的(non-standard)） 。 詳細情形請看 /usr/doc/g77/DOC or /usr/doc/f2c/f2c.ps。

您的系統管理員已經安裝了一個叫做 g77 的 compiler (不錯但是, 在 0.5.21 版, 還是沒有完全和 DEC 的 Fortran 百分百地相容) ，或是一個把 Fortran 轉成 C 語言的轉換器 ，叫做 f2c。 加上她的一個前端(front-ends)裝置 ，模擬就像是一個 Fortran 編譯器。 依我的經驗 ，yaf77 套件提供最好的解決方案。

欲使用 g77 來 compile 您的 Fortran 程式, 先用任意編輯記編輯程式碼 ，以副檔名 .f 存檔 ， 然後執行如下：

```
$ g77 myprog.f
```

她會自動幫您產稱 a.out 的執行檔。(您不需要作任何 link 的動作） 如果想要產生不同的執行檔檔名 ，可以下參數：

```
$ g77 -O2 -o myprog myprog.f
```

請注意最佳化！要求您的系統管理員細讀編譯器的說明文件 ，並且告知您是否還有任何的存在問題。

編譯中間碼(subroutine)：

```
$ g77 -c mysub.f
```

這會產生 mysub.o 這個檔案。 然後您需要作連結(link)的動作。

```
$ g77 -o myprog myprog.f mysub.o
```

如果您想作成函數庫(library) ，您可以這樣作：

```
$ cd subroutines/
$ cat *f >mylib.f ; g77 -c mylib.f
```

這會產生出 mylib.o ，您可以使用她來連結您的程式。

最後 ，如果要 link 其他的函數庫(library) ，我們假設 libdummy.so:

```
$ g77 -o myprog myprog.f -ldummy
```

如果您使用 f2c, 您只能使用 f77 或是 fort77 ，而不能使用 g77。

另一個有用的工具是 make, 詳述如下：

## 5.2 如何使用 make

make 可以用來編輯很多分散於個檔案的原始碼。在 VMS 上我們稱為 MMS 和 MMK, 但是她們和 Linux 有不同的語法。

設您有一些原始程式 ,需要作些例行程序。程式分別為 file_1.f, file_2.f, file_3.f, 主程式為 myprog.f。 如果您是手動編譯您的程式 ,每當您修改您的程式碼後 ,您必須知道哪個檔案和哪個檔案有關連 ,哪個檔案必須先編譯 ,等等。

與其發瘋 ,不如您可以寫個 'makefile'。 這是一個文字檔 ,裡面記錄著您程式碼與程式碼之間的關連性。 當其中一個檔案被修改後 ,只有與這個被修改的檔案有相關的檔案需要被重新編譯。

例如 ,您寫了一個 makefile 如下：

```
# 這是一個 makefile
# 使用 <TAB> 鍵 ,當您見到 <TAB> 標籤時!
# 這非常的重要 ,請不要使用空白鍵代替。

myprog: myprog.o file_1.o file_2.o file_3.o
<TAB>g77 -o myprog myprog.o file_1.o file_2.o file_3.o
# myprog depends on four object files

myprog.o: myprog.f
<TAB>g77 -c myprog.f
# myprog.o depends on its source file

file_1.o: file_1.f
<TAB>g77 -c file_1.f
# file_1.o depends on its source file

file_2.o: file_2.f file_1.o
<TAB>g77 -c file_2.f file_1.o
# file_2.o depends on its source file and an object file

file_3.o: file_3.f file_2.o
<TAB>g77 -c file_3.f file_2.o
# file_3.o depends on its source file and an object file

# end of makefile.
```

儲存這個檔案 ,命名為 Makefile 且在命令列輸入 make 來編譯您的程式; 或是您也可以將她存成 myprog.mak 然後使用 make -f myprog.mak 來編譯. 接下來的後續動作, RMP.（還記得什麼是 RMP 吧！）

## 5.3 Shell Scripts

Shell scripts 就像 VMS 上的 command files。, 這可以建構出非常有用的功能。

要寫一個 script, 您所要作的就只是寫一個包含一些指令的文字檔 ，然後存檔 ，改變成可執行的模式（使用 `chmod +x <scriptfile>`）。 只要輸入該 Script 的名字就可以執行了。

一個 Script 是一個非常重大的工程 ，這需要一本書。這裡我就不再多做說明了。

我只給您一個或多或少的綜合的（希望）有用的例子 ，您或許可以從這個例子中得到一些基本的規則。

EXAMPLE: sample.sh

```
#!/bin/sh
# sample.sh
# I am a comment
# 不要修改第一行 ，她必須以這種形式存在在第一行！！！
echo "This system is: `uname -a`" # use the output of the command
echo "My name is $0" # built-in variables
echo "You gave me the following $# parameters: "$*
echo "First parameter is: "$1
echo -n "What's your name? " ; read your_name
echo notice the difference: "hi $your_name" # quoting with "
echo notice the difference: 'hi $your_name' # quoting with '
DIRS=0 ; FILES=0
for file in `ls .` ; do
  if [ -d ${file} ] ; then # if file is a directory
    DIRS=`expr $DIRS + 1`  # this means DIRS = DIRS + 1
  elif [ -f ${file} ] ; then
    FILES=`expr $FILES + 1`
  fi
  case ${file} in
    *.gif|*jpg) echo "${file}: graphic file" ;;
    *.txt|*.tex) echo "${file}: text file" ;;
    *.c|*.f|*.for) echo "${file}: source file" ;;
    *) echo "${file}: generic file" ;;
  esac
done
echo "there are ${DIRS} directories and ${FILES} files"
ls | grep "ZxY--!!!WKW"
if [ $? != 0 ] ; then # exit code of last command
  echo "ZxY--!!!WKW not found"
fi
echo "enough... type 'man bash' if you want more info."
```

## 5.4 C 語言

Linux 是一個寫 C 語言的好地方。就假設您會 C 語言吧！這裡也幾個指導方針。 編譯您的程式`hello.c` ，您會使用到 gcc compiler, 這已是 Linux 的一部份了。 而且和 `g77` 有相同的使用方法:

```
$ gcc -O2 -o hello hello.c
```

連結(link)函數庫(library) ，加入參數 -l<libname>。 例如 ，要 link 數學函數庫和最佳化 ，可以下如下指令：

```
$ gcc -O2 -o mathprog mathprog.c -lm
```

( -l<libname> 參數强迫 gcc 連結(link)函數庫(library) /usr/lib/lib<libname>.a; 所以 -lm 就連結了 /usr/lib/libm.a).

如果您的程式是由許多檔案組成的 ，您可能也需要使用前面所提到的 make 這個工具。 在 makefile 使用 gcc 和 C 語言的原始碼檔案就可以了。

您也可以獲得有關 C 語言的函數的説明。這些説明文件都已經被編成 man pages ，第三節(section 3) 了。 例如:

```
$ man 3 printf
```

同時 ，有許許多多的函數庫可供使用。其中您第一想要使用的是 ncurses, 這可以用來處理文字模式下的特效。或是 svgalib, 這可以用來處理圖形模式。

# 6 圖形界面

在許多圖形套件中 ，gnuplot 真是其中之翹楚。進入 X Window 後輸入 gnuplot, 可是著讀入兩個範例檔: 2D-data.dat (每行有兩筆資料), 和 3D-data.dat (每行有三筆資料).

例：2-D graphs:

```
gnuplot> set title "my first graph"
gnuplot> plot '2D-data.dat'
gnuplot> plot '2D-data.dat' with linespoints
gnuplot> plot '2D-data.dat', sin(x)
gnuplot> plot [-5:10] '2D-data.dat'
```

例：3-D graphs (每一行中的資料以空白鍵最為分隔):

```
gnuplot> set parametric ; set hidden3d ; set contour
gnuplot> splot '3D-data.dat' using 1:2:3 with linespoints
```

只有一行資料的檔案(如時間序列)也可以被繪製成 2-D 的圖形：

```
gnuplot> plot [-5:15] '2D-data-1col.dat' with linespoints
```

或是繪成 3-D graph :

```
gnuplot> set noparametric ; set hidden3d
gnuplot> splot '3D-data-1col.dat' using 1 with linespoints
```

列印圖形：假設您的 Postscript 列表機的指令是 lpr -Pps file.ps ，步驟如下:

```
gnuplot> set term post
gnuplot> set out '| lpr -Pps'
gnuplot> replot
```

然後輸入 `set term x11` 回復成 x11 模式。 別懷疑 ，當您離開 `gnuplot` 後 ，文件將會自動列印出來。多訊息 ，可輸入 `help` 或是參見目錄中的範例。 （`/usr/lib/gnuplot/demos/`）

# 7 Mail 和 Internet 工具

因為網路是誕生在 Unix 的機器上 ，您可以在 Linux 上發現許多好用且易上手的軟體。 者裡所列 ，只是其中的一部份而已：

- **Mail**: 使用 `elm` 或是 `pine` 來處理您的 email; 這兩個應用程式都有線上說明(Online Help)。 對於較短的訊息 ，您可能使用 `mail`, 如 `mail -s "hello mate" user@somewhere < msg.txt`. 您可能會愛上像 `xmail` 這樣的應用程式 ，或是類似的程式。
- **Newsgroups**: 可使用 `tin` 或是 `slrn`, 兩者都是直覺式 ，且自我解讀(self-explanatory).
- **ftp**: 欲脫離文字界面的 `ftp` 應用程式, 要求您的系統管理員安裝全螢幕(full-screen)模式的 ftp 軟體 ，如 `ncftp` ，或是圖形界面的 `xftp`.
- **WWW**: 普遍存在的 `netscape`, 或是 `xmosaic`, `chimera`, 和 `arena` 都是圖形界面的瀏覽器。另外還有一個文字界面的瀏覽器 `lynx`, 非常的快速而且又有效率。

# 8 進階

到這個階段 ，遊戲就比較複雜了。學完這些東西 ，您就可以說 ，我懂一些 Linux 的東西 ;-)

## 8.1 允許權(Permissions) 及 所有權(Ownership)

檔案和目錄都有允許權(Permission) (即 VMS 上的 'protections') 和所有權(ownership), 這個觀念不難理解 ，因為 VMS 上也有。如果您不能執行一隻程式 ，或是您不能修改某一檔案 ，或是不能進入某目錄 ，這是因為您沒有被賦予使用這些資源的權力的關係（沒有允許權） ，或者是這個檔案的根本就不屬於您的（沒有所有權） ，讓我們看看下面的例子吧：

```
$ ls -l /bin/ls
-rwxr-xr-x   1 root     bin         27281 Aug 15  1995 /bin/ls*
```

第一個欄位顯示出這個檔案的使用權限 `ls` (擁有者(owner) 是 root, 群組(group) 是 bin). 擁有權可以分成三種模式: 擁有者(owner), 群組(group), 和 其他使用者(others)。（和 VMS 的 owner, group, world 相似）；有三種不同的允許權(permissions): 讀(read), 寫(write) (包含移除(delete)), 和執行(execute).

從左到右 ，`-` 是檔案模式 (`-` = 一般檔案(ordinary file), `d` = 目錄(directory), `l` = 連結(link), 等); `rwx` 是允許檔案擁有者可以對該檔案作讀 ，寫 ，和執行的動作； `r-x` 是允許群組使用者可以對該檔案作讀和執行的動作 ，但是不可以作寫的動作。； `r-x` 是限制其他的使用者對該檔案只有讀和執行的權限 ，卻沒有寫的權限。

如何改變檔案的使用權限：

```
$ chmod <whoXperm> <file>
```

其中 who 是 u (user（使用者）, 即檔案的擁有者), g (群組 group), o (其他 other), X 表示可以為 + 或 -, perm 可以為 r (讀 read), w (寫 write), 或是 x (可執行 execute). 舉例說明：

```
$ chmod u+x file
```

這是針對該檔案 ，開放可執行的權限給該檔案的擁有者。可縮寫成： `chmod +x file`.

```
$ chmod go-wx file
```

這是對該檔案 ，移除群組使用者和其他使用者對該檔案的“寫”的權限和“執行”的權限。

```
$ chmod ugo+rwx file
```

這是開放該檔案的讀 ，寫 ，和執行的權限給所有的使用者。

另一個快速表示法為使用一組數字：`rwxr-xr-x` 可以用 755 來表示 (每一位數字對應一個控制位元: `--` 表是 0, `-x` 表示 1, `-w-` 表示 2...).

所以 ，rwx = 4+2+1 = 7 ，r-x = 4+0+1 = 5 ，所以 ，`rwxr-xr-x` 可以用 755 來表示。

對目錄而言 ，`rx` 表示您可以切換到該目錄： `cd` , 而 `w` 表示您可以刪除該目錄的檔案(當然需要視該檔案的使用權限而定), 或是目錄本身. 這些還只是冰山的一角而已 ，想要知道更多—RMP.

如何改變檔案的擁有權：

```
$ chown username file
```

我整理成下表：

```
VMS                                Linux                    Notes
------------------------------------------------------------------------------


SET PROT=(O:RW) file.txt         $ chmod u+rw file.txt
                                 $ chmod 600 file.txt
SET PROT=(O:RWED,W) file         $ chmod u+rwx file
                                 $ chmod 700 file
SET PROT=(O:RWED,W:RE) file      $ chmod 755 file
SET PROT=(O:RW,G:RW,W) file      $ chmod 660 file
SET FILE/OWNER_UIC=JOE file      $ chown joe file
SET DIR/OWNER_UIC=JOE [.dir]     $ chown joe dir/
```

## 8.2 多工(Multitasking): 程序(Processes) 和 工作(Jobs)

欲執行很多程式 ，在 Linux 上沒有所謂的 ‘batch queues’ ; 多工其實以非常不一樣的手法處理。 以下是一般常見的命令列：

```
$ command -s1 -s2 ... -sn par1 par2 ... parn < input > output &
```

其中 `-s1`, ..., `-sn` 是該程式的參數（switch）, `par1`, ..., `parn` 是程式的輸入變數(parameters).

現在讓我們來看看在 Linux 上如何多工。程序（processes）是一隻不論在前景或是背景被執行的程式。

- 在背景執行程序:

  ```
  $ progname [-switches] [parameters] [< input] [> output] &
  [1] 234
  ```

  shell 會提示您該程序(process)的工作號碼（'job number'） (第一個數字; 請看下文) 和該程序的程序識別碼 PID (Process IDentifier). 每一個程序都有一組獨一無二的 PID.

- 檢視有多少 Processes 被執行：

  ```
  $ ps -ax
  ```

  這會輸出一大串正在執行的程序(processes)。

- 殺掉程序：

  ```
  $ kill <PID>
  ```

  如果您不知道要如何正確的終止某一個程序 ，這時您可能就要手動把程序給殺死喔！... ;-). 有些時候 ，一個程序只能被下列一個指令殺死：

  ```
  $ kill -15 <PID>
  $ kill -9 <PID>
  ```

附帶一提 ，Sell 允許您停止或是暫時停止一個程序 ，將程序送到背景工作 ，或是將程序從背景帶到前景。 在這裡 ，程序(processes)我們稱之為工作('jobs').

- 檢視有多少工作（jobs）正在執行：

  ```
  $ jobs
  ```

  工作(jobs)是由 Shell 給的工作號碼（job number）來作區別 ，注意 ，不是 PID 喔.

- 終止在前景工作的程序(processes):

  ```
  $ CTRL-C
  ```

  (不一定每一次奏效)

- 暫停在前景工作的程序(process):

  ```
  $ CTRL-Z
  ```

  (同上)

- 將暫停的程序轉換成背景工作(background jobs):

  ```
  $ bg <job>
  ```

- 將背景工作帶到前景:

  ```
  $ fg <job>
  ```

- 殺掉一個工作(job):

  ```
  $ kill <%job>
  ```

### 8.3 在訪檔案(files)

更多有關檔案的訊息。

- **stdin, stdout, stderr**: 在 UNIX 中 ,任何一個系統元件都被視為是一個檔案。 指令或是應用程式讀入一個所謂標準輸入的檔案（`stdin`：通常為鍵盤） ，輸出結果到所謂標準輸出的檔案（`stdout`：通成為螢幕） ,而錯誤輸出也是輸出到一個檔案 ，我們稱之為 `stderr` (通常也是螢幕).

  使用 < 和 > ,您可以將輸出和輸入重新導向到另外一個檔案。 然而 ，>> 是將結果附加到檔案的檔尾 ,而不是覆蓋原來的檔案。; 2> 重導錯誤訊息(stderr); 2>&1 重導錯誤訊息到標準輸出, 而 1>&2 則是重導標準輸出到錯誤輸出。 有一個黑洞（'black hole'）叫做 `/dev/null`: 所有導到這個檔案的東西 ，都將消失的無影無蹤。
- **wildcards**: 萬用字元 '`*`' 有相同的功能。用法: * 符合所有檔案 ,除了隱藏檔以外。 .* 符合所有隱藏檔; *.* 符合只有一個點 '.' 在中間的檔案。跟隨著其他字元： p*r 表示 'peter' 或是 'piper' 都有可能; *c* 可能符合 'picked' 或是 'peck'。 '`%`' 換成 '`?`'。 還有另外一種萬用字元：`[]`. 用法: `[abc]*` 表示檔案必須以字元 a, b, c 最為開頭; `*[I-N,1,2,3]` 表示檔案名稱由 I, J, K, L, M, N, 1, 2, 3 作為結尾。
- `mv` (重新命名) 不可以一次多個檔案。即 `mv *.xxx *.yyy` 將不會執行成功;
- 使用 `cp -i` 和 `mv -i` 可以提醒您是否要覆蓋以存在的檔案。

### 8.4 列印佇列

就像 VMS ,列印工作會先被放到佇列當中(queues) ,當您下列印指令的時候 ,您或許會指定列表機的名稱。 例如：

```
$ lpr file.txt          # 這將指用到預設的列表機
$ lpr -Plaser file.ps   # 這將使用名為 'laser' 的列表機
```

處理列印佇列(print queues), 您可能需要下列指令：

```
VMS                                     Linux
------------------------------------------------------------------------------

$ PRINT file.ps                         $ lpr file.ps
$ PRINT/QUEUE=laser file.ps             $ lpr -Plaser file.ps
$ SHOW QUEUE                            $ lpq
$ SHOW QUEUE/QUEUE=laser                $ lpq -Plaser
$ STOP/QUEUE                            $ lprm <item>
```

## 9 設定

您的系統管理員已經提供您一些組態設定檔 ,如： `.xinitrc`, `.bash_profile`, `.inputrc`, 等等。 其中您想要編輯的是：

- `.bash_profile` 或是 `.profile`: 在 login 時 ，這個檔案就被 Shell 給讀進來了。就像 VMS 上的 `LOGIN.COM`;
- `.bash_logout`: 當要登出的時候 ，讀取這個檔案。這就像 VMS 上的 `LOGOUT.COM`;
- `.bashrc`: 被 non–login shells 讀取.
- `.inputrc`: 這個檔案可以定做一組合鍵和設定 shell 的行為。

我將以我的 `.bash_profile` 最為實例。

---

```
# $HOME/.bash_profile

# 如果不需要 ，請不要重新定義 $PATH 變數。
echo $PATH | grep $LOGNAME > /dev/null
if [ $? != 0 ]
then
  export PATH="$PATH:/home/$LOGNAME/bin"  # add my dir to the PATH
fi

export PS1='LOGNAME:\w\$ '
export PS2='Continued...>'

# aliases 別名

alias bin="cd ~/bin" ; alias cp="cp -i" ; alias d="dir"
alias del="delete" ; alias dir="/bin/ls $LS_OPTIONS --format=vertical"
alias ed="jed" ; alias mv='mv -i'
alias u="cd .." ; alias undel="undelete"

# A few useful functions 一些有用的函數！酷！

inst() # Install a .tar.gz archive in current directory.
{
  gzip -dc $1 | tar xvf -
}
cz() # List the contents of a .zip archive.
{
  unzip -l $*
}
ctgz() # List the contents of a .tar.gz archive.
{
  for file in $* ; do
    gzip -dc ${file} | tar tf -
  done
}
tgz() # Create a .tgz archive a la zip.
{
```

```
  name=$1 ; tar -cvf $1 ; shift
  tar -rf ${name} $* ; gzip -S .tgz ${name}
}
```

這是我的 .inputrc:

```
# $HOME/.inputrc
# 上次修改日期: 16 January 1997.
#
# 以下是註解 ,說明這個 Script 的用意。
# This file is read by bash and defines key bindings to be used by the shell;
# what follows fixes the keys END, HOME, and DELETE, plus accented letters.
# For more information, man readline.

"\e[1~": beginning-of-line
"\e[3~": delete-char
"\e[4~": end-of-line

set bell-style visible
set meta-flag On
set convert-meta Off
set output-meta On
set horizontal-scroll-mode On
set show-all-if-ambiguous On

# (F1 .. F5) are "\e[[A" ... "\e[[E"

"\e[[A": "info "
```

# 10 有用的程式

## 10.1 瀏覽檔案: less

您將會需要每天使用這個檔案瀏覽器。所以我將給您一些祕笈讓您能好好的利用這個工具。 首先 ,要求您的系統管理員這定 less 使她不只可以顯示文字檔 ,更可以壓縮檔案文件 ,等。

就像現在版本的 TYPE, less 可以讓您瀏覽您目前目錄下的檔案。 她同時也可以接受許多額外由鍵盤輸入的按鍵指令 ,比較有用的是：

- 第一, 按下 q 可以離開檔案瀏覽;
- h 提供您許多的線上說明;
- g 跳到檔頭, G 移至檔尾, 數字 n 加 g 表示到弟 n 行。(例如 125g), 數字加 % 表示一到該檔案之百分比大小的地方;
- /pattern 向下搜尋 'pattern'; n 向下搜尋下筆資料; ?pattern 和 N 表示向上尋找。

- m+letter 標示目前地方 (如 ma); '+letter 到被標示的地方.
- :e 檢查新檔案;
- !command 執行命令列檔案.

## 10.2 在 Linux 下的版本備份

唉, Linux 並沒有提供檔案版本的功能。但是您可以用兩種發法解決這個問題。 第一 ,使用 RCS, 版本管理系統（the Revision Control System）, 她可以讓您管理先前的檔案版本。RCS 可以在 “The RCS MINI-HOWTO” ( <http://sunsite.unc.edu/mdw/HOWTO/mini/RCS.html>) 中找到.

第二種方法 ,使用知道如何處理版本編號的編輯器：emacs 和 jed 都有這個功能。 在 emacs, 在 .emacs 那加入下列幾行:

```
(setq version-control t)
(setq kept-new-versions 15) ;;; or any other value
(setq kept-old-versions 15)
(setq backup-by-copying-when-linked t)
(setq backup-by-copying-when-mismatch t)
```

如果使用 jed, 請確定您的版本在 0.98.7 以上; 她的更新檔 ,可以使其支援數字版本備份。 相關文件可以在 <http://ibogeo.df.unibo.it/guido/slang/backups.sl> 中找到.

## 10.3 檔案館（Archiving）: tar & gzip

在 UNIX 裡 ,有一些常用到的應用程式來集中管理及壓縮檔案。 tar 可以用來包封檔案。欲包封檔案可用下列指令。

```
$ tar -cvf <archive_name.tar> <file> [file...]
```

解開包封檔案可用：

```
$ tar -xpvf <archive_name.tar> [file...]
```

列書包封檔案中所有的檔案：

```
$ tar -tf <archive_name.tar> | less
```

可以使用 compress 或是 gzip 來壓縮檔按以節省硬碟空間:

```
$ compress <file>
$ gzip <file>
```

如果使用 compress 來壓縮檔案 ,她為幫您產生副檔名為 .Z 的壓縮檔 ,而 gzip 則是以 .gz 為副檔名。這些應用程式並不能用來包封檔案 ,但是可以用來壓縮檔案。解壓縮可用：

```
$ compress -d <file.Z>
$ gzip -d <file.gz>
```

詳細情形 ，請 RMP.

此外 ，unarj, zip 和 unzip 些工具也可以利用。 檔案的副檔名如果是 .tar.gz 或是 .tgz (表示這是一個封裝檔案 tar, 然後 ，使用 gzip) 壓縮軟體再壓縮而成的。在 UNIX 的世界 ，這種格式常常會會見到。 以下是示範如何察看 .tar.gz 檔內的內容。

```
$ tar -ztf <file.tar.gz> | less
```

以下是示範如何解開 .tar.gz 檔:

```
$ tar -zxf <file.tar.gz>
```

## 11 應用

UNIX 的主要概念是希望透過管線(piping)和重導(redirection)的觀念將所有簡單的指令串連起來而可以足夠應付真實生活上所有複雜的工作。 讓我們來看看下面的例子 ，我將只解釋比較複雜的例子；其他簡單的例子 ，請利用上面我所介紹的觀念和說明文件(man pages) ，相信您一定可以很快的進入情況。

問題: 如果只使用 ls 這個指令來顯示檔案 ，如果檔案太多的話 ，常常就一飛而逝 ，還來不及看完 ，結果就被捲上去了：

解決方案:

```
$ ls | less
```

問題: 我有一個檔案 ，裡面包含了許多文字 ，我想將之反向排序後列印出來：

解決方案:

```
$ cat myfile.txt | sort -r | lpr
```

問題: 我的資料檔內有許多重複的資料 ，我要如何刪除重複的資料呢？

解決方案:

```
$ sort datafile.dat | uniq > newfile.dat
```

問題: 我有一個叫做 'mypaper.txt' 或是 'mypaper.tex' 的檔案 ，或是類似這樣的檔案名稱 ，但是我忘記我將她們存放在什麼地方了 ，我要如何找到她們呢？

解決方案:

```
$ find ~ -name "mypaper*"
```

說明: find 是一個非常有用的指令 ，她可以列出樹狀目錄下的所有檔案 ，(在本例中 ，我們是從 ~ 這裡開始尋找 ，也就是 $HOME 的目錄)。 她的輸出結果可以透過許多條件設定 ，而達到許多不一樣的需求。例如 -name。

問題: 我想找一個檔案內的某一個字 ，例如 'entropy' ，我要如何作呢？有沒有類似像 SEARCH 這種指令呢?

解決方案: 有的, 試試看 grep 這個指令：

```
$ grep -l 'entropy' *
```

說明：*表示所有的檔案。

題: 在某個地方，的某個檔案的內容有 'entropy' 這個字, 我想知道是哪一個檔案，且放在哪裡，在 VMS 上我可以使用 `search entropy [...]*.*;*`, 但是 `grep` 這個指令不可以收尋子目錄，現在我要如何解決呢？

解決方案:

```
$ find . -exec grep -l "entropy" {} \; 2> /dev/null
```

說明: `find .` 輸出從目前這個目錄下所有的檔案（包含這個目錄下所有子目錄），`-exec grep -l "entropy"` 是對每一筆由 find 傳出來的檔案重複執行執行(represented by {}), \ 為結束這個指令。如果您覺得的些東西太可怕了，沒錯，不過您可以試著寫寫下面的 Script。

---

```
#!/bin/sh
# rgrep: recursive grep
# 遞迴式擷取

if [ $# != 3 ]
then
  echo "Usage: rgrep --switches 'pattern' 'directory'"
  exit 1
fi
find $3 -name "*" -exec grep $1 $2 {} \; 2> /dev/null
```

---

說明: `grep` 就像 `search`, 結合了 `find`，我們得到了兩者的精華。

問題: 我有一個資料檔案，這個檔案共有兩列檔頭(header lines)（也就是說，第三筆資料才是我真正的資料），如果我想要擷取每筆資料的第二和第五欄的資料，我需要寫一個 Fortran 的程式嗎？

解決方案: 不需要。來看看這個具有殺傷力的指令！

```
$ awk 'NL > 2 {print $2, "\t", $5}' datafile.dat > newfile.dat
```

說明: `awk` 這個指令實際上可以說是一種程式語言：上面的意思是說：在 `datafile.dat` 這個檔案裡，從檔案的第三行開始的每一行，印出每一行的第二和第五欄位，以 tab 為分隔符號。學會了 `awk` 一定會讓您事半功倍的。

問題: 我從 FTP 站下載了 `ls-lR.gz` 這個檔案，想要察看檔案內容。對每一個子目錄而言，她有一行寫說 "total xxxx", 其中 xxxx 表示檔案大小（kbytes）我想知道所有 xxxx 值的總和，我要如何作呢？

解決方案:

```
$ zcat ls-lR.gz | awk ' $1 == "total" { i += $2 } END {print i}'
```

說明: `zcat` 可以列出 `.gz` 的檔案內容。然後將 zcat 得到的結果轉送(pipe)到 `awk`。看到了吧！了解 awk 真的對您有很大的幫助喔。RMP！

問題: 我已經寫了一隻 `myprog` 的 Fortran 的程式 ，可以計算由命令列傳進來的檔案的資料 ，可是 ，我有很多個資料檔需要輸入到這隻程式 ，每一個檔案將會有一個結果檔案。但是每次都要輸入檔名實在很煩人 ，在 VMS 上 ，我需要寫一個囉唆的命令檔(command file)才可以解決 ，那麼在 Linux 上呢？

解決方案: 只要一個小小的 Script 就可以解決了。修改您的程式 ，讓您的程式可以預定讀入 '`mydata.dat`' 這個檔案 ，將結果輸出到螢幕(stdout), 然後編輯下面的 Script：

```
#!/bin/sh
# myprog.sh: run the same command on many different files
# usage: myprog.sh *.dat
for file in $*  # for all parameters (e.g. *.dat)
do
  # append the file name to result.dat
  echo -n "${file}:    " >> results.dat
  # copy current argument to mydata.dat, run myprog
  # and append the output to results.dat
  cp ${file} mydata.dat ; myprog >> results.dat
done
```

問題: 我希望將所有檔案內的 'geology' 字替換成 'geophysics' ，我需要手動編輯嗎？！

解決方案: 不需要。 下面的 Shell Script 可以幫您辦到：

```
#!/bin/sh
# replace $1 with $2 in $*
# usage: replace "old-pattern" "new-pattern" file [file...]
OLD=$1          # first parameter of the script
NEW=$2          # second parameter
shift ; shift   # discard the first 2 parameters: the next are the file names
for file in $*  # for all files given as parameters
do
# replace every occurrence of OLD with NEW, save on a temporary file
  sed "s/$OLD/$NEW/g" ${file} > ${file}.new
# rename the temporary file as the original file
  /bin/mv ${file}.new ${file}
done
```

問題: 我有一些檔案 ，我不知道她們的檔案長度 ，我必須移除這些檔案內的倒數第二和倒數第三行 ，需要手動嗎？

解決方案: 當然不需要 ，還是使用 Shell Script:

```
#!/bin/sh
# prune.sh: removes n-1th and n-2th lines from files
# usage: prune.sh file [file...]
for file in $*   # for every parameter
do
  LINES=`wc -l $file | awk '{print $1}'`  # 計算總共有幾行
```

```
  LINES=`expr $LINES - 3`            # LINES = LINES - 3
  head -n $LINES $file > $file.new   # 輸出前 LINES 行
  tail -n 1 $file >> $file.new       # 再將最後一行附加到檔案最後
done
```

希望以上這些實例增加您不少的食慾...

# 12 不可少的小偏方

- **Command completion**: 當您在命令列欲輸入某一指令的時候，您不需要完全輸入這個指令的所有名稱，輸入前幾個字元後，再按 <TAB>，系統會幫您完成輸入所有的指令名稱; 例如: 您必須輸入 `less this_is_a_long_name`; 現在您只要輸入 `less thi<TAB>` 系統就會幫您輸入剩下的部份，不需要再書如全部的檔名了。(如果您有很多檔案都是以這些字元作開頭，當然系統就會顯示出所有以這幾個字元開頭的檔案，但是，您可以輸入足夠的字元，就可以得到您需要的唯一檔案。)
- **Back-scrolling**: 同時按 SHIFT–PAG UP 可以讓您的螢幕往回捲到以前顯示過的，但是可以往回捲幾頁，則需要視您的顯示卡的記憶體而定了。
- **Resetting the screen**: 不果您不小心用 `more` 或 `cat` 開啓了一個二進位檔(binary file), 那您的螢幕可能為因此而亂掉。這時候您可以使用 `reset` 或是使用 `echo CTRL-V ESC c RETURN` 可以讓您的螢幕恢復正常。
- **Pasting text**: 在 X 下，以滑鼠在 `xterm` 視窗拖曳出一個區域，然後按滑鼠中鍵（如果您的滑鼠沒有中鍵，可以同時按滑鼠的左右鍵模擬中鍵）就可以貼上您複製的文字。
- **Using the mouse**: 在終端機模式下，您可能需要尋為您的系統管理員有沒有安裝 `gpm` 這隻程式, 如果有，您就可以用滑鼠拖曳出一個區域，然後按滑鼠的左鍵，就可以貼上您剛剛拖曳出來的文字區域。（不同的虛擬終端機(VC)可能會有不一樣的操作方式）。

# 13 在 Linux 上讀取 VMS 的磁帶

(以下這段是由 Mike Miller 先生所著)

## 13.1 簡介

有時候您可能需要讀由 VMS 做出來的磁帶（或是為了讓 VMS 的機器和 *nix 系統可以讀取的磁帶），大致上，對 DECFILES11A 磁帶並不會太難。

或許您曾讀過 Linux 的 mini-HOWTO, 但是，我相信，我在這裡提及的一些觀念也是和任何一種 *nix 系統。我曾經在 Linux, HP, Sun and DEC *nix 等系統中試過。比較不一樣的地方是，每種作業系統對裝置名稱（device names）可能會有所不同，而 mt 在指明裝置名稱(device name)的時候，其選項(options)也可能因作業系統而異。例如：在 Linux 上是 mt -f，但是在 HPUX 9 上卻是 mt -t。

注意 - 我只以 Exabyte 8mm SCSI tape drives 實作過，如果您已經讀過其他的格式，請讓我知道，我將會把她增加到這份 Note 中。

## 13.2 一些基本常識

當讀取 VMS 由 “copy” 指令作成的磁帶時 ，（或是看起來像是由 copy 指令作成的磁帶） ，您必須要知道 ，每個實際的檔案將會分成三的檔案存放在磁帶上 ，分別是“檔頭(header)”, “資料(data)”, 和“檔尾(trailer)”。 其中檔頭(header) 和 檔尾(trailer)存放著檔案的訊息 ，和在 VMS 上的位置。而資料(data)當然就是存放原來的資料啦！ 這些檔案都可以由 dd 這個指令從磁帶(tape)上萃取出來。 由 mt 指令 ，可以將磁帶轉到所需要的位置。

範例：我有一個連續存放著檔案的磁帶。最前面兩個在 VMS 上的檔名為 ce66-2.evt 和 ce66-3.evt 磁帶的標籤(label)是 c66a2。

如果我執行下列指令：

```
> dd if=$TAPE bs=16k of=header1
> dd if=$TAPE bs=16k of=data1
> dd if=$TAPE bs=16k of=trailer1
> dd if=$TAPE bs=16k of=header2
> dd if=$TAPE bs=16k of=data2
> dd if=$TAPE bs=16k of=trailer2
```

我 dd 了六個檔案：header1, data1, trailer1, header2, data2 和 trailer2. 使用語法如下：if="input file", bs="block size" 和 of="output file". TAPE 這個便是須視您的機器上的裝置名稱而定 ，例如：如果您在 Linux 上使用 SCSI 的磁帶裝置 ，您的裝置名稱可能是 /dev/nts0。

如果您想要讀第二個檔案 ，不是第一個。您不在意檔頭(header) ，而且您想要使用原來的檔名 ，您可以這麼作：

```
> mt -f $TAPE fsf 4
> dd if=$TAPE bs=16k of=ce66-2.evt
> mt -f $TAPE fsf 1
```

注意 “4” - 跳過第一個檔案所附屬的三個檔案 ，所以下一比（第二筆）的檔頭將會是從第四筆開始。 第二個 mt 為跳過第二筆檔案的檔尾(tailer)並且指到下一筆檔案(第三筆資料的檔頭)。 當然 ，您也可以使用 mt 使磁帶快轉(bsf), 倒轉(rewind) ，或是 磁頭歸位(offline, rewoffl).

## 13.3 更多細節

檔頭(header) 和 檔尾(trailer) 包含了給 VMS 儲存檔案的一些訊息 ，例如區塊大小(block size)。 同時她們也包含了檔案名稱 ，方便您建立 Script 檔來自動讀取檔案或是搜尋特定的檔案。 然而 ，在磁帶上的第一個檔頭檔(header file) ，和其他磁帶上的檔頭檔有些許的差異。

對於在磁帶上的第一個檔案(假設為 header1) ，最前面的四個字元將會是 "VOL1" ，其後伴隨著版本名稱(volume name)。 本例中 ，header1 的最初內容應該是 "VOL1C66A2"。隨後是一個字串"HDR1" ，表示這是一個檔頭檔(header file)。 然後才是檔案名稱(file name)。本例中 ，她會是 "HDR1CE66-2.EVT"。下一個欄位又是版本名稱(volume name)。

對於其他的不是磁帶上第一個檔頭檔的檔頭檔 ，最初的 VOL1 欄位將不會存在。 其他的欄位就和前面所介紹的第一個檔頭檔的結構一模一樣。另一個比較有用的欄位是第七個欄位。 她是以 "DECFILES11A" 為結尾。她必須存在 ，用還確認 DEC Files11A standard.

```
field     initial header         subsequent headers
=====     ============           =================
  1       VOL1 + volume name     HDR1 + file name
  2       3HDR1 + file name      volume name
  3       volume name

  6                              ...DECFILES11A
  7      ...DECFILES11A
```

其他更詳細的檔頭和檔尾的結構，請參考 DEC FILES11A 的說明文件。 (on the orange/grey wall - ask your local VMS folks :-).

## 13.4 區塊大小的建議(Block Size)

在例子中，我使用的區塊大小是 16K。在 *nix 系統上，磁碟上的檔案並沒有和區塊大小有關連，然而，在 VMS 上，每一個檔案都有一個明確的區塊大小(block size)。也就是說，在 Linux 端，區塊大小並沒多太大的關係，除非當不容易從磁帶中讀出檔案的時候。如果您找不到區塊大小(block size)，您可以試著在磁碟裝置上，用 'mt -f $TAPE setblk 0' 來設定硬體的區塊大小。 用 setblk 選項的萃取的形式可能會隨著 mt 的版本不同而有所不一樣，您的磁帶裝置的硬體界面，還有您喜愛的 *nix 系統也想相當的關係。

(感謝 Wojtek Skulski ( <skulski@nsrlc6.nsrl.rochester.edu>) 提出 setblk 相關部份.)

# 14 總結

## 14.1 版權



## 14.2 後序

這份作品是我在 Settore di Geofisica of the Universita' di Bologna (Italy) 的實作經驗。 其 VAX 4000 也已經被以 Linux 為作業系統的奔騰(pentium)的個人電腦所代替了。 我大部分的同事以前都是 VMS 的愛用者，可是現在她們都已經轉換到 Linux 上了。

"From VMS to Linux HOWTO"（從 VMS 到 Linux）這份文件主要是由 Guido Gonzato 所撰寫的, <guido''at''ibogeo.df.unibo.it>, Mike Miller 先生, <miller5@uiuc.edu>, 他貢獻如何從 Linux 上讀

取 VMS 的備份磁帶這一部份文章。 十分感謝我的同事和朋友們 ，他們幫我解釋了大部分 VMS 使用者的需要和習慣 ，尤其感謝 Dr. Warner Marzocchi。

請幫我改善這份 HOWTO 文件。因為我不是 VMS 專家 ，而且也不可能是 ，所以十分歡迎您的建議 ，和錯誤回報。

歡迎您進入 Linux 的殿堂,

Guido =8-)

## 14.3 譯者序

在翻譯這篇文章的時候 ，我邊翻譯邊想 ，台灣現在還有人在使用 VMS 嗎？ 甚至還不是很多人知道 VMS呢！當我翻譯完後發現 ，其實這篇文章不只適合 VMS 的使用者 ，任何想要進入 Linux 世界的人都可以參考看看 ，因為 ，這篇文章的觀念還不算太難 ，讀完後 ，大致上可以了解 Linux 的基本操作。(甚是您可以把他想像成從 dos 到 Linux）。 但是 ，說真的 ，光憑這篇文章還是不夠的 ，我想 ，如果 Linux 的路要走得平順一點的話 ，買一本好書絕對是必要的。

或許 ，有些地方翻的不是很貼切 ，如果有好的建議 ，或是有任何地方的誤謬 ，請不吝賜教！Linux 的世界 ，是一個隨時隨地都在學習的世界！

Linux 愛好者,

峰舟 email: <sonnystone@kimo.com.tw>

Dec 27, 2000 于 MI。