

Linux 程式設計

Department of Computer Science and Information Engineering
Chaoyang University of Technology
Taichung, Taiwan, Republic of China

Instructor: De-Yu Wang (王德譽)

E-mail: dywang@csie.cyut.edu.tw

Homepage: <http://dywang.csie.cyut.edu.tw>

Phone: (04)23323000 ext 4538

Office: E738



Wednesday 8th June, 2022

-
- 教材宗旨：提供同學上網預習、複習及自我測驗。
 - 學習目標：
 1. 了解 Bash Shell 的功能及使用環境，進而學會撰寫 Shell Scripts。
 2. 了解 Linux 軟體開發與發行，進而學會套件之管理。
 3. 啟發學生學習 Linux 程式設計之興趣。
 4. 加強學生撰寫、管理及發行 Linux 套件之能力。
 5. 培養學生對 Linux 套件安裝與使用之除錯能力。
 6. 培養學生撰寫 Linux GUI 程式能力。
 - 先修科目或先備能力：具備 Linux 系統之基本知識。
 - Instructor: De-Yu Wang
 1. Email: dywang@csie.cyut.edu.tw
 2. Homepage: <http://dywang.csie.cyut.edu.tw>
 3. 本文件 pdf 檔下載
 4. Phone: (04)23323000 ext 4538
 5. Office: E738
 - 參考資料
 1. Neil Matthew Richard Stones 著，江俊龍譯，“Linux 程式設計教學手冊（第三版）(Beginning Linux Programming (Third Edition))”，碁峰資訊股份有限公司。
 2. vim 官方網站亦稱 vim 為程式開發工具。
 3. 大家來學 VIM（一個歷久彌新的編輯器）
 4. Bash shell
 5. 中央研究院計算中心 RCS 簡介網頁
 6. C++ GUI Programming with Qt 3 by Jasmin Blanchette and Mark Summerfield, Published by Prentice Hall.
 7. Maximum RPM 網頁

Contents

1	vi 編輯器	1
1.1	vi 與 vim	1
1.2	vi 的使用	1
1.3	vim 的額外功能	4
1.4	vi 實機練習題	6
1.4.1	練習一	6
1.4.2	練習二	7
1.4.3	練習三	7
2	Shell 變數	9
2.1	前言	9
2.2	變數設定規則	10
2.3	環境變數 env	13
2.4	環境變數使用範例	14
2.5	自訂變數導出 export	16
2.6	實機練習題	17
3	Shell 變數使用	18
3.1	鍵盤讀取 read	18
3.2	變數宣告 declare	18
3.3	陣列	19
3.4	字典陣列	21
3.5	變數的變化與取代	22
3.6	變數的設定	25
3.7	實機練習題	26
4	資料導向與管線處理	28
4.1	前言	28
4.2	資料流重導向	29
4.3	連續命令	31

4.4	管線命令 pipe	32
4.5	tee 雙向重導向	34
4.6	實機練習題	35
5	正規表示法	37
5.1	前言	37
5.2	基礎正規表示法	37
5.3	Windows 斷行符號	40
5.4	正規表示法與萬用字元	42
5.5	延伸正規表示法	42
5.6	基礎與延伸比較一	44
5.7	基礎與延伸比較二	45
5.8	實機練習題	46
6	sed 工具	49
6.1	sed 命令簡介	49
6.2	sed 範例一	51
6.3	sed 範例二	52
6.4	sed 範例三	52
6.5	sed 範例四	53
6.6	sed 與正規表示法	55
6.7	實機練習題	56
7	awk 工具	59
7.1	awk 命令簡介	59
7.2	awk 範例一	61
7.3	awk 範例二	63
7.4	awk 範例三	65
7.5	awk 與正規表示法	66
7.6	實機練習題	67
8	Shell Scripts	70
8.1	前言	70
8.2	第一支腳本	71
8.3	Shell 是一種程式語言	73
8.4	shell script 練習	74
8.5	set 命令	75
8.6	set 基本用法	77

8.7	追蹤與除錯	79
8.8	腳本預設參數	80
8.9	實機練習題	81
9	條件判斷	83
9.1	test 命令	83
9.2	test 命令使用	85
9.3	判斷符號 []	86
9.4	判斷符號 [[]]	87
9.5	判斷符號 (())	88
9.6	if 條件判斷式	89
9.7	case 條件判斷式	91
9.8	實機練習題	93
10	迴圈	95
10.1	for 迴圈	95
10.2	while 迴圈	97
10.3	until 迴圈	98
10.4	function 功能	100
10.5	實機練習題	102
11	gcc 編譯	105
11.1	為何要編譯	105
11.2	編譯步驟	106
11.3	編譯器 gcc	107
11.4	多檔編譯	109
11.5	多檔重編譯	110
11.6	實機練習題	111
12	make 與 makefile	112
12.1	前言	112
12.2	Make 命令	113
12.3	Makefile 語法	114
12.4	Makefile 相依性項目	116
12.5	多重目標項目	117
12.6	Makefile 內建法則	118
12.7	Makefile 自訂法則	119
12.8	實機練習題	120

13 Makefile 變數	123
13.1 前言	123
13.2 makefile 變數實例一	124
13.3 makefile 常用變數	125
13.4 makefile 變數實例二	126
13.5 實機練習題	127
14 使用者手冊	129
14.1 撰寫手冊	129
14.2 groff 處理手冊	132
14.3 man 查詢手冊	134
14.4 實機練習題	134
15 Makefile 函式庫	136
15.1 函式庫	136
15.2 ldd 函式庫工具	136
15.3 ldconfig 函式庫工具	137
15.4 建立函式庫工具	140
15.5 管理函式庫實例	142
15.6 實機練習題	144
16 *RCS 版本控制系統	146
16.1 版本控制	146
16.2 RCS 基本功能	149
16.3 識別關鍵字串	166
16.4 標記符號	174
16.5 版本比較與整合	179
16.6 存取名單	183
17* 使用 QT 設計 KDE 視窗程式	188
17.1 KDE 和 QT 介紹	188
17.2 Qt 開發環境建立	190
17.3 gtk+ 開發環境建立	192
17.4 Signals 和 Slots	194
17.5 QT Widget	200
17.6 對話窗 dialog	213
17.7 選單和工具列	218

18*Tarball 套件發行	222
18.1 套件發行	222
18.2 Tarballs 簡介	223
18.3 KPlayer Tarball 實例	233
19*RPM 與 SRPM 套件發行	241
19.1 RPM 與 SRPM 套件	241
19.2 建立 RPM 套件	246
19.3 KPlayer RPM 實例	255
20* 套件修補、檢驗與管理	259
20.1 patch 程式	259
20.2 檢驗軟體正確性	269
20.3 RPM 套件管理程式	274

Chapter 1

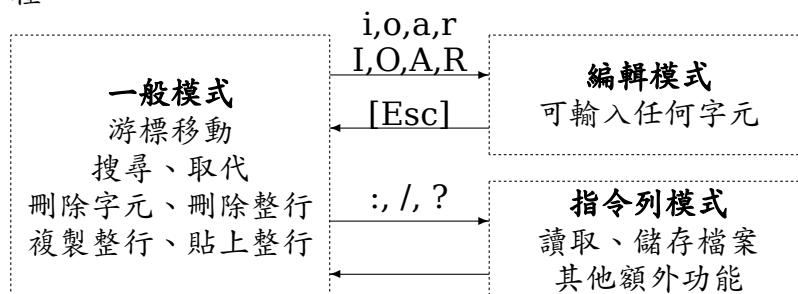
vi 編輯器

1.1 vi 與 vim

1. Linux 的系統中使用文字編輯器來編輯您的 Linux 參數設定檔。
2. Linux 與 Unix 系統中的參數檔幾乎都是 ASCII 碼的『純文字』檔。
3. vi 為文書編輯器，vim 是 vi 的進階軟體。
4. vim 加入了支援正規表示法的搜尋架構、多檔編輯、區塊複製等功能。
5. vim 官方網站亦稱 vim 為程式開發工具。
6. 大家來學 VIM（一個歷久彌新的編輯器）

1.2 vi 的使用

1. vi 共分為三種模式，分別是『一般模式』、『編輯模式』與『指令列命令模式』三種。



2. 簡易範例

(a) 使用 vi 進入一般模式


```
1 [dywang@dywOffice tmp]$ vim test.txt
```

(b) 按下 `i` 進入編輯模式，開始編輯文字。

(c) 按下 `[Esc]` 按鈕回到一般模式。

(d) 在一般模式中按下 `:wq` 儲存後離開 `vi`。

3. 常用指令列表

一般模式	移動游標
<code>h</code> 或向左方向鍵	游標向左移動一個字元
<code>j</code> 或向下方向鍵	游標向下移動一個字元
<code>k</code> 或向上方向鍵	游標向上移動一個字元
<code>l</code> 或向右方向鍵	游標向右移動一個字元
<code>[Ctrl]+f</code>	螢幕『向前』移動一頁
<code>[Ctrl]+b</code>	螢幕『向後』移動一頁
<code>[Ctrl]+d</code>	螢幕『向前』移動半頁
<code>[Ctrl]+u</code>	螢幕『向後』移動半頁
<code>+</code>	游標移動到非空白字元的下一行
<code>-</code>	游標移動到非空白字元的上一行
<code>n<space></code>	按下數字後再按空白鍵，游標會向右移動這一行的 <code>n</code> 個字元。 例如 <code>20<space></code> 則游標會向後面移動 20 個字元距離。
<code>0</code>	這是數字『0』：移動到這一行的最前面字元處。
<code>\$</code>	移動到這一行的最後面字元處
<code>H</code>	游標移動到這個螢幕的最上方那一行
<code>M</code>	游標移動到這個螢幕的中央那一行
<code>L</code>	游標移動到這個螢幕的最下方那一行
<code>G</code>	游標移動到這個檔案的最後一行
<code>nG</code>	游標移動到這個檔案的第 <code>n</code> 行。 例如 <code>20G</code> 則會移動到檔案的第 20 行 (可配合 <code>:set nu</code>)
<code>n<Enter></code>	游標向下移動 <code>n</code> 行
一般模式	搜尋與取代
<code>/word</code>	向游標之後尋找一個字串名稱爲 <code>word</code> 的字串。
<code>?word</code>	向游標之前尋找一個字串名稱爲 <code>word</code> 的字串。
<code>n</code>	重複前一個搜尋。
<code>N</code>	反向進行前一個搜尋動作。
<code>:n1,n2s/word1/word2/g</code>	在第 <code>n1</code> 與 <code>n2</code> 行之間尋找 <code>word1</code> 這個字串，並將該字串取代爲 <code>word2</code> 。
<code>:1,\$s/word1/word2/g</code>	從第一行到最後一行尋找 <code>word1</code> 字串，並將該字串取代爲 <code>word2</code> 。

:1,\$s/word1/word2/gc	從第一行到最後一行尋找 word1 字串，並將該字串取代為 word2 且在取代前顯示提示字元給使用者確認是否需要取代。
一般模式	刪除、複製與貼上
x, X	x 為向後刪除一個字元，X 為向前刪除一個字元
nx	向後刪除 n 個字元
dd	刪除游標所在的那一整行
ndd	刪除游標所在的向下 n 行，例如 20dd 則是刪除 20 行
d1G	刪除游標所在到第一行的所有資料
dG	刪除游標所在到最後一行的所有資料
dw	刪除一個字 (delete word)。不適用於中文。
yy	複製游標所在的那一行 (常用)
nyy	複製游標所在的向下 n 行，例如 20yy 則是複製 20 行
y1G	複製游標所在行到第一行的所有資料
yG	複製游標所在行到最後一行的所有資料
p, P	p 為複製的資料在游標下一行貼上，P 則為貼在游標上一行。
J	將游標所在行與下一行的資料結合成同一行
u	復原前一個動作。
[Ctrl]+r	反復原前一個動作。
.	重做上一個動作。
進入編輯模式	
i, I	插入：i(I) 在目前的游標所在處 (所在行第一個非空白字元處) 插入輸入之文字；
a, A	增加：a(A) 由目前游標所在的下一個 (最後一個) 字元處開始輸入；
o, O	插入新的一行：o(O) 從游標所在的下 (上) 一行插入新的一行；
r, R	取代：r 會取代游標所在的那一個字元；R 會一直取代游標所在的文字，直到按下 ESC 為止；
Esc	退出編輯模式，回到一般模式中。
指令列命令模式	
:w	將編輯的資料寫入硬碟檔案中
:w!	若檔案屬性為『唯讀』時，強制寫入該檔案
:q	離開 vi
:q!	若曾修改過檔案，又不想儲存，使用! 為強制離開不儲存檔案。
:wq	儲存後離開，若為:wq! 則為強制儲存後離開
:x	與 wq 相同
:e!	將檔案還原到最原始的狀態。

ZZ	若檔案沒有更動，則不儲存離開，若檔案已經經過更動，則儲存後離開。
:w [filename]	將編輯的資料儲存成另一個檔案（類似另存新檔）
:r [filename]	在編輯的資料中，讀入另一個檔案的資料。亦即將『file-name』這個檔案內容加到游標所在行後面。
:set nu	顯示行號，設定之後，會在每一行的字首顯示該行的行號
:set nonu	與 set nu 相反，為取消行號。
:n1,n2 w [filename]	將 n1 到 n2 的內容儲存成 filename 這個檔案。
:! command	暫時離開 vi 到指令列模式下執行 command 的顯示結果。 例如 [:! ls /home]

4. vi 編輯中回復與暫存檔

- (a) 如果編輯檔案為 /home/csie/vitest.txt，則同時會有一個暫存檔 /home/csie/.vitest.txt.swp 產生，其為隱藏檔。
- (b) 結束 vi 編輯時暫存檔也會刪除。
- (c) 若不知因素導致 vi 程式中斷，則在下次編輯該檔案時會出現是否回復的訊息。

1.3 vim 的額外功能

1. 如果使用 vi，在畫面右下角有目前游標所在行列號碼，則 vi 已被 vim 取代了。

```

1 [dywang@dywOffice tmp]$ ll /bin/vi*
lrwxrwxrwx 1 root root 20 Oct 17 13:11 /bin/vi -> /etc/alternatives/vi*
3 lrwxrwxrwx 1 root root 21 Oct 17 13:11 /bin/vim -> /etc/alternatives/
  vim*
[dywang@dywOffice tmp]$ ll /etc/alternatives/vi*
5 lrwxrwxrwx 1 root root 21 Oct 17 13:11 /etc/alternatives/vi ->
  /usr/bin/vim-enhanced*
7 lrwxrwxrwx 1 root root 21 Oct 17 13:11 /etc/alternatives/vim ->
  /usr/bin/vim-enhanced*
```

2. vim 具有顏色顯示的功能，並且還支援許多的程式語法。

3. vim /etc/man.config 出現訊息說明：

```

"/etc/man.config" [readonly] 150L, 4900C          1,1
Top
```

- (a) 編輯檔案名稱爲 /etc/man.config；
- (b) 檔案爲唯讀檔；
- (c) 檔案共有 150 行，4900 字元；
- (d) 目前游標所在位置爲第一行，第一列；
- (e) 目前頁面在最前頁。

4. 區塊選擇

區塊選擇的按鍵意義	
v	字元選擇，會將游標經過的地方反白選擇。
V	行選擇，會將游標經過的反白選擇。
[Ctrl]+v	區塊選擇，可以用長方形的方式選擇資料。
y	將反白的地方複製起來。
d	將反白的地方刪除掉。

5. 多檔案編輯

- (a) vi 內使用:r filename 可將檔案 filename 的內容在游標處插入。
- (b) 可於 vim 後接多個檔案來同時開啓多個檔案，例如：vim filename1 filename2 filename3。其相關按鍵有：

多檔案編輯的按鍵	
:n	編輯下一個檔案。
:N	編輯上一個檔案。
:files	列出目前開啓的所有檔案。

6. 多視窗功能

- (a) 在指令列模式輸入:sp filename。
- (b) 如果省略 filename 則兩視窗爲同一檔案。

多視窗下的按鍵功能	
:sp	開啓同一檔案於新視窗。
:sp filename	開啓檔案 filename 於新視窗。
[Ctrl]+wj	游標移動到下方的視窗。按法爲：先按下 [Ctrl] 不放，再下 w 後放開所有的按鍵，然後再按下 j。
[Ctrl]+wk	游標移動到上方的視窗。按法爲同上。
[Ctrl]+wq	結束下方視窗，與 [Ctrl]+w 移動到下方視窗後，再按下:q 離開相同。

7. vim 環境設定

- (a) 個人動作記錄檔案：`~/.viminfo`。例如：編輯同一檔案時，游標會在上次退出時的位置。
- (b) 整體 vim 的設定值放在 `/etc/vimrc`。
- (c) 若要更改 vim 設定，建議自行建立 `~/.vimrc`。

	vim 的環境設定參數
<code>:set nu</code>	設定行號。
<code>:set nonu</code>	取消定行號。
<code>:set hlsearch</code>	將搜尋的字串反白。
<code>:set autoindent</code>	自動縮排。
<code>:set noautoindent</code>	不自動縮排。
<code>:set backup</code>	自動儲存備份。備份檔名為 <code>filename~</code> 。
<code>:set rule</code>	顯示右下角的狀態說明。
<code>:set showmode</code>	顯示 <code>-insert-</code> 等字眼在左下角的狀態列。
<code>:set backspace=(012)</code>	2 利用 <code>backspace</code> 例退鍵除任意字元； 0 或 1 僅可刪除剛剛輸入的字元。
<code>:set all</code>	顯示目前所有的環境參數設定值。
<code>:syntax (off on)</code>	是否依據程式相關語法顯示不同顏色。

1.4 vi 實機練習題

1.4.1 練習一

1. 在自己的家目錄建立一個新的目錄 `zzz`。
2. 進入目錄 `zzz`。
3. 下載檔案 `viex1.txt`
4. 將 `viex1.txt` 重新命名為 `vi1.txt`
5. 使用 `vi` 或 `vim` 編輯 `vi1.txt`，執行以下動作：
 - (a) 先到第 `n` 行，`n` 為學號除 37 取餘數 +10。
 - (b) 複製 3 行。
 - (c) 再到第 20 行，往下貼上。
 - (d) 再到最後一行，往下產生一行空白行方式進入“插入”模式進行編輯。
 - (e) 加入文字 `'ABCDEabcde'`。
 - (f) 換行再加入 `'1234567890'`。

- (g) 搜尋字串"MANPATH"，並將其取代為"manpath"。
- (h) 存檔後退出。

1.4.2 練習二

1. 在自己的家目錄建立一個新的目錄 zzz。
2. 進入目錄 zzz。
3. 下載檔案 viex2.txt
4. 將 viex2.txt 重新命名為 vi2.txt
5. 使用 vi 或 vim 編輯 vi2.txt，執行以下動作：
 - (a) 先到第 n 行，n 為學號除 27 取餘數 +8。
 - (b) 刪除 3 行。
 - (c) 再到第 30 行，往上貼上。
 - (d) 再到第一行，往上產生一行空白行方式進入"插入"模式進行編輯。
 - (e) 加入文字'ABCDEabcde'。
 - (f) 換行再加入'1234567890'。
 - (g) 從第 30 行至最後一行搜尋字串「MANPATH」，並將其取代為「manpath」。
 - (h) 存檔後退出。

1.4.3 練習三

1. 在自己的家目錄建立一個新的目錄 zzz。
2. 進入目錄 zzz。
3. 下載檔案 viex3.txt
4. 將 viex3.txt 重新命名為 vi3.txt
5. 使用 vi 或 vim 編輯 vi3.txt，執行以下動作：
 - (a) 先到第 n 行，n 為學號除 44 取餘數 +11。
 - (b) 將下一行合併至這一行
 - (c) 再到第 20 行，刪除這一行到第一行的資料。
 - (d) 再到最後一行，往下貼上。

- (e) 再到第 13 行，在這行的最後加入文字'ABCDEabcde'。
- (f) 換行再加入'1234567890'。
- (g) 從第 30 行至第 55 行搜尋字串"man"，並將其取代為"MAN"。
- (h) 存檔後退出。

Chapter 2

Shell 變數

2.1 前言

1. 何謂變數：

- (a) 以一組文字或符號等，來取代一些設定或者是一串保留的資料。
- (b) 例如：每個帳號的郵件信箱預設是以 MAIL 這個變數來進行存取。

```
1 [root@dyw219 ~]# cat /etc/profile
3 LOGNAME=$USER
  MAIL="/var/spool/mail/$USER"
```

2. 取變數內容：變數前加符號 \$，變數用大括號括起來，變數沒接其他「英文字元或數字」時，大括號可以省略。下面變數 variable 沒設定，變數 PATH 為環境變數。

```
2 [root@dyw219 ~]# echo ${variable}
  [root@dyw219 ~]# echo $variable
4 [root@dyw219 ~]# echo ${PATH}
  /sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin:/usr/local/sbin
6 [root@dyw219 ~]# echo $PATH
  /sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin:/usr/local/sbin
```

3. 變數的用途之一：簡化路徑名稱。

```
1 [dywang@dyw219 ~]$ linuxpro=\
  /home/dywang/Documents/latex/cjk-tex/linuxprogram/
3 [dywang@dyw219 ~]$ cd linuxpro
```



```
bash: cd: linuxpro: No such file or directory
5 [dywang@dyw219 ~]$ cd $linuxpro
[dywang@dyw219 linuxprogram]$ pwd
7 /home/dywang/Documents/latex/cjk-tex/linuxprogram
```

2.2 變數設定規則

1. 變數與變數內容以等號『=』來連結；

```
1 [root@dyw219 ~]# name=dywang
```

2. 使用 \${} 取變數內容用。

```
1 [root@dyw219 ~]# echo ${name}
dywang
```

3. 取變數內容，變數後沒接數字或英文字母，大括號可以不用。

```
2 [root@dyw219 ~]# echo $name
dywang
```

4. 等號兩邊不能直接接空白字元；

```
2 [root@dywmac ~]# name = dywang
-bash: name: command not found
```

5. 變數名稱只能是英文字母與數字，但是數字不能是開頭字元；

```
2 [root@dyw219 ~]# 12name=dywang
-bash: 12name=dywang: command not found
```

6. 若有空白字元可以使用雙引號『"』或單引號『'』來將變數內容結合起來。

(a) 雙引號內的特殊字元可以保有變數特性，

```
2 [root@dyw219 ~]# name=dywang; myname="variable name is $name"
  [root@dyw219 ~]# echo $myname
variable name is dywang
```

(b) 單引號內的特殊字元則僅為一般字元；

```
1 [root@dyw219 ~]# name=dywang; myname='variable name is $name'
  [root@dyw219 ~]# echo $myname
3 variable name is $name
```

7. 必要時需要以跳脫字元『\』來將特殊符號（如 Enter, \$, \, 空白字元, ' 等）變成一般符號。

(a) 變數 name 設定為 dywang's name，沒有加單引號或雙引號，則 dywang 及 s 之間的單引號沒有配對，出現 > 要繼續輸入，按 Ctrl+C 中斷輸入。

```
1 [root@dywmac ~]# name=dywang's name
  > -bash: unexpected EOF while looking for matching `''
3 -bash: syntax error: unexpected end of file
```

(b) 利用雙引號將包含單引號及空白的字串，設定給變數 name。

```
1 [root@dywmac ~]# name="dywang's name" ; echo $name
dywang's name
```

(c) 利用反斜線 \ 跳脫單引號與空白鍵也可以。

```
2 [root@dywmac ~]# name=dywang\'s\ name ; echo $name
dywang's name
```

8. 在一串指令中，還需要藉由其他的指令提供的資訊，指令先執行：

(a) 使用 quote 『`command`』，符號 ` 是鍵盤上方的數字鍵 1 左邊那個按鍵，而不是單引號。以指令 uname 查詢目前核心版本，並切換到此版本的模組目錄。

```
2 [root@dywmac ~]# uname -r
4 4.4.5-1.el6.elrepo.x86_64
6 [root@dywmac ~]# cd /lib/modules/`uname -r`
8 [root@dywmac 4.4.5-1.el6.elrepo.x86_64]# pwd
10 /lib/modules/4.4.5-1.el6.elrepo.x86_64
12 [root@dywmac 4.4.5-1.el6.elrepo.x86_64]# cd
14 [root@dywmac ~]#
```

(b) 使用 `$(command)` 一樣可以先執行 `command`，取得結果。

```
1 [root@dywmac ~]# cd /lib/modules/$(uname -r)
3 [root@dywmac 4.4.5-1.el6.elrepo.x86_64]# pwd
5 /lib/modules/4.4.5-1.el6.elrepo.x86_64
7 [root@dywmac 4.4.5-1.el6.elrepo.x86_64]# cd
9 [root@dywmac ~]#
```

9. 擴增變數內容

(a) 變數 `var` 後不是數字或英文字元，可以不加雙引號或不用大括號。

```
1 [root@kvm6 ~]# var='123456'
3 [root@kvm6 ~]# var=$var:7890 ; echo $var
123456:7890
```

(b) 變數 `var` 後是數字或英文字元，此例是 1，取變數時會取 `var1`，但 `var1` 沒設定，所以輸出只有:7890。

```
1 [root@kvm6 ~]# var=$var1:7890 ; echo $var
:7890
```

(c) 要避免變數與後面的字元混淆，必須用大括號括起變數。

```
2 [root@kvm6 ~]# var=${var}1:7890 ; echo $var
123456:7890
```

10. unset 取消變數。

```
2 [root@kvm6 ~]# var=1237890 ; echo $var
1237890
4 [root@kvm6 ~]# unset var; echo $var
```

2.3 環境變數 env

1. Linux 是一個多人多功的系統，每個用戶登入時讀取設定檔 `/etc/bashrc` 及 `/etc/profile` 取得工作的環境變數，登入後執行 `env` 可以列出環境變數。

```

1  [dywang@dywmac ~]$ env
ORBIT_SOCKETDIR=/tmp/orbit-dywang
3  HOSTNAME=dywmac
IMSETTINGS_INTEGRATE_DESKTOP=yes
5  TERM=xterm
SHELL=/bin/bash
7  HISTSIZE=1000
XDG_SESSION_COOKIE=ac14e52ecfb2492f78ca6a6900000012
-1552852991.392630-81397834
9  GTK_RC_FILES=/etc/gtk/gtkrc:/home/dywang/.gtkrc-1.2-gnome2
WINDOWID=69206020
11 QTDIR=/usr/lib64/qt-3.3
QTINC=/usr/lib64/qt-3.3/include
13 IMSETTINGS_MODULE=gcin
USER=dywang
15 LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd
=40;33;...
GNOME_KEYRING_SOCKET=/tmp/keyring-8bg08w/socket
17 SSH_AUTH_SOCK=/tmp/keyring-8bg08w/socket.ssh
USERNAME=dywang
19 SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/3513,unix/unix:/tmp/.ICE-unix
/3513
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin
21 MAIL=/var/spool/mail/dywang
DESKTOP_SESSION=gnome
23 QT_IM_MODULE=gcin
PWD=/home/dywang
25 XMODIFIERS=@im=gcin
GDM_KEYBOARD_LAYOUT=us
27 GNOME_KEYRING_PID=3503
LANG=en_US.UTF-8
29 GDM_LANG=en_US.UTF-8
GDMSESSION=gnome
31 SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HISTCONTROL=ignoredups
33 HOME=/home/dywang
SHLVL=2
35 GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=dywang
37 QTLIB=/usr/lib64/qt-3.3/lib
CVS_RSH=ssh
39 DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-CSMOPauJQO,guid=51
ed8f7a36376c036368676d0000002e
LESSOPEN=||/usr/bin/lesspipe.sh %s
41 WINDOWPATH=1
DISPLAY=:0.0
43 G_BROKEN_FILENAMES=1
COLORTERM=gnome-terminal

```

```
45 | XAUTHORITY=/var/run/gdm/auth-for-dywang-xnorl4/database  
   | _=/usr/bin/env
```

2. 除了系統預設的環境變數，用戶也可以在自己的家目錄中 `.bashrc` 及 `.bash_profile` 設定專屬的環境變數。變數設定在 `/etc/bashrc` 或 `/etc/profile` 兩個檔案 `ssh` 登入都會生效，但圖形界面開啓 `gnome-terminal` 不會讀取 `/etc/profile`，同樣的狀況，用戶設定在家目錄中的 `.bashrc` 或 `.bash_profile`，`ssh` 登入都生效，但 `gnome-terminal` 不會讀取 `.bash_profile`。

```
[dywang@dywmac ~]$ vim .bashrc  
2 [dywang@dywmac ~]$ tail -1 .bashrc  
myname=dywang  
4 [dywang@dywmac ~]$ vim .bash_profile  
[dywang@dywmac ~]$ tail -1 .bash_profile  
6 doc=Documents  
[dywang@dywmac ~]$ bash  
8 [dywang@dywmac ~]$ echo $myname  
dywang  
10 [dywang@dywmac ~]$ echo $doc  
  
12 [dywang@dywmac ~]$ exit  
exit  
14 [dywang@dywmac ~]$ ssh localhost  
Last login: Mon Mar 18 04:56:47 2019 from localhost  
16 [dywang@dywmac ~]$ echo $myname  
dywang  
18 [dywang@dywmac ~]$ echo $doc  
Documents
```

2.4 環境變數使用範例

1. `?`：『上個執行的指令所回傳的值』，成功執行該指令會回傳一個 0 值，如果執行過程發生錯誤，就會回傳『錯誤代碼』。例如：`ls` 列出存在的檔案 `/etc/issue`，成功列出，`echo $?` 印出回傳值是 0；但 `ls` 列出不存在的檔案 `/etc/dywang`，回傳值是 2。

```
1 [dywang@dyw219 ~]$ ls /etc/issue  
/etc/issue  
3 [dywang@dyw219 ~]$ echo $?  
0  
5 [dywang@dyw219 ~]$ ls /etc/dywang  
ls: cannot access /etc/dywang: No such file or directory  
7 [dywang@dyw219 ~]$ echo $?
```

2

2. 變數 RANDOM：echo \$RANDOM 系統就會主動的隨機取出一個介於 0~32767 的數值。例如輸出一個 0 到 9 的值。

```

[dywang@dyw219 ~]$ echo $((RANDOM*10/32767))
2 3
[dywang@dyw219 ~]$ echo $((RANDOM*10/32767))
4 9
[dywang@dyw219 ~]$ echo $((RANDOM*10/32767))
6 2

```

3. PS1：提示字元的設定

- (a) \d：代表日期，格式為 Weekday Month Date，例如"Mon Aug 1"
- (b) \H：完整的主機名稱。例如 dyw219.deyu.wang。
- (c) \h：僅取主機名稱的第一個名字。上例為 dyw219，.deyu.wang 被省略。
- (d) \t：顯示時間，為 24 小時格式，如：HH:MM:SS
- (e) \T：顯示時間，12 小時的時間格式。
- (f) \A：顯示時間，24 小時格式，HH:MM
- (g) \u：目前使用者的帳號名稱；
- (h) \v：BASH 的版本資訊；
- (i) \w：完整的工作目錄名稱。家目錄會以 ~ 取代；
- (j) \W：利用指令 basename 取得工作目錄名稱，所以僅會列出最後一個目錄名。
- (k) \#：下達的第幾個指令。
- (l) \\$：提示字元，如果是 root 時，提示字元為 #，否則就是 \$。

4. PS1 修改範例

```

[dywang@dyw219 ~]$ PS1='[\u@\h \w \A #\#]\$ '
2 [dywang@dyw219 ~ 18:15 #16]$ echo $PS1
  [\u@\h \w \A #\#]\$
4 [dywang@dyw219 ~ 18:15 #17]$ ls /etc/issue
  /etc/issue
6 [dywang@dyw219 ~ 18:15 #18]$ source ~/.bashrc

```

2.5 自訂變數導出 export

1. 取得 shell 後自訂的變數，只限目前的 shell 有效，其他 shell 或其子程序 shell 都無法使用。

```
[dywang@dywmac ~]$ myip=192.168.122.1
2 [dywang@dywmac ~]$ echo $myip
192.168.122.1
4 [dywang@dywmac ~]$ bash
[dywang@dywmac ~]$ echo $myip
6
[dywang@dywmac ~]$ exit
8 exit
```

2. 若要讓目前設定的變數可以讓子程序 shell 使用，必須使用 export 導出。export 不加參數列出所有導出 (export) 的變數。

```
[dywang@dywmac ~]$ export
2 declare -x CVS_RSH="ssh"
declare -x G_BROKEN_FILENAMES="1"
4 declare -x HISTCONTROL="ignoredups"
declare -x HISTSIZE="1000"
6 declare -x HOME="/home/dywang"
declare -x HOSTNAME="dywmac"
8 declare -x LANG="en_US.UTF-8"
declare -x LESSOPEN="||/usr/bin/lesspipe.sh %s"
10 declare -x LOGNAME="dywang"
declare -x LS_COLORS="rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do
=01;35:bd=40;33;01:..."
12 declare -x MAIL="/var/spool/mail/dywang"
declare -x OLDPWD
14 declare -x /usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin"
declare -x PWD="/home/dywang"
16 declare -x SELINUX_LEVEL_REQUESTED=""
declare -x SELINUX_ROLE_REQUESTED=""
18 declare -x SELINUX_USE_CURRENT_RANGE=""
declare -x SHELL="/bin/bash"
20 declare -x SHLVL="1"
declare -x SSH_ASKPASS="/usr/libexec/openssh/gnome-ssh-askpass"
22 declare -x SSH_CLIENT=":::1 50346 22"
declare -x SSH_CONNECTION=":::1 50346 :::1 22"
24 declare -x SSH_TTY="/dev/pts/11"
declare -x TERM="xterm"
26 declare -x USER="dywang"
declare -x XMODIFIERS="@im=gcin"
```

3. 設定 myip 變數並以 export 來導出變數，進入子程式後 myip 變數仍然有效。

```
1 [dywang@dywmac ~]$ export myip=192.168.122.1
[dywang@dywmac ~]$ export | grep myip
3 declare -x myip="192.168.122.1"
[dywang@dywmac ~]$ bash
5 [dywang@dywmac ~]$ echo $myip
192.168.122.1
7 [dywang@dywmac ~]$ exit
exit
```

2.6 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 zzz 目錄。
- (b) 切換工作目錄到 zzz，所有練習腳本皆存在此目錄。
- (c) 腳本必須可執行。

2. 練習題

- (a) 下載 vartest.sh，依據檔案內的說明修改變數，不要增加任何多餘的字元或空白，使 vartest.sh 可執行。
- (b) 修改 dywang 的 .bashrc 檔增加變數 doc，使登入後執行 cd \$doc 可以切換到工作目錄 /home/dywang/Documents。
- (c) 寫一腳本 varenv1.sh，利用變數 RANDOM 產生 0 或 1 隨機數字。
- (d) 寫一腳本 varenv2.sh，執行 ls /tmp/myreturn，並印出其回傳值。
- (e) 寫一腳本 varenv3.sh，執行 source varenv3.sh 命令列的提示會變成 [您的學號 下達的第幾個指令]\$ 。

Chapter 3

Shell 變數使用

3.1 鍵盤讀取 read

1. read 指令後接變數名稱，執行時會等待鍵盤輸入，輸入完成會將內容存在指定的變數。

```
[dywang@dywmac ~]$ cd zzz
2 [dywang@dywmac zzz]$ read var
123456
4 [dywang@dywmac zzz]$ echo $var
123456
```

2. read 加選 -p 可以接提示字元，提示使用者輸入資料。

```
1 [dywang@dywmac zzz]$ read -p 'keyin a number: ' var
keyin a number: 2100
3 [dywang@dywmac zzz]$ echo $var
2100
```

3. read 加選 -t 可以接等待秒數，使用者在這秒數內沒輸入，就會繼續程式。

```
1 [dywang@dywmac zzz]$ read -p 'keyin a number: ' -t 3 var
2 keyin a number: [dywang@dywmac zzz]$ echo $var
```

3.2 變數宣告 declare

1. shell 變數預設為字串，設定變數 sum=1+2+3 不會加總。

```
1 [dywang@dywmac zzz]$ sum=1+2+3
[dywang@dywmac zzz]$ echo $sum
3 1+2+3
```

2. `declare` 指令加選項 `-i` 可以宣告變數為整數，宣告 `sum` 是整數且 `sum=1+2+3` 就是計算 `sum` 為 6。

```
1 [dywang@dywmac zzz]$ declare -i sum=1+2+3
[dywang@dywmac zzz]$ echo $sum
3 6
```

3. 不用指令 `declare` 宣告 `sum` 為整數，可以用雙小括號做計算，再以 `$` 取出計算結果存到變數 `sum`。

```
1 [dywang@dywmac zzz]$ unset sum
[dywang@dywmac zzz]$ sum=$((1+2+3+4))
3 [dywang@dywmac zzz]$ echo $sum
10
```

3.3 陣列

1. 以小括號 `()` 指定元素給陣列變數 `arr`。

```
[dywang@dywmac zzz]$ unset arr
2 [dywang@dywmac zzz]$ arr=(A B C)
```

2. `${arr[@]}` 取出陣列所有元素。

```
[dywang@dywmac zzz]$ echo ${arr[@]}
2 A B C
```

3. `${#arr[@]}` 取出陣列 `arr` 的元素個數。

```
[dywang@dywmac zzz]$ echo ${#arr[@]}
2 3
```

4. 變更陣列 arr 第 3 個元素為 R

```
[dywang@dywmac zzz]$ arr=(A B C)
2 [dywang@dywmac zzz]$ arr[2]=R
  [dywang@dywmac zzz]$ echo ${arr[@]}
4 A B R
```

5. 陣列 arr 只有 3 個元素，設定第 4 個元素為 D，會增加一個元素。

```
[dywang@dywmac zzz]$ arr[3]=D
2 [dywang@dywmac zzz]$ echo ${arr[@]}
  A B R D
```

6. 現在陣列 arr 有 4 個元素，設定第 7 個元素為 Y，元素個數為 5，但第 5 及 6 個元素沒設定。

```
1 [dywang@dywmac zzz]$ arr[6]=Y
  [dywang@dywmac zzz]$ echo ${arr[@]}
3 A B R D Y
  [dywang@dywmac zzz]$ echo ${#arr[@]}
5 5
  [dywang@dywmac zzz]$ echo ${arr[4]}
7
  [dywang@dywmac zzz]$ echo ${arr[6]}
9 Y
```

7. 陣列後面累加一個元素

```
1 [dywang@dywIssd zzz]$ arr=(1 12 13 14)
  [dywang@dywIssd zzz]$ arr1=(${arr[@]} 15)
3 [dywang@dywIssd zzz]$ echo ${arr1[@]}
  1 12 13 14 15
```

8. 陣列前面累加一個元素

```
[dywang@dywIssd zzz]$ arr=(1 12 13 14)
2 [dywang@dywIssd zzz]$ arr1=(0 ${arr[@]})
  [dywang@dywIssd zzz]$ echo ${arr1[@]}
4 0 1 12 13 14
```

9. 從陣列 arr 第 N 個元素開始取 M 個元素

```
arr1=${arr[@]:N-1:M})
```

10. 例如：從陣列 `arr` 第 3 個元素取 5 個元素

```
1 [dywang@dywIssd zzz]$ arr=(1 12 13 14 15 16 17)
  [dywang@dywIssd zzz]$ arr1=${arr[@]:2:5})
3 [dywang@dywIssd zzz]$ echo ${arr1[@]}
  13 14 15 16 17
```

11. 例如：從陣列 `arr` 第 4 個元素取到最後一個元素 (M 沒設定表示取到最後一個)

```
2 [dywang@dywIssd zzz]$ arr=(1 12 13 14 15 16 17)
  [dywang@dywIssd zzz]$ arr1=${arr[@]:3})
  [dywang@dywIssd zzz]$ echo ${arr1[@]}
4 14 15 16 17
```

12. 例如：從陣列 `arr` 第 1 個元素取到倒數第二個元素

```
2 [dywang@dywIssd zzz]$ arr=(1 12 13 14 15 16 17)
  [dywang@dywIssd zzz]$ arr1=${arr[@]:0:${#arr[@]}-1})
  [dywang@dywIssd zzz]$ echo ${arr1[@]}
4 1 12 13 14 15 16
```

3.4 字典陣列

1. python 有 `dictionary` 使用非數字序列的陣列 `index`，shell 一樣可以使用這樣的陣列，使用前必須以選項 `-A` 宣告陣列。以下設定 `grades` 是字典陣列。

```
[dywang@dywmac ~]$ declare -A grades
```

2. 設定 `grades` 有四個元素，`index` 是姓名，元素值是成績。

```
1 [dywang@dywmac ~]$ grades=(["dywang"]='60' ["linda"]='80' ["peter"]='92'
  ["rita"]='87')
```

3. 取出 linda 的成績。

```
1 [dywang@dywmac ~]$ echo "${grades['linda']}"  
80
```

4. 取陣列 grades 所有的元素值。

```
2 [dywang@dywmac ~]$ echo "${grades[@]}"  
87 92 80 60
```

5. 取陣列 grades 所有的 keys。

```
2 [dywang@dywmac ~]$ echo "${!grades[@]}"  
rita peter linda dywang
```

6. 取陣列 grades 的元素個數。

```
2 [dywang@dywmac ~]$ echo "${#grades[@]}"  
4
```

7. 重新設定 linda 的成績。

```
2 [dywang@dywmac ~]$ grades['linda']=85  
[dywang@dywmac ~]$ echo "${grades['linda']}"  
85
```

3.5 變數的變化與取代

1. 刪除變數前後部分字串

(a) blkid 列出 /dev/vda2 包含 uuid 的字串訊息

```
1 [dywang@ip114 ~]$ sudo /sbin/blkid /dev/vda2  
/dev/vda2: LABEL="crt-boot" UUID="364e680a-fb91-4783-8238-  
f7c83af8a1e3"  
3 BLOCK_SIZE="1024" TYPE="ext4" PARTUUID="68f5d32f-4de5-43a1-9a67-  
c7f303e9eee8"
```

- (b) 從最前面開始比對，刪除符合的字元。一個井號# 開頭，表示取『最小的那一段』。

```

1 [dywang@ip114 ~]$ uuid=$(sudo /sbin/blkid /dev/vda2); echo $uuid
/dev/vda2: LABEL="crt-boot" UUID="364e680a-fb91-4783-8238-
    f7c83af8a1e3"
3 BLOCK_SIZE="1024" TYPE="ext4" PARTUUID="68f5d32f-4de5-43a1-9a67-
    c7f303e9eee8"
[dywang@ip114 ~]$ uuid=${uuid##*D=\\}; echo $uuid
5 364e680a-fb91-4783-8238-f7c83af8a1e3" BLOCK_SIZE="1024" TYPE="ext4"
PARTUUID="68f5d32f-4de5-43a1-9a67-c7f303e9eee8"

```

- (c) 從最前面開始比對，刪除符合的字元。兩個井號## 開頭，表示符合的最長的一段。

```

2 [dywang@ip114 ~]$ uuid=$(sudo /sbin/blkid /dev/vda2); echo $uuid
/dev/vda2: LABEL="crt-boot" UUID="364e680a-fb91-4783-8238-
    f7c83af8a1e3"
BLOCK_SIZE="1024" TYPE="ext4" PARTUUID="68f5d32f-4de5-43a1-9a67-
    c7f303e9eee8"
4 [dywang@ip114 ~]$ uuid=${uuid###*D=\\}; echo $uuid
68f5d32f-4de5-43a1-9a67-c7f303e9eee8"

```

- (d) 從後面開始比對，刪除符合的字元。一個百分比符號% 開頭，表示取『最小的那一段』。

```

1 [dywang@ip114 ~]$ uuid=$(sudo /sbin/blkid /dev/vda2); echo $uuid
/dev/vda2: LABEL="crt-boot" UUID="364e680a-fb91-4783-8238-
    f7c83af8a1e3"
3 BLOCK_SIZE="1024" TYPE="ext4" PARTUUID="68f5d32f-4de5-43a1-9a67-
    c7f303e9eee8"
[dywang@ip114 ~]$ uuid=${uuid%*=\\*}; echo $uuid
5 /dev/vda2: LABEL="crt-boot" UUID="364e680a-fb91-4783-8238-
    f7c83af8a1e3"
BLOCK_SIZE="1024" TYPE="ext4" PARTUUID

```

- (e) 從後面開始比對，刪除符合的字元。兩個百分比符號%% 開頭，表示取『最長的那一段』。

```

2 [dywang@ip114 ~]$ uuid=$(sudo /sbin/blkid /dev/vda2); echo $uuid
/dev/vda2: LABEL="crt-boot" UUID="364e680a-fb91-4783-8238-
    f7c83af8a1e3"
BLOCK_SIZE="1024" TYPE="ext4" PARTUUID="68f5d32f-4de5-43a1-9a67-
    c7f303e9eee8"
4 [dywang@ip114 ~]$ uuid=${uuid%%*=\\*}; echo $uuid
/dev/vda2: LABEL

```

2. 取代變數部分字串

(a) blkid 取得 /dev/vda2 的 UUID，存在變數 uuid。

```
1 [dywang@ip114 ~]$ uuid=$(sudo /sbin/blkid /dev/vda2); echo $uuid
/dev/vda2: LABEL="crt-boot" UUID="364e680a-fb91-4783-8238-
f7c83af8a1e3"
3 BLOCK_SIZE="1024" TYPE="ext4" PARTUUID="68f5d32f-4de5-43a1-9a67-
c7f303e9eee8"
```

(b) 一個斜線 / 將字串變數 uuid 中的等號"UUID=" 取代成冒號":"，只取代第一個。

```
1 [dywang@ip114 ~]$ uuid=$(sudo /sbin/blkid /dev/vda2); echo ${uuid/
UUID=:}
/dev/vda2: LABEL="crt-boot" : "364e680a-fb91-4783-8238-f7c83af8a1e3"
3 BLOCK_SIZE="1024" TYPE="ext4" PARTUUID="68f5d32f-4de5-43a1-9a67-
c7f303e9eee8"
```

(c) 二個斜線 / 將字串變數 uuid 中的等號"UUID=" 取代成冒號":"，取代所有。

```
1 [dywang@ip114 ~]$ uuid=$(sudo /sbin/blkid /dev/vda2); echo ${uuid//
UUID=:}
/dev/vda2: LABEL="crt-boot" : "364e680a-fb91-4783-8238-f7c83af8a1e3"
3 BLOCK_SIZE="1024" TYPE="ext4" PART: "68f5d32f-4de5-43a1-9a67-
c7f303e9eee8"
```

3. 例題：blkid 列出檔案系統/dev/vg_crt/swap 如下，使用變數變化請取出 uuid 編號，不含雙引號。

```
1 [dywang@ip114 ~]$ uuid=$(sudo /sbin/blkid /dev/vg_crt/swap); echo $uuid
/dev/vg_crt/swap: UUID="659acc86-0a0f-4983-bdf7-b3e660f1d1ed" TYPE="swap"
"
```

3.6 變數的設定

1. 變數設定方式

變數設定方式	str 沒有設定	str 為空字串	str 為非空字串
var=\${str-expr}	var=expr	var=	var=\$str
var=\${str:-expr}	var=expr	var=expr	var=\$str
var=\${str+expr}	var=	var=expr	var=expr
var=\${str:+expr}	var=	var=	var=expr
var=\${str=expr}	str=expr	str 不變	str 不變
	var=expr	var=	var=\$str
var=\${str:=expr}	str=expr	str=expr	str 不變
	var=expr	var=expr	var=\$str
var=\${str?expr}	expr 輸出至 stderr	var=	var=\$str
var=\${str:?expr}	expr 輸出至 stderr	expr 輸出至 stderr	var=\$str

2. 若 str 這個變數內容存在，則 var 設定為 str，否則 var 設定為 "newvar"。str 不存在時 var 為 newvar，str 存在時 var 等於 str 的內容。

```

1 [root@dyw219 ~]# unset str; var=${str:-newvar}
2 [root@dyw219 ~]# echo var="$var", str="$str"
var=newvar, str=
4 [root@dyw219 ~]# str="oldvar"; var=${str-newvar}
5 [root@dyw219 ~]# echo var="$var", str="$str"
6 var=oldvar, str=oldvar

```

3. 若 str 不存在，則 var 與 str 均設定為 newvar，否則僅 var 為 newvar。str 不存在時 var/str 均為 newvar，str 存在時 var 等於 str 的內容。

```

1 [root@dyw219 ~]# unset str; var=${str=newvar}
2 [root@dyw219 ~]# echo var="$var", str="$str"
var=newvar, str=newvar
4 [root@dyw219 ~]# str="oldvar"; var=${str=newvar}
5 [root@dyw219 ~]# echo var="$var", str="$str"
6 var=oldvar, str=oldvar

```

4. 若 str 這個變數存在，則 var 等於 str，否則輸出 "novar"。str 不存在時輸出錯誤訊息，str 存在時 var 等於 str 的內容。

```

1 [root@dyw219 ~]# unset str; var=${str?novar}
2 -bash: str: novar
3 [root@dyw219 ~]# str="oldvar"; var=${str?novar}
4 [root@dyw219 ~]# echo var="$var", str="$str"
var=oldvar, str=oldvar

```



```
var=oldvar, str=oldvar
```

5. 上面這三個案例都沒有提到當 `str` 有設定，且為空字串的情況，請練習。

3.7 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 `zzz` 目錄。
- (b) 切換工作目錄到 `zzz`，所有練習腳本皆存在此目錄。
- (c) 腳本必須可執行。

2. 練習題

- (a) 寫一腳本 `var1.sh`，使用 `read` 從鍵盤輸入讀入一數字，並將其乘 2 後輸出到螢幕。
- (b) 寫一腳本 `var2.sh`，建立一個 1 至 9 共 9 個元素的陣列，使用 `read` 從鍵盤輸入讀入一個 5 至 8 的數字 `N`，並列出陣列第 2 元素至將第 `N` 個元素。
- (c) 下載檔案 `gradelist.txt`
- (d) 寫一腳本 `arr1.sh`，將 `gradelist.txt` 所列的「名字」依序存成陣列 `name`，「成績」依序存成陣列 `grade`，並完成以下要求：
 - i. 輸出 `name` 的第 2 個元素。
 - ii. 輸出 `name` 的第 1 到第 3 個元素。
 - iii. 輸出 `grade` 的第 3 個元素。
 - iv. 輸出 `grade` 的第 2 到最後一個元素。
- (e) 寫一腳本 `arr2.sh`，將 `gradelist.txt` 所列的「名字」依序存成陣列 `name`，「成績」依序存成陣列 `grade`，並完成以下要求：
 - i. 輸出陣列 `name` 所有的元素值。
 - ii. 輸出陣列 `grade` 所有元素值。
 - iii. 輸出陣列 `grade` 的元素個數。
 - iv. 重新設定 `grade` 的第 3 個成績，增加 5 分，再次輸出第 3 個成績。
- (f) 寫一腳本 `dict1.sh`，將 `gradelist.txt` 所列的「名字成績」存在字典陣列 `grades`，並完成以下要求：
 - i. 輸出 `cora` 的成績。

- ii. 設定 `betty` 的成績為 88 分。
 - iii. 輸出陣列 `grades` 所有的元素值。
- (g) 寫一腳本 `dict2.sh`，將 `gradelist.txt` 所列的「名字成績」存在字典陣列 `grades`，並完成以下要求：
- i. 輸出陣列 `grades` 所有的 `keys`。
 - ii. 輸出陣列 `grades` 的元素個數。
 - iii. 重新設定 `cora` 的成績，增加 5 分，再次輸出 `cora` 的成績。
- (h) 寫一腳本 `var3.sh`，取得根目錄 `/` 的檔案系統的 UUID，不含雙引號及其他資訊。腳本使用 `sudo blkid` 命令取得 UUID。
- (i) 寫一腳本 `var3-1.sh`，使用 `ifconfig enXXX | grep 'inet '` 取得網卡 IP 資訊存放在變數 `ip`，其中 `enXXX` 為 HOST 對外網卡名稱，刪除變數 `ip` 前後不要的字元，取得您的 IP，並輸出「`myip=xxx.xxx.xxx.xxx`」。
- (j) 寫一腳本 `var4.sh`，利用變數設定方法設定以下變數，並將變數 `vb` 輸出到螢幕。
- i. 變數 `va:str` 沒有設定，則 `va` 設定為 111；`str` 為空字串，則設定為空字串；其餘設定為 `$str`。
 - ii. 變數 `vb`：`va` 沒有設定或空字串，則將 222 輸出到 `stderr`；其餘設定為 `$va`。
- (k) 寫一腳本 `var4-1.sh`，利用變數設定方法設定以下變數，並將變數 `va` 輸出到螢幕。變數 `va:str` 沒有設定或空字串時，變數 `va` 設定為空字串，`str` 有設定時 `va` 設定 444。

執行腳本前使用 `export` 導出變數 `str` 給腳本 `var4-1.sh`，即可測試。

Chapter 4

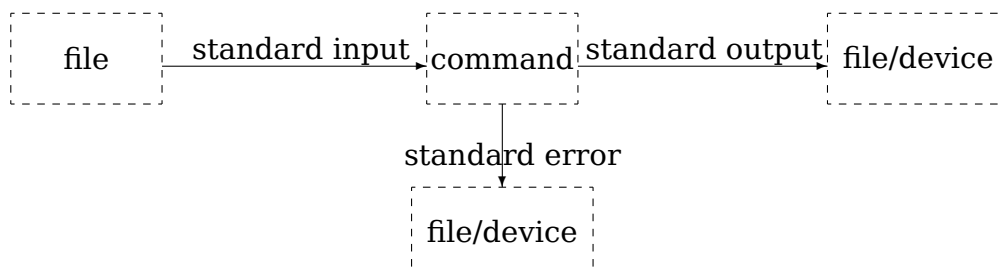
資料導向與管線處理

4.1 前言

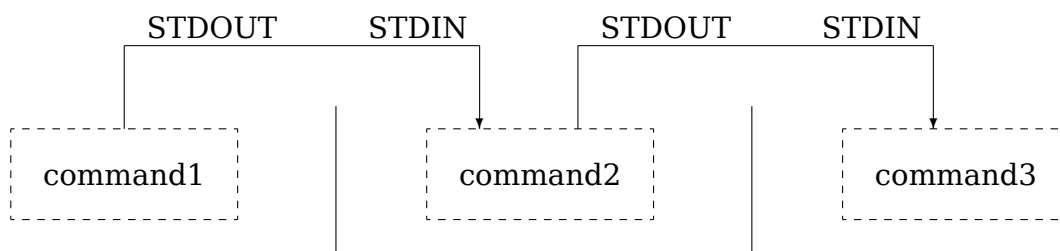
1. 爲何要使用命令輸出重導向？

- (a) 儲存螢幕輸出的資訊；
- (b) 背景執行中的程式，不希望他干擾螢幕正常的輸出結果時；
- (c) 儲存系統例行命令（例如寫在 `/etc/crontab` 中的檔案）的執行結果；
- (d) 將已知的可能錯誤訊息丟掉『`2> /dev/null`』；
- (e) 錯誤訊息與正確訊息需要分別輸出時。

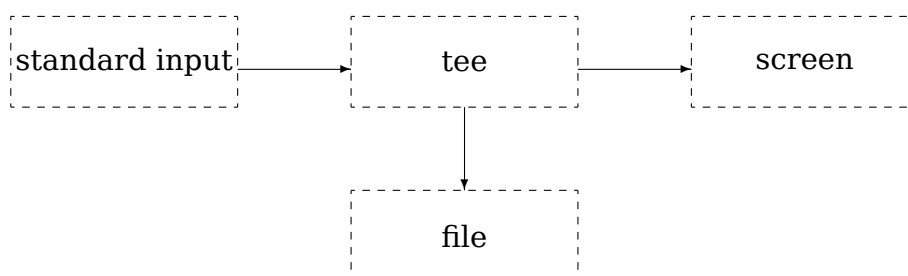
2. 指令執行過程之資料傳輸



3. 管線命令的處理



4. 雙向重導向流程



4.2 資料流重導向

1. 輸入輸出代碼

- (a) 標準輸入 (stdin) : 代碼為 0 , 使用 < 或 << ;
- (b) 標準輸出 (stdout) : 代碼為 1 , 使用 > 或 >> ;
- (c) 標準錯誤輸出 (stderr) : 代碼為 2 , 使用 2> 或 2>> ;

	>, 1>	標準輸出至檔案，該檔案被覆蓋或建立。
	>>, 1>>	標準輸出至檔案，該檔案被建立或累加。
command	2> 裝置或檔案	錯誤輸出至檔案，該檔案被覆蓋或建立。
	2>>	錯誤輸出至檔案，該檔案被建立或累加。
	<	輸入
	<<	結束的輸入字元

2. ls 列出 /etc/crontab，/etc/crontab 檔案存在，STDOUT 輸出 /etc/crontab 到螢幕。

```
1 [dywang@dywmac ~]$ ls /etc/crontab
/etc/crontab
```

3. ls 列出 /etc/dywang，檔案 /etc/dywang 不存在，STDERR 輸出 /etc/dywang 不存在的錯誤訊息也會輸出到螢幕。

```
2 [dywang@dywmac ~]$ ls /etc/dywang
ls: cannot access /etc/dywang: No such file or directory
```

4. ls 同時列出 /etc/crontab /etc/dywang，STDOUT 及 STDERR 輸出都在螢幕上。

```
2 [dywang@dywmac ~]$ ls /etc/crontab /etc/dywang
ls: cannot access /etc/dywang: No such file or directory
/etc/crontab
```

5. STDOUT 導向到 /dev/null，則螢幕只剩 STDERR 輸出；STDERR 導向到 /dev/null，則螢幕只剩 STDOUT 輸出。

```
1 [dywang@dywmac ~]$ ls /etc/crontab /etc/dywang > /dev/null
ls: cannot access /etc/dywang: No such file or directory
3 [dywang@dywmac ~]$ ls /etc/crontab /etc/dywang 2> /dev/null
/etc/crontab
```

6. 使用 > 將 STDOUT 導向到檔案 lsout，再導向時檔案內的資料會被覆蓋；使用 >> 導向時則 STDOUT 會累加到檔案。

```
1 [dywang@dywmac ~]$ ls /etc/crontab > lsout
2 [dywang@dywmac ~]$ cat lsout
/etc/crontab
4 [dywang@dywmac ~]$ ls /etc/dywang-release > lsout
[dywang@dywmac ~]$ cat lsout
6 /etc/dywang-release
[dywang@dywmac ~]$ ls /etc/crontab >> lsout
8 [dywang@dywmac ~]$ cat lsout
/etc/dywang-release
10 /etc/crontab
```

7. 使用 2> 將 STDERR 導向到檔案 lserr，再導向時檔案內的資料會被覆蓋；使用 2>> 導向時則 STDERR 會累加到檔案。

```
1 [dywang@dywmac ~]$ ls /etc/dywang 2> lserr
2 [dywang@dywmac ~]$ cat lserr
ls: cannot access /etc/dywang: No such file or directory
4 [dywang@dywmac ~]$ ls /etc/err 2> lserr
[dywang@dywmac ~]$ cat lserr
6 ls: cannot access /etc/err: No such file or directory
[dywang@dywmac ~]$ ls /etc/dywang 2>> lserr
8 [dywang@dywmac ~]$ cat lserr
ls: cannot access /etc/err: No such file or directory
10 ls: cannot access /etc/dywang: No such file or directory
```

8. 要將 STDOUT 及 STDERR 同時導向到檔案 lsfile，以下做法不正確。

```
[dywang@dywmac ~]$ ls /etc/crontab /etc/dywang > lsfile 2> lsfile
2 [dywang@dywmac ~]$ cat lsfile
/etc/crontab
4 cess /etc/dywang: No such file or directory
```

9. 要將 STDOUT 及 STDERR 同時導向到檔案 lsfile，正確做法為 2>&1。

```
[dywang@dywmac ~]$ ls /etc/crontab /etc/dywang > lsfile 2>&1
2 [dywang@dywmac ~]$ cat lsfile
ls: cannot access /etc/dywang: No such file or directory
4 /etc/crontab
```

10. 將螢幕輸入結果存入檔案，<< 後接的字串 EOF 為結束輸入符號。

```
[dywang@dywmac ~]$ cat > catfile << EOF
2 > 123
> 456
4 > 789
> EOF
6 [dywang@dywmac ~]$ cat catfile
123
8 456
789
```

11. < 可以將檔案內容導向前面指令，cat 可再將收到的內容導向到 catfile。

```
1 [dywang@dywmac ~]$ cat < /etc/dywang-release
De-Yu Wang Linux 6.4 release (Taiwan)
3 [dywang@dywmac ~]$ cat > catfile < /etc/dywang-release
[dywang@dywmac ~]$ cat catfile
5 De-Yu Wang Linux 6.4 release (Taiwan)
```

4.3 連續命令

1. 指令間以分號 (;) 隔開：連續執行指令。編輯 str.sh 只有一行指令，輸出變數 str 內容，執行前用 export 將變數 str 導出，所以 str.sh 腳本可以使用 str 變數。

```
1 [dywang@dywmac ~]$ vim str.sh
[dywang@dywmac ~]$ cat str.sh
```

```
3 echo $str  
[dywang@dywmac ~]$ export str='abc' ; sh str.sh  
5 abc
```

2. 指令間以 `&&` 隔開：前面指令執行結果正確，就接著執行後續的指令，否則就略過；指令間以 `||` 隔開：前面指令執行結果不正確，就接著執行後續的指令，否則就略過。

```
1 [dywang@dywmac ~]$ ls /etc/dywang-release && cat /etc/dywang-release  
/etc/dywang-release  
3 De-Yu Wang Linux 6.4 release (Taiwan)  
[dywang@dywmac ~]$ ls /etc/dywang-release || cat /etc/dywang-release  
5 /etc/dywang-release
```

3. 以 `ls` 測試 `abcd` 是否存在？`&&` 及 `||` 連續執行，若存在則顯示其內容，若不存在，則顯示產生空檔案 `abcd`，再次執行時檔案 `abcd` 已存在，顯示其內容為空的。

```
1 [dywang@dywmac ~]$ ls abcd && cat abcd || touch abcd  
ls: cannot access abcd: No such file or directory  
3 [dywang@dywmac ~]$ cat abcd  
[dywang@dywmac ~]$ ls abcd && cat abcd || touch abcd  
5 abcd
```

4. 習題：以 `ls` 測試目錄 `/tmp/s9427???` 是否存在，若存在則進入目錄，若不存在，則建立目錄。

4.4 管線命令 pipe

1. 命令 `grep` 可以分析一行訊息，若當中有需要的資訊，就將該行取出。`ifconfig` 列出網卡訊息，經由管線命令送到指令 `grep` 過濾出 192 字串的行。

```
1 [dywang@dywmac ~]$ ifconfig | grep 192  
    inet addr:192.168.122.1  Bcast:192.168.122.255  Mask  
    :255.255.255.0  
3    inet addr:192.168.1.104  Bcast:192.168.1.255  Mask  
    :255.255.255.0
```

2. 命令 `ps aux` 列出執行的程序，經由管線命令送到指令 `grep` 過濾出 `ssh` 字串的行。

```
1 [dywang@dywmac ~]$ ps aux | grep ssh
root      2526  0.0  0.0  66488  2684 ?        Ss   06:42   0:00 /usr/
          sbin/sshd
3 dywang   4358  0.0  0.0  60088  5764 pts/5    S+   06:50   0:00 ssh
          dyw219
dywang    4469  0.0  0.0  110308  2280 pts/6    S+   06:57   0:00 grep --
          color ssh
```

3. 命令 `wc` 可以計算行數、字數、字元數，`cat` 列出 `/etc/man.config` 內容，經由管線命令送到命令 `wc` 計算。

```
2 [dywang@dywmac ~]$ cat /etc/man.config | wc
    152      765    4940
```

4. `cat` 列出 `/etc/passwd` 帳號表，經由管線命令送到命令 `grep` 找到 `bash`，也就是登入使用 `bash` 的用戶。

```
2 [dywang@dywmac ~]$ cat /etc/passwd | grep bash
root:x:0:0:root:/root:/bin/bash
dywang:x:500:500:De-Yu Wang:/home/dywang:/bin/bash
4 linda:x:503:503::/home/linda:/bin/bash
```

5. `grep` 找到登入使用 `bash` 的用戶，再一次管線命令用 `sort` 以冒號“:”分隔欄位，以第三個欄位排序。

```
2 [dywang@dywmac ~]$ cat /etc/passwd | grep bash | sort -t: -k3
root:x:0:0:root:/root:/bin/bash
dywang:x:500:500:De-Yu Wang:/home/dywang:/bin/bash
4 linda:x:503:503::/home/linda:/bin/bash
```

6. `sort` 排序後再一次管線命令，使用 `cut` 命令以冒號“:”分隔欄位，取出第 1 及 3 兩個欄位。

```
2 [dywang@dywmac ~]$ cat /etc/passwd | grep bash | sort -t: -k3 | cut -d:
-f 1,3
root:0
dywang:500
4 linda:503
```


4.5 tee 雙向重導向

1. 重複上節管線命令例子，換行找 `nologin` 的用戶，`grep -m5` 只列找到的前 5 筆，`sort` 排序 `-nk3` 是以數字排序第 3 欄位，最後以 `cut` 取出第 1 及 3 欄位，結果輸出在螢幕上。

```
[dywang@dywmac ~]$ cat /etc/passwd | grep -m5 nologin | sort -t: -nk3 |  
cut -d: -f 1,3  
2 bin:1  
daemon:2  
4 adm:3  
lp:4  
6 mail:8
```

2. 要將螢幕上的資料存檔，可以用 `>`, `>>`, `2>`, `2>>` 將 `STDOUT` 或 `STDERR` 導向到檔案，但螢幕就看不到訊息。若要維持螢幕輸出並存檔，要使用 `tee` 命令做「雙向重導向」。

```
[dywang@dywmac ~]$ cat /etc/passwd | grep -m5 nologin | sort -t: -nk3 \  
2 | cut -d: -f 1,3 | tee user.log  
bin:1  
4 daemon:2  
adm:3  
6 lp:4  
mail:8  
8 [dywang@dywmac ~]$ cat user.log  
bin:1  
10 daemon:2  
adm:3  
12 lp:4  
mail:8
```

3. 改找註解有“Daemon”字串的用戶，再雙向重導向到 `user.log`，`user.log` 內容被覆蓋成新的資料。

```
1 [dywang@dywmac ~]$ cat /etc/passwd | grep Daemon | sort -t: -nk3 \  
3 | cut -d: -f 1,3 | tee user.log  
nscd:28  
rpc:32  
5 pulse:497
```

```
[dywang@dywmac ~]$ cat user.log
7 nscd:28
  rpc:32
9  pulse:497
```

4. tee 加上 -a 選項雙向重導向的檔案就不會被覆蓋，而是累加上去。

```
1 [dywang@dywmac ~]$ cat /etc/passwd | grep -m5 nologin | sort -t: -nk3 \
  | cut -d: -f 1,3 | tee user.log
3 bin:1
  daemon:2
5 adm:3
  lp:4
7 mail:8
  [dywang@dywmac ~]$ cat /etc/passwd | grep Daemon | sort -t: -nk3 \
9  | cut -d: -f 1,3 | tee -a user.log
  nscd:28
11 rpc:32
   pulse:497
13 [dywang@dywmac ~]$ cat /etc/pauser.log
   bin:1
15 daemon:2
   adm:3
17 lp:4
   mail:8
19 nscd:28
   rpc:32
21 pulse:497
```

4.6 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 zzz 目錄。
- (b) 切換工作目錄到 zzz，所有練習腳本皆存在此目錄。
- (c) 腳本必須可執行。
- (d) 檔案導向都是執行.sh 腳本命令時導向，不是寫在腳本內導向。檔案導向到 /tmp/\$mysid.\$evalname 目錄，其中變數 \$mysid 為自己的學號，\$evalname 為評分程式名稱不加附檔名，目錄若不存在，必須自行建立。

2. 練習題

- (a) 寫一腳本 `redirect1.sh`，使用 `cat` 命令產生一個 10 行以上的檔案，檔名使用 `read` 從鍵盤輸入，每行都至少包含一個數字，以 EOF 做為結束輸入字元。測試產生 `/tmp/$mysid.$evalname/redirect1.txt` 看看。
- (b) 寫一腳本 `pipe1.sh`，使用 `read` 從鍵盤輸入讀入檔名 (使用絕對路徑)。以 `ls` 命令列出檔名或目錄，標準錯誤導向到 `perror.txt`，標準輸出導向到 `priht.txt`。
- (c) 寫一腳本 `pipe2.sh`，使用 `read` 從鍵盤輸入讀入檔名 (使用絕對路徑)。以 `ls` 命令列出檔名或目錄，以 `&&`，`||` 連續執行，如果檔案或目錄存在輸出 'FILE exist'，否則輸出 'FILE not exist'，其中 FILE 是輸入的檔名。
- (d) 寫一腳本 `pipe3.sh`，使用 `read` 從鍵盤輸入讀入一指定字串。
 - i. 執行 `ifconfig` 命令。
 - ii. 管線處理以 `grep` 命令過濾出指定的字串。
 - iii. 管線處理以 `sort` 命令指定分隔符號為 ':'，再以第二欄位排序。
 - iv. 管線處理以 `tee` 命令螢幕輸出同時導向到檔案，檔案名稱以指定字串命名。

Chapter 5

正規表示法

5.1 前言

1. 什麼是正規表示法

- (a) 處理字串的方法，以行為單位進行字串的處理行為。
- (b) 透過一些特殊符號的輔助，讓使用者輕易的達到搜尋、取代，限定某特定字串的處理程序。
- (c) 延伸的正規表示法：可以作群組的字串『或 (or)』及『且 (and)』的處理。
- (d) 正規表示法與萬用字元不一樣，不可混淆。

2. 正規表示法用途

- (a) 系統管理員的可以透過『正規表示法』的功能，將登錄的資訊進行處理，僅取出『有問題』的資訊來進行分析。
- (b) 一般支援正規表示法的軟體可利用其進行字串的處理，例如：『郵件伺服器』以『字串』的比對來過濾郵件。
- (c) 軟體中輸入欄位的格式限制，例如：只能為數字。

3. vi, grep, awk ,sed 等工具皆支援正規表示法。

5.2 基礎正規表示法

1. 重要特殊字元 (characters)

1	RE字元	意義與範例
2	^word	待搜尋的字串(word)在行首。
3	word\$	待搜尋的字串(word)在行尾。

	.	代表『任意一個』字符，一定是一個任意字符。
5	\	跳脫字符，將特殊符號的特殊意義去除。
	*	重複零個或多個的前一個 RE 字符
7	\{n,m\}	連續 n 到 m 個的『前一個 RE 字符』
		若為 \{n\} 則是連續 n 個的前一個 RE 字符，
9		若是 \{n,\} 則是連續 n 個以上的前一個 RE 字符。
	[]	在 [] 當中『僅代表一個待搜尋的字元』
11		若 [^]，^ 在中括號中表示 NOT。

2. 找檔案 /etc/man.config 中行首是大寫 C 的行。

```
1 [dywang@dywmac ~]$ grep -n ^C /etc/man.config
108:CAT      /bin/cat
3 114:CMP      /usr/libexec/man-cmp.sh
118:COMPRESS /usr/bin/lzma
5 119:COMPRESS_EXT .lzma
```

3. 查詢行尾是 s.。

```
1 [dywang@dywmac ~]$ grep -n 's\.$' /etc/man.config
37:# and to determine the correspondence between extensions and
   decompessors.
3 63:# lots of other nearby files and directories.
144:# Enable/disable makewhatis database cron updates.
```

4. 計算行首是 # 且只有這個註解符號，共有 34 行。

```
1 [dywang@dywmac ~]$ grep '^#\$' /etc/man.config | wc -l
2 34
```

5. 計算行首不是 # 的行數，也就是不是註解的行，共有 39 行。

```
1 [dywang@dywmac ~]$ grep -v '^#' /etc/man.config | wc -l
2 113
```

6. 搜尋 C 接 H 或 M

```
1 [dywang@dywmac ~]$ grep -n 'C[HM]' /etc/man.config
2 80:# NOCACHE keeps man from creating cache pages ("cat pages")
   84:#NOCACHE
4 111:# When CMP is defined man will try to avoid showing the same
```

```
114:CMP      /usr/libexec/man-cmp.sh
```

7. 搜尋行首 C 但不接 H 或 M

```
1 [dywang@dywmac ~]$ grep -n '^C[~HM]' /etc/man.config
108:CAT      /bin/cat
3  118:COMPRESS    /usr/bin/lzma
   119:COMPRESS_EXT .lzma
```

8. 搜尋數字後至少接一個以上英文字母

```
2 [dywang@dywmac ~]$ grep '[0-9][a-zA-Z][a-zA-Z]*' /etc/man.config
# man.conf from man-1.6f
MANPATH /usr/X11R6/man
4 MANPATH_MAP /usr/X11R6/bin      /usr/X11R6/man
  MANPATH_MAP /usr/bin/X11        /usr/X11R6/man
6 # and the MANSECT environment variable is not set (1x-8x sections are
  used by
MANSECT      1:1p:8:2:3:3p:4:5:6:7:9:0p:n:1:p:o:1x:2x:3x:4x:5x:6x:7x:8x
```

9. 搜尋連續 12 到 15 個英文字母的字串

```
1 [dywang@dywmac ~]$ grep '[a-zA-Z]\{12,15\}' /etc/man.config
# Generated automatically from man.conf.in by the
3 # pages corresponding to given man pages should be stored,
# MANPATH      manpath_element [corresponding_catdir]
5 # and to determine the correspondence between extensions and
  decompressors.
# Every automatically generated MANPATH includes these fields
7 # NOAUTOPATH keeps man from automatically adding directories that look
  like
#MANDEFOPTIONS  -a
9 # Decompress with given decompressor when input file has given extension
# If MAKEWHATISDBUPDATES variable is uncommented
11 #MAKEWHATISDBUPDATES      n
```

10. 搜尋連續 14 個以上的英文字母

```
1 [dywang@dywmac ~]$ grep '[a-zA-Z]\{14,\}' /etc/man.config
# and to determine the correspondence between extensions and
  decompressors.
3 # If MAKEWHATISDBUPDATES variable is uncommented
#MAKEWHATISDBUPDATES      n
```

11. 搜尋剛好連續 13 個英文字母

```
[dywang@dywmac ~]$ grep '[^a-zA-Z][a-zA-Z]\{13\}[^a-zA-Z]' /etc/man.  
config  
2 # Generated automatically from man.conf.in by the  
# pages corresponding to given man pages should be stored,  
4 # MANPATH      manpath_element [corresponding_catdir]  
# and to determine the correspondence between extensions and  
# decompressors.  
6 # Every automatically generated MANPATH includes these fields  
# NOAUTOPATH keeps man from automatically adding directories that look  
# like  
8 #MANDEFOPTIONS -a
```

12. 搜尋 x 開始 y 結束的字串

```
[dywang@dywmac ~]$ grep 'x.*y' /etc/man.config  
2 # Explicitly given catdirs override.  
# The command "man -a xyzzy" will show all man pages for xyzzy.  
4 # and the MANSECT environment variable is not set (1x-8x sections are  
# used by
```

13. 搜尋多個字串，例如：this 及 will 兩個字串。

```
[root@dywssd bin]# grep 'this\\|will' /etc/man.config  
2 # For more information about this file, see the man pages man(1)  
# (so that this dir has both man1 etc. and cat1 etc. subdirs).  
4 # The keyword FSSTND will cause this behaviour.  
# The keyword FHS will cause this behaviour (and overrides FSSTND).  
6 # in the mandatory manpath already, but will keep man from statting  
# The command "man -a xyzzy" will show all man pages for xyzzy.  
8 # When CMP is defined man will try to avoid showing the same  
# will not update makewhatis database.  
10 # Otherwise the database will be updated.
```

5.3 Windows 斷行符號

1. 切換工作目錄到 zzz，編輯一個簡單的腳本 dos.sh 複製 /etc/issue 到此目錄，使用 cat 查看檔案內容，並使用選項 -E 顯示結束符號、-v 顯示非列印符號、-T 顯示 TAB 空白。

```
[dywang@dywmac ~]$ cd zzz  
2 [dywang@dywmac zzz]$ vim dos.sh
```

```
[dywang@dywmac zzz]$ cat -EvT dos.sh
4 #/bin/bash$
a=123$
6 echo^I$a$
$
```

2. 試著執行 dos.sh，成功輸出變數 a 的內容 123。

```
1 [dywang@dywmac zzz]$ sh dos.sh
123
```

3. 如果您是使用 windows 的 notepad 編輯 dos.sh，存檔時每行行尾會有斷行符號 ^M。為測試此不同，先以指令 unix2dos 將 dos.sh 轉成 DOS 格式，再以 cat -EvT 顯示檔案內容，可查看到 windows 格式的文件結尾符號 ^M。

```
[dywang@dywmac zzz]$ unix2dos dos.sh
2 unix2dos: converting file dos.sh to DOS format ...
[dywang@dywmac zzz]$ cat -EvT dos.sh
4 #/bin/bash^M$
a=123^M$
6 echo^I$a^M$
^M$
```

4. grep 要找 windows 的斷行符號，必須使用 c-style escape '\$'\r'' 解釋 ...，例如 \n 是換行，而 windows 的斷行符號是 \r。因此可以找到有斷行符號的總行數 4 行，也就是 issue 有 4 行，空白行 1 行。

```
1 [dywang@dywmac zzz]$ grep '$'\r' dos.sh | wc -l
4
3 [dywang@dywmac zzz]$ grep '$'^\r' dos.sh | wc -l
1
```

5. 執行 dos.sh 時，空白行不應該執行，但因有斷行符號，所以出現錯誤訊息。

```
2 [dywang@dywmac zzz]$ sh dos.sh
123
: command not found
```

6. 再將 dos.sh 轉成 unix 格式並成功執行。


```

1 [dywang@dywmac zzz]$ dos2unix dos.sh
dos2unix: converting file dos.sh to UNIX format ...
3 [dywang@dywmac zzz]$ sh dos.sh
123

```

5.4 正規表示法與萬用字元

1. 『正規表示法的特殊字元』與『萬用字元』之差異：

字元或代表意義	萬用字元	正規表示法的特殊字元
*	0 至無限多個字元	重複 0 到多個的前一個 RE 字符
反向選擇	[!range]	[^range]

2. ls 不支援正規表示法，使用的是萬用字元。

```

[dywang@dywmac zzz]$ ls dos*
2 dos.sh
[dywang@dywmac zzz]$ touch dos1 dos2
4 [dywang@dywmac zzz]$ ls dos*
dos1 dos2 dos.sh

```

3. find 不支援正規表示法，使用的是萬用字元。

```

1 [dywang@dywmac zzz]$ find . -name 'dos*'
./dos1
3 ./dos2
./dos.sh

```

4. 例題：想要查出來檔案中含有! 與 > 的字行 (! 在正規表示法中並不是特殊字元)：

```
grep -n '[!>]' re.txt
```

5.5 延伸正規表示法

1. grep 支援的是基礎型的正規表示法，延伸正規表示法 egrep 是 grep -E 的命令別名。

2. 延伸型正規表示法之特殊符號

RE 字符	意義與範例
+	重複『一個或一個以上』的前一個 RE 字符
?	『零個或一個』的前一個 RE 字符
	用或(or)的方式找出數個字串
()	找出『群組』字串或作為『多個重複群組』的判別

3. + : 找 C 後接一個以上小寫英文字母。

```

1 [dywang@dywIssd zzz]$ grep -E 'C[a-z]+' /etc/man.config --color
# Certain versions of the FSSTND recommend putting formatted versions
3 # Certain versions of the FHS recommend putting formatted versions of
# Compress cat pages

```

4. (|) : 找 local 或 share 或 foo 字串。

```

[dywang@dywIssd zzz]$ egrep '(local|share|foo)' /etc/man.config --color
2 # /usr/.../share/man/[locale/]manx/page.x into
# /var/cache/man/.../[locale/]catx/page.x.
4 # MANBIN /usr/local/bin/man
MANPATH /usr/share/man
6 MANPATH /usr/local/man
MANPATH /usr/local/share/man
8 # MANPATH /usr/share/*/man
# If people ask for "man foo" and have "/dir/bin/foo" in their PATH
10 MANPATH_MAP /bin /usr/share/man
MANPATH_MAP /sbin /usr/share/man
12 MANPATH_MAP /usr/bin /usr/share/man
MANPATH_MAP /usr/sbin /usr/share/man
14 MANPATH_MAP /usr/local/bin /usr/local/share/man
MANPATH_MAP /usr/local/sbin /usr/local/share/man
16 MANPATH_MAP /usr/bin/mh /usr/share/man

```

5. ? : 找 inet 後接一個或 0 個數字"6"。

```

[dywang@dywIssd zzz]$ ifconfig | egrep 'inet6? ' --color
2 inet addr:192.168.1.140 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::8ad7:f6ff:fe53:3525/64 Scope:Link
4 inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
6 inet addr:192.168.122.1 Bcast:192.168.122.255 Mask:255.255.255.0
inet6 addr: 2001:ac18::131/64 Scope:Global

```

5.6 基礎與延伸比較一

1. 編譯一個測試檔如下：

```
1 [dywang@dywmac ~]$ vim a.txt
[dywang@dywmac ~]$ cat a.txt
3 asdt
  ast
5 asbt
  as?t
7 asbt
  as?t
9 as\bt
  as\?t
```

2. 查詢 `as?t` 不加單引號，基礎正規表示法中 `?` 不是特殊字元，使不使用 `\` 跳脫都一樣查到 `as?t`。

```
[dywang@dywmac ~]$ grep as?t a.txt
2 as?t
  as?t
4 [dywang@dywmac ~]$ grep as\?t a.txt
  as?t
6 as?t
```

3. 查詢 `as?t` 加單引號，則單引號中每一個字符都要匹配，基礎正規表示法中 `?` 不是特殊字元，查到 `as?t`。

```
[dywang@dywmac ~]$ grep 'as?t' a.txt
2 as?t
  as?t
```

4. 查詢 `as\?t` 加單引號，則單引號中 `\?` 為延伸正規表示法的 `?`，也就是重複前一個字符 0 或 1 次，所以查到 `ast`。

```
1 [dywang@dywmac ~]$ grep 'as\?t' a.txt
  ast
```

5. 查詢 `a\?t` 或 `h\?t` 加單引號，則單引號中 `\?` 為延伸正規表示法的 `?`，也就是重複前一個字符 `a(h)` 0 或 1 次，所以有 `t` 都查到。

```
[dywang@dywmac ~]$ grep 'a\?t' a.txt
2 asdt
  abt
4 afbt
  ar?t
6 asbt
  as?t
8 as\bt
  as\?t
10 [dywang@dywmac ~]$ grep 'h\?t' a.txt
   asdt
12   ast
   asbt
14   as?t
   asbt
16   as?t
   as\bt
18   as\?t
```

5.7 基礎與延伸比較二

1. 編輯測試檔 b.txt。

```
[dywang@dywmac zzz]$ vim b.txt
2 [dywang@dywmac zzz]$ cat b.txt
boot
4 345 bot
root 390 booooot
6 aoot sort 390 booooot
coot boooooooooot
8 roooooooooot 390 xyzbt
root 134not390 booooot
```

2. 查詢 3 至 5 個連續字元 o，使用延伸表示時大括號不加倒斜線。

```
1 [dywang@dywmac zzz]$ grep --color 'o\{3,5\}' b.txt
root 390 booooot
3 aoot sort 390 booooot
coot boooooooooot
5 roooooooooot 390 xyzbt
root 134not390 booooot
7 [dywang@dywmac zzz]$ egrep --color 'o{3,5}' b.txt
root 390 booooot
9 aoot sort 390 booooot
coot boooooooooot
11 roooooooooot 390 xyzbt
```

```
root 134not390 booooot
```

3. 查詢 b 與 t 之間 1 個以上連續字元 o。

```
[dywang@dywmac zzz]$ grep --color 'boo*t' b.txt
2 boot
345 bot
4 root 390 booooot
aoot sort 390 booooot
6 coot boooooooooot
root 134not390 booooot
8 [dywang@dywmac zzz]$ egrep --color 'bo+t' b.txt
boot
10 345 bot
root 390 booooot
12 aoot sort 390 booooot
coot boooooooooot
14 root 134not390 booooot
```

4. 查詢 b 與 t 之間 0 或 1 個字元 o。

```
[dywang@dywmac zzz]$ grep --color 'bo\{0,1\}t' b.txt
2 345 bot
roooooooooot 390 xyzbt
4 [dywang@dywmac zzz]$ egrep --color 'bo?t' b.txt
345 bot
6 roooooooooot 390 xyzbt
```

5.8 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 zzz 目錄。
- (b) 切換工作目錄到 zzz，所有練習腳本皆存在此目錄。
- (c) 腳本必須可執行。
- (d) 檔案導向都是執行.sh 腳本命令時導向，不是寫在腳本內導向。檔案導向到 /tmp/\$mysid.\$evalname 目錄，其中變數 \$mysid 為自己的學號，\$evalname 為評分程式名稱不加附檔名，目錄若不存在，必須自行建立。

2. 練習題

- (a) 下載檔案 `re1.txt`。寫 `.sh` 腳本，以 `grep` 對 `re1.txt` 執行以下搜尋，不需要列印行號，不要做任何的更動，所有結果導向到 `.txt` 檔案：
- i. `myre1.1.sh` 導向到 `myre1.1.txt`
 - A. `you` 其中 `y` 大小寫都可以。
 - B. `tast` 或 `test`。
 - C. `oo` 前不是 `g`，也不是 `t`。
 - ii. `myre1.2.sh` 導向到 `myre1.2.txt`
 - A. 剛好三個阿拉伯數字。
 - B. 兩個以上阿拉伯數字。
 - iii. `myre1.3.sh` 導向到 `myre1.3.txt`
 - A. 行首是大寫英文字母。
 - B. 行尾不是 `'` 句點。
- (b) 下載檔案 `dos.sh`，`dos.sh` 是用 windows notepad 編輯檔案，下載下來卻無法執行，請轉換成可在 Linux 下執行的腳本。
- (c) 下載檔案 `re2.txt`，寫 `.sh` 腳本，以 `grep` 對 `re2.txt` 使用一般正規表示法執行以下搜尋，不需要列印行號，不要做任何的更動，所有結果依序累加到 `.txt` 檔案：
- i. `myre2.1.sh` 導向到 `myre2.1.txt`。
 - A. 開頭 `b` 結尾是 `t`，中間 1 個以上 `a` 的字串。
 - B. 開頭 `b` 結尾是 `t`，中間 0 個或 1 個 `a` 的字串。
 - C. `dog`, `feet`, `good` 三個字串。
 - ii. `myre2.2.sh` 導向到 `myre2.2.txt`。
 - A. `boot` 或 `babt`。
 - B. 開頭 `b` 結尾是 `t`，中間 `pqr` 重複 1 次以上。
- (d) 寫 `.sh` 腳本，以 `grep` 對 `re2.txt` 使用延伸正規表示法執行以下搜尋，不需要列印行號，不要做任何的更動，所有結果依序累加到 `.txt` 檔案：
- i. `myre2e.1.sh` 導向到 `myre2e.1.txt`。
 - A. 開頭 `b` 結尾是 `t`，中間 1 個以上 `a` 的字串。
 - B. 開頭 `b` 結尾是 `t`，中間 0 個或 1 個 `a` 的字串。
 - C. `dog`, `feet`, `good` 三個字串。
 - ii. `myre2e.2.sh` 導向到 `myre2e.2.txt`。
 - A. `boot` 或 `babt`。
 - B. 開頭 `b` 結尾是 `t`，中間 `pqr` 重複 1 次以上。

- (e) 下載檔案 `sid.txt`，寫一腳本 `resid.sh` 使用 `grep` 對 `sid.txt` 執行學號姓名檔的搜尋，不需要列印行號，輸出存到檔案 `residresult1.txt`，不要做任何的更動。`sid` 學號姓名檔格式要求如下：
- i. 格式：「電腦位置 IPV4 最後一組數字學號姓名」
 - ii. 每個欄位用一個空白隔開。
 - iii. `resid1.sh` 查詢行首電腦位置 A-B，A: 1-6 B: 1-12，導向到 `resid1.txt`。
 - iv. `resid2.sh` 查詢 IPV4 最後一組數字：50-199 且前後都空白，導向到 `resid2.txt`。
 - v. `resid3.sh` 查詢學號：「1xxxxxxx」，x 是 0-9 數字，且前後都空白，導向到 `resid3.txt`。
 - vi. `resid4.sh` 查詢姓名：「OOO」，O 是中文字，要求行尾 2-5 個中文字，導向 `resid4.txt`。
 - vii. 提示：`grep` 找中文字用 `-P` (`--perl-regexp`) 選項，中文字正規表示 `\p{Han}`，`-P` 已支援延伸正規表示法的群組 `()` 及或 `|`。

Chapter 6

sed 工具

6.1 sed 命令簡介

1. sed 命令輔助說明如下：

```
[dywang@dywmac zzz]$ sed --help
2 Usage: sed [OPTION]... {script-only-if-no-other-script} [input-file]...

4  -n, --quiet, --silent
           suppress automatic printing of pattern space
6  -e script, --expression=script
           add the script to the commands to be executed
8  -f script-file, --file=script-file
           add the contents of script-file to the commands to be
           executed
10 --follow-symlinks
           follow symlinks when processing in place; hard links
12           will still be broken.
-i[SUFFIX], --in-place[=SUFFIX]
14           edit files in place (makes backup if extension supplied
           ).
           The default operation mode is to break symbolic and
           hard links.
16           This can be changed with --follow-symlinks and --copy.
-c, --copy
18           use copy instead of rename when shuffling files in -i
           mode.
           While this will avoid breaking links (symbolic or hard)
           , the
20           resulting editing operation is not atomic. This is
           rarely
           the desired mode; --follow-symlinks is usually enough,
           and
22           it is both faster and more secure.
-l N, --line-length=N
24           specify the desired line-wrap length for the 'l'
           command

--posix
```



```

26      disable all GNU extensions.
    -r, --regexp-extended
28      use extended regular expressions in the script.
    -s, --separate
30      consider files as separate rather than as a single
        continuous
        long stream.
32    -u, --unbuffered
        load minimal amounts of data from the input files and
        flush
34      the output buffers more often
    --help    display this help and exit
36    --version output version information and exit

38 If no -e, --expression, -f, or --file option is given, then the first
non-option argument is taken as the sed script to interpret. All
40 remaining arguments are names of input files; if no input files are
specified, then the standard input is read.
42
GNU sed home page: <http://www.gnu.org/software/sed/>.
44 General help using GNU software: <http://www.gnu.org/gethelp/>.
E-mail bug reports to: <bug-gnu-utils@gnu.org>.
46 Be sure to include the word ``sed'' somewhere in the ``Subject:'' field.

```

2. sed (stream editor) 可以分析 Standard Input (STDIN) 的資料，進行取代、刪除、新增、擷取特定行等處理後，再輸出到 standard out (STDOUT)，功能簡介如下：

```

[root@linux ~]# sed [-nefr] [動作]
2 選項：
    -n    : 使用安靜 (silent) 模式。在一般 sed 的用法中，所有來自 STDIN
4         的資料一般都會被列出到螢幕上。但如果加上 -n 參數後，則只有經過
         sed 特殊處理的那一行(或者動作)才會被列出來。
6    -e    : 直接在指令列模式上進行 sed 的動作編輯；
    -f    : -f filename 可以執行 filename 內的 sed 動作；
8    -r    : sed 的動作支援的是延伸型正規表示法的語法。(預設是基礎正規表示法語
        法)
    -i    : 直接修改讀取的檔案內容，而不是由螢幕輸出。
10
動作說明： [n1[,n2]]function
12 n1, n2  : 選擇進行動作的行數，例如，『10,20[動作行爲]』
14 function:
    a      : 新增，a 的後面可以接字串，這些字串會在目前的下一行出現。
16    c      : 取代，c 的後面可以接字串，這些字串可以取代 n1,n2 之間的行。
    d      : 刪除，d 後面通常不接任何字串；
18    i      : 插入，i 的後面可以接字串，這些字串會在目前的上一行出現；
    p      : 列印，將某個選擇的資料印出。通常 p 會與參數 sed -n 一起運作。
20    s      : 取代，s 的動作可以搭配正規表示法。例如 1,20s/old/new/g。

```

6.2 sed 範例一

1. `cat -n` 顯示行號列出 `/etc/passwd` 的內容，管線命令給 `head` 取前 10 行，再管線處理以 `sed` 刪除第 2 5 行。

```
[dywang@dywmac ~]$ cat -n /etc/passwd | head | sed '2,5d'
2      1 root:x:0:0:root:/root:/bin/bash
      6 sync:x:5:0:sync:/sbin:/bin/sync
4      7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
      8 halt:x:7:0:halt:/sbin:/sbin/halt
6      9 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
      10 uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

2. `cat -n` 顯示行號列出 `/etc/passwd` 的內容，管線命令給 `head` 取前 5 行，再管線處理以 `sed` 刪除第 3 行。

```
1 [dywang@dywmac ~]$ cat -n /etc/passwd | head -5 | sed '3d'
      1 root:x:0:0:root:/root:/bin/bash
3      2 bin:x:1:1:bin:/bin:/sbin/nologin
      4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5      5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

3. `cat -n` 顯示行號列出 `/etc/passwd` 的內容，管線命令給 `tail` 取最後 10 行，再管線處理以 `sed` 刪除第 5 至最後一行。

```
1 [dywang@dywmac ~]$ cat -n /etc/passwd | tail | sed '5,$d'
      43 hsqldb:x:96:96::/var/lib/hsqldb:/sbin/nologin
3      44 openvpn:x:495:491:OpenVPN:/etc/openvpn:/sbin/nologin
      45 toranon:x:494:490:Tor anonymizing user:/var/lib/tor:/sbin/
      nologin
5      46 mockbuild:x:502:502::/home/mockbuild:/sbin/nologin
```

4. 在第二行後面加入兩行字，例如『Drink tea or』『drink beer?』。可以新增好幾行，但每一行之間都必須要以反斜線 `"\"` 跳脫 Enter 進行新行的增加。

```
1 [dywang@dywmac ~]$ cat -n /etc/passwd | head -5 | sed '2a Drink tea or
      .....\'
> drink beer?'
3      1 root:x:0:0:root:/root:/bin/bash
      2 bin:x:1:1:bin:/bin:/sbin/nologin
5 Drink tea or .....
  drink beer?
7      3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

```

9      4  adm:x:3:4:adm:/var/adm:/sbin/nologin
      5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

```

5. 將第 2-5 行的內容取代成為『No 2-5 rows』?

```

1  [dywang@dywmac ~]$ cat -n /etc/passwd | head -6 | sed '2,5c No 2-5 rows'
      1  root:x:0:0:root:/root:/bin/bash
3  No 2-5 rows
      6  sync:x:5:0:sync:/sbin:/bin/sync

```

6. 僅列出第 4-6 行，sed 沒有加 -n 參數時 sed '4,6p'，4-6 行會重複輸出。

```

2  [dywang@dywmac ~]$ cat -n /etc/passwd | sed -n '4,6p'
      4  adm:x:3:4:adm:/var/adm:/sbin/nologin
      5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
4      6  sync:x:5:0:sync:/sbin:/bin/sync

```

6.3 sed 範例二

1. 練習：使用 ifconfig eth0 查詢 eth0 網卡訊息，grep 'inet' 查詢其 ipv4 的 IP。接著可以再用兩次的管線命令，以 sed 將 IP 前後不要的資料刪除。

```

2  [dywang@dywmac ~]$ ifconfig eth0 | grep 'inet '
      inet addr:192.168.1.104 Bcast:192.168.1.255 Mask
      :255.255.255.0

```

2. 取 ipv4 IP 解答

```

2  [dywang@dywmac ~]$ ifconfig eth0 | grep 'inet ' | sed 's/.*addr://' |
      sed 's/ .*//'
192.168.1.104

```

6.4 sed 範例三

1. 利用 sed 直接在 ~/.bashrc 最後一行加入 z="/home/dywang/zzz"，\$ 代表最後一行，所以 \$a 是在最後一行才新增。以下命令只是由螢幕輸出，若要寫到檔案必須加 -i 參數。

```

[dywang@dywmac ~]$ sed 's#a z=\"/home/dywang/zzz\"' ~/.bashrc
2 # .bashrc
4 # Source global definitions
  if [ -f /etc/bashrc ]; then
6     . /etc/bashrc
  fi
8 # User specific aliases and functions
10 z=\"/home/dywang/zzz\"

```

6.5 sed 範例四

1. 練習：取出 /etc/man.config 檔案的內容中有 MANPATH 的設定，但其中包含以 "#" 開始的註解，請再利用管線命令以 sed 刪除註解行。

```

[dywang@dywmac ~]$ grep MANPATH /etc/man.config
2 # when MANPATH contains an empty substring), to find out where the cat
  # MANPATH      manpath_element [corresponding_catdir]
4 # MANPATH_MAP   path_element   manpath_element
  # Every automatically generated MANPATH includes these fields
6 MANPATH /usr/man
  MANPATH /usr/share/man
8 MANPATH /usr/local/man
  MANPATH /usr/local/share/man
10 MANPATH /usr/X11R6/man
  # MANPATH /opt/*/man
12 # MANPATH /usr/lib/*/man
  # MANPATH /usr/share/*/man
14 # MANPATH /usr/kerberos/man
  # Set up PATH to MANPATH mapping
16 MANPATH_MAP /bin      /usr/share/man
  MANPATH_MAP /sbin     /usr/share/man
18 MANPATH_MAP /usr/bin  /usr/share/man
  MANPATH_MAP /usr/sbin /usr/share/man
20 MANPATH_MAP /usr/local/bin /usr/local/share/man
  MANPATH_MAP /usr/local/sbin /usr/local/share/man
22 MANPATH_MAP /usr/X11R6/bin /usr/X11R6/man
  MANPATH_MAP /usr/bin/X11 /usr/X11R6/man
24 MANPATH_MAP /usr/bin/mh /usr/share/man

```

2. 取出 /etc/man.config 檔案的內容中有 MANPATH 的設定解答。且刪除註解說明。

```

[dywang@dywmac ~]$ grep MANPATH /etc/man.config | sed '/^#/d'

```

```

2 | MANPATH /usr/man
   | MANPATH /usr/share/man
4 | MANPATH /usr/local/man
   | MANPATH /usr/local/share/man
6 | MANPATH /usr/X11R6/man
   | MANPATH_MAP /bin /usr/share/man
8 | MANPATH_MAP /sbin /usr/share/man
   | MANPATH_MAP /usr/bin /usr/share/man
10 | MANPATH_MAP /usr/sbin /usr/share/man
    | MANPATH_MAP /usr/local/bin /usr/local/share/man
12 | MANPATH_MAP /usr/local/sbin /usr/local/share/man
    | MANPATH_MAP /usr/X11R6/bin /usr/X11R6/man
14 | MANPATH_MAP /usr/bin/X11 /usr/X11R6/man
    | MANPATH_MAP /usr/bin/mh /usr/share/man

```

3. sed 加參數 -e 同時完成刪除與取代命令。例如：grep -v 取出 /etc/man.config 不是註解的設定行，再利用 sed 刪除檔案第 10 行至最後一行，且將字串 MAN 改成 man。

```

1 | [dywang@dywmac ~]$ grep -v '^#' /etc/man.config | sed -e '10,$d' -e 's/
   | MAN/man/g'
   | FHS
3 | manPATH /usr/man
   | manPATH /usr/share/man
5 | manPATH /usr/local/man
   | manPATH /usr/local/share/man
7 | manPATH /usr/X11R6/man
   | manPATH_MAP /bin /usr/share/man
9 | manPATH_MAP /sbin /usr/share/man
   | manPATH_MAP /usr/bin /usr/share/man

```

4. 如果 sed 要同時執行很多動作，可以將這些動作寫在一個檔案中，再以 -f 選項指令檔案執行 sed 命令。例如將上例刪除與取代命令編輯至檔案 ds.src，執行結果一樣。

```

2 | [dywang@dywmac zzz]$ vim ds.src
   | [dywang@dywmac zzz]$ cat ds.src
   | 10,$d
   | s/MAN/man/g
6 | [dywang@dywmac zzz]$ grep -v '^#' /etc/man.config | sed -f ds.src
   | FHS
8 | manPATH /usr/man
   | manPATH /usr/share/man
10 | manPATH /usr/local/man
    | manPATH /usr/local/share/man
12 | manPATH /usr/X11R6/man

```

```

14 manPATH_MAP /bin          /usr/share/man
   manPATH_MAP /sbin        /usr/share/man
   manPATH_MAP /usr/bin      /usr/share/man

```

6.6 sed 與正規表示法

1. sed 配合正規表示法進行搜尋 `sed -n '/pattern/=' filename`，`-n` 選項只列印行號，不列印檔案內容。

```

1 [dywang@dywmac zzz]$ grep -n '[nb]zip' /etc/man.config
136:.gz      /usr/bin/gunzip -c
3 137:.bz2    /usr/bin/bzip2 -c -d
[dywang@dywmac zzz]$ sed -n '/[nb]zip/=' /etc/man.config
5 136
  137

```

2. sed 配合正規表示法刪除一整行 `sed -n '/pattern/d' filename`，刪除 ssh 遠端連線紀錄檔中 IP 為 192.168.122.0/24 的紀錄，sed 沒有加 `-i` 選項不會真的刪除檔案中的內容，只是螢幕輸出中刪除。

```

[dywang@dywmac ~]$ sed '/192\.168\.122\.[0-9]\{1,3\}/d' .ssh/known_hosts
\
2 | grep 192.168.122 --color

```

3. 群組字串的取代：找到 `\(patternN\)`，可用 `\N` 將其放置取代結果。

```

sed 's/\(pattern1\)...\(pattern2\)/\2\1/g'
2 [dywang@dywmac zzz]$ echo 'AxB' | sed 's/\([A-Z]\)x\([A-Z]\)/\2--\1/g'
  B--A
4 [dywang@dywmac zzz]$ echo 'DxU' | sed 's/\([A-Z]\)x\([A-Z]\)/\2--\1/g'
  U--D

```

4. 加減乘除符號 `+` `-` `*` `/` 的取代，使用雙引號時，減號要使用兩個反斜線跳脫。

```

1 [dywang@deyu moodle]$ echo '+-*/+*/' | sed "s/\([+\\-*/\\)\)/\1A/g"
sed: -e expression #1, char 19: Invalid range end
3 [dywang@deyu moodle]$ echo '+-*/+*/' | sed "s/\([+*\\/\\-\\)\)/\1A/g"
+A-A*A/A+A-A*A/A

```

5. 加減乘除符號 $+$ $-$ $*$ $/$ 的取代，使用單引號時，減號就可以使用一個反斜線跳脫。

```
[dywang@dywIssd zzz]$ echo '+-*/+-*/' | sed 's/\([+*/\]\)/\1A/g'
2 +A-A*A/A+A-A*A/A
[dywang@dywIssd zzz]$ echo '+-*/+-*/' | sed 's/\([+*/\]\)/\1A/g'
4 +A-A*A/A+A-A*A/A
```

6. 將 (a,b,x) 取代成 $axb=$ 。

```
[dywang@dywmac zzz]$ echo '(23,10,+)' | sed 's|(\([0-9]*\)),\([0-9]*\)|\1\3\2=|g'
2 23+10=
```

6.7 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 `zzz` 目錄。
- (b) 切換工作目錄到 `zzz`，所有練習腳本皆存在此目錄。
- (c) 腳本必須可執行。
- (d) 檔案導向都是執行 `.sh` 腳本命令時導向，不是寫在腳本內導向。檔案導向到 `/tmp/$mysid.$evalname` 目錄，其中變數 `$mysid` 為自己的學號，`$evalname` 為評分程式名稱不加附檔名，目錄若不存在，必須自行建立。

2. 練習題

- (a) 下載檔案 `man.conf`
- (b) 寫腳本以 `sed` 處理 `man.conf` 執行以下動作，執行腳本將結果導向到「輸出檔案」，不要做任何的更動。
 - i. 腳本 `sed1.1.sh`，輸出檔案 `sed1.1.txt`
 - A. 刪除 50 至 120 行。
 - B. 第 63 行後增加三行。

```
2 123ASD
   KJH983
   apem34kq98
```

C. 第 30 至 145 行取代成：

```
1 abcdefgh
  9876543
3 3333333
```

ii. 腳本 sed1.2.sh，輸出檔案 sed1.2.txt

A. 只列出第 103 至 120 行。

B. 刪除註解行及空白行。

C. 第 17 行至最後一行刪除，且將 man 取代成 SSEEDD。

(c) 使用 sed 處理 man.conf 檔，將以下 sed 要執行的動作寫在 src 檔，再寫腳本執行 sed 時以 -f 選項指定 src 檔，執行腳本將結果導向到「輸出檔案」，不要做任何的更動。

i. 腳本 sed2.1.sh，src 檔 sed2.1.src，輸出檔案 sed2.1.txt

A. 刪除 5 至 6 行。

B. 第 33 行後增加二行。

```
1 BBKJH983
  Capem34kq98
```

C. 第 40 至 45 行取代成：

```
2 abcdefgh
  9876543
  3333333
```

ii. 腳本 sed2.2.sh，src 檔 sed2.2.src，輸出檔案 sed2.2.txt

A. 列出第 3 至 142 行。

B. 刪除註解行及空白行。

C. 第 102 行至最後一行刪除

D. 將 man 取代成 SSEEDD。

(d) 下載檔案 re3.txt，撰寫以下腳本：

i. re3.txt 的算術式格式：(a,b,X)，其中 a b 是正負數字，X 是 +-* / 加減乘除運算子。

ii. sed3.1.sh 處理 re3.txt 將 (a,b,X) 取代成 aXb=，結果導向到 sed3.1.txt。

- iii. sed3.2.sh 處理 sed3.1.txt，將算術式中減負數 (--)，取代成 +，結果導向到 sed3.2.txt。
 - iv. sed3.3 處理 sed3.2.txt，將算術式中加負數 (+-)，取代成 -，結果導向到 sed3.3.txt。
 - v. sed3.sh 一次處理 re3.txt，完成 sed3.1.sh sed3.2.sh sed3.3.sh 所做動作，結果存成 sed3.txt，不要做任何的更動。
- (e) 下載檔案 gradelist.txt，撰寫以下腳本：
- i. 使用 sed, tr 命令，配合正規表示法、管線處理。
 - ii. sed4.1.sh 處理 gradelist.txt，轉換成字典陣列需要的格式，名字及成績都用雙引號括起來，結果導向到 sed4.1.txt。
 - iii. sed4.2.sh 處理 sed4.1.txt，tr 命令將換行符號改成空白字元，結果導向到 sed4.2.txt。
 - iv. sed4.3.sh 處理 sed4.2.txt，sed 命令將行尾的空白字元改成換行符號，結果導向到 sed4.3.txt。
 - v. sed4.sh 處理 gradelist.txt，完成 sed4.1.sh sed4.2.sh sed4.3.sh 所有動作，輸出轉換後的結果導向到 sed4.txt，輸出範例如下：

```
1 ["dywang"]="60" ["linda"]="80" ["peter"]="92" ["rita"]="87"
```

Chapter 7

awk 工具

7.1 awk 命令簡介

1. awk 相較於 sed 作用於一整個行的處理，awk 是『以行為一次處理的單位』，而『以欄位為最小的處理單位』。awk 命令輔助說明如下：

```
1 [dywang@dywmac zzz]$ awk --help
Usage: awk [POSIX or GNU style options] -f progfile [--] file ...
3 Usage: awk [POSIX or GNU style options] [--] 'program' file ...
POSIX options:      GNU long options:
5   -f progfile      --file=progfile
   -F fs             --field-separator=fs
7   -v var=val       --assign=var=val
   -m[fr] val
9   -O               --optimize
   -W compat         --compat
11  -W copyleft      --copyleft
   -W copyright      --copyright
13  -W dump-variables[=file] --dump-variables[=file]
   -W exec=file      --exec=file
15  -W gen-po        --gen-po
   -W help          --help
17  -W lint[=fatal]  --lint[=fatal]
   -W lint-old       --lint-old
19  -W non-decimal-data --non-decimal-data
   -W profile[=file] --profile[=file]
21  -W posix         --posix
   -W re-interval    --re-interval
23  -W source=program-text --source=program-text
   -W traditional    --traditional
25  -W usage         --usage
   -W use-lc-numeric --use-lc-numeric
27  -W version       --version

29 To report bugs, see node 'Bugs' in 'gawk.info', which is
   section 'Reporting Problems and Bugs' in the printed version.
31
```

```

33 gawk is a pattern scanning and processing language.
    By default it reads standard input and writes standard output.
35 Examples:
    gawk '{ sum += $1 }; END { print sum }' file
37 gawk -F: '{ print $1 }' /etc/passwd

```

2. `awk --dump-variables dump` 所有變數，指定的檔案空字串，則預設存在 `awkvars.out`，其中 `FS: string (" ")` 表示欄位分隔符號為空白 " "。

```

1 [dywang@dywmac ~]$ awk --dump-variables ''
  [dywang@dywmac ~]$ cat awkvars.out
3 ARGV: array, 1 elements
  ARGIND: number (0)
5 ARGV: array, 1 elements
  BINMODE: number (0)
7 CONVFMT: string ("%0.6g")
  ERRNO: number (0)
9 FIELDWIDTHS: string ("")
  FILENAME: string ("")
11 FNR: number (0)
  FS: string (" ")
13 IGNORECASE: number (0)
  LINT: number (0)
15 NF: number (0)
  NR: number (0)
17 OFMT: string ("%0.6g")
  OFS: string (" ")
19 ORS: string ("\n")
  RLENGTH: number (0)
21 RS: string ("\n")
  RSTART: number (0)
23 RT: string ("")
  SUBSEP: string ("\034")
25 TEXTDOMAIN: string ("messages")

```

3. 查看 `awk` 命令連結到 `gawk`，也就是在 Linux 上 `awk` 是 `gawk`。

```

1 [dywang@dywmac zzz]$ ll /usr/bin/awk
lrwxrwxrwx. 1 root root 14 Jun  2  2018 /usr/bin/awk -> ../../bin/gawk

```

4. `gawk --version` (GNU Awk) 版本宣告，是自由軟體。

```

2 [dywang@dywmac zzz]$ gawk -W version
GNU Awk 3.1.7
Copyright (C) 1989, 1991-2009 Free Software Foundation.

```

```
4 | This program is free software; you can redistribute it and/or modify
6 | it under the terms of the GNU General Public License as published by
8 | the Free Software Foundation; either version 3 of the License, or
   | (at your option) any later version.
10 | This program is distributed in the hope that it will be useful,
   | but WITHOUT ANY WARRANTY; without even the implied warranty of
12 | MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
   | GNU General Public License for more details.
14 |
16 | You should have received a copy of the GNU General Public License
   | along with this program. If not, see http://www.gnu.org/licenses/.
```

7.2 awk 範例一

1. vim 編輯 sdata.txt 包含五筆資料，每筆資分別有「編號姓名成績 1 成績 2 成績 3」三個欄位，每個欄位以空白隔開。

```
2 | [dywang@dywmac zzz]$ vim sdata.txt
   | [dywang@dywmac zzz]$ cat sdata.txt
   | 11 dywang 81 12 A
   | 4 152 linda 90 58 C
   | 33 peter 72 95 C
   | 6 4 rita 65 34 E
   | 58 cora 5 85 D
```

2. gawk 查看輔助說明的最後一段，有兩個範例。

```
1 | [dywang@dywmac zzz]$ gawk -W help | grep ^Example -A2
   | Examples:
3 |     gawk '{ sum += $1 }; END { print sum }' file
   |     gawk -F: '{ print $1 }' /etc/passwd
```

3. 使用第一個範例，計算 sdata.txt 中「成績 1」的總和。

```
2 | [dywang@dywmac zzz]$ gawk '{sum+=$3}; END {print sum}' sdata.txt
   | 313
```

4. 延續第一個範例，計算 sdata.txt 中「成績 1」的平均。

```

[dywang@dywmac zzz]$ gawk '{sum+=$3; count+=1}; END {print sum/count}'
sdata.txt
2 62.6

```

5. 沒有 END，不等到 sum 加總完成，所有動作皆連續執行，所以可以只用一個大括號，每行皆列出「筆數姓名成績 1 目前筆數平均成績」，每個欄位皆以 "\t" TAB 定位隔開，整齊輸出。

```

[dywang@dywmac zzz]$ gawk '{ sum += $3; count+=1; \
2 print count "\t" $2 "\t" $3 "\t" sum "\t" sum/count}' sdata.txt
1 dywang 81 81 81
4 2 linda 90 171 85.5
3 peter 72 243 81
6 4 rita 65 308 77
5 cora 5 313 62.6

```

6. awk 常用內建變數

變數名稱	代表意義
NF	每一行 (\$0) 擁有的欄位總數
\$NF	每一行的最後一個欄位內容
NR	目前 awk 所處理的是『第幾行』資料
FS	目前的分隔字元，預設是空白鍵

7. 查詢內建變數預設值。

```

1 [dywang@dywmac zzz]$ egrep 'NF|NR|FS' ../awkvars.out
FNR: number (0)
3 FS: string (" ")
NF: number (0)
5 NR: number (0)
OFS: string (" ")

```

8. 承上例：count 變數用 NR 取代。

```

[dywang@dywmac zzz]$ gawk '{ sum += $3; \
2 print NR "\t" $2 "\t" $3 "\t" sum "\t" sum/NR}' sdata.txt
1 dywang 81 81 81
4 2 linda 90 171 85.5
3 peter 72 243 81
6 4 rita 65 308 77
5 cora 5 313 62.6

```

7.3 awk 範例二

1. gawk 查看輔助說明的最後一段，有兩個範例。

```
1 [dywang@dywmac zzz]$ gawk -W help | grep ^Example -A2
Examples:
3   gawk '{ sum += $1 }; END { print sum }' file
   gawk -F: '{ print $1 }' /etc/passwd
```

2. 使用第二範例：`/etc/passwd` 以冒號`:`作為欄位的分隔，列出第一欄位 (帳號名稱) 及第三欄位 (uid)。

```
2 [dywang@dywmac zzz]$ gawk -F: '{print $1 "\t" $3}' /etc/passwd | head -4
root    0
bin 1
4 daemon 2
adm 3
```

3. 如果不使用 `-F` 選項指定 `passwd` 以冒號`:`作為欄位的分隔，而是在單引號內，也就是 `awk` 程式本身設定分隔符號，則執行結果如下，第一行還是以「空白」做為分隔符號，所以無法正確輸出。

```
1 [dywang@dyw219 zzz]$ gawk 'FS=":" {print $1 "\t" $3}' /etc/passwd | head
-4
root:x:0:0:root:/root:/bin/bash
3 bin 1
daemon 2
5 adm 3
```

4. `awk` 的 `END` 是指先執行所有行數，再進行下一個動作，`BEGIN` 則是先做設定或處理再執行下一個動作。所以 `FS=":"` 的設定以 `BEGIN` 先設定就可以解決第一行輸出不正確的問題。

```
1 [dywang@dyw219 zzz]$ gawk 'BEGIN{FS=":"} {print $1 "\t" $3}' /etc/passwd
| head -4
root    0
3 bin 1
daemon 2
5 adm 3
```

5. awk 的邏輯運算字元

運算單元	代表意義
>	大於
<	小於
>=	大於或等於
<=	小於或等於
==	等於
!=	不等於

6. 查閱第三欄 uid 大於等於 500 的帳號

```

1 [dywang@dywmac zzz]$ gawk 'BEGIN{FS=":"} $3<=4 {print $1 "\t" $3}' /etc/
  passwd
root    0
3 bin    1
daemon  2
5 adm    3
lp      4

```

7. 查閱第三欄 uid 大於等於 500 的帳號

```

[dywang@dywmac zzz]$ gawk 'BEGIN{FS=":"} $3>=500 {print $1 "\t" $3}' /
etc/passwd
2 nfsnobody 65534
dywang     500
4 mockbuild 502
linda      503

```

8. 「查閱第三欄 uid 大於等於 500 的帳號」的 awk 程式寫在 passwd.awk。

```

1 [dywang@dywIssd zzz]$ vim passwd.awk
[dywang@dywIssd zzz]$ cat passwd.awk
3 BEGIN{
  FS=":"
5 }
$3>=500 {
7   print $1 "\t" $3
}

```

9. 執行 awk 以選項指定程式在 passwd.awk，一樣可以達到要求。

```

[dywang@dywIssd zzz]$ awk -f passwd.awk /etc/passwd
2 nfsnobody 65534

```

```

4 dywang 500
  linda 501

```

7.4 awk 範例三

1. 編輯 sdata.txt，第一行加上欄位抬頭。

```

[dywang@dywmac zzz]$ vim sdata.txt
2 [dywang@dywmac zzz]$ cat sdata.txt
no. name score1 score2 grade
4 11 dywang 81 12 A
  152 linda 90 58 C
6 33 peter 72 95 C
  4 rita 65 34 E
8 58 cora 5 85 D

```

2. 如果第一行不做特別處理，平均會多算一行，且第一行的後兩個欄位，抬頭都是 0。

```

[dywang@dywmac zzz]$ gawk '{ sum += $3; \
2 print NR "\t" $2 "\t" $3 "\t" sum "\t" sum/NR}' sdata.txt
1 name score1 0 0
4 2 dywang 81 81 40.5
  3 linda 90 171 57
6 4 peter 72 243 60.75
  5 rita 65 308 61.6
8 6 cora 5 313 52.1667

```

3. 大括號 { } 內動作加 if 條件，將一列個別處理。

```

[dywang@dywmac zzz]$ gawk '{if(NR==1) print "count\tname\tscore1\tsum\t
2   avg."} \
{sum += $3; if(NR>=2) print NR-1 "\t" $2 "\t" $3 "\t" sum "\t" sum/(NR
  -1}}' sdata.txt
count name score1 sum avg.
4 1 dywang 81 81 81
  2 linda 90 171 85.5
6 3 peter 72 243 81
  4 rita 65 308 77
8 5 cora 5 313 62.6

```

4. 改成列印每個人的兩項成績，並計算其平均值。


```

[dywang@dywmac zzz]$ gawk '{if(NR==1) print "count\tname\tscore1\tscore2\tavg."} \
2 {sum = $3+$4; if(NR>=2) print NR-1 "\t" $2 "\t" $3 "\t" $4 "\t" sum/2}'
  sdata.txt
count  name    score1 score2  avg.
4 1   dywang   81    12  46.5
  2   linda   90    58  74
6 3   peter   72    95  83.5
  4   rita    65    34  49.5
8 5   cora     5    85  45

```

5. 若 $NR==1$, $NR>=2$ 的判斷放在大括號 `{}` 動作外，則表示「條件」，不用再用 `if` 處理。

```

[dywang@dywmac zzz]$ gawk 'NR==1 {print "count\tname\tscore1\tscore2\tavg."} \
2 NR>=2 {sum = $3+$4; print NR-1 "\t" $2 "\t" $3 "\t" $4 "\t" sum/2}'
  sdata.txt
count  name    score1 score2  avg.
4 1   dywang   81    12  46.5
  2   linda   90    58  74
6 3   peter   72    95  83.5
  4   rita    65    34  49.5
8 5   cora     5    85  45

```

7.5 awk 與正規表示法

1. `awk` 配合正規表示法進行搜尋 `sed '/pattern/' filename`，找到符合 `pattern` 的資料。例如：找檔案 `/etc/man.config` 中含 `nzip` 及 `bzip` 的資料。

```

[dywang@dywmac zzz]$ grep -n '[nb]zip' /etc/man.config
2 136:.gz      /usr/bin/gunzip -c
  137:.bz2     /usr/bin/bzip2 -c -d
4 [dywang@dywmac zzz]$ awk '/[nb]zip/' /etc/man.config
  .gz      /usr/bin/gunzip -c
6  .bz2     /usr/bin/bzip2 -c -d

```

2. `awk` 找到 `/etc/man.config` 行首是 `C` 的資料。

```

[dywang@dywmac zzz]$ awk '/^C/' /etc/man.config
2 CAT      /bin/cat
  CMP      /usr/libexec/man-cmp.sh

```

```
4 | COMPRESS    /usr/bin/lzma
   | COMPRESS_EXT .lzma
```

3. awk 從 sdata.txt 中找到名字是 d 或 p 開頭的資料，印出名字及成績。

```
1 | [dywang@dywmac zzz]$ cat sdata.txt
   | no. name score1 score2 grade
3 | 11 dywang 81 12 A
   | 152 linda 90 58 C
5 | 33 peter 72 95 C
   | 4 rita 65 34 E
7 | 58 cora 5 85 D
   | [dywang@dywmac zzz]$ awk '/\ [dp]/ {print $2 "\t" $5}' sdata.txt
9 | dywang A
   | peter C
```

4. awk 從 sdata.txt 中找到成績是 A, B 或 C 的資料，算出數量並印出總數。

```
1 | [dywang@dywmac zzz]$ awk '/[ABC]/ {count+=1} END {print "count="count}'
   | sdata.txt
2 | count=3
```

7.6 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 zzz 目錄。
- (b) 切換工作目錄到 zzz，所有練習腳本皆存在此目錄。
- (c) 腳本必須可執行。
- (d) .txt 檔案都存到 /tmp/\$mysid 目錄，其中變數 \$mysid 為自己的學號，目錄若不存在，必須自行建立。

2. 下載檔案 awk-a.csv，以 awk 處理 awk-a.csv 檔，第一欄位 no，第二欄位 name，第三欄位是 score1，第四欄位是 score2，第五欄位是 grade，撰寫以下腳本：

- (a) awk-a1.sh：計算 awk-a.csv 檔中所有學生的 score1 平均分數及 score2 的平均分數，輸出只列出所有學生 score1 平均分數及所有學生 score2 的平均分數，中間用 TAB 隔開，存成 awk-a1.txt。

- (b) `awk-a2.sh`：計算 `awk-a.csv` 檔中學生的 `score1` 及 `score2` 的平均分數，列出「目前處理第幾行 TAB 姓名 TAB 目前 `score1` 平均 TAB 目前 `score2` 平均」，存成 `awk-a2.txt`。
 - (c) `awk-a3.sh`：找到 `no` 大於 10 的學生，列印「目前處理第幾行 TAB `grade` TAB 姓名 TAB `score1` 與 `score2` 平均值」，存成 `awk-a3.txt`。
 - (d) `awk-a4.sh`：配合正規表示式找到名字是 `r` 開頭的學生，列印「目前處理第幾行 TAB `grade` TAB 姓名 TAB `score1` 與 `score2` 平均值」，存成 `awk-a4.txt`。
3. 下載檔案 `awk-b.csv`，以 `awk` 處理 `awk-b.csv` 檔，第一行是各欄位名稱，不算筆數而且必須特別處理，撰寫以下腳本：
- (a) `awk-b1.sh`：計算 `awk-b.csv` 檔中所有學生的 `score1` 平均分數，及 `score2` 的平均分數，只列出 `score1` 平均分數及 `score2` 的平均分數，中間用 TAB 隔開，第一行列印「平均一 TAB 平均二」，存成 `awk-b1.txt`。
 - (b) `awk-b2.sh`：計算 `awk-b.csv` 檔中學生的 `score1` 及 `score2` 的平均分數，列出「目前筆數 TAB 姓名 TAB 目前 `score1` 平均 TAB 目前 `score2` 平均」，第一行列出「筆數 TAB 姓名 TAB 累計平均 1 TAB 累計平均 2」，存成 `awk-b2.txt`。
 - (c) `awk-b3.sh`：找到 `no` 大於 10 的學生，列印「目前筆數 TAB `grade` TAB 姓名 TAB `score1` 與 `score2` 平均值」，第一行列出「筆數 TAB 成績 TAB 姓名 TAB 兩科平均」，存成 `awk-b3.txt`。
 - (d) `awk-b4.sh`：配合正規表示式找到名字是 `r` 開頭的學生，列印「目前筆數 TAB `grade` TAB 姓名 TAB `score1` 與 `score2` 平均值」，第一行列出「筆數 TAB 成績 TAB 姓名 TAB 兩科平均」，存成 `awk-b4.txt`。
4. 下載檔案 `passwd`，以 `awk` 處理 `passwd` 檔，`passwd` 檔的以冒號`:`分隔欄位，撰寫以下腳本：
- (a) `passwd1.sh`：列印第 1 及 3 欄位，中間以 TAB 分隔，存成 `passwd1.txt`。
 - (b) `passwd2.sh`：列出 `UID` 大於 10 且小於 100 的帳號，印出格式為「目前處理的行數 TAB 帳號名稱 TAB `UID` TAB 該行有多少欄位」，存成 `passwd2.txt`。
 - (c) `passwd3.sh`：查閱第 3 欄位小於 7 以下的數據，並且僅列出第 3, 1, 5 欄位，中間以 TAB 分隔，但是分隔符號的設定必須在 `awk` 程式本身，而不是使用 `-F` 選項，觀察其第一行輸出，存成 `passwd3.txt`。

- (d) passwd4.sh : 同上題，但利用關鍵字 BEGIN 讓第一行正確顯示，存成 passwd4.txt。
- (e) passwd5.sh : 將 passwd4.sh 的 awk 程式寫在 passwd5.awk，passwd5.sh 以 awk -f 選項指定執行 passwd.awk 完成上一題的要求，結果存成 passwd5.txt。

Chapter 8

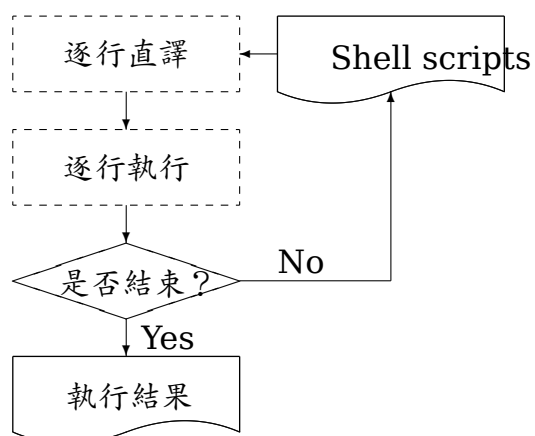
Shell Scripts

8.1 前言

1. 何謂 Shell Script

- (a) 文字介面下讓使用者與系統溝通的一個工具介面。
- (b) 利用 shell 的功能所寫的一支『程式 (program)』。
- (c) 將一些 shell 的語法與指令寫成純文字檔；
- (d) 可搭配正規表示法、管線命令與資料流重導向等功能；
- (e) shell script 更提供陣列、迴圈、條件與邏輯判斷等重要功能；
- (f) 使用者也可以直接以 shell 來撰寫程式。

2. Shell scripts 執行流程：



3. 為什麼要學習 shell scripts?

- (a) 自動化管理的重要依據：自動處理分析主機狀態，若有問題才通知。

- (b) 追蹤與管理系統的重要工作：Linux 系統的服務 (services) 啟動的介面，在目錄 `/etc/init.d/` 下，所有的檔案都是 scripts。
- (c) 簡單入侵偵測功能：主動分析系統登錄檔。
- (d) 連續指令單一化：彙整在 `command line` 下達的連續指令。例如：`/etc/rc.d/rc.local` 裡的資料。
- (e) 簡易的資料處理：處理數據資料的比對，文字資料的處理等。
- (f) 跨平台支援與學習歷程較短：幾乎所有的 Unix Like 上面都可以跑 shell script。
- (g) shell script 是一項很好的系統管理工具，但數值運算的速度較慢，且使用的 CPU 資源較多。

4. Scripts 撰寫注意事項：

- (a) 指令與參數間的多個空白會被忽略掉；
- (b) 空白行及 `[tab]` 不會被理會；
- (c) 讀到 `[Enter]` 符號，就開始執行該行命令；
- (d) 一行的內容太多，可以使用 `\[Enter]` 來延伸至下一行；
- (e) 加在 `#` 後面的字，全部被視為註解。

5. 如何執行檔案 `shell.sh` ？

- (a) 將 `shell.sh` 加上可讀與執行 (rx) 的權限，就能夠以 `./shell.sh` 執行；
- (b) 直接以 `sh shell.sh` 方式執行。

8.2 第一支腳本

1. 建立撰寫 shell script 的良好習慣，在每個 script 的檔頭記錄下列資訊，以助程式的改寫與 debug：

- (a) 功能；
- (b) 版本資訊；
- (c) 作者與聯絡方式；
- (d) 版權宣告方式；
- (e) History (歷史紀錄)；
- (f) script 內較特殊的指令，使用絕對路徑的方式來下達；

(g) script 運作時需要的環境變數預先宣告與設定。

2. 撰寫第一支 script 印出"Hello World"

```
[dywang@dywmac zzz]$ cd
2 [dywang@dywmac ~]$ cd zzz
[dywang@dywmac zzz]$ vim sh01.sh
4 [dywang@dywmac zzz]$ cat sh01.sh
#!/bin/bash
6 # Program:
#   This program is used to show "Hello World !" in screen.
8 # History:
# 2019/04/25 dywang First release
10
12 echo -e "Hello World ! \a \n"
exit 0
```

3. 腳本說明：

- (a) 程式內容的宣告：script 當中，除了第一行的 #! 是用來宣告 shell 外，其他的 # 都是『註解』。
- (b) 主要環境變數的宣告：例如 PATH。
- (c) 主要程式部分：echo 選項 -e 代表解釋倒斜線 \ 跳脫字元。 \a 為警告； \n 為換行。
- (d) 執行成果告知：
 - i. 可以利用指令 exit 讓程式中斷，並且回傳一個數值給系統。
 - ii. 在此例中使用 exit 0，代表離開 script，並且回傳 0 給系統。
 - iii. 指令執行成功與否，可下達 echo \$? 觀察回傳值。
 - iv. 利用 exit n 的功能，可以自訂錯誤訊息。

4. 使用 sh 執行 sh01.sh

```
[dywang@dywmac zzz]$ sh sh01.sh
2 Hello World !
```

5. 成功執行 sh01.sh，回傳值 0。

```
1 [dywang@dywmac zzz]$ echo $?
0
```

6. 設定 sh01.sh 為可執行檔，直接執行 sh01.sh，此時會以腳本中第一行宣告的命令 /bin/bash 執行。

```
[dywang@dywmac zzz]$ chmod +x sh01.sh
2 [dywang@dywmac zzz]$ ./sh01.sh
Hello World !
```

8.3 Shell 是一種程式語言

1. 將一段在 shell 下的命令，集結成一個檔案來執行，就是一支腳本。
2. 例如：在 shell 用 cat 將螢幕輸入的資料存成 shdata1.txt，再用 awk 算出總和並印出 total= 總和，其中 << EOF 表示輸入 EOF 時結束輸入。

```
[dywang@dywmac zzz]$ cat > shdata1.txt << EOF
2 > 23
> 43
4 > 156
> 32
6 > EOF
[dywang@dywmac zzz]$ cat shdata1.txt
8 23
43
10 156
32
12 [dywang@dywmac zzz]$ awk '{sum+=$1} END {print "total=sum}" shdata1.txt
total=254
```

3. 將以上幾行命令寫成腳本 sh02.sh。

```
1 [dywang@dywmac zzz]$ vim sh02.sh
[dywang@dywmac zzz]$ cat sh02.sh
3 #!/bin/bash
cat > shdata1.txt << EOF
5 23
43
7 156
32
9 EOF
11 cat shdata1.txt
awk '{sum+=$1;} END {print "total=sum}" shdata1.txt
13 exit 0
```


4. 執行腳本 sh02.sh，結果與螢幕輸入的一樣。

```
1 [dywang@dywmac zzz]$ sh sh02.sh
23
3 43
156
5 32
total=254
```

8.4 shell script 練習

1. 以 read 指令讀入一串數字，使用命令 sed 將數字中間的空白取代成加號"+", 再管線命令用 bc 計算這算術式的結果。

```
[dywang@dywmac zzz]$ cat sh03.sh
2 #!/bin/bash
read -p "請輸入一串數字： " -t 5 num
4 num=$(echo "$num" | sed 's/ /+/g')
total=$(echo "$num" | bc)
6 echo $num=$total
exit 0
```

2. 執行結果，輸入四筆或三筆數字都成功計算總和並輸出。

```
1 [dywang@dywmac zzz]$ sh sh03.sh
請輸入一串數字： 12 34 2 4
3 12+34+2+4=52
[dywang@dywmac zzz]$ sh sh03.sh
5 請輸入一串數字： 1 3 5 7
1+3+5+7=16
```

3. 執行結果，沒輸入時等號兩邊的變數都是空字串。

```
[dywang@dywmac zzz]$ sh sh03.sh
2 請輸入一串數字： =
```

4. 利用變數設定 {num:-expr}，當 num 沒設定或空字串時，num 變數就設成 expr，以避免使用者沒輸入的狀況。

```

[dywang@dywmac zzz]$ cat sh03.sh
2  #!/bin/bash
4  read -p "請輸入一串數字： " -t 5 num
6  num=${num:-"1 2 3 4 5 6 7 8 9"}
7  num=$(echo "$num" | sed 's/ /+/g')
8  total=$(echo "$num" | bc)
9  echo $num=$total
10 exit 0

```

5. 執行結果，沒輸入時自動加總 1 到 9 等於 45。

```

1 [dywang@dywmac zzz]$ sh sh03.sh
   請輸入一串數字： 1+2+3+4+5+6+7+8+9=45

```

8.5 set 命令

1. set 指令用來修改 shell 環境變數，是 Bash 腳本的重要工具，適當的使用可以增加腳本的安全性和可維護性。set 不加任何選項會列出所有環境變數及 shell 函式。

```

[dywang@dywmac zzz]$ set | grep ^H
2 HISTCONTROL=ignoredups
3 HISTFILE=/home/dywang/.bash_history
4 HISTFILESIZE=1000
5 HISTSIZE=1000
6 HOME=/home/dywang
7 HOSTNAME=dywmac
8 HOSTTYPE=x86_64

```

2. set 是 shell 內建的命令，要用 sh -c "help set" 查看輔助說明，選項說明如下：

```

[dywang@dywmac zzz]$ sh -c "help set"
2 set: set [--abefhkmnptuvxBCHP] [-o option-name] [arg ...]
   Set or unset values of shell options and positional parameters.
4
   Change the value of shell attributes and positional parameters, or
6   display the names and values of shell variables.
8
   Options:
   -a Mark variables which are modified or created for export.

```

```

10  -b Notify of job termination immediately.
    -e Exit immediately if a command exits with a non-zero status.
12  -f Disable file name generation (globbing).
    -h Remember the location of commands as they are looked up.
14  -k All assignment arguments are placed in the environment for a
    command, not just those that precede the command name.
16  -m Job control is enabled.
    -n Read commands but do not execute them.
18  -o option-name
    Set the variable corresponding to option-name:
20      allexport      same as -a
        braceexpand  same as -B
22      emacs         use an emacs-style line editing interface
        errexit      same as -e
24      errtrace      same as -E
        functrace    same as -T
26      hashall       same as -h
        histexpand   same as -H
28      history        enable command history
        ignoreeof    the shell will not exit upon reading EOF
30      interactive-comments
                        allow comments to appear in interactive
                        commands
32      keyword        same as -k
        monitor      same as -m
34      noclobber      same as -C
        noexec       same as -n
36      noglob         same as -f
        nolog        currently accepted but ignored
38      notify         same as -b
        nounset      same as -u
40      onecmd         same as -t
        physical     same as -P
42      pipefail       the return value of a pipeline is the status
                        of
                        the last command to exit with a non-zero
                        status,
44                        or zero if no command exited with a non-zero
                        status
        posix        change the behavior of bash where the default
46                        operation differs from the Posix standard to
                        match the standard
        privileged   same as -p
48      verbose        same as -v
        vi          use a vi-style line editing interface
50      xtrace         same as -x
52  -p Turned on whenever the real and effective user ids do not
    match.
    Disables processing of the $ENV file and importing of shell
54  functions. Turning this option off causes the effective uid
    and
    gid to be set to the real uid and gid.
56  -t Exit after reading and executing one command.
    -u Treat unset variables as an error when substituting.

```

```

58      -v Print shell input lines as they are read.
      -x Print commands and their arguments as they are executed.
60      -B the shell will perform brace expansion
      -C If set, disallow existing regular files to be overwritten
62      by redirection of output.
      -E If set, the ERR trap is inherited by shell functions.
64      -H Enable ! style history substitution. This flag is on
      by default when the shell is interactive.
66      -P If set, do not follow symbolic links when executing commands
      such as cd which change the current directory.
68      -T If set, the DEBUG trap is inherited by shell functions.
      - Assign any remaining arguments to the positional parameters.
70      The -x and -v options are turned off.

72      Using + rather than - causes these flags to be turned off. The
      flags can also be used upon invocation of the shell. The current
74      set of flags may be found in $-. The remaining n ARGs are
      positional
      parameters and are assigned, in order, to $1, $2, .. $n. If no
76      ARGs are given, all shell variables are printed.

78      Exit Status:
      Returns success unless an invalid option is given.
80
      選項說明：
82      -a 標示已修改的變數，以供輸出至環境變數。
      -b 使被中止的後台程序立刻回報執行狀態。
84      -e 若指令傳回值不等於 0，則立即退出 shell。
      -H Shell 可利用"!"+<指令編號>的方式來執行 history 中記錄的指令。
86      -k 指令所給的參數都會被視為此指令的環境變數。
      -l 記錄 for 迴圈的變數名稱。
88      -m 使用監視模式。
      -n 不要執行 script，僅查詢語法的问题。
90      -p 啟動優先順序模式。
      -P 啟動 -P 選項後，執行指令時，會以實際的文件或目錄來取代符號連
      接。
92      -t 執行完隨後的指令，即退出 shell。
      -u 當執行時使用到未定義過的變數，則顯示錯誤信息。
94      -v 執行 sccript 前，先將 script 的內容輸出到螢幕上；
      -x 執行指令前，先顯示該指令及所下的選項。

```

8.6 set 基本用法

1. 執行腳本 set1.sh 時，遇到不存在的變數 abc，bash 預設忽略它，繼續往下執行。

```

1 [dywang@dywmac zzz]$ vim set1.sh
  [dywang@dywmac zzz]$ cat set1.sh
3 #!/bin/bash

```

```
echo $abc
5 echo $USER
[dywang@dywmac zzz]$ chmod +x set1.sh
7 [dywang@dywmac zzz]$ ./set1.sh
9 dywang
```

2. 在腳本 set2.sh 加上 set -u，遇到不存在的變數 abc，腳本會報錯並停止執行。

```
1 [dywang@dywmac zzz]$ vim set2.sh
[dywang@dywmac zzz]$ cat set2.sh
3 #!/bin/bash
set -u
5 echo $abc
echo $USER
7 [dywang@dywmac zzz]$ ./set2.sh
./set2.sh: line 3: abc: unbound variable
```

3. 執行腳本時只螢幕輸出執行結果，無法得知是那行命令執行的？set3.sh 加上 set -x，就可以先印命令再輸出結果，以利追蹤及除錯腳本，命令行開頭都以加號“+”開頭。

```
[dywang@dywmac zzz]$ vim set3.sh
2 [dywang@dywmac zzz]$ cat set3.sh
#!/bin/bash
4 set -x
echo $abc
6 echo $USER
[dywang@dywmac zzz]$ ./set3.sh
8 + echo
10 + echo dywang
dywang
```

4. 腳本 sh03.sh 加入 set -t 及 echo "set -t test"。

```
1 [dywang@dywmac zzz]$ cat sh03.sh
#!/bin/bash
3 read -p "請輸入一串數字： " -t 5 num

5 num=${num:-"1 2 3 4 5 6 7 8 9"}
num=$(echo "$num" | sed 's/ /+/g')
7 total=$(echo "$num" | bc)
echo $num=$total
9 set -t
```

```

11 echo "set -t test"
    exit 0

```

5. 執行到 `set -t` 後退出腳本，所以 `echo "set -t test"` 沒輸出。

```

1 [dywang@dywmac zzz]$ sh sh03.sh
  請輸入一串數字： 1+2+3+4+5+6+7+8+9=45

```

8.7 追蹤與除錯

1. 在腳本中可以用 `set` 命令來追蹤除錯腳本。`sh` 執行腳本時一樣可以使用 `set` 選項進行追蹤除錯。複習幾個 `set` 選項：

```

2 選項說明：
   -n  不要執行 script，僅查詢語法的問題。
   -t  執行完隨後的指令，即退出 shell。
4  -u  當執行時使用到未定義過的變數，則顯示錯誤信息。
   -v  執行 sccript 前，先將 script 的內容輸出到螢幕上；
6  -x  執行指令前，先顯示該指令及所下的選項。

```

2. `sh03.sh` 第 4 行刪除一個雙引號，存成 `sh03-1.sh`。

```

2 [dywang@dywmac zzz]$ cat -n sh03-1.sh
   1  #!/bin/bash
   2  read -p "請輸入一串數字： " -t 5 num
4   3
   4  num=${num:-"1 2 3 4 5 6 7 8 9"}
6   5  num=$(echo "$num" | sed 's/ /+/g')
   6  total=$(echo "$num" | bc)
8   7  echo $num=$total

```

3. `sh -n` 測試 `sh03-1.sh`，出現第 4 行找不到配對的雙引號，第 8 行檔案結束語法錯誤。

```

2 [dywang@dywmac zzz]$ sh -n sh03-1.sh
sh03-1.sh: line 4: unexpected EOF while looking for matching `"'
sh03-1.sh: line 8: syntax error: unexpected end of file

```

4. `sh -v` 執行 `sh03.sh`，每行執行前會先將執行命令內容輸出到螢幕上。

```

1 [dywang@dywmac zzz]$ sh -v sh03.sh
  #!/bin/bash
3 read -p "請輸入一串數字： " -t 5 num
  請輸入一串數字：
5 num=${num:-"1 2 3 4 5 6 7 8 9"}
  num=$(echo "$num" | sed 's/ /+/g')
7 total=$(echo "$num" | bc)
  echo $num=$total
9 1+2+3+4+5+6+7+8+9=45
  exit 0

```

5. `sh -x` 執行 `sh03.sh`，執行過程會列印出來，且會在行首加上`+` 號表示程式段落。

```

2 [dywang@dywmac zzz]$ sh -x sh03.sh
+ read -p '請輸入一串數字： ' -t 5 num
  請輸入一串數字： + num='1 2 3 4 5 6 7 8 9'
4 ++ echo '1 2 3 4 5 6 7 8 9'
  ++ sed 's/ /+/g'
6 + num=1+2+3+4+5+6+7+8+9
  ++ echo 1+2+3+4+5+6+7+8+9
8 ++ bc
  + total=45
10 + echo 1+2+3+4+5+6+7+8+9=45
  1+2+3+4+5+6+7+8+9=45
12 + exit 0

```

8.8 腳本預設參數

1. Shell 腳本執行時參數的預設變數為 `$0`，`$1`，`$2`，...，對應如下：

```

2 /path/to/scriptname  opt1  opt2  opt3  opt4  ...
  $0                  $1    $2    $3    $4    ...

```

2. `sh04.sh` 列出執行命令名稱、前三個參數：

```

2 [dywang@dywmac zzz]$ cat sh04.sh
  #!/bin/bash
  echo "The script name is ==> $0"
4 [ -n "$1" ] && echo "The 1st paramter is ==> $1" || exit 0
  [ -n "$2" ] && echo "The 2nd paramter is ==> $2" || exit 0

```

```
6 [ -n "$3" ] && echo "The 3th paramter is ==> $3" || exit 0
```

3. 執行 sh04.sh 加三個參數，列出執行命令名稱及三個參數：

```
2 [dywang@dywmac zzz]$ sh sh04.sh abc 123 cde
The script name is ==> sh04.sh
The 1st paramter is ==> abc
4 The 2nd paramter is ==> 123
The 3th paramter is ==> cde
```

4. 執行 sh04.sh 加兩個參數，列出執行命令名稱及兩個參數後退出腳本。

```
1 [dywang@dywmac zzz]$ sh sh04.sh abc 123
The script name is ==> sh04.sh
3 The 1st paramter is ==> abc
The 2nd paramter is ==> 123
```

8.9 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 zzz 目錄。
- (b) 切換工作目錄到 zzz，所有練習腳本皆存在此目錄。
- (c) 腳本必須宣告執行命令 /bin/bash，且可執行。

2. 寫一腳本 script1.sh，要求如下：

- (a) 腳本版本宣告「日期學號第一版」，日期格式：yyyymmdd，學號不加 s。
- (b) 列印 mysid=xxxxxxxx，其中 xxxxxxxx 是您的學號。
- (c) 成功執行後回傳 0。

3. 寫一腳本 script2.sh，要求如下：

- (a) read 讀入一數字 nu。
- (b) cat 將「12 34 nu 56 nu-3 nu*2」等數字逐行寫入檔案 script2.txt。
- (c) awk 計算 script2.txt 中所有數字的平均 avg，並印列「average=avg」。

4. 寫一腳本 script3.sh，要求如下：

- (a) read 讀入一串包含正負數字的字串。
 - (b) 將此字串轉換成相加運算式，再以 bc 計算結果。
 - (c) 例如輸入字串「2 -3 6 7」，輸出「2-3+6+7=12」。
5. 寫一腳本 script4.sh，要求如下：
- (a) read 讀入檔名 file。
 - (b) 以 source 讀取 file 中的變數設定。
 - (c) 輸出變數 abc 內容，若 abc 變數沒設定，則中斷程式 (提示 set -u)。
 - (d) 請同時測試 file 中有設定變數 abc 及沒設定的狀況。
6. 承上題，寫一腳本 script5.sh，在腳本中使用 set 命令，讓執行腳本時先印命令，行首會加上 "+" 號。
7. 承上題，寫一腳本 script6.sh，腳本最後增加一行「echo abcde」，但不能輸出。
8. 寫一腳本 script7.sh，使用執行時指定參數的預設變數，要求如下：
- (a) 預設變數 \$0 印出執行腳本名稱。
 - (b) 預設變數 \$1 指定一個檔案名。
 - (c) 預設變數 \$2 指定要寫入 \$1 檔案的第一個數字。
 - (d) 預設變數 \$3 指定要寫入 \$1 檔案的第二個數字。
 - (e) 預設變數 \$4 指定要寫入 \$1 檔案的第三個數字。
 - (f) 在腳本中以 cat 將三個數字寫入到檔案 \$1。
 - (g) awk 計算檔案 \$1 中的三個數字總和並輸出。

Chapter 9

條件判斷

9.1 test 命令

1. test 命令可以進行各種測試，功能可執行 `man test` 查詢：

```
TEST(1)                                User Commands                                TEST(1)
2 NAME
   test - check file types and compare values
4 SYNOPSIS
   test EXPRESSION
   [ EXPRESSION ]
6 DESCRIPTION
8     Exit with the status determined by EXPRESSION.
   ( EXPRESSION )
10     EXPRESSION is true
   ! EXPRESSION
12     EXPRESSION is false
   EXPRESSION1 -a EXPRESSION2
14     both EXPRESSION1 and EXPRESSION2 are true
   EXPRESSION1 -o EXPRESSION2
16     either EXPRESSION1 or EXPRESSION2 is true
   -n STRING
18     the length of STRING is nonzero
   STRING equivalent to -n STRING
20     -z STRING
       the length of STRING is zero
22 .....
```

2. 關於某個檔名的『類型』偵測 (存在與否)，如 `test -e filename`：

測試選項	代表意義
-e	該『檔名』是否存在？(常用)
-f	該『檔名』是否為檔案 (file)？(常用)
-d	該『檔名』是否為目錄 (directory)？(常用)
-b	該『檔名』是否為一個 block device 裝置？
-c	該『檔名』是否為一個 character device 裝置？
-S	該『檔名』是否為一個 Socket 檔案？
-p	該『檔名』是否為一個 FIFO (pipe) 檔案？
-L	該『檔名』是否為一個連結檔？

3. 關於檔案的權限偵測，如 `test -r filename`：

測試選項	代表意義
-r	偵測該檔名是否具有『可讀』的屬性？
-w	偵測該檔名是否具有『可寫』的屬性？
-x	偵測該檔名是否具有『可執行』的屬性？
-u	偵測該檔名是否具有『SUID』的屬性？
-g	偵測該檔名是否具有『SGID』的屬性？
-k	偵測該檔名是否具有『Sticky bit』的屬性？
-s	偵測該檔名是否為『非空白檔案』？

4. 兩個檔案之間的比較，如：`test file1 -nt file2`：

測試選項	代表意義
-nt	(newer than) 判斷 file1 是否比 file2 新
-ot	(older than) 判斷 file1 是否比 file2 舊
-ef	判斷 file1 與 file2 是否為同一檔案。

5. 關於兩個整數之間的判定，例如 `test n1 -eq n2`：

測試選項	代表意義
-eq	兩數值相等 (equal)
-ne	兩數值不等 (not equal)
-gt	n1 大於 n2 (greater than)
-lt	n1 小於 n2 (less than)
-ge	n1 大於等於 n2 (greater than or equal)
-le	n1 小於等於 n2 (less than or equal)

6. 判定字串的資料

測試選項	代表意義
<code>test -z string</code>	判定字串 <code>string</code> 是否為空字串？
<code>test -n string</code>	判定字串 <code>string</code> 是否不為空字串？
<code>test str1 == str2</code>	判定 <code>str1</code> 是否等於 <code>str2</code> ？
<code>test str1 != str2</code>	判定 <code>str1</code> 是否不等於 <code>str2</code> ？

7. 多重條件判定，例如：`test -r filename -a -x filename`：

測試選項	代表意義
<code>-a</code>	(and) 兩狀況同時成立。例如 <code>test -r file -a -x file</code> ，則 <code>file</code> 同時具有 <code>r</code> 與 <code>x</code> 權限時，才回傳 <code>true</code> 。
<code>-o</code>	(or) 兩狀況任何一個成立。例如 <code>test -r file -o -x file</code> ，則 <code>file</code> 具有 <code>r</code> 或 <code>x</code> 權限時，就可回傳 <code>true</code> 。
<code>!</code>	反相狀態，如 <code>test ! -x file</code> ，當 <code>file</code> 不具有 <code>x</code> 時，回傳 <code>true</code>

9.2 test 命令使用

1. `test -f` 測試檔案 `sh03.sh` 及 `sh0y.sh` 是否存在？

```

[dywang@dywmac zzz]$ test -e sh03.sh && echo "exist" || echo "not exist"
2 exist
[dywang@dywmac zzz]$ test -e sh0y.sh && echo "exist" || echo "not exist"
4 not exist

```

2. `sh05.sh` 從鍵盤輸入一個數字，`test` 測試這個數字是否大於 10。

```

[dywang@dywmac zzz]$ vim sh05.sh
2 [dywang@dywmac zzz]$ cat sh05.sh
#!/bin/bash
4 read -p "請輸入一數字： " -t 5 num
6 test "$num" -gt "10" && echo "$num>10" || echo "$num<=10"
exit 0

```

3. 執行 `sh05.sh`。

```

1 [dywang@dywmac zzz]$ sh sh05.sh
請輸入一數字： 10
3 10<=10
[dywang@dywmac zzz]$ sh sh05.sh
5 請輸入一數字： 45

```

```

7 45>10
   [dywang@dywmac zzz]$ sh sh05.sh
   請輸入一數字： -3
9  -3<=10

```

9.3 判斷符號 []

1. 在中括號 [] 內的每個元件都需要有空白鍵來分隔，以下說明空白鍵使用『□』來表示：

```

1  [ "$HOME" == "dywang" ]
3  [ "$HOME" == "dywang" ]
   ↑      ↑  ↑      ↑

```

2. [-z "\$name"] 判斷變數 name 是否是空字串？

```

1  [dywang@dywmac zzz]$ [ -z "$name" ] && echo "No name" || echo "name=
   $name"
   No name
3  [dywang@dywmac zzz]$ name=dywang; [ -z "$name" ] && echo "No name" ||
   echo "name=$name"
   name=dywang

```

3. 在中括號內，變數沒用引號括起來，當變數沒設定、是空字串、含空隔或特殊字元，判斷會出錯。

```

2  [dywang@dywmac zzz]$ name="De Yu"; [ $name == "De Yu" ] && echo "Y" ||
   echo "N"
   bash: [: too many arguments
   N

```

4. 在中括號內，變數要用單引號括起來，不會解析變數內容，所以判斷出錯。

```

1  [dywang@dywmac zzz]$ [ '$name' == "De Yu" ] && echo "Y" || echo "N"
   N

```

5. 在中括號內，變數要用雙引號括起來，解析變數內容後成功判斷。

```

2 [dywang@dywmac zzz]$ [ "$name" == "De Yu" ] && echo "Y" || echo "N"
Y

```

9.4 判斷符號 [[]]

1. 雙中括號 [[]] 比較時，右邊不加引號時是一個模式，用 == 符號比較是否相等，其中 ? 代表任意一個字元。

```

2 [dywang@dywmsi zzz]$ [[ abcd == abc ]] && echo Y || echo N
N
4 [dywang@dywmsi zzz]$ [[ abcd == abc. ]] && echo Y || echo N
N
6 [dywang@dywmsi zzz]$ [[ abcf == abc? ]] && echo Y || echo N
Y
8 [dywang@dywmsi zzz]$ [[ abcd == abc? ]] && echo Y || echo N
Y

```

2. 用 == 符號比較是否相等，其中 * 代表任意字元 0 個以上。

```

2 [dywang@dywmsi zzz]$ [[ abc == abc* ]] && echo Y || echo N
Y
4 [dywang@dywmsi zzz]$ [[ abc., == abc* ]] && echo Y || echo N
Y
6 [dywang@dywmsi zzz]$ [[ abcdf == abc* ]] && echo Y || echo N
Y
8 [dywang@dywmsi zzz]$ [[ abcdfe == abc* ]] && echo Y || echo N
Y
10 [dywang@dywmsi zzz]$ [[ abcd == abc* ]] && echo Y || echo N
Y

```

3. 用 > < 符號比較是大小時，是比較字串的 ASCII 碼大小。

```

2 [dywang@dywmsi zzz]$ [[ abc > abe ]] && echo Y || echo N
N
4 [dywang@dywmsi zzz]$ [[ abf > abe ]] && echo Y || echo N
Y
6 [dywang@dywmsi zzz]$ [[ 12 > 13 ]] && echo Y || echo N
N
8 [dywang@dywmsi zzz]$ [[ 192 > 53 ]] && echo Y || echo N
N

```

4. 使用 `=~` 符號判斷是否匹配，支援正規表示式。

```

2 [dywang@dywmsi zzz]$ [[ abcd =~ abc. ]] && echo Y || echo N
Y
4 [dywang@dywmsi zzz]$ [[ 1abcd2 =~ abc. ]] && echo Y || echo N
Y

```

5. 使用 `=~` 符號判斷字串是否是正負數字？

```

2 [dywang@dyw219 zzz]$ num='-19' ; [[ $num =~ ^-[0-9]+$ ]] && echo Y ||
echo N
Y
4 [dywang@dyw219 zzz]$ num='19' ; [[ $num =~ ^-[0-9]+$ ]] && echo Y ||
echo N
Y
6 [dywang@dyw219 zzz]$ num='19 ' ; [[ $num =~ ^-[0-9]+$ ]] && echo Y ||
echo N
N

```

9.5 判斷符號 (())

1. 雙小括號 (()) 主要是進行算術運算，適合整數的 `<` `>` `=` 比較。

```

1 [dywang@dywmsi zzz]$ (( 192 > 53 )) && echo Y || echo N
Y
3 [dywang@dywmsi zzz]$ (( -192 > 53 )) && echo Y || echo N
N
5 [dywang@dywmsi zzz]$ (( 10+44 > 53 )) && echo Y || echo N
Y

```

2. 雙小括號 (()) 進行算術運算，與 `[]` 或 `[[]]` 判斷不同，符號 (()) 中參數間有無空白隔開，結果都一樣。

```

2 [dywang@dywmsi zzz]$ ((10+44>53)) && echo Y || echo N
Y
4 [dywang@dywmsi zzz]$ ((-192>53)) && echo Y || echo N
N
6 [dywang@dywmsi zzz]$ ((10*5>53)) && echo Y || echo N
N
8 [dywang@dywmsi zzz]$ ((10>53-90)) && echo Y || echo N
Y

```

9.6 if 條件判斷式

1. 語法一：if ... then 當符合條件判斷時，就進行某項工作。

```
1 if [ 條件判斷式 ]; then
2     當條件判斷式成立時，可以進行的指令工作內容；
3 fi
```

2. 範例：輸入大小寫 y 或 n，條件判斷使用 && 做 AND，|| 做 OR 的條件判斷。

```
1 [dywang@dywmac zzz]$ vim sh06.sh
2 [dywang@dywmac zzz]$ cat sh06.sh
3 #!/bin/bash
4
5 read -p "Please input (Y/N): " yn
6
7 if [ "$yn" == "Y" ] || [ "$yn" == "y" ]; then
8     echo "OK, continue"
9     exit 0
10 fi
11 if [ "$yn" == "N" ] || [ "$yn" == "n" ]; then
12     echo "Oh, interrupt!"
13     exit 0
14 fi
15 echo "I don't know what is your choise" && exit 0
```

3. 執行結果。

```
1 [dywang@dywmac zzz]$ sh sh06.sh
2 Please input (Y/N): y
3 OK, continue
4 [dywang@dywmac zzz]$ sh sh06.sh
5 Please input (Y/N): n
6 Oh, interrupt!
7 [dywang@dywmac zzz]$ sh sh06.sh
8 Please input (Y/N): q
9 I don't know what is your choise
```

4. 語法二：if .. else ..fi

```
1 if [ 條件判斷式 ]; then
2     當條件判斷式成立時，可以進行的指令工作內容；
3 else
4     當條件判斷式不成立時，可以進行的指令工作內容；
5 fi
```


5. 語法三：if .. elif.. else.. fi

```
1 if [ 條件判斷式一 ]; then
    當條件判斷式一成立時，可以進行的指令工作內容；
3 elif [ 條件判斷式二 ]; then
    當條件判斷式二成立時，可以進行的指令工作內容；
5 else
    當條件判斷式一與二均不成立時，可以進行的指令工作內容；
7 fi
```

6. 例題：改寫 sh06.sh 成 sh06-1.sh：

```
1 [dywang@dywmac zzz]$ vim sh06-1.sh
[dywang@dywmac zzz]$ cat sh06-1.sh
3 #!/bin/bash
5 read -p "Please input (Y/N): " yn
7 if [ "$yn" == "Y" ] || [ "$yn" == "y" ]; then
    echo "OK, continue"
9 elif [ "$yn" == "N" ] || [ "$yn" == "n" ]; then
    echo "Oh, interrupt!"
11 else
    echo "I don't know what is your choise"
13 fi
```

7. 執行結果。

```
1 [dywang@dywmac zzz]$ sh sh06-1.sh
Please input (Y/N): Y
3 OK, continue
[dywang@dywmac zzz]$ sh sh06-1.sh
5 Please input (Y/N): N
Oh, interrupt!
7 [dywang@dywmac zzz]$ sh sh06-1.sh
Please input (Y/N):
9 I don't know what is your choise
```

8. 例題：偵測輸入的參數是否為 hello。

(a) 如果是，顯示"Hello, how are you?"；

- (b) 如果沒有加任何參數，提示使用者必須要使用的參數下達法；
- (c) 而如果加入的參數不是 `hello`，就提醒使用者僅能使用 `hello` 為參數。

```
1 [dywang@dywmac zzz]$ vim sh07.sh
[dywang@dywmac zzz]$ cat sh07.sh
3 #!/bin/bash
5 if [ "$1" == "hello" ]; then
    echo "Hello, how are you?"
7 elif [ -z "$1" ]; then
    echo "You MUST input parameters, ex> $0 someword"
9 else
    echo "The only parameter is 'hello'"
11 fi
```

9. 執行結果。

```
1 [dywang@dywmac zzz]$ sh sh07.sh
You MUST input parameters, ex> sh07.sh someword
3 [dywang@dywmac zzz]$ sh sh07.sh someword
The only parameter is 'hello'
5 [dywang@dywmac zzz]$ sh sh07.sh hello
Hello, how are you?
```

9.7 case 條件判斷式

1. case 判斷語法：

```
case $變數名稱 in
2     "變數內容1")
    程式段
4     ;;    ###兩個分號 (;;) 來代表該程式段落的結束
    "變數內容2")
6     程式段
    ;;
8     *)
    不包含變數內容1與變數內容2的其他程式執行段
10    exit 1
    ;;
12 esac
```

2. 改寫 `sh07.sh` 輸入 `hello` 的腳本，改用 `case` 條件判斷。

```
[dywang@dywmac zzz]$ cat sh08.sh
2  #!/bin/bash
4  case $1 in
    "hello")
6      echo "Hello, how are you?"
        ;;
8      "")
        echo "You MUST input parameters, ex> $0 someword"
        ;;
10     *)
        echo "Usage $0 {hello}"
        ;;
12     esac
```

3. 執行結果。

```
[dywang@dywmac zzz]$ sh sh08.sh
2  You MUST input parameters, ex> sh08.sh someword
[dywang@dywmac zzz]$ sh sh08.sh someword
4  Usage sh08.sh {hello}
[dywang@dywmac zzz]$ sh sh08.sh hello
6  Hello, how are you?
```

4. if, case 都可以做條件判斷，就看使用哪種方式腳本可讀性、維護性比較高。以下例題使用 if 判斷星期幾？以決定使用的練習虛擬機，使用 case 判斷外部第 1 個參數，決定要在虛擬機執行什麼程式？

```
[dywang@dywmac zzz]$ cat sh09.sh rhce
2  #!/bin/bash
4  k=$(date +%u)
   if [ "$k" -eq "4" ]; then
6      kvm=kvm3
   elif [ "$k" -eq "1" ]; then
8      kvm=kvm5
   else
10     kvm=kvm6
   fi
12  kvm="${kvm}.deyu.wang"
14  case $1 in
    'lpvar'|'rhcsa'|'rhce'|'mysql'|'py')
16      cmd="sh /tmp/$1.sh"
        ;;
18     'halt'|'reboot')
        cmd="$1"
```

```
20     ;;  
    'hello')  
22     cmd="cd ~/zzz && ./hello"  
    ;;  
24     'ntp')  
        cmd="ntpdate -u 192.168.1.140"  
26     ;;  
    *)  
28     cmd="exit"  
        echo "Nothing to do!"  
30     ;;  
esac  
32 echo "ssh $kvm \"$cmd\""
```

5. 執行結果。

```
[dywang@dywmac zzz]$ sh sh09.sh rhce  
2 ssh kvm6.deyu.wang "sh /tmp/rhce.sh"  
[dywang@dywmac zzz]$ sh sh09.sh py  
4 ssh kvm6.deyu.wang "sh /tmp/py.sh"  
[dywang@dywmac zzz]$ sh sh09.sh halt  
6 ssh kvm6.deyu.wang "halt"  
[dywang@dywmac zzz]$ sh sh09.sh ntp  
8 ssh kvm6.deyu.wang "ntpdate -u 192.168.1.140"  
[dywang@dywmac zzz]$ sh sh09.sh  
10 Nothing to do!  
ssh kvm6.deyu.wang "exit"
```

9.8 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 zzz 目錄。
- (b) 切換工作目錄到 zzz，所有練習腳本皆存在此目錄。
- (c) 腳本必須宣告執行命令 /bin/bash，且可執行。

2. 寫一腳本 testif1.sh，要求如下：

- (a) 使用腳本執行的預設第一個參數 \$1 讀入檔名，假設是 filename。
- (b) 測試 filename 是否存在，不存在則輸出 "No filename" 到 stderr，且回傳值 1。
- (c) 測試 filename 是目錄或檔案？如果是檔案印出「filename is file」，是目錄則印出「filename is directory」。

3. 寫一腳本 `testif2.sh`，要求如下：

- (a) 使用腳本執行的預設第一個參數 `$1`，假設是 `nu`。
- (b) 使用 `&& ||` 單行完成：如果 `nu > 0` 輸出 `nu*2` 結果，否則輸出 `100-nu` 結果。

4. 寫一腳本 `testif3.sh`，要求如下：

- (a) 使用腳本執行的預設第一個參數 `$1`，假設是 `nu`。
- (b) 判斷 `nu < 50`？範例輸出：`Ans:nu<50`
- (c) 判斷 `nu > 50`？範例輸出：`Ans:nu>50`
- (d) 判斷 `nu = 50`？範例輸出：`Ans:nu=50`
- (e) 判斷 `nu` 是空字串？範例輸出：`Ans:str=null`
- (f) 判斷 `nu` 是非數字字串？範例輸出：`Ans:str=nu`
- (g) 提示：先判斷是否空字串，再判斷是否數字。

5. 寫一腳本 `testif4.sh`，要求如下：

- (a) 使用腳本執行的預設第一個參數 `$1`。
- (b) 使用 `case` 判斷變數 `$1`，`0,7` 代表星期日，`1,2,3,4,5,6` 分別代表星期一至六。
- (c) `$1` 是 `0` 至 `7` 數字則列出正確的星期英文，第一個字母大寫，其餘小寫。例如：星期日 `Sunday`。
- (d) `$1` 不是 `0` 至 `7` 的數字，則列出「請輸入 `0` 至 `7` 的數字」。

Chapter 10

迴圈

10.1 for 迴圈

1. for 迴圈適合已知迴圈次數狀況，語法：

```
1 for ((初始值; 限制值; 執行步階))
  do
3   程式段
  done
```

2. for 括號內的三串內容意義為：

- (a) 初始值：某個變數在迴圈當中的起始值，例如 $i=1$ ；
- (b) 限制值：當變數的值在這個限制值的範圍內，就繼續進行迴圈。例如 $i \leq 100$ ；
- (c) 執行步階：每作一次迴圈時，變數的變化量。例如 $i=i+1$ 。

3. 例題：從 1 加到第 1 個指定的數字，但必須判斷參數是大於 1 且小於 100 的數字。

```
[dywang@dywmac zzz]$ cat sh11.sh
2 #!/bin/bash
num=$(echo $1 | egrep -o "[0-9]+")
4 [ -n "$num" ] && [ "$num" -gt 1 ] && [ "$num" -lt "100" ] \
  && max="$num" || max="100"
6 sum=0
  for ((i=1; i<=max; i++)); do
8     ((sum+=i))
  done
10 echo "1+..+$max=$sum"
```

4. 執行結果。

```
[dywang@dywmac zzz]$ sh sh11.sh
2 1+..+100=5050
[dywang@dywmac zzz]$ sh sh11.sh num
4 1+..+100=5050
[dywang@dywmac zzz]$ sh sh11.sh -12
6 1+..+12=78
[dywang@dywmac zzz]$ sh sh11.sh 201
8 1+..+100=5050
[dywang@dywmac zzz]$ sh sh11.sh 56
10 1+..+56=1596
```

5. 非數字的迴圈語法：

```
for var in con1 con2 con3 ...
2 do
    程式段
4 done
```

6. 例題：從腳本執行參數取出所有數字加總，並印出數字加總數學式及總和。先使用 `egrep` 過濾參數中的數字，非數字 `for` 迴圈將這些以空白隔開的數字加總。

```
[dywang@dywmac zzz]$ cat sh12.sh
2 #!/bin/bash
num=$(echo $@ | egrep -o "[0-9]+")
4 sum=0
for i in $num; do
6 ((sum+=i))
done
8 echo "$(echo $num | sed 's/\ \ */+/g')=$sum"
```

7. 執行結果。

```
[dywang@dywmac zzz]$ sh sh12.sh 12x34f56fe328fad
2 12+34+56+328=430
[dywang@dywmac zzz]$ sh sh12.sh 12 74n 23d wf3
4 12+74+23+3=112
```

10.2 while 迴圈

1. for 迴圈有固定數量的次數，while 不固定迴圈次數，直到 condition 條件不成立時才停止，語法如下：

```
while [ condition ]  
2 do  
    程式段落  
4 done
```

2. 例題：隨機產生一個 0 到 9 的數字，使用者輸入數字直到猜中為止。

```
[dywang@dywmac zzz]$ vim sh13.sh  
2 [dywang@dywmac zzz]$ cat sh13.sh  
#!/bin/bash  
4 rnum=$((RANDOM%10))  
num="-10"  
6 count=0  
while [ "$rnum" -ne "$num" ]; do  
8     read -p "請輸入一個 0 到 9 的數字： " -t 5 num  
    ((count++))  
10 done  
echo "你答對了，隨機數字 $num，你猜了 $count 次。"
```

3. 執行結果。

```
1 [dywang@dywmac zzz]$ sh sh13.sh  
請輸入一個 0 到 9 的數字： 5  
3 你答對了，隨機數字 5，你猜了 1 次。  
[dywang@dywmac zzz]$ sh sh13.sh  
5 請輸入一個 0 到 9 的數字： 5  
請輸入一個 0 到 9 的數字： 3  
7 你答對了，隨機數字 3，你猜了 2 次。  
[dywang@dywmac zzz]$ sh sh13.sh  
9 請輸入一個 0 到 9 的數字： 6  
請輸入一個 0 到 9 的數字： 5  
11 請輸入一個 0 到 9 的數字： 4  
請輸入一個 0 到 9 的數字： 3  
13 請輸入一個 0 到 9 的數字： 2  
請輸入一個 0 到 9 的數字： 1  
15 請輸入一個 0 到 9 的數字： 8  
你答對了，隨機數字 8，你猜了 7 次。
```

4. 先產生如下 sdata.csv 檔，每個欄位以逗號"," 隔開。


```

[dywang@dywmac zzz]$ cat sdata.csv
2 no.,name,score1,score2,grade
  11,dywang,81,12,A
4 152,linda,90,58,C
  33,peter,72,95,C
6 4,rita,65,34,E
  58,cora,5,85,D

```

5. 例題：使用 `while` 逐行讀入一個以逗號`,`分隔欄位的 `csv` 檔，再計算平均成績。IFS=`'` 設定逗號為欄位分隔符號，`read -r` 選項關閉倒斜線 `"\"` 跳脫動作，`f1 f2...` 指定各欄位依序放在 `f1 f2 ..` 等變數。

```

1 [dywang@dywmac zzz]$ cat sh14.sh
  #!/bin/bash
3 [ -z "$1" ] && echo "Usage: $0 <datafile>" && exit 1
  ! test -f "$1" && echo "Input file not found" && exit 2
5 while IFS=' ' read -r f1 f2 f3 f4 f5; do
    [ "$f1" == "no." ] && echo -e "name\tavg." && continue
7     avg=$((f3+f4)/2)
    echo -e "$f2\t$avg"
9 done < $1

```

6. 執行結果。

```

1 [dywang@dywmac zzz]$ sh sh14.sh
  Usage: sh14.sh <datafile>
3
  [dywang@dywmac zzz]$ sh sh14.sh sdata.csv
5 name      avg.
  dywang    46
7 linda     74
  peter     83
9 rita      49
  cora      45

```

10.3 until 迴圈

1. `while` 直到 `condition` 條件不成立時才停止，`until` 則當 `condition` 條件成立時終止迴圈，否則持續進行迴圈的程式段。

```

2 until [ condition ]
  do

```

程式段落

4 done

2. 例題：將 sh13.sh 的 while 迴圈改為 until，隨機產生一個 0 到 9 的數字，使用者輸入數字直到猜中為止。

```
[dywang@dywmac zzz]$ cat sh13-1.sh
2 #!/bin/bash
  rnum=$((RANDOM%10))
4 num="-10"
  count=0
6 until [ "$rnum" -eq "$num" ]; do
    read -p "請輸入一個 0 到 9 的數字： " -t 5 num
8    ((count++))
  done
10 echo "你答對了，隨機數字 $num，你猜了 $count 次。"
```

3. 執行結果。

```
[dywang@dywmac zzz]$ sh sh13-1.sh
2 請輸入一個 0 到 9 的數字： 4
  請輸入一個 0 到 9 的數字： 6
4 請輸入一個 0 到 9 的數字： 5
  你答對了，隨機數字 5，你猜了 3 次。
6 [dywang@dywmac zzz]$
[dywang@dywmac zzz]$ sh sh13-1.sh
8 請輸入一個 0 到 9 的數字： 1
  請輸入一個 0 到 9 的數字： 2
10 請輸入一個 0 到 9 的數字： 3
  請輸入一個 0 到 9 的數字： 4
12 請輸入一個 0 到 9 的數字： 5
  請輸入一個 0 到 9 的數字： 6
14 你答對了，隨機數字 6，你猜了 6 次。
```

4. 例題：將 sh14.sh 的 while 迴圈改為 until，逐行讀入一個以逗號"," 分隔欄位的 csv 檔，再計算平均成績。

```
[dywang@dywmac zzz]$ cat sh14-1.sh
2 #!/bin/bash
  [ -z "$1" ] && echo "Usage: $0 <datafile>" && exit 1
4 ! test -f "$1" && echo "Input file not found" && exit 2
  until ! IFS=',' read -r f1 f2 f3 f4 f5; do
6    [ "$f1" == "no." ] && echo -e "name\tavg." && continue
    avg=$((f3+f4)/2))
8    echo -e "$f2\t$avg"
```

```
done < $1
```

5. 執行結果。

```
1 [dywang@dywmac zzz]$ sh sh14-1.sh
Usage: sh14-1.sh <datafile>
3 [dywang@dywmac zzz]$ sh sh14-1.sh sdata.csv
name      avg.
5 dywang   46
  linda    74
7 peter    83
  rita     49
9 cora     45
```

10.4 function 功能

1. 利用 function 功能，語法：

```
1 function fname() {
   程式段
3 }
```

2. 計算數字階層 (factorial) 是程式使用函式的典型例子，以下範例輸入一字串，取出數字部分並分別計算其「階層」值。

```
1 [dywang@dywmac zzz]$ vim sh15.sh
[dywang@dywmac zzz]$ cat sh15.sh
3 #!/bin/bash
5 factorial(){
   if [ "$1" -gt "1" ]; then
7     i=$((1 - 1))
     j=$(factorial $i)
9     k=$((1 * j))
     echo $k
11  else
     echo 1
13  fi
   }
15
17 num=$(echo $@ | egrep -o "[0-9]+")
   for i in $num; do
     result=$(factorial $i)
19     echo "$i!=$result"
```

```
done
```

3. 執行結果。

```
[dywang@dywmac zzz]$ sh sh15.sh 23nmdf4sd5 7u8
2 23!=8128291617894825984
  4!=24
4 5!=120
  7!=5040
6 8!=40320
```

4. shell 腳本是直譯程式，使用的函式一定要寫在呼叫程式碼之前，否則無法呼叫。

```
[dywang@dywmac zzz]$ cat sh15.sh
2 #!/bin/bash

4 num=$(echo $@ | egrep -o "[0-9]+")
  for i in $num; do
6     result=$(factorial $i)
      echo "$i!=$result"
8  done

10 factorial(){
      if [ "$1" -gt "1" ]; then
12         i=$((1 - 1))
          j=$(factorial $i)
14         k=$((1 * j))
          echo $k
16     else
          echo 1
18     fi
  }
```

5. 執行結果，找不到命令 (函式) factroial。

```
1 [dywang@dywmac zzz]$ sh sh15.sh 23nmdf
sh15.sh: line 5: factorial: command not found
3 23!=
```

10.5 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 `zzz` 目錄。
- (b) 切換工作目錄到 `zzz`，所有練習腳本皆存在此目錄。
- (c) 腳本必須宣告執行命令 `/bin/bash`，且可執行。

2. 撰寫以下腳本：

- (a) `loop01.sh`：從 1 加到由命令列第一個參數指定的數字，但必須判斷參數是大於 1 且小於 10 的數字，若不是則預設為 10，假設輸入的輸字為 8，輸出範例如下：

```
1 1+2+3+4+5+6+7+8=28
```

- (b) `loop02.sh`：從命令列第一個參數指定的數字，加到 12，但必須判斷參數是大於 1 且小於 7 的數字，若不是則預設為 7，假設輸入的輸字為 5，輸出範例如下：

```
1 5+6+7+8+9+10+11+12=68
```

- (c) `loop03.sh`：從 1 加到由命令列第一個參數指定的數字，迴圈步階為 2，但必須判斷參數是大於 1 且小於 10 的數字，若不是則預設為 10，假設輸入的輸字為 8，輸出範例如下：

```
1 1+3+5+7=16
```

- (d) `loop04.sh`：從命令列第一個參數指定的數字，加到 1，迴圈步階為 -1，但必須判斷參數是大於 5 且小於 10 的數字，若不是則預設為 10，假設輸入的輸字為 8，輸出範例如下：

```
1 8+7+6+5+4+3+2+1=36
```

3. 寫一腳本 `loop1.sh` 使用 `for` 迴圈，要求如下：

- (a) 使用腳本執行的預設第一、二個參數 `$1`，`$2` 讀入兩個數字字串。

- (b) 判斷字串中的每個數字，若是兩個字串在同一個位置的數字相同，則記載一個 A。
- (c) 若是兩個字串都有這個數字但是位置不同，則記載一個 B。
- (d) 當沒有 A 也沒有 B 時則輸出 0A0B。
- (e) 輸入範例: 14685 47653，輸出結果: 14685 47653 Ans=1A2B

4. 寫一腳本 loop2.sh 使用 while 迴圈，要求如下：

- (a) 有一運算式，其運算規則為 (運算子, 運算元, 運算元)。例如 (*,3,5) 表示 $3*5$ ，結果等於 15。
- (b) 運算子可為 +, -, *, /，運算元則是一個正或負整數，若有多重括弧，則以內部括弧先處理，每個括弧運算皆取整數。
- (c) 使用腳本執行的預設第一個參數 \$1 讀入一個運算式，
例如: "(+, (*, 3, 5), (-, (*, 2, 4), 6))"
範例輸出: Ans=17
- (d) 範例輸入: "(+, (-, (+, 14, 19), (*, -7, 9)), (/ , 4, (/ , 14, 12)))"
範例輸出: Ans=100
- (e) 範例輸入 "(/ , 0, (/ , (+, -1, (*, 7, (*, -7, 18))), (/ , -3, 10)))"
範例輸出: No solution

5. 寫一腳本 loop3.sh 使用 until 迴圈，要求如下：

- (a) 對於一個字串若起始字元為 A，最後字元為 L，且其間不含任何 A 或 L 字元則稱 AL 字串，AL 不分大小寫。
- (b) 例: A_PENCIL 或 A_XXBBHCL (_ 表示空白字元)
- (c) 使用腳本執行的預設第一個參數 \$1 讀入一長字串。
- (d) 消去所有可能的 AL 字串，使得消去後之輸出不包含 AL 字串。
- (e) 範例輸入: This is a ball, not a bell.
範例輸出:
This is , not l.

6. 寫一腳本 loop4.sh，要求如下：

- (a) 使用腳本執行的預設第一個參數 \$1 讀入一個目錄字串變數 dirname。
- (b) 如果變數 dirname 不是目錄，輸出「NO \$dirname directory」到 stderr、退出、回傳值 1。

- (c) 建立一個 function `txtrename`，將預設變數 `$1` 檔名 `*.txt` 改名為 `*.bak`，例如：`$1` 是 `abc.txt`，則被改名為 `abc.bak`。
- (d) 使用 `for` 迴圈讀取目錄 `dirname` 中所有的 `.txt` 檔，逐一呼叫 `txtrename` 函式改名為 `.bak`。

Chapter 11

gcc 編譯

11.1 爲何要編譯

1. Linux 系統上真正認識的可執行檔其實是二進位檔案 (binary file)，例如 /usr/bin/passwd, /bin/touch, /bin/bash。以 file 命令查看檔案 /bin/bash 型態，顯示執行檔類別 (ELF 64-bit LSB executable)，同時說明是使用動態函式庫 (uses shared libs)。(ELF: Extensible Linking Format, LSB: Linux Standard Base)

```
1 [dywang@dywIssd zzz]$ file /bin/bash
/bin/bash: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
3 dynamically linked (uses shared libs), for GNU/Linux 2.6.18, stripped
```

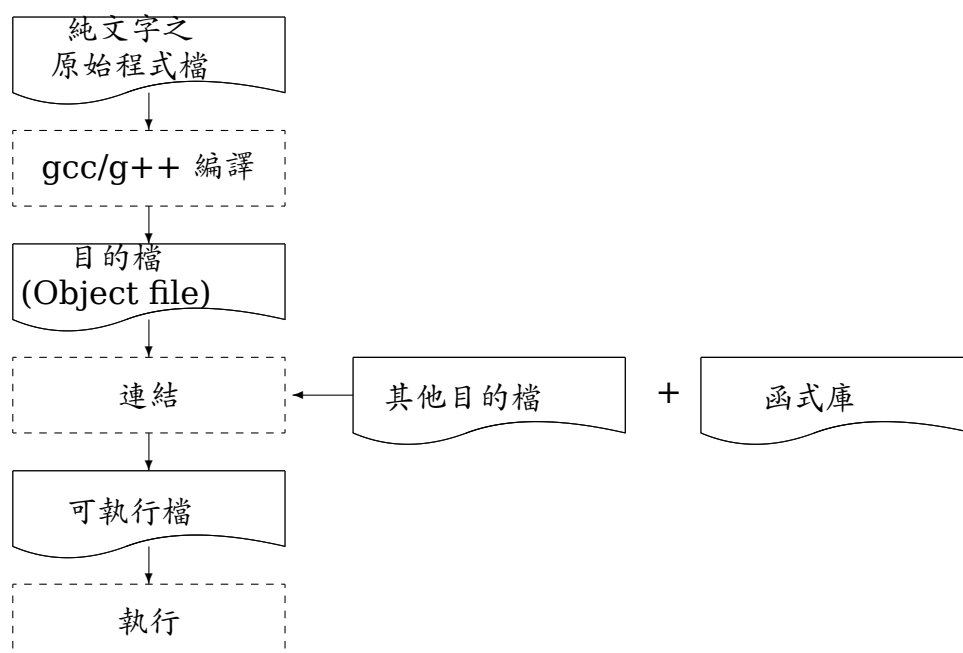
2. Shell scripts 只是利用 shell (例如 bash) 程式的功能進行一些判斷式，及呼叫一些已經編譯好的 binary 檔案來執行。以 file 命令查看檔案 /etc/init.d/autofs 型態，顯示 shell script 腳本檔類別。

```
1 [dywang@dywIssd zzz]$ file /etc/init.d/autofs
/etc/init.d/autofs: Bourne-Again shell script text executable
```

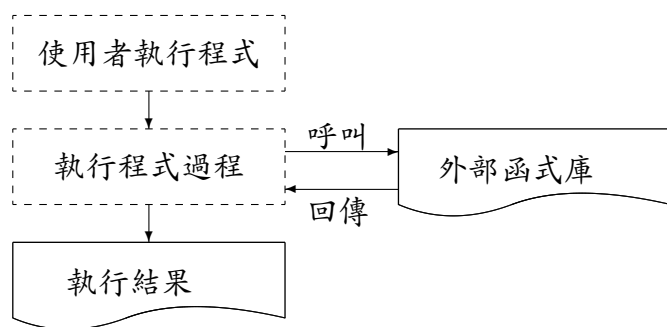
3. 編譯器：將純文字 (text file) 之原始碼檔案『編譯』成作業系統看得懂的 binary file。例如：以編譯器 gcc 將使用 C 語法進行的原始程式碼，編譯成可執行的 binary file。
4. 寫一個 C/C++ 程式，要有 C/C++ 的編譯器 (例如：gcc/g++)，將原始碼 (source code) 翻譯成機器可以執行的程式碼。

11.2 編譯步驟

1. 使用任何的文字編輯器產生 C/C++ 原始碼 (source code) ；
2. 編譯器 gcc/g++ 將 C/C++ 的原始碼編譯成目的檔 (object file) ，目的檔為機器可了解的機器語言但還無法執行 ；
3. 將目的檔和函式庫中的程式連結成可執行檔，函式庫存放一些 C/C++ 常用的 function，可執行檔為機器可執行的程式 ；
4. 執行可執行檔。
5. gcc/g++ 編譯流程：編譯的過程中，會產生以 *.o 的附檔名樣式存在的目標檔 (Object file) 。



6. 程式中『引用、呼叫』其他的外部副程式，必須在編譯的過程中，將該函式庫給加進以將所有的程式碼與函式庫作一個連結 (Link) 以產生正確的執行檔，程式執行引用函式庫流程：



11.3 編譯器 gcc

1. 以 vi 編輯程式檔 hello.c

```
[dywang@dyw219 zzz]$ vi hello.c
2  #include <stdio.h>
   main()
4  {
       printf("Hello!\n");
6  }
```

2. 僅將原始碼編譯成爲目標檔，並不製作連結等功能。

```
[dywang@dyw219 zzz]$ gcc -c hello.c
```

3. 自動的產生 hello.o 這個檔案，但是並不會產生 binary 執行檔。

```
1 [dywang@dyw219 zzz]$ ll hello.*
-rw-rw-r--. 1 dywang dywang 58 Apr 26 15:20 hello.c
3 -rw-rw-r--. 1 dywang dywang 1488 Apr 26 15:20 hello.o
```

4. 在編譯的時候，依據作業環境給予最佳化執行速度，以下命令會自動的產生 hello.o 這個檔案，並且進行最佳化。

```
1 [dywang@dyw219 zzz]$ gcc -O hello.c -c
```

5. 將編譯的結果輸出成某個特定檔名，-o 後接的是要輸出的 binary file 檔名。

```
1 [dywang@dyw219 zzz]$ gcc -o hello hello.c
[dywang@dyw219 zzz]$ ll hello*
3 -rwxrwxr-x. 1 dywang dywang 6663 Apr 26 15:22 hello
-rw-rw-r--. 1 dywang dywang 58 Apr 26 15:20 hello.c
5 -rw-rw-r--. 1 dywang dywang 1488 Apr 26 15:20 hello.o
```

6. 在編譯的時候，輸出較多的訊息說明，加入 `-Wall`，程式的編譯時會顯示警告訊息，`-Wall` 或 `-O` 等參數為旗標 (FLAGS)，簡稱這些旗標為 CCFLAGS。

```
1 [dywang@dyw219 zzz]$ gcc -o hello hello.c -Wall
hello.c:2:1: warning: return type defaults to 'int' [-Wreturn-type]
3   main()
   ~
5 hello.c: In function 'main' :
hello.c:5:1: warning: control reaches end of non-void function [-Wreturn
   -type]
7   }
   ~
```

7. 在進行 binary file 製作時，將連結的函式庫與相關的路徑填入。

- (a) `-lm` 指的是函式庫檔案 `libm.so` 或 `libm.a`；
- (b) `-L` 後面接函式庫的搜尋目錄路徑；
- (c) `-I` 後面接原始碼內的 `include` 檔案之所在目錄。

```
[dywang@dyw219 zzz]$ gcc -o hello1 hello.c -lm -L/usr/lib -I/usr/include
```

8. 產生的 `hello1` 與先前的 `hello` 不同，包含了 `libm.so.6` 函式。

```
1 [dywang@dyw219 zzz]$ ldd hello
   linux-vdso.so.1 => (0x00007fff308dd000)
3   libc.so.6 => /lib64/libc.so.6 (0x000000394aa00000)
   /lib64/ld-linux-x86-64.so.2 (0x0000560fe1b54000)
5 [dywang@dyw219 zzz]$ ldd hello1
   linux-vdso.so.1 => (0x00007ffce89bb000)
7   libm.so.6 => /lib64/libm.so.6 (0x000000394ae00000)
   libc.so.6 => /lib64/libc.so.6 (0x000000394aa00000)
9   /lib64/ld-linux-x86-64.so.2 (0x000055a2b0e7f000)
```

11.4 多檔編譯

1. 假設有一些標頭檔案 a.h、b.h 和 c.h，以及 C 原始碼檔案 main.c、2.c 和 3.c。

- (a) 下載測試檔案。

```
1 [dywang@dyw219 zzz]$ wget http://dywang.csie.cyut.edu.tw/dywang/download/make1.tar.gz
```

- (b) 解打包壓縮測試檔 make1.tar.gz。

```
1 [dywang@dyw219 zzz]$ tar zxvf make1.tar.gz
make1/
3 make1/main.c
  make1/b.h
5 make1/c.h
  make1/2.c
7 make1/3.c
  make1/a.h
```

2. 如何讓程式可以執行？

- (a) 切換工作目錄到 make1。

```
2 [dywang@dyw219 zzz]$ cd make1
  [dywang@dyw219 make1]$
```

- (b) 先編譯所有.c 檔產生.o 檔。

```
2 [dywang@dyw219 make1]$ gcc -c main.c
  [dywang@dyw219 make1]$ gcc -c 2.c
  [dywang@dyw219 make1]$ gcc -c 3.c
```

- (c) 再由.o 檔產生可執行檔。

```
1 [dywang@dyw219 make1]$ gcc -o main main.o 2.o 3.o
```

- (d) 執行程式 main。

```
1 [dywang@dyw219 make1]$ ./main
2222[dywang@dyw219 make1]$
```

11.5 多檔重編譯

1. 查看 main.c 2.c 3.c 三個程式 include 的標頭檔。如果程式設計人員改變 c.h，檔案 main.c 和 2.c 不需要重新編譯；3.c 需要被重新編譯。如果 b.h 被改變，程式設計人員忘了重新編譯 2.c，最後的程式可能不會正確運作。

```
2 [dywang@dyw219 make1]$ grep ^# main.c
3 #include <stdlib.h>
4 #include "a.h"
5
6 [dywang@dyw219 make1]$ grep ^# 2.c
7 #include <stdio.h>
8 #include "a.h"
9 #include "b.h"
10
11 [dywang@dyw219 make1]$ grep ^# 3.c
12 #include "b.h"
13 #include "c.h"
```

2. 變更 2.c 檔，輸出 2222 後換行。

```
2 [dywang@dyw219 make1]$ vim 2.c
3 [dywang@dyw219 make1]$ grep 2222 2.c
4 printf("2222\n");
```

3. 要讓 main 可執行，必須先重新編譯 2.o，再重新產可執行檔 main。

```
1 [dywang@dyw219 make1]$ gcc -c 2.c
2 [dywang@dyw219 make1]$ gcc -o main main.o 2.o 3.o
```

4. 執行 main，輸出 2222 後換行。

```
2 [dywang@dyw219 make1]$ ./main
2222
```

11.6 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 `zzz` 目錄。
- (b) 切換工作目錄到 `zzz`，所有練習皆存在此目錄。
- (c) 遠距教學期間，在家練習，`sid` 檔為「Public-IP 學號姓名」。

2. 寫一 C 語言程式 `gcc1.c`，要求如下：

- (a) 印出與你的註冊的學號姓名檔 `sid` 一樣的內容，要換行。
- (b) 將 `gcc1.c` 編譯成 `gcc1.o` 檔。
- (c) 編譯成可執行檔 `gcc1.b1`
- (d) 連結數學函式 `libm.so` 編譯成 `gcc1.b2`。

3. 下載檔案 `make1.tar.gz`。

- (a) 解打包壓縮 `make1.tar.gz`。
- (b) 進入 `make1` 工作目錄。
- (c) 修改程式讓程式只輸出與你的註冊的學號姓名檔 `sid` 一樣的內容，要換行，不要做其他更動或增加空白字元。
- (d) 編譯所有 `.c` 檔，產生 `.o` 檔。
- (e) 使用 `.o` 檔編譯成可執行檔 `main.b1`。

Chapter 12

make 與 makefile

12.1 前言

1. 何謂 makefile ?

- (a) make 命令雖然有很多內建的功能，但它也無法知道如何建立應用程式。故必須提供一個檔案，即 makefile，告訴 make 如何建立應用程式。
- (b) makefile 與專案的原始碼檔案，通常放在同一個目錄中。
- (c) 可以同時有很多不同的 makefile 管理專案的不同部分。
- (d) make 命令和 makefile 的結合，不僅控制原始碼的編譯，也可以用來準備使用手冊文件、安裝應用程式到目的目錄中。

2. make 使用 makefile 的好處：

- (a) 簡化編譯時所需要下達的指令；
- (b) 若在編譯完成之後，修改了某個原始碼檔案，則 make 僅會針對被修改了的檔案進行編譯，其他的 object file 不會被更動；
- (c) 最後可以依照相依性來更新 (update) 執行檔。

3. make 與 configure

- (a) 當執行 make 時，make 會在當時的目錄下搜尋文字檔 makefile (or Makefile)，其記錄了原始碼如何編譯的詳細資訊。
- (b) make 會自動的判別原始碼是否經過變動，而自動更新執行檔。
- (c) 偵測程式 configure
 - i. 軟體開發者會寫一支偵測程式來偵測使用者的作業環境，以及該作業環境是否有軟體開發商所需要的其他功能；

- ii. 偵測完畢後，會主動的建立 Makefile 的規則檔案。
- iii. 偵測程式的檔名為 configure 或者是 config。

4. makefile 規則

- (a) makefile 是由很多相依性項目 (dependencies) 和法則 (rules) 所組成。
- (b) 相依性項目，描述目標項目 (target, 要產生的檔案) 和產生該檔案之相關的原始碼檔案。
- (c) 法則是說明如何根據相依性檔案，來建立目標項目。
- (d) make 命令利用 makefile，先決定依序建立哪些目標項目，再決定依序喚起哪些法則。

12.2 Make 命令

1. make 輔助說明

```

[dywang@dyw219 make1]$ make --help
2 Usage: make [options] [target] ...
Options:
4  -b, -m                                Ignored for compatibility.
   -B, --always-make                     Unconditionally make all targets.
6  -C DIRECTORY, --directory=DIRECTORY
                                         Change to DIRECTORY before doing anything.
8  -d                                    Print lots of debugging information.
   --debug[=FLAGS]                      Print various types of debugging
                                         information.
10 -e, --environment-overrides
                                         Environment variables override makefiles.
12 -f FILE, --file=FILE, --makefile=FILE
                                         Read FILE as a makefile.
14 -h, --help                             Print this message and exit.
   -i, --ignore-errors                  Ignore errors from commands.
16 -I DIRECTORY, --include-dir=DIRECTORY
                                         Search DIRECTORY for included makefiles.
18 -j [N], --jobs[=N]                    Allow N jobs at once; infinite jobs with
                                         no arg.
   -k, --keep-going                     Keep going when some targets can't be made
                                         .
20 -l [N], --load-average[=N], --max-load[=N]
                                         Don't start multiple jobs unless load is
                                         below N.
22 -L, --check-symlink-times             Use the latest mtime between symlinks and
                                         target.
   -n, --just-print, --dry-run, --recon
                                         Don't actually run any commands; just
                                         print them.
24 -o FILE, --old-file=FILE, --assume-old=FILE

```



```

26      Consider FILE to be very old and don't
      remake it.
      -p, --print-data-base      Print make's internal database.
28      -q, --question            Run no commands; exit status says if up to
      date.
      -r, --no-builtin-rules     Disable the built-in implicit rules.
30      -R, --no-builtin-variables Disable the built-in variable settings.
      -s, --silent, --quiet      Don't echo commands.
32      -S, --no-keep-going, --stop
      Turns off -k.
34      -t, --touch               Touch targets instead of remaking them.
      -v, --version              Print the version number of make and exit.
36      -w, --print-directory     Print the current directory.
      --no-print-directory       Turn off -w, even if it was turned on
      implicitly.
38      -W FILE, --what-if=FILE, --new-file=FILE, --assume-new=FILE
      Consider FILE to be infinitely new.
40      --warn-undefined-variables Warn when an undefined variable is
      referenced.

42 This program built for x86_64-redhat-linux-gnu
Report bugs to <bug-make@gnu.org>

```

2. make 常用的選項及參數：

- (a) -j N：讓 make 在同一個時間執行 N 個命令，以加速編譯的時間。
- (b) -k：讓 make 在遇到錯誤時，仍然繼續運行，不停止在第一個問題點。
- (c) -n：告訴 make 只印出將會進行的工作，而不真正去編譯。
- (d) -f <filename>：告訴 make 該使用的 makefile 檔案。如果不使用這個選項，make 會依序尋找目錄中的 GNUmakefile, makefile, Makefile。

12.3 Makefile 語法

1. makefile 的語法 (syntax)

```

1  標的(target): 目標檔1 目標檔2
    <tab> gcc -o 欲建立的執行檔 目標檔1 目標檔2

```

- (a) 標的 (target) 與相依檔案 (就是目標檔) 之間需以『:』隔開。
- (b) <tab> 需要在命令行的第一個字元；
- (c) makefile 語法中之 <tab> 與空白：

- i. 所有的法則必須在同一行，而且行首必須為 <tab>；不能為空白。
- ii. 在 makefile 中，行尾如果有一個空白，會造成 make 命令執行錯誤。

(d) makefile 的註解 (comment)：

- i. 如同 C 原始碼檔案一般，在 makefile 中，以 # 為行首的文字都是註解。
- ii. makefile 中的註解只是協助作者和其它人，了解 makefile 的內容。

2. 第一支 makefile。

```
1 [dywang@dyw219 zzz]$ cd make1/  
2 [dywang@dyw219 make1]$ vim makefile  
[dywang@dyw219 make1]$ cat makefile  
4 main: main.o 2.o 3.o  
    gcc -o main main.o 2.o 3.o
```

(a) 刪除先前產生的.o 檔及執行檔 main。

```
1 [dywang@dyw219 make1]$ rm *.o main
```

(b) 執行 make 不加任何參數，會自動搜尋 makefile 進行編譯。

```
1 [dywang@dyw219 make1]$ make  
cc      -c -o main.o main.c  
3 cc      -c -o 2.o 2.c  
cc      -c -o 3.o 3.c  
5 gcc -o main main.o 2.o 3.o
```

(c) 執行結果：

```
1 [dywang@dyw219 make1]$ ./main  
2222
```

(d) 再執行一次 make，沒有任何變動，所以沒有必要進行任何重編譯。

```
1 [dywang@dyw219 make1]$ make  
2 make: `main' is up to date.
```

12.4 Makefile 相依性項目

1. 再查看 main.c 2.c 3.c 程式包含的標頭檔。

```
[dywang@dyw219 make1]$ grep ^# main.c
2 #include <stdlib.h>
  #include "a.h"
4 [dywang@dyw219 make1]$ grep ^# 2.c
  #include <stdio.h>
6 #include "a.h"
  #include "b.h"
8 [dywang@dyw219 make1]$ grep ^# 3.c
  #include "b.h"
10 #include "c.h"
```

2. 以查到的標頭檔確認：目標項目 2.o 與 2.c a.h b.h 三個檔有關，main 與 main.o、2.o、3.o 相關，main.o 與 main.c、a.h 相關，以此類推。

```
2 main.o: main.c a.h
  2.o: 2.c a.h b.h
  3.o: 3.c b.h c.h
```

3. 其實可以不用這麼麻煩，以 gcc 的 -MM 選項查詢 main.c 2.c 3.c，就可以 makefile 格式輸出相依性項目。

```
1 [dywang@dyw219 make1]$ gcc -MM main.c 2.c 3.c
  main.o: main.c a.h
3 2.o: 2.c a.h b.h
  3.o: 3.c b.h c.h
```

4. 輸出結果可插入 makefile 中，成為相依性的法則。

```
[dywang@dyw219 make1]$ gcc -MM main.c 2.c 3.c >> makefile
```

5. 查看 makefile 除原先的 main 目標項目外，增加了 main.o 2.o 3.o 三個目標項目，但都沒有指定編譯法則。

```
1 [dywang@dyw219 make1]$ cat makefile
  main: main.o 2.o 3.o
3 main.o: main.c a.h
  2.o: 2.c a.h b.h
```

```
5 3.o: 3.c b.h c.h
```

6. 執行 `make` 自動找到 `makefile` 編譯，沒有指定編譯法則，則使用 `make` 內建法則。

```
1 [dywang@dyw219 make1]$ make
cc      -c -o main.o main.c
3 cc      -c -o 2.o 2.c
cc      -c -o 3.o 3.c
5 cc  main.o 2.o 3.o      -o main
```

7. 成功產生 `main` 執行檔，並執行成功。

```
1 [dywang@dyw219 make1]$ ./main
2222
```

12.5 多重目標項目

1. 可在 `makefile` 中建立多個目標項目，並於 `make` 時指定目標項目。例如：增加 `clean` 目標項目，移除不想要的目的檔，`clean` 冒號之後是空白，目標項目永遠會被認為過期，所以它的法則一定會被執行。

```
[dywang@dyw219 make1]$ vim makefile
2 [dywang@dyw219 make1]$ cat makefile
main: main.o 2.o 3.o
4 main.o: main.c a.h
2.o: 2.c a.h b.h
6 3.o: 3.c b.h c.h
clean:
8 rm -f main main.o 2.o 3.o
```

2. 測試目標項目 `clean`：

```
[dywang@dyw219 make1]$ make clean
2 rm -f main main.o 2.o 3.o
[dywang@dyw219 make1]$ ls
4 2.c 3.c a.h b.h c.h main.c makefile
```

3. 再 make 建立 main 目標檔

```

[dywang@dyw219 make1]$ make
2 cc -c -o main.o main.c
cc -c -o 2.o 2.c
4 cc -c -o 3.o 3.c
cc main.o 2.o 3.o -o main

```

4. 此時 main main.o 2.o 3.o 等目標檔已存在，make 可先清除目標檔再編譯程式 main。

```

1 [dywang@dyw219 make1]$ make clean main
rm -f main main.o 2.o 3.o
3 cc -c -o main.o main.c
cc -c -o 2.o 2.c
5 cc -c -o 3.o 3.c
cc main.o 2.o 3.o -o main

```

12.6 Makefile 內建法則

1. make 內建的法則是借助檔尾 (suffix) 得知使用的法則。檔尾和符號 (pattern) 法則有下列兩項表示法，將 .old_suffix 的檔案變成 .new_suffix 的檔案。

```

2 .<old_suffix>.<new_suffix>:
%.<new_suffix>: %.<old_suffix>

```

2. 可以利用 -p 選項，要求 make 印出內建法則，過濾第一種法則 .c.o: 將.c 檔變成.o 檔。

```

1 [dywang@dyw219 make1]$ make -p | sed -r '/^(#|$)/d' | grep -A1 "^c.o:"
.c.o:
3 $(COMPILE.c) $(OUTPUT_OPTION) $<

```

3. 過濾第二種法則 %.o: %.c 將.c 檔變成.o 檔。

```

1 [dywang@dyw219 make1]$ make -p | sed -r '/^(#|$)/d' | grep -A1 "%.o: %.c$"
%.o: %.c
3 gcc -Wall -c $<

```

4. 刪除 main main.o 2.o 3.o 後再執行 make，顯示 cc -c ...，以 cc 編譯，表示內建法則一生效。

```
1 [dywang@dyw219 make1]$ rm -f main main.o 2.o 3.o
  [dywang@dyw219 make1]$ make
3 cc -c -o main.o main.c
  cc -c -o 2.o 2.c
5 cc -c -o 3.o 3.c
  cc main.o 2.o 3.o -o main
```

5. 再清除 main 執行檔及所有.o 檔。

```
2 [dywang@dywIssd make1]$ make clean
  rm -f main main.o 2.o 3.o
```

6. 若不使用內建法則，可以於 make 時，直接加入參數

```
2 [dywang@dywIssd make1]$ make CC=gcc CFLAGS="-Wall -g" main
gcc -Wall -g -c -o main.o main.c
gcc -Wall -g -c -o 2.o 2.c
4 gcc -Wall -g -c -o 3.o 3.c
  gcc -o main main.o 2.o 3.o
```

12.7 Makefile 自訂法則

1. 在 makefile 增加目標項目，以法則一將.c 檔案變成.o 檔案。其中變數 \$< 代表目前的相依性項目，會被展開成為原本的檔案名稱（含舊的檔尾）。

```
1 [dywang@dyw219 make1]$ vim makefile
  [dywang@dyw219 make1]$ grep ^.c.o -A1 makefile
3 .c.o:
    gcc -W -c $<
```

2. make clean 刪除 main main.o 2.o 3.o。

```
[dywang@dyw219 make1]$ make
2 make: `main' is up to date.
[dywang@dywIssd make1]$ make clean
4 rm -f main main.o 2.o 3.o
```

3. 再執行 make，gcc 以 -W 選項編譯，表示法則一生效。

```
[dywang@dyw219 make1]$ make
2 gcc -W -c main.c
gcc -W -c 2.c
4 gcc -W -c 3.c
cc main.o 2.o 3.o -o main
```

4. 將 makefile 將.c 檔案變成.o 檔案法則改用法則二。

```
1 [dywang@dyw219 make1]$ grep ~%.o -A1 makefile
%.o: %.c
3 gcc -W -c $<
```

5. make clean 刪除 main main.o 2.o 3.o。

```
1 [dywang@dyw219 make1]$ make
make: `main' is up to date.
3 [dywang@dywIssd make1]$ make clean
rm -f main main.o 2.o 3.o
```

6. 再執行 make，gcc 以 -W 選項編譯，表示法則二生效。

```
[dywang@dyw219 make1]$ rm -f main main.o 2.o 3.o
2 [dywang@dyw219 make1]$ make
gcc -W -c main.c
4 gcc -W -c 2.c
gcc -W -c 3.c
6 cc main.o 2.o 3.o -o main
```

12.8 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 `zzz` 目錄。
- (b) 切換工作目錄到 `zzz`，所有練習皆存在此目錄。
- (c) 檔案導向都是執行 `.sh` 腳本命令時導向，不是寫在腳本內導向。檔案導向到 `/tmp/$mysid.$evalname` 目錄，其中變數 `$mysid` 為自己的學號，`$evalname` 為評分程式名稱不加附檔名，目錄若不存在，必須自行建立。
- (d) 下載檔案 `make1.tar.gz`。
- (e) 解打包壓縮 `make1.tar.gz`。
- (f) 進入 `make1` 工作目錄。
- (g) `mf1-mf5` 新增的項目都列在檔案最後，且不要更動前面的項目，題目沒要求的，也不要多做。

2. 寫一 makefile `mf1`，要求如下：

- (a) 修改程式讓程式印出你的「位置 IP 最後一段學號姓名」，要換行。
- (b) 使用 `.o` 檔編譯成可執行檔 `mf1.bin`。
- (c) 編譯法則：使用 `gcc` 編輯器，依序為 `-o mf1.bin` 生成執行檔，使用 `main.o 2.o 3.o` 檔。

3. 承上題，產生 makefile `mf2`，要求如下：

- (a) 包含原 `mf1` 的目標項目。
- (b) 使用 `gcc -MM` 查詢所有標的及目標項目，並累加導向到 `mf2`，不做其他更動。

4. 承上題，產生 makefile `mf3`，要求如下：

- (a) 包含原 `mf2` 的目標項目。
- (b) 增加目標項目 `remove` 移除產生的執行檔及 `.o` 檔，依序為 `mf1.bin main.o 2.o 3.o`。

5. 承上題，產生 makefile `mf4`，要求如下：

- (a) 包含原 `mf3` 的目標項目。
- (b) 使用法則一 `.c.o` 自訂 `.c` 檔處理成 `.o` 檔法則，使用 `gcc` 編譯器，緊接使用數學函式庫選項。

6. 承上題，產生 makefile `mf5`，要求如下：

- (a) 包含原 `mf3` 的目標項目。

- (b) 使用法則二 `%.o: %.c` 自訂.c 檔處理成.o 檔法則，使用 `gcc` 編譯器，緊接使用數學函式庫選項。
7. 使用 `make -p` 查詢法則，去除註解行及空白行，列出所有法則 (有幾行列幾行)。
- (a) 第一種法則 `%.<old_suffix>.<new_suffix>:`
- 過濾目標項目副檔名.sh 檔案的處理法則，導向到 `mk1-1.txt`。
 - 過濾目標項目副檔名.cpp 處理成.o 的法則，導向到 `mk1-2.txt`。
- (b) 第二種法則 `%.<new_suffix>: %.<old_suffix>`
- 過濾目標項目副檔名.sh 檔案的處理法則，導向到 `mk2-1.txt`。
 - 過濾目標項目副檔名.cpp 處理成.o 的法則，導向到 `mk2-2.txt`。

Chapter 13

Makefile 變數

13.1 前言

1. 變數用途：

- (a) 簡化 makefile
- (b) 開發過程與最終版本使用之編譯參數不同
 - i. 開發應用程式過程中，不會進行最佳化處理，而須連結一些除錯資訊；
 - ii. 最終版本則應是一個最小的二進位檔案，且不含除錯資訊。
- (c) 讓 makefile 也適用不同的編譯器。

2. 變數定義：MACRONAME=value

- (a) 變數與變數內容以『=』隔開，同時兩邊可以具有空格；
(shell 中的變數設定，兩邊不可以具有空格)
- (b) 變數左邊不可以有 <tab> ；
- (c) 變數與變數內容在『=』兩邊不能具有『:』；
- (d) 習慣上，變數最好是以『大寫字母』為主；
- (e) 等號後面的 value 變成空白時，代表將變數清成空白。

3. 變數存取

- (a) \$(MACRONAME)
- (b) \${MACRONAME}
- (c) 有些 make 版本也接受 \$MACRONAME

4. 環境變數取用規則：

- (a) make 指令列後面加上的環境變數為優先；
- (b) makefile 裡面指定的環境變數第二；
- (c) shell 原本具有的環境變數第三。

13.2 makefile 變數實例一

1. 複製 makefile 成 makefile1，並設定 CC CFLAGS OBJS 三個變數。若要改變編譯器命令，只需改變 CC 變數，改變編譯旗標變數 CFLAGS 就能統一變更編譯選項，OBJS 變數指定所有的.o 檔。

```
[dywang@dyw219 make1]$ cp makefile makefile1
2 [dywang@dyw219 make1]$ vim makefile1
[dywang@dyw219 make1]$ cat makefile1
4 CC = gcc
  CFLAGS = -Wall
6 OBJS = main.o 2.o 3.o

8 main: $(OBJS)
  main.o: main.c a.h
10     $(CC) $(CFLAGS) -c main.c
  2.o: 2.c a.h b.h
12     $(CC) $(CFLAGS) -c 2.c
  3.o: 3.c b.h c.h
14     $(CC) $(CFLAGS) -c 3.c
  clean:
16     rm -f $(OBJS)
```

2. 執行結果：

```
[dywang@dyw219 make1]$ make -f makefile1 clean main
2 rm -f main.o 2.o 3.o
  gcc -Wall -c main.c
4 gcc -Wall -c 2.c
  gcc -Wall -c 3.c
6 gcc  main.o 2.o 3.o  -o main
```

13.3 makefile 常用變數

1. 常用的變數或符號

符號	意義
\$?	代表需要重建（被修改）的相依性項目。
\$@	目前的目標項目名稱。
\$<	代表目前的相依性項目。
\$*	代表目前的相依性項目，不過不含副檔名。
-	make 會忽略命令的錯誤。
@	make 不會在標準輸出 stdout 顯示要執行的命令。

2. 以下例子，目標項目 `main` 的相依性項目有 `main.o 2.o 3.o`，則法則中的 `$<` 代表目前的相依項目 `main.o 2.o 3.o`。

```
2 main: main.o 2.o 3.o
   gcc -o main $<
```

3. `makefile1` 的目標項目 `main`，使用內建法則（沒有指定法則），現在增加如下法則，其中 `$@` 代表目前的目標項目 `main`。

```
2 [dywang@dyw219 make1]$ vim makefile1
3 [dywang@dyw219 make1]$ grep ^main: -A1 makefile1
4 main: $(OBJS)
   $(CC) -o $@ $(OBJS)
```

4. `clean` 目標項目刪除所有 `.o` 檔，當 `.o` 檔不存在時忽略錯誤。

```
2 [dywang@dyw219 make1]$ vim makefile1
3 [dywang@dyw219 make1]$ grep ^clean: -A1 makefile1
4 clean:
   -rm -f $(OBJS)
```

5. 判斷式 `if` 起始為符號 `@`，讓 `make` 在執行該法則時，停止印出標準輸出的文字。

```
2 install: main
3   @if [ -d $(INSTDIR) ]; \
4     then \
   ...;\
   fi
```

13.4 makefile 變數實例二

1. 加入 install 目標項目，將完成的應用程式安裝到指定的目錄，判斷式 if 起始為符號 @，讓 make 在執行該法則時，停止印出標準輸出的文字。因為 makefile 中每一行命令都會啟動一個新的 shell，為了讓判斷式中所有命令都在同一個 shell 執行，必須加上反斜線 \，讓所有 script 命令在同一行。

```
1 [dywang@dyw219 make1]$ cp makefile1 makefile2
[dywang@dyw219 make1]$ vim makefile2
3 [dywang@dyw219 make1]$ grep ^install: -A10 makefile2
install: main
5     @if [ -d $(INSTDIR) ]; \
      then \
7         cp main $(INSTDIR);\
          chmod a+x $(INSTDIR)/main;\
9         chmod og-w $(INSTDIR)/main;\
          echo "Installed in $(INSTDIR)";\
11        else \
          echo "Sorry, $(INSTDIR) does not exist";\
13        fi
[dywang@dyw219 make1]$ grep ^INSTDIR makefile2
15 INSTDIR = /home/dywang/bin
```

2. 在此例中；分號連續執行，也可以換成 && 前個命令成功，才執行下個命令

```
1 [dywang@dyw219 make1]$ grep ^install: -A10 makefile2
install: main
3     @if [ -d $(INSTDIR) ]; \
      then \
5         cp main $(INSTDIR) &&\
          chmod a+x $(INSTDIR)/main &&\
7         chmod og-w $(INSTDIR)/main &&\
          echo "Installed in $(INSTDIR)";\
9         else \
          echo "Sorry, $(INSTDIR) does not exist";\
11        fi
```

3. 執行 clean 目標項目

```
1 [dywang@dyw219 make1]$ make -f makefile2 clean
rm -f main.o 2.o 3.o
```

4. 執行 main 目標項目

```
[dywang@dyw219 make1]$ make -f makefile2 main
2 gcc -Wall -c main.c
  gcc -Wall -c 2.c
4 gcc -Wall -c 3.c
  gcc -o main main.o 2.o 3.o
```

5. 執行 install 目標項目

```
1 [dywang@dyw219 make1]$ make -f makefile2 install
  Installed in /home/dywang/bin
3 [dywang@dyw219 make1]$ ll /home/dywang/bin/ma
  mac_desktop.sh  main
```

6. 查看安裝的檔案 main

```
[dywang@dyw219 make1]$ ll /home/dywang/bin/main
2 -rwxr-xr-x. 1 dywang dywang 7036 Apr 27 18:35 /home/dywang/bin/main
```

13.5 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 zzz 目錄。
- (b) 切換工作目錄到 zzz，所有練習皆存在此目錄。
- (c) 下載檔案 make1.tar.gz。
- (d) 解打包壓縮 make1.tar.gz。
- (e) 進入 make1 工作目錄。
- (f) 每題的執行檔 target 都改成與指定的 makefile 檔名加副檔名 bin，例如：
mfv1.bin, mfv2.bin, mfv3.bin ...。
- (g) 題目沒要求的，不要多做。

2. 承上一章的 makefile mf4，新增 mfv1，所有的.o 檔設定成變數 OBJS。

3. 承上題 mfv1，產生 makefile mfv2

- (a) 包含原 mfv1 的目標項目。

- (b) 增加所有相依性法則，.o 目標項目皆不使用內建法則。
 - (c) 將 main.o 2.o 3.o 的「相依項目」改用內建變數 \$<，其他不要更動。
4. 承上題 mfv2，產生 makefile mfv3，要求如下：
- (a) 設定編譯器變數 CC = gcc，編譯選項變數 CFLAGS = -W。
5. 承上題 mfv3，產生 makefile mfv4，要求如下：
- (a) 包含原 mfv3 的目標項目。
 - (b) 使用變數 \$@ 產生目標項目 mfv4.bin。
 - (c) 增加目標項目 remove 移除產生的執行檔及所有.o 檔，當.o 檔不存在時忽略錯誤。
6. 承上題 mfv4，產生 makefile mfv5，要求如下：
- (a) 增加目標項目 mdir 建立目錄 /tmp/mfv5.mdir.test，只建一層目錄，請用最精簡方式建立。
 - (b) 當要建的目錄存在時忽略錯誤，使用以下命令測試生效：
- ```
make -f mfv5 mdir && echo Yes
```
7. 承上題 mfv5，產生 makefile mfv6，要求如下：
- (a) 包含原 mfv5 的目標項目。
  - (b) 新增目標項目 install
    - i. 安裝目錄變數 DYBIN = /tmp/dywang/bin。
    - ii. 判斷安裝目錄 DYBIN 是否存在？如果不存在，必須先建目錄。不要使用 if...then，判斷後以 && 或 || 執行建立目錄即可。
    - iii. 判斷腳本不輸出到 stdout。
    - iv. 所有命令在同一個 shell。
    - v. 除判斷式內腳本外，其他腳本以「前一命令成功才執行下一個命令」的方式進行。
    - vi. 安裝檔案已存在時，直接覆蓋。
    - vii. 以單一命令完成「執行檔必須所有人可執行，但都不可寫入」的設定。
    - viii. 目標項目 install 的法則以最精簡的腳本完成，「判斷目錄是否存在的判斷式」只做建立目錄，其他命令都不可放在判斷式中。不 echo 輸出到螢幕，一個判斷式 + 三個命令即可完成。

# Chapter 14

## 使用者手冊

### 14.1 撰寫手冊

1. Makefile 假造目標項目 `all` 為 makefile 的第一個目標項目，`all` 列出所有要產生的檔案為相依性項目。如果不指定 `all` 目標項目，`make` 就會產生 makefile 中的第一個目標項目。
2. 在 makefile2 增加假造目標項目 `all`，建立多個目標檔案。例如：產生二進位執行檔 `main` 和使用手冊文件 `main.1`。

```
1 [dywang@dyw219 make1]$ vim makefile2
 [dywang@dyw219 make1]$ grep ^all makefile2
3 all: main main.1
```

3. 大部分使用手冊的設計都有如下樣式：

- (a) Header
- (b) Name
- (c) Synopsis
- (d) Description
- (e) Options
- (f) Files
- (g) See also
- (h) Bugs

4. 編輯簡單 `main` 應用程式使用手冊之原始碼 `main.1`。



```

1 [dywang@dyw219 make1]$ vim main.1
[dywang@dyw219 make1]$ cat main.1
3 .TH MAIN 1
.SH NAME
5 Main \- A simple demonstration application that does very little.
.SH SYNOPSIS
7 .B main
[\-option ...]
9 .SH DESCRIPTION
.PP
11 \fImain\fP is a complete application that does nothing useful.
.PP
13 It was written for demonstration purposes.
.SH OPTIONS
15 .PP
It doesn't have any, but let's pretend, to make this template
17 complete:
.TP
19 .BI \-option
If there was an option, it would not be -option.
21 .SH RESOURCES
.PP
23 main uses almost no resources.
.SH DIAGNOSTICS
25 The program shouldn't output anything, so if you find it doing so
there's probably something wrong. The return value is zero.
27 .SH SEE ALSO
The only other program we know with this this little functionality is
29 the ubiquitous hello world application.
.SH COPYRIGHT
31 main is Copyright (c) 2019
.SH BUGS
33 There probably are some, but we don't know what they are yet.
.SH AUTHORS
35 ABC123

```

5. man 手冊格式巨集都以 (.) 開頭，而且都很簡短。執行 `man 7 man` 可查詢格式。

```

1 [root@dyw219 ~]# man -f man
man (1) - format and display the on-line manual pages
3 man.config [man] (5) - configuration data for man
man (7) - macros to format man pages
5 [root@dyw219 ~]# man 7 man | cat -

7 MAN(7) Linux Programmer's Manual MAN(7)
NAME
9 man - macros to format man pages
SYNOPSIS
11 groff -Tascii -man file ...
 groff -Tps -man file ...

```

```

13 DESCRIPTION
15 PREAMBLE
 The first command in a man page should be
17 .TH title section date source manual,
SECTIONS
19 Sections are started with .SH followed by the heading name.
 .SH NAME
21 chess \- the game of chess
FONTS
23 The commands to select the type face are:
 .B Bold
25 .BI Bold alternating with italics
 .BR Bold alternating with Roman
27 .I Italics
 .IB Italics alternating with bold
29 .IR Italics alternating with Roman
 .RB Roman alternating with bold
31 .RI Roman alternating with italics
 .SB Small alternating with bold
33 .SM Small (useful for acronyms)
35 OTHER MACROS AND STRINGS
 Normal Paragraphs
37 .LP Same as .PP (begin a new paragraph).
 .P Same as .PP (begin a new paragraph).
39 .PP Begin a new paragraph and reset prevailing indent.
 Relative Margin Indent
41 .RS i Start relative margin indent:
 moves the left margin i to the right.
43 .RE End relative margin indent and restores
 the previous value of the prevailing indent.
45 Indented Paragraph Macros
 .HP i Begin paragraph with a hanging indent.
47 .IP x i Indented paragraph with optional hanging tag x.
 .TP i Begin paragraph with hanging tag.
49 Hypertext Link Macros
 Miscellaneous Macros
51 .DT Reset tabs to default tab values (every 0.5 inches)..
 .PD d Set inter-paragraph vertical distance to d (if omitted,
53 d=0.4v).
 .SS t Subheading t (like .SH, but used for a subsection).
55 Predefined Strings
57 SAFE SUBSET
 Font changes (ft and the \f escape sequence) should only have the
59 values 1, 2, 3, 4, R, I, B, P, or CW.
NOTES
61 FILES
 /usr/share/groff/[*/]tmac/tmac.an
63 /usr/man/whatis
BUGS
65 AUTHORS
SEE ALSO

```

```

67 apropos(1), groff(1), man(1), man2html(1), mdoc(7), mdoc.samples
 (7),
 groff_man(7), groff_www(7), whatis(1)
69 Linux 2004-07-27 MAN(7)

```

## 14.2 groff 處理手冊

1. UNIX 使用手冊是藉由 nroff 工具，格式處理後所形成，在 Linux 系統上也有個類似的 GNU 專案 groff 可處理使用手冊的原始碼。

2. groff 輔助說明

```

1 [dywang@dyw219 make1]$ groff --help
usage: groff [-abceghilpstvzCENRSUVXZ] [-Fdir] [-mname] [-Tdev] [-ffam]
3 [-wname] [-Wname] [-Mdir] [-dcs] [-rcn] [-nnum] [-olist] [-Parg]
 [-Larg] [-Idir] [files...]
5
6 -h print this message
7 -t preprocess with tbl
8 -p preprocess with pic
9 -e preprocess with eqn
10 -g preprocess with grn
11 -G preprocess with grap
12 -s preprocess with soelim
13 -R preprocess with refer
14 -Tdev use device dev
15 -X use X11 previewer rather than usual postprocessor
16 -mname read macros tmac.name
17 -dcs define a string c as s
18 -rcn define a number register c as n
19 -nnum number first page n
20 -olist output only pages in list
21 -ffam use fam as the default font family
22 -Fdir search dir for device directories
23 -Mdir search dir for macro files
24 -v print version number
25 -z suppress formatted output
26 -Z don't postprocess
27 -a produce ASCII description of output
28 -i read standard input after named input files
29 -wname enable warning name
30 -Wname inhibit warning name
31 -E inhibit all errors
32 -b print backtraces with errors or warnings
33 -l spool the output
34 -c disable color output
35 -C enable compatibility mode
36 -V print commands on stdout instead of running them
37 -Parg pass arg to the postprocessor

```

```

39 -Larg pass arg to the spooler
 -N don't allow newlines within eqn delimiters
 -S enable safer mode (the default)
41 -U enable unsafe mode
 -Idir search dir for soelim. Implies -s

```

### 3. groff 選項簡要說明：

```

2 -Tascii：產生 ASCII 文字。
 -Tps：產生 PostScript。
 -man：告訴 groff 輸入一個使用手冊。

```

### 4. 將使用手冊 main.1 以 ASCII 文字產生：

```

1 [dywang@dyw219 make1]$ groff -Tascii -man main.1
 MAIN(1)
 MAIN(1)
3 NAME
 Main - A simple demonstration application that does very little.
5 SYNOPSIS
 main [-option ...]
7 DESCRIPTION
 main is a complete application that does nothing useful.
 It was written for demonstration purposes.
9 OPTIONS
11 It doesn't have any, but let's pretend, to make this template
 complete:
 -option
13 If there was an option, it would not be -option.
15 RESOURCES
 main uses almost no resources.
17 DIAGNOSTICS
 The program shouldn't output anything, so if you find it
 doing so
 there's probably something wrong. The return value is zero.
19 SEE ALSO
 The only other program we know with this this little
 functionality is
21 the ubiquitous hello world application.
23 COPYRIGHT
 main is Copyright (c) 2019
25 BUGS
 There probably are some, but we don't know what they are yet.
27 AUTHORS
 ABC123

```

MAIN(1)

## 14.3 man 查詢手冊

1. Linux 的使用手冊都放在 `/usr/share/man` 目錄下，次目錄 `man1` 下放一般使用者可存取的 1 號手冊，`man5` 放設定檔相關的 5 號手冊，依此類推，而且都經過壓縮。例如 `man` 的 1 號手冊 `man.1.gz`。

```
2 [dywang@dyw219 make1]$ ll /usr/share/man/man1/man.1.gz
-rw-r--r--. 1 root root 5009 Mar 22 2017 /usr/share/man/man1/man.1.gz
```

2. 先以指令 `gzip` 將使用手冊 `main.1` 壓縮成 `main.1.gz`。

```
2 [dywang@dyw219 make1]$ gzip main.1
[dywang@dyw219 make1]$ ll main.1.gz
-rw-rw-r--. 1 dywang dywang 533 Apr 28 11:01 main.1.gz
```

3. 使用 `root` 權限將壓縮之使用手冊 `main.1.gz` 放到 `/usr/share/man/man1` 目錄下。

```
1 [dywang@dyw219 make1]$ sudo cp main.1.gz /usr/share/man/man1/
```

4. 再執行 `man main` 就可以查詢應用程式 `main` 的手冊了。

```
1 [dywang@dyw219 make1]$ man main
```

## 14.4 實機練習題

1. 共同要求：

- (a) 在家目錄下建立 `zzz` 目錄。
- (b) 切換工作目錄到 `zzz`，所有練習皆存在此目錄。
- (c) 下載檔案 `make1.tar.gz`。
- (d) 解打包壓縮 `make1.tar.gz`。
- (e) 進入 `make1` 工作目錄。
- (f) 下載檔案 `main.1`。

- (g) 題目沒要求的，不要多做。
2. main.1 更名為 mfm1.1。
  3. 修改 mfm1.1：在 section RESOURCES 內容改為 `https://dywang.csie.cyut.edu.tw`。
  4. 使用 groff 處理 mfm1.1，產生 PostScript 並存成 mfm1.1.ps 檔。
  5. 使用 groff 處理 mfm1.1，產生 ASCII 文字並存成 mfm1.1.txt 檔。
  6. 寫一 makefile mfm1，要求如下：
    - (a) 承上章的 mfv6
    - (b) 增加目標項目 man，壓縮 mfm1.1 成 mfm1.1.gz，並保留原始檔 mfm1.1。
    - (c) 增加目標項目 maninst，將 mfm1.1.gz 安裝到 /tmp/dywang/man/man1，當安裝目錄不存在時產生目錄。
    - (d) 判斷安裝目錄 /tmp/dywang/man/man1 是否存在？不存在則產生，且不顯示此執行過程。
    - (e) 使用 man /tmp/dywang/man/man1/mfm1.1.gz 查詢手冊。
  7. 寫一手冊 mfm2.1 及 makefile mfm2，要求如下：
    - (a) mfm2.1 承上題 mfm1.1
    - (b) 修改 mfm2.1：AUTHORS 改為您的 sid 學號姓名檔一樣的內容，但不包含中文姓名，且設定為粗體字，行尾改為原先字型設定。提示：改變字型以 `\f escape sequence` 開頭接字型設定值，返回原先設定是 `\fP`。
    - (c) makefile mfm2 承上題 mfm1
    - (d) mfm2 中所有 mfm1 都改成 mfm2
    - (e) 將 mfm2.1 設成變數 MAN。
    - (f) 手冊安裝目錄 /tmp/dywang/man/man1 設成變數 MANINST。
  8. 寫一手冊 mfm3.1 及 makefile mfm3，要求如下：
    - (a) makefile mfm3 承上題 mfm2
    - (b) mfm3 中所有 mfm2 都改成 mfm3
    - (c) 將目標項目 man 及 maninst 中可以使用常用變數 `$<` 的部分，都用變數 `$<` 取代。
    - (d) 修改 mfm3.1：AUTHORS 改為今天日期 (格式為 yyyy-mm-dd)，且設定為粗體字，行尾改回原字型設定。

# Chapter 15

## Makefile 函式庫

### 15.1 函式庫

#### 1. 靜態函式庫：

- (a) 附檔名：通常為 `libxxx.a` 的類型；
- (b) 編譯行為：整個函式庫的資料被整合到執行檔中，所以利用編譯成的檔案會比較大。
- (c) 獨立執行的狀態：編譯成功的可執行檔可以獨立執行，而不需要再向外部要求讀取函式庫的內容。
- (d) 升級難易度：函式庫升級後，連執行檔也需要重新編譯過一次，將新的函式庫整合到執行檔當中。

#### 2. 動態函式庫：

- (a) 附檔名：通常為 `libxxx.so` 的類型；
- (b) 編譯行為：編譯時執行檔中僅具有指向動態函式庫所在的指標而已，並不包含函式庫的內容，所以檔案會比較小。
- (c) 獨立執行的狀態：不能被獨立執行，程式讀取函式庫時，函式庫『必須要存在』，且函式庫的『所在目錄也不能改變』。
- (d) 升級難易度：函式庫升級後，執行檔不需要進行重新編譯，故目前的 Linux distribution 比較傾向使用動態函式庫。

### 15.2 ldd 函式庫工具

#### 1. Linux 放置函式庫之目錄：

```
1 [dywang@dyw219 make1]$ ll -d /lib{,64} /usr/lib{,64}
dr-xr-xr-x. 12 root root 4096 Nov 29 18:10 /lib
3 dr-xr-xr-x. 10 root root 12288 Sep 29 2018 /lib64
dr-xr-xr-x. 33 root root 4096 Nov 29 18:10 /usr/lib
5 dr-xr-xr-x. 146 root root 77824 Feb 14 03:33 /usr/lib64
```

## 2. ldd 輔助說明

```
1 [dywang@dyw219 make1]$ ldd --help
Usage: ldd [OPTION]... FILE...
3 --help print this help and exit
 --version print version information and exit
5 -d, --data-relocs process data relocations
 -r, --function-relocs process data and function relocations
7 -u, --unused print unused direct dependencies
 -v, --verbose print all information
9
For bug reporting instructions, please see:
11 <http://www.gnu.org/software/libc/bugs.html>.
```

## 3. 找出執行檔 main 使用的那些動態函式庫？

```
1 [dywang@dyw219 make1]$ ldd main
linux-vdso.so.1 => (0x00007ffe741f9000)
3 libc.so.6 => /lib64/libc.so.6 (0x000000394aa00000)
/lib64/ld-linux-x86-64.so.2 (0x000055836ce6b000)
```

## 4. 再找出函式 /lib64/libc.so.6 的相關其他函式庫。

```
1 [dywang@dyw219 make1]$ ldd /lib64/libc.so.6
2 /lib64/ld-linux-x86-64.so.2 (0x0000558d8c22f000)
linux-vdso.so.1 => (0x00007ffd3d158000)
```

# 15.3 ldconfig 函式庫工具

1. ldconfig 可以將動態函式庫載入快取記憶體 ( cache )，以增進動態函式庫的讀取速度。

```
1 [dywang@dyw219 make1]$ ldconfig --help
Usage: ldconfig [OPTION...]
```



```

3 | Configure Dynamic Linker Run Time Bindings.
5 | -c, --format=FORMAT Format to use: new, old or compat (default)
 | -C CACHE Use CACHE as cache file
7 | -f CONF Use CONF as configuration file
 | -i, --ignore-aux-cache Ignore auxiliary cache file
9 | -l Manually link individual libraries.
 | -n Only process directories specified on the
 | command
11 | line. Don't build cache.
 | -N Don't build cache
13 | -p, --print-cache Print cache
 | -r ROOT Change to and use ROOT as root directory
15 | -v, --verbose Generate verbose messages
 | -X Don't generate links
17 | -?, --help Give this help list
 | --usage Give a short usage message
19 | -V, --version Print program version
21 | Mandatory or optional arguments to long options are also mandatory or
 | optional
 | for any corresponding short options.
23 |
25 | For bug reporting instructions, please see:
 | <http://www.gnu.org/software/libc/bugs.html>.

```

## 2. 參數簡要說明

```

1 | -f conf : conf 預設為 /etc/ld.so.conf
 | -C cache : cache 預設為 /etc/ld.so.cache
3 | -p : 列出目前在 cache 內的資料

```

## 3. /etc/ld.so.conf 記錄要讀入快取記憶體的動態函式庫所在的目錄，顯示檔案內容只有一行，include ld.so.conf.d 目錄下的所有.conf 檔。

```

1 | [dywang@dyw219 make1]$ cat /etc/ld.so.conf
 | include ld.so.conf.d/*.conf
3 | [dywang@dyw219 make1]$ ls /etc/ld.so.conf.d/
 | atlas-x86_64.conf kernel-ml-4.4.5-1.el6.elrepo.x86_64.
 | conf
5 | compat-mysql51-x86_64.conf kernel-ml-4.8.12-1.el6.elrepo.x86_64.
 | conf
 | ctapi-x86_64.conf mysql-x86_64.conf
7 | kernel-2.6.32-696.el6.x86_64.conf qt-x86_64.conf

```

## 4. /etc 目錄必須是 root 才能寫入，所以先轉換身份為 root。

```
1 [dywang@dyw219 make1]$ su -
```

5. 在 `ld.so.conf.d` 目錄下新增一個檔案 `xorg.conf`，寫入 `xorg` 模組儲存位置。

```
1 [root@dyw219 ~]# echo '/usr/lib64/xorg/modules/' >> /etc/ld.so.conf.d/
 xorg.conf
[root@dyw219 ~]# cat /etc/ld.so.conf.d/xorg.conf
3 /usr/lib64/xorg/modules/
```

6. `ldconfig` 以選項 `-p` 查詢目前 `cache` 中沒有 `xorg` 相關的函式庫。

```
1 [root@dyw219 ~]# ldconfig -p | grep xorg
```

7. `ldconfig` 重新載入函式庫到 `cache`。

```
1 [root@dyw219 ~]# ldconfig
```

8. `ldconfig` 以選項 `-p` 再查詢目前 `cache` 中有 `xorg` 相關的函式庫。

```
1 [root@dyw219 ~]# ldconfig -p | grep xorg
 libwfb.so (libc6,x86-64) => /usr/lib64/xorg/modules/libwfb.so
3 libvgahw.so (libc6,x86-64) => /usr/lib64/xorg/modules/libvgahw.so
 libvbe.so (libc6,x86-64) => /usr/lib64/xorg/modules/libvbe.so
5 libshadowfb.so (libc6,x86-64) => /usr/lib64/xorg/modules/libshadowfb
 .so
 libshadow.so (libc6,x86-64) => /usr/lib64/xorg/modules/libshadow.so
7 libint10.so (libc6,x86-64) => /usr/lib64/xorg/modules/libint10.so
 libglamoregl.so (libc6,x86-64) => /usr/lib64/xorg/modules/
 libglamoregl.so
9 libfbdevhw.so (libc6,x86-64) => /usr/lib64/xorg/modules/libfbdevhw.
 so
 libfb.so (libc6,x86-64) => /usr/lib64/xorg/modules/libfb.so
11 libexa.so (libc6,x86-64) => /usr/lib64/xorg/modules/libexa.so
```

9. 刪除 `xorg.conf`，再重新載入，`cache` 中沒有 `xorg` 相關函式庫。

```
1 [root@dyw219 ~]# rm /etc/ld.so.conf.d/xorg.conf
[root@dyw219 ~]# ldconfig
3 [root@dyw219 ~]# ldconfig -p | grep xorg
```

10. 資料同時也記錄一份在檔案 `/etc/ld.so.cache` 中。

```
1 [root@dyw219 ~]# ll /etc/ld.so.cache
-rw-r--r--. 1 root root 98744 Apr 27 20:37 /etc/ld.so.cache
```

## 15.4 建立函式庫工具

### 1. ar 輔助說明

```
[dywang@dywmac ~]$ ar --help
2 Usage: ar [emulation options] [-]{dmpqrstx}[abcDfilMNoPsSTuvV] [--plugin
 <name>] [member-name] [count] archive-file file...
 ar -M [<mri-script>]
4 commands:
 d - delete file(s) from the archive
6 m[ab] - move file(s) in the archive
 p - print file(s) found in the archive
8 q[f] - quick append file(s) to the archive
 r[ab][f][u] - replace existing or insert new file(s) into the archive
10 s - act as ranlib
 t - display contents of archive
12 x[o] - extract file(s) from the archive
command specific modifiers:
14 [a] - put file(s) after [member-name]
 [b] - put file(s) before [member-name] (same as [i])
16 [D] - use zero for timestamps and uids/gids
 [U] - use actual timestamps and uids/gids (default)
18 [N] - use instance [count] of name
 [f] - truncate inserted file names
20 [P] - use full path names when matching
 [o] - preserve original dates
22 [u] - only replace files that are newer than current archive
 contents
generic modifiers:
24 [c] - do not warn if the library had to be created
 [s] - create an archive index (cf. ranlib)
26 [S] - do not build a symbol table
 [T] - make a thin archive
28 [v] - be verbose
 [V] - display the version number
30 @<file> - read options from <file>
 --target=BFDNAME - specify the target object format as BFDNAME
32 optional:
 --plugin <p> - load the specified plugin
34 emulation options:
 No emulation specific options
36 ar: supported targets: elf64-x86-64 elf32-i386 elf32-x86-64 a.out-i386-
 linux pei-i386 pei-x86-64 elf64-l1om elf64-k1om elf64-little elf64-
 big elf32-little elf32-big plugin srec symbolsrec verilog tekhex
```

```
binary ihex
Report bugs to <http://bugzilla.redhat.com/bugzilla/>
```

## 2. ar 選項簡易說明

- 1 **r** : 在函式庫中插入目標檔。當插入的目標檔名已在庫中存在，則替換。  
**v** : 顯示執行操作選項的附加信息。

## 3. nm 輔助說明

```
[dywang@dywmac ~]$ nm --help
2 Usage: nm [option(s)] [file(s)]
 List symbols in [file(s)] (a.out by default).
4 The options are:
 -a, --debug-syms Display debugger-only symbols
6 -A, --print-file-name Print name of the input file before every
 symbol
 -B Same as --format=bsd
8 -C, --demangle[=STYLE] Decode low-level symbol names into user-level
 names
 The STYLE, if specified, can be `auto' (the
 default),
10 `gnu', `lucid', `arm', `hp', `edg', `gnu-v3',
 `java'
 or `gnat'
 --no-demangle Do not demangle low-level symbol names
12 -D, --dynamic Display dynamic symbols instead of normal
 symbols
14 --defined-only Display only defined symbols
 -e (ignored)
16 -f, --format=FORMAT Use the output format FORMAT. FORMAT can be `
 bsd',
 `sysv' or `posix'. The default is `bsd'
18 -g, --extern-only Display only external symbols
 -l, --line-numbers Use debugging information to find a filename
 and
20 line number for each symbol
 -n, --numeric-sort Sort symbols numerically by address
22 -o Same as -A
 -p, --no-sort Do not sort the symbols
24 -P, --portability Same as --format=posix
 -r, --reverse-sort Reverse the sense of the sort
26 --plugin NAME Load the specified plugin
 -S, --print-size Print size of defined symbols
28 -s, --print-arnmap Include index for symbols from archive members
 --size-sort Sort symbols by size
30 --special-syms Include special symbols in the output
 --synthetic Display synthetic symbols as well
32 -t, --radix=RADIX Use RADIX for printing symbol values
```

```

34 --target=BFDNAME Specify the target object format as BFDNAME
 -u, --undefined-only Display only undefined symbols
 -X 32_64 (ignored)
36 @FILE Read options from FILE
 -h, --help Display this information
38 -V, --version Display this program's version number

40 nm: supported targets: elf64-x86-64 elf32-i386 elf32-x86-64 a.out-i386-
 linux pei-i386 pei-x86-64 elf64-l1om elf64-k1om elf64-little elf64-
 big elf32-little elf32-big plugin srec symbolsrec verilog tekhex
 binary ihex
 Report bugs to <http://bugzilla.redhat.com/bugzilla/>.

```

4. ar 將 2.o 3.o 兩個目標檔案組合成一個函式庫檔案 mylib.a。

```

1 [dywang@dyw219 make1]$ ar rv mylib.a 2.o 3.o
ar: creating mylib.a
3 a - 2.o
 a - 3.o

```

5. nm：查看目標檔庫檔案 mylib.a 的內容，包含了 2.o 及 3.o。

```

2 [dywang@dyw219 make1]$ nm mylib.a
2
2.0:
4 0000000000000000 T function_two
 U puts
6
3.0:
8 0000000000000000 T function_three

```

## 15.5 管理函式庫實例

1. 先複製 makefile2 成 makefile3，新增變數 MYLIB = mylib.a，新增目標項目 mylib.a，其相依項目包含 2.o 及 3.o，但沒有給產生法則。

```

2 [dywang@dyw219 make1]$ cp makefile2 makefile3
2 [dywang@dyw219 make1]$ vim makefile3
[dywang@dyw219 make1]$ grep MYLIB -A1 makefile3
4 MYLIB = mylib.a

6 main: main.o $(MYLIB)
 $(CC) -o $@ $(OBJS)
8 $(MYLIB): $(MYLIB)(2.o) $(MYLIB)(3.o)

```

```
main.o: main.c a.h
```

2. 查詢以 `makefile3` 執行 `make` 的預設法則，過濾變數 `AR = ar`, `ARFLAGS = rv`。

```
1 [dywang@dyw219 make1]$ make -p -f makefile3 | sed '/^#/d' | sed '/^$/d'
 | grep "^AR.*=" --color
ARFLAGS = rv
3 AR = ar
```

3. 再查詢以 `makefile3` 執行 `make` 的預設法則，過濾 `mylib.a`，找到目標項目 `mylib.a(2.o)`, `mylib.a(3.o)` 的預設法則是 `$(AR) $(ARFLAGS) $@ $<`。其中變數為：

- (a) `$(AR)`: `ar`
- (b) `$(ARFLAGS)`: `rv`
- (c) `$<` 代表目前的相依性項目，目標項目 `mylib.a(2.o)` 的相依項目就是 `2.o`；目標項目 `mylib.a(3.o)` 的相依項目則是 `3.o`。
- (d) `$@` 代表目前的目標項目，就是 `mylib.a` 函式庫。

```
1 [dywang@dyw219 make1]$ make -p -f makefile3 | sed '/^#/d' | sed '/^$/d'
 | grep -A1 "mylib.a" --color
MYLIB = mylib.a
3 COMPILE.C = $(COMPILE.cc)
--
5 mylib.a: mylib.a(2.o) mylib.a(3.o)
3.c:
7 mylib.a(3.o): 3.o
 $(AR) $(ARFLAGS) $@ $<
9 --
main: main.o mylib.a
11 $(CC) -o $@ $(OBJS)
--
13 mylib.a(2.o): 2.o
 $(AR) $(ARFLAGS) $@ $<
```

4. 若 `mylib.a` 存在就先刪除，再以 `makefile3` 執行 `make`，可以驗證上述法則確實執行。

```
1 [dywang@dyw219 make1]$ rm mylib.a
2 [dywang@dyw219 make1]$ make -f makefile3
```

```
ar rv mylib.a 2.o
4 ar: creating mylib.a
 a - 2.o
6 ar rv mylib.a 3.o
 a - 3.o
8 gcc -o main main.o 2.o 3.o
```

5. 刪除 3.o，再執行一次 make，與 3.o 相關的程式都必須重新產生，包含 mylib.a。

```
[dywang@dyw219 make1]$ rm 3.o
2 [dywang@dyw219 make1]$ make -f makefile3
gcc -Wall -c 3.c
4 ar rv mylib.a 3.o
 r - 3.o
6 gcc -o main main.o 2.o 3.o
```

## 15.6 實機練習題

- 共同要求：
  - 在家目錄下建立 zzz/make1 目錄。
  - 切換工作目錄到 zzz/make1，所有練習皆存在此目錄。
  - 檔案導向都是執行.sh 腳本命令時導向，不是寫在腳本內導向。檔案導向到 /tmp/\$mysid.\$evalname 目錄，其中變數 \$mysid 為自己的學號，\$evalname 為評分程式名稱不加附檔名，目錄若不存在，必須自行建立。
- 查詢執行檔 /usr/bin/passwd 使用的動態函式庫，只取函式庫名稱，並導向到 ldd1.so。
- 查詢函式檔 /lib64/libcrypt.so.1 相關的函式庫，只取函式庫名稱，並導向到 ldd2.so。
- 查詢 cache 中的函式庫，過濾 art 字串，並導向到 art1.txt。
- 將 2.o 及 3.o 兩個目標檔組合成一個函式庫檔案 myar1.a。
- nm 查看 myar1.a 的內容，並導向到 mynm1.txt。
- make 查詢 AR 開頭的變數或法則，並導向到 myar2.txt。

8. 寫一 makefile myar2，要求如下：

- (a) 增加目標項目 myar2.bin，由 main.o 2.o 3.o 三個檔產生可執行檔 myar2.bin。
- (b) 增加目標項目 myar2lib.a，將 2.o 3.o 組合成 mya2rlib.a。
- (c) 以 makefile myar2 查詢預設法則，去除註解行及空白行後過濾 myar2lib 字串，顯示包含下一行，並導向到 myar2lib.txt。

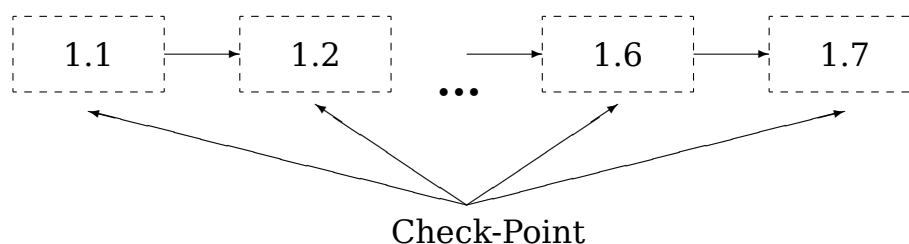


# Chapter 16

## \*RCS 版本控制系統

### 16.1 版本控制

- 版本控制 Version Control( VC )
  1. VC 是指發展程式文件檔案時，檔案版本的儲存和管理的功能。
  2. 程式發展過程，需要重複的修改、暫存、測試以期得到正確的結果。
  3. 每次修改的結果分以不同的檔案儲存起來，以避免重要訊息的流失和除錯時的困難。
  4. RCS ( Revision Control System ) 為提供 VC 環境的工具。
- RCS 的功能
  1. 提供一些命令來管理原始碼檔案。
  2. 透過一個檔案來管理一系列的修改動作，檔案紀錄足以恢復先前的版本。
  3. 只儲存不同版本之間的修正部分，所以非常節省空間。
  4. 能比較檔案各版本之間的不同。
  5. 將一些完成除錯的版本內容與其它檔案版本整合 ( merge )。
- 產生檢查點 ( Check-Point ) 與除錯
  1. 設立檢查點：在檔案完成階段目標後，將其內容當成一個版本，給予一個版本序號 (revision number) 存起來。



2. 除錯 ( debug ) 時可取出先前的版本，找出錯誤 ( bug ) 發生前的階段，
3. 比較錯誤 ( bug ) 發生前後版本間的差異，進而找出錯誤發生的原因。

- 產生標記符號

1. 專案發展需要結合數個程式檔案，每個版本的檔案個數不一定相同。
2. 不同的專案亦可能用到相同的檔案。
3. RCS 除了以版本序號代表該版本外，也可產生標記符號 ( mark symbol ) 來代表某些特別的版本。
4. 版本序號通常是由 RCS 自動編排，而標記符號則需要使用者自己針對特別的版本產生。
5. 若需要指出某些版本為相關 ( 例如：同專案 )，則可給序特定符號以供辨識。

- 版本比較與合併

1. RCS 能比較檔案各版本之不同，幫助使用者瞭解版本間的差異。
2. 檔案版本的比較之訊息，往往是除錯的重要訊息。
3. RCS 也可將兩個版本的差異與正在工作的檔案版本整合 ( merge )。

- 存取名單

1. 支援多人工作環境中檔案存取權的管理以及避免同時編輯相同檔案的發生。
2. 提供存取名單 ( access list ) 的設立，記錄有存取權使用者。
3. 存取名單設定
  - (a) 記錄在存取名單的使用者才有讀寫檔案的權限。
  - (b) 存取名單內的使用者同時具有讀、寫、執行三種權限。

- 鎖檔

1. 鎖檔 ( lock ) 的功能主要避免同時多人寫入同一檔案。
2. 一個檔案同時只能有一個使用者做編輯的動作;

3. 在 RCS 中若要對檔案做修改的動作，必先鎖住 ( lock ) 該檔案;
4. 被鎖住的檔案，不可以再被其它使用者鎖住。
5. 鎖檔方式:
  - (a) strick lock：只有被鎖住的檔案版本，才能有寫入的動作，此為預設鎖檔方式。
  - (b) nonstrick lock：可寫入檔案的任一版本。因容易發生錯誤，所以通常不與考慮使用。

### 練習題

1. 何謂版本控制 Version Control( VC )？

Sol. 指發展程式文件檔案時，檔案版本的儲存和管理的功能。

2. 發展程式文件檔案時，為什麼要做版本控制？

Sol. 程式發展過程，需要重複的修改、暫存、測試以期得到正確的結果，而每次修改的結果均以不同的檔案儲存起來，以避免重要訊息的流失和除錯時的困難。

3. 請簡述三項 RCS 功能。

Sol. 1. 提供命令管理原始碼檔案。2. 能比較檔案各版本之間的不同。3. 能將完成除錯的版本與其它版本整合。

4. 在檔案完成階段目標後，RCS 會在此設立檢查點，將其內容存起來，並給予一個號碼，稱之為何？

Sol. 版本序號 (revision number)

5. RCS 中除了自動編排的版本序號外，使用者還可以依個別需求自訂符號，稱之為何？

Sol. 標記符號 ( mark symbol )

6. 請說明 RCS 中，存取名單 ( access list ) 的功能。

Sol. 1. 支援多人工作環境以避免同時編輯相同檔案的發生。2. 存取名單記錄有存取權使用者。

7. 請說明 RCS 中，鎖檔 ( lock ) 的功能。

Sol. 1. 避免同時多人寫入同一檔案。2. 若要對檔案做修改，必先鎖住該檔案。3. 被鎖住的檔案，不能再被其它使用者鎖住。

8. 請說明 RCS 中，鎖檔 ( lock ) 的方式。

Sol. 1.strick lock：只有被鎖住的檔案版本，才能有寫入的動作，此為預設鎖檔方式。2.nonstrick lock：可寫入檔案的任一版本。因容易發生錯誤，所以通常不與考慮使用。

9. RCS 指令 `ci` 之功能為何？

Sol. 將檔案版本寫入管理檔案。

10. RCS 指令 `co` 之功能為何？

Sol. 把檔案的某個版本從管理檔案讀出。

11. RCS 中版本主支的序號如何設定？

Sol. 1. 以 `m.n` 為單位，`m` 為主要號碼，`n` 為次要號碼；2. 預設的版本序號從 1.1 開始。

12. RCS 中版本分支的序號如何設定？

Sol. 1. 在原版本的版本序號後加上 `.m.n`。例如：版本 1.2 產生的分支為 1.2.1.1，版本 1.2.1.1 產生的分支為 1.2.1.1.1.1。

13. 請說明 RCS 中，附記（`log`）的功能。

Sol. 版本在寫入時，產生該版本的附記，可以記錄該版本的一些特性。

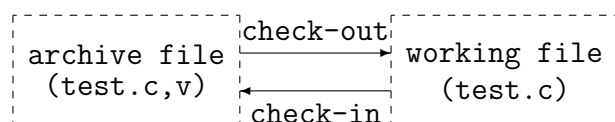
## 16.2 RCS 基本功能

### • 管理檔案（`archive file`）

1. 必先針對個別檔案產生屬於該檔的管理檔案，以儲存該檔案的各個版本。
2. 各檔案 `archive file` 的檔名是在該檔案名後加上 `,v` 來標示。
3. 例如：檔名 `test.c` 的管理檔案即為 `test.c,v`。

### • 讀出（`check-out`）與寫入（`check-in`）

1. 一個檔案的管理檔案存放該檔的各個版本，可以透過讀出（`check-out`）與寫入（`check-in`）動作，取出已存入的版本修改或存入新的版本。



(a) `check in (ci)`：將檔案版本寫入管理檔案的動作。

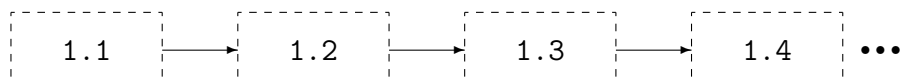
(b) `check out (co)`：把檔案的某個版本從管理檔案讀出的動作。

2. 預設為唯讀狀態取出版本，不可修改。若要修改檔案，必須將其鎖住。

### • 版本序號（`revision number`）

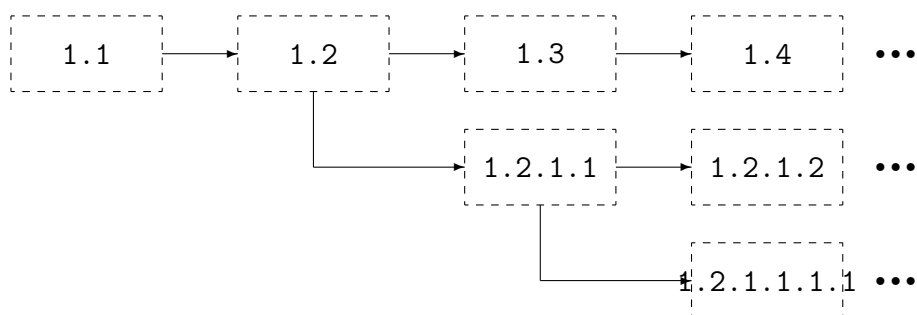
1. 在 RCS 的檔案中每一個版本，都會有一個版本序號以區分各個版本；

2. 對於特定版本的存取都利用版本序號來指定；
3. 版本序號的設定是以 m.n 為單位，m 為主要號碼 (major number)，n 為次要號碼 (minor number)；
4. 預設的版本序號從 1.1 開始。



- 分支 ( branch )

1. 檔案的發展可能因為修改或是目的的不同，會從一個版本產生不同的分支。
2. 分支的序號為在原版本的版本序號後加上.m.n，其中 m 用以區分不同的分支，n 區分在同一分支下的不同版本。
3. 例如：版本 1.2 產生的分支為 1.2.1.1，版本 1.2.1.1 產生的分支為 1.2.1.1.1.1。



- 附記 ( log )

1. 每個版本在寫入 ( check in ) archive file 時，都會要求使用者產生該版本的附記以記錄該版本的一些特性。
2. 例如：為何產生該版本、該版本解決了那些問題、甚至該版本用了那些想法與技巧等等。

- 產生 RCS archive file

1. rcs 命令

```

2 [root@dywHome2 ~]# rcs [-Aaebilnnotu] parameter
 -A file 將其它檔案的可存取名單寫入該檔的可存取名單
 -a name 將使用者寫入可存取名單
 4 -b branch number 設定主支(default branch)
 -e name 將使用者自可存取名單內剔除
 6 -i file 產生及初始化一個新的 RCS 檔案

```

```

-1[number] lock 該版本但不取出
8 -m 更改檔案版本的附注(log) 的內容
-n symbol 產生或去除標記符號(mark symbol)
10 -o[number] 從檔案的 archive file 中移除某版本
-t 改檔案版本的描述(description) 的內容
12 -u[number] 去除該版本 lock 的狀態
u[number] 表示選項 u 後面馬上接版本序號

```

2. 以 vi 編輯 important.c :

```

1 [dywang@dywOffice testrcs]$ vi important.c
/*
3 This is an important file for managing the project.
 It implements the canonical "Hello World" program.
5 */

7 #include <stdlib.h>
 #include <stdio.h>
9
11 int main()
 {
13 printf("Hello World\n");
 exit(EXIT_SUCCESS);
 }

```

3. 初始 RCS 檔案管理：使用 rcs -i 命令初始一個 RCS 管理檔案。

```

[dywang@dywOffice testrcs]$ rcs -i important.c
2 RCS file: important.c,v
 enter description, terminated with single '.' or end of file:
4 NOTE: This is NOT the log message!
 >> This is an important demonstration file
6 >> .
 done
8 # 可以輸入很多註解。要跳離提示符號，必須輸入(.)，或者在檔案字元的結
 尾輸入 Ctrl+D。

```

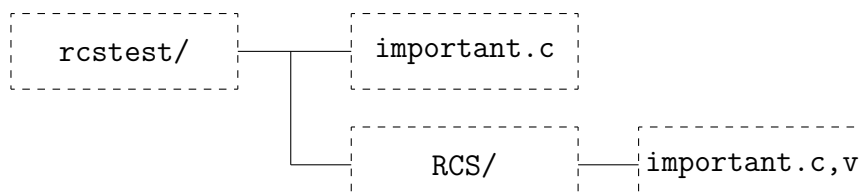
4. rcs 命令後會產生一個新的唯讀檔案，其檔尾為 (,v)：

```

[dywang@dywOffice testrcs]$ ll
2 total 12
 -rw-r--r-- 1 dywang users 226 Mar 5 13:45 important.c
4 -r--r--r-- 1 dywang users 105 Mar 5 13:55 important.c,v

```

5. 如果要將 RCS 檔案放在不同的目錄，可先產生一個 RCS 子目錄。所有的 `rscs` 命令都會自動將 `rscs` 檔案儲存在 RCS 子目錄。



```

[dywang@dywOffice testrcs]$ mkdir RCS
2 [dywang@dywOffice testrcs]$ ll
total 8
4 -rw-r--r-- 1 dywang users 226 Mar 5 13:45 important.c
 drwxr-xr-x 2 dywang users 4096 Mar 5 13:49 RCS/
6
[dywang@dywOffice testrcs]$ rcs -i important.c
8 RCS file: RCS/important.c,v
 enter description, terminated with single '.' or end of file:
10 NOTE: This is NOT the log message!
 >> This is an important demonstration file
12 >> .
 done
14
[dywang@dywOffice testrcs]$ ll . RCS
16 .:
total 8
18 -rw-r--r-- 1 dywang users 226 Mar 5 13:45 important.c
 drwxr-xr-x 2 dywang users 4096 Mar 5 13:49 RCS/
20
RCS:
22 total 4
 -r--r--r-- 1 dywang users 105 Mar 5 13:49 important.c,v

```

6. 查看 `important.c,v` 內容

```

1 [dywang@dywOffice testrcs]$ cat RCS/important.c,v
head ;
3 access;
 symbols;
5 locks; strict;
 comment @ * @;
7
 desc
9 @This is an important demonstration file
 @

```

- 將程式內容寫入 `archive file`

## 1. ci (check in) 命令

```

[root@dywHome2 ~]# ci [-dfwmnrtlu] parameter
2 -d date 給定版本生成時間
 -f 強制產生新版本(不管是否寫入相同版本內容)
4 -w name 給定生成該版本的作者名
 -m log-message 直接以 log-message 寫入附記(log)
6 -n symbol 在寫入 archive file 同時, 給定標記符號(mark symbol
)
 -r[number] 寫入時指定版本的 revision number
8 -t description 直接以檔案的形式寫入描述(description)
 -l 寫入後再取出該版本為工作檔案(lock)
10 -u 寫入後再取出該版本為工作檔案(unlock)

```

## 2. 利用 ci 命令寫入 (check in) 檔案，儲存現在的版本。

```

[dywang@dywOffice testrcs]$ ci important.c
2 RCS/important.c,v <-- important.c
 initial revision: 1.1
4 done

```

## 3. important.c 的內容和管理資訊，會被存在 important.c,v 中。

```

[dywang@dywOffice testrcs]$ ll . RCS
2 .:
 total 4
4 drwxr-xr-x 2 dywang users 4096 Mar 5 14:06 RCS/
6 RCS:
 total 4
8 -r--r--r-- 1 dywang users 446 Mar 5 14:06 important.c,v

```

## 4. 查看 important.c,v 內容

```

[dywang@dywOffice testrcs]$ cat RCS/important.c,v
2 head 1.1;
 access;
4 symbols;
 locks; strict;
6 comment @ * @;

8 1.1
 date 2008.03.05.06.06.15; author dywang; state Exp;
10 branches;
 next ;
12 desc

```



```

14 |@This is an important demonstration file
 |@
16 |
 |1.1
18 |log
 |@Initial revision
20 |@
 |text
22 |@/*
 | This is an important file for managing the project.
24 | It implements the canonical "Hello World" program.
 |*/
26 |
 |#include <stdlib.h>
28 |#include <stdio.h>
30 |int main()
 |{
32 | printf("Hello World\n");
 | exit(EXIT_SUCCESS);
34 |}
36 |@

```

- 產生新版本：將工作檔案從 archive file 讀出、修改，再寫入 archive file

#### 1. co (check out) 命令

```

[root@dywHome2 ~]# co [-dfjlprwu] parameter
2 -d date 取出在指定時間前最後生成的版本
 -f 強制取出版本內容取代目前工作檔案
4 -j 取出並整合指定的版本內容
 -l lock 取出版本
6 -p 取出版本內容至標準輸出(standard output)
 -r number 指定取出版本
8 -w name 取出指定作者名之版本
 -u 取出 unlock 的版本內容

```

#### 2. 將檔案 important.c 讀出 (check out)。

```

1 [dywang@dywOffice testrcs]$ co important.c
 RCS/important.c,v --> important.c
3 revision 1.1
 done
5 [dywang@dywOffice testrcs]$ ll
 total 8
7 -r--r--r-- 1 dywang users 226 Mar 5 14:15 important.c
 drwxr-xr-x 2 dywang users 4096 Mar 5 14:06 RCS/

```

3. 將檔案 `important.c` 讀出 (check out)，以 `co -l` 鎖定取出版本。

```
[dywang@dywOffice testrcs]$ co -l important.c
2 RCS/important.c,v --> important.c
 revision 1.1 (locked)
4 done
[dywang@dywOffice testrcs]$ ll
6 total 8
-rw-r--r-- 1 dywang users 226 Mar 5 14:18 important.c
8 drwxr-xr-x 2 dywang users 4096 Mar 5 14:18 RCS/
```

4. 編輯 `important.c`，加入一行，並儲存新的版本：

```
[dywang@dywOffice testrcs]$ vi important.c
2 [dywang@dywOffice testrcs]$ cat important.c
/*
4 This is an important file for managing the project.
 It implements the canonical "Hello World" program.
6 */
8 #include <stdlib.h>
 #include <stdio.h>
10
12 int main()
 {
14 printf("Hello World\n");
 printf("This is an extra line added later\n");
16 exit(EXIT_SUCCESS);
 }
```

5. 利用 `ci` 命令儲存改變處：

```
[dywang@dywOffice testrcs]$ ci important.c
2 RCS/important.c,v <-- important.c
 new revision: 1.2; previous revision: 1.1
4 enter log message, terminated with single '.' or end of file:
 >> Added an extra line to be printed out
6 >> .
 done
```

6. 查看目錄看到 `important.c` 再次被刪除。

```
1 [dywang@dywOffice testrcs]$ ll . RCS
 .:
3 total 8
-rw-r--r-- 1 dywang users 226 Mar 5 14:18 important.c~
5 drwxr-xr-x 2 dywang users 4096 Mar 5 14:24 RCS/
```

```

7 RCS:
 total 4
9 -r--r--r-- 1 dywang users 639 Mar 5 14:24 important.c,v

```

### 7. 查看 important.c,v 内容

```

1 [dywang@dywOffice testrcs]$ cat RCS/important.c,v
 head 1.2;
3 access;
 symbols;
5 locks; strict;
 comment @ * @;
7
 1.2
9 date 2008.03.05.06.23.52; author dywang; state Exp;
 branches;
11 next 1.1;
13
 1.1
14 date 2008.03.05.06.06.15; author dywang; state Exp;
15 branches;
16 next ;
17
 desc
19 @This is an important demonstration file
 @
21
 1.2
23 log
 @Added an extra line to be printed out
25 @
 text
27 @/*
 This is an important file for managing the project.
29 It implements the canonical "Hello World" program.
 */
31
 #include <stdlib.h>
33 #include <stdio.h>
35 int main()
 {
37 printf("Hello World\n");
 printf("This is an extra line added later\n");
39 exit(EXIT_SUCCESS);
 }
41
 @
43
 1.1
45 log

```

```

47 | @Initial revision
 | @
 | text
49 | @d12 1
 | @

```

- 加上自己對該版本的附註 ( log )

1. check in 時 RCS 會自動要求輸入:

(a) 初始 RCS archive file

```

[dywang@dywOffice testrcs]$ rcs -i important.c
2 RCS file: important.c,v
 enter description, terminated with single '.' or end of file:
4 NOTE: This is NOT the log message!
 >> This is an important demonstration file
6 >> .
 done
8 # 可以輸入很多註解。要跳離提示符號，必須輸入(.)，或者在檔案字元
 的結尾輸入 Ctrl+D。

```

(b) 寫入 archive file

```

[dywang@dywOffice testrcs]$ ci important.c
2 RCS/important.c,v <-- important.c
 new revision: 1.2; previous revision: 1.1
4 enter log message, terminated with single '.' or end of file:
 >> Added an extra line to be printed out
6 >> .
 done

```

2. 使用指令 `ci -m"comment" file.c`，將版本附註 `comment` 直接寫入 archive file 中，而不會再產生提示訊息。

```

1 [dywang@dywOffice testrcs]$ co -l important.c
 RCS/important.c,v --> important.c
3 revision 1.3
 done
5 [dywang@dywOffice testrcs]$ vi important.c
[dywang@dywOffice testrcs]$ ci -m"test for comment" important.c
7 RCS/important.c,v <-- important.c
 new revision: 1.4; previous revision: 1.3
9 done
[dywang@dywOffice testrcs]$ rlog -r1.4 important.c
11 RCS file: RCS/important.c,v

```

```

13 Working file: important.c
 head: 1.4
15 branch:
 locks: strict
17 access list:
 symbolic names:
19 keyword substitution: kv
 total revisions: 5; selected revisions: 1
21 description:
 This is an important demonstration file
23 -----
 revision 1.4
25 date: 2008/03/07 02:30:48; author: dywang; state: Exp; lines: +1 -0
 test for comment
27 -----

```

- 查詢檔案各版本附記 ( log )

### 1. rlog 命令

```

1 [root@dywHome2 ~]# rlog [-bhLRt] parameter
-b 列出檔案主分支(default branch)的附記(log)一覽表
3 -h 列出檔案附記(log)的標頭(header)
-L 若該檔有被 lock 則列出各版本附記(log)一覽表
5 -R 列出檔案的 archive file 路徑與名稱
-t 列出檔案附記(log)的標頭(header)+ 描述(description)

```

### 2. 透過 rlog 命令查看檔案修改的摘要。

```

2 [dywang@dywOffice testrcs]$ rlog important.c
4 RCS file: RCS/important.c,v
 Working file: important.c
 head: 1.2
6 branch:
 locks: strict
8 access list:
 symbolic names:
10 keyword substitution: kv
 total revisions: 2; selected revisions: 2
12 description:
 This is an important demonstration file
14 -----
 revision 1.2
16 date: 2008/03/05 06:23:52; author: dywang; state: Exp; lines: +1 -0
 Added an extra line to be printed out
18 -----
 revision 1.1
20 date: 2008/03/05 06:06:15; author: dywang; state: Exp;

```

```

22 Initial revision
=====
在 1.2 版中的第一行行尾 lines:+1 -0 表示增加了一行，但沒有刪除任何行。

```

3. 透過 `rlog` 命令查看檔案附記 ( `log` ) 的標頭 ( `header` )。

```

1 [dywang@dywOffice testrcs]$ rlog -h important.c
3 RCS file: RCS/important.c,v
 Working file: important.c
5 head: 1.2
 branch:
7 locks: strict
 access list:
9 symbolic names:
 keyword substitution: kv
11 total revisions: 2
=====

```

4. 透過 `rlog` 命令查看檔案的 `archive file` 路徑與名稱。

```

2 [dywang@dywOffice testrcs]$ rlog -R important.c
 RCS/important.c,v

```

5. 透過 `rlog` 命令查看檔案附記 ( `log` ) 的標頭 ( `header` ) + 描述 ( `description` )。

```

2 [dywang@dywOffice testrcs]$ rlog -t important.c
4 RCS file: RCS/important.c,v
 Working file: important.c
6 head: 1.2
 branch:
8 locks: strict
 access list:
10 symbolic names:
 keyword substitution: kv
 total revisions: 2
12 description:
 This is an important demonstration file
14 =====

```

- 解決檔案未鎖住之錯誤

1. 讀出 `important.c`，但未鎖住。

```
[dywang@dywOffice testrcs]$ co important.c
2 RCS/important.c,v --> important.c
 revision 1.2
4 done
[dywang@dywOffice testrcs]$ ll
6 total 12
-r--r--r-- 1 dywang users 276 Mar 6 09:56 important.c
8 -rw-r--r-- 1 dywang users 277 Mar 5 21:09 important.c~
drwxr-xr-x 2 dywang users 4096 Mar 6 09:56 RCS/
```

2. 修改 `important.c`，並強制存檔後，無法寫入 archive file。

```
1 [dywang@dywOffice testrcs]$ vi important.c
[dywang@dywOffice testrcs]$ ci important.c
3 RCS/important.c,v <-- important.c
 ci: RCS/important.c,v: no lock set by dywang
```

3. 以 `rcs` 鎖住後，即可寫入 archive file。

```
[dywang@dywOffice testrcs]$ rcs -l1.2 important.c
2 RCS file: RCS/important.c,v
 1.2 locked
4 done
[dywang@dywOffice testrcs]$ ci important.c
6 RCS/important.c,v <-- important.c
 new revision: 1.3; previous revision: 1.2
8 enter log message, terminated with single '.' or end of file:
>> test for lock
10 >> .
 done
```

- 版本之刪除

1. 讀出 `important.c`，並鎖住。

```
1 [dywang@dywOffice testrcs]$ co -l important.c
RCS/important.c,v --> important.c
3 revision 1.2 (locked)
 done
```

2. 修改 `important.c`，存檔後，指定為原先版本（1.2），則無法寫入 archive file。

```
[dywang@dywOffice testrcs]$ vi important.c
2 [dywang@dywOffice testrcs]$ ci -r1.2 important.c
RCS/important.c,v <-- important.c
4 ci: RCS/important.c,v: revision 1.2 too low; must be higher than
 1.2
```

3. 寫入 archive file，版本為 1.3。

```
[dywang@dywOffice testrcs]$ ci important.c
2 RCS/important.c,v <-- important.c
new revision: 1.3; previous revision: 1.2
4 enter log message, terminated with single '.' or end of file:
>> test for lock
6 >> .
done
```

4. 重新讀出 important.c ( 版本為 1.3 )，並鎖住，修改後存檔。

```
1 [dywang@dywOffice testrcs]$ co -l important.c
RCS/important.c,v --> important.c
3 revision 1.3 (locked)
done
5 [dywang@dywOffice testrcs]$ vi important.c
```

5. 先移除原先版本 ( 1.3 )。

```
1 [dywang@dywOffice testrcs]$ rcs -o1.3 important.c
RCS file: RCS/important.c,v
3 rcs: RCS/important.c,v: can't remove locked revision 1.3
[dywang@dywOffice testrcs]$ rcs -u1.3 important.c
5 RCS file: RCS/important.c,v
1.3 unlocked
7 done
[dywang@dywOffice testrcs]$ rcs -o1.3 important.c
9 RCS file: RCS/important.c,v
deleting revision 1.3
11 done
```

6. 最後寫入 archive file，版本仍為 1.3。

```
1 [dywang@dywOffice testrcs]$ ci important.c
RCS/important.c,v <-- important.c
3 ci: RCS/important.c,v: no lock set by dywang
[dywang@dywOffice testrcs]$ rcs -l1.2 important.c
5 RCS file: RCS/important.c,v
```



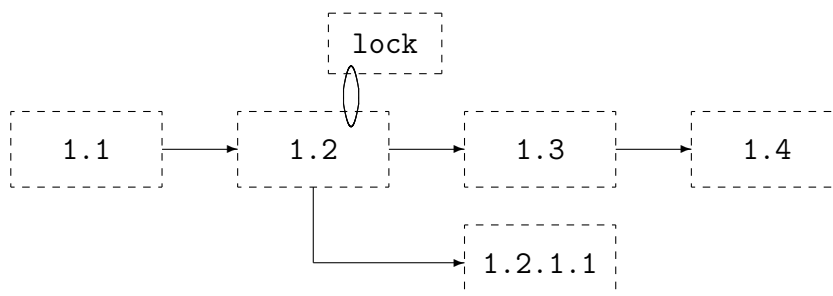
```

1.2 locked
7 done
[dywang@dywOffice testrcs]$ ci important.c
9 RCS/important.c,v <-- important.c
 new revision: 1.3; previous revision: 1.2
11 enter log message, terminated with single '.' or end of file:
 >> test for deletes
13 >> .
 done

```

- 產生分支 ( branch )

如果在版本 1.2 產生版本 1.3 之後，又修改版本 1.2 的內容而欲生成新的版本，則需在版本 1.2 處產生分支 ( branch ) 1.2.1.1。



1. 取出可寫入的版本 1.2

```

[dywang@dywOffice testrcs]$ co -r1.2 -l important.c
2 RCS/important.c,v --> important.c
 revision 1.2 (locked)
4 done

```

2. 如果沒有變動，直接寫入，不會產生版本 1.2.1.1

```

[dywang@dywOffice testrcs]$ ci important.c
2 RCS/important.c,v <-- important.c
 file is unchanged; reverting to previous revision 1.2
4 done
[dywang@dywOffice testrcs]$ ll
6 total 8
 -rw-r--r-- 1 dywang users 276 Mar 6 10:12 important.c~
8 drwxr-xr-x 2 dywang users 4096 Mar 6 13:05 RCS/

```

3. 重新讀出，並更改版本 1.2 的內容

```

[dywang@dywOffice testrcs]$ co -r1.2 -l important.c

```

```

2 RCS/important.c,v --> important.c
 revision 1.2 (locked)
4 done
[dywang@dywOffice testrcs]$ vi important.c
6 [dywang@dywOffice testrcs]$ cat important.c
/*
8 This is an important file for managing the project.
 It implements the canonical "Hello World" program.
10 */
12 #include <stdlib.h>
 #include <stdio.h>
14
16 int main()
 {
18 printf("Hello World\n");
 printf("This is an extra line added later\n");
20 printf("test for branch\n");
 exit(EXIT_SUCCESS);
 }

```

#### 4. 寫入產生版本 1.2.1.1

```

1 [dywang@dywOffice testrcs]$ ci important.c
 RCS/important.c,v <-- important.c
3 new revision: 1.2.1.1; previous revision: 1.2
 enter log message, terminated with single '.' or end of file:
5 >> test for branch
 >> .
7 done

```

#### 5. 查看版本訊息

```

[dywang@dywOffice testrcs]$ rlog important.c
3 RCS file: RCS/important.c,v
 Working file: important.c
5 head: 1.3
 branch:
7 locks: strict
 access list:
9 symbolic names:
 keyword substitution: kv
11 total revisions: 4; selected revisions: 4
 description:
13 This is an important demonstration file

15 revision 1.3
 date: 2008/03/06 03:17:00; author: dywang; state: Exp; lines: +1 -0
17 test for deletes

19 revision 1.2

```

```

21 | date: 2008/03/05 06:23:52; author: dywang; state: Exp; lines: +1 -0
 | branches: 1.2.1;
 | Added an extra line to be printed out
23 | -----
 | revision 1.1
25 | date: 2008/03/05 06:06:15; author: dywang; state: Exp;
 | Initial revision
27 | -----
 | revision 1.2.1.1
29 | date: 2008/03/06 05:07:38; author: dywang; state: Exp; lines: +1 -0
 | test for branch
31 | =====

```

### 練習題

1. 如何以 `rcs` 指令初始程式 `important.c` 的 RCS 管理檔案。  
Sol. `rcs -i important.c`
2. 若程式檔名為 `important.c`，則其 RCS 管理檔案（archive file）檔名為何？  
Sol. `important.c,v`
3. 以 `rcs` 指令初始 RCS 管理檔案或以 `ci` 指令寫入 RCS 管理檔案，會被要求輸入附記，請問如何結束輸入？  
Sol. 輸入一點（.），或按鍵 `Ctrl+D`。
4. 如果希望 RCS 的管理檔案自動存放在一個子目錄，則必須建立怎樣的子目錄？  
Sol. 檔名為 `RCS`（不可為小寫）的子目錄。
5. 如何將程式 `important.c` 從 RCS 管理檔案中讀出。  
Sol. `co important.c`
6. 如何將程式 `important.c` 從 RCS 管理檔案中讀出，且鎖定取出版本。  
Sol. `co -l important.c`
7. 如果從 RCS 管理檔案中讀出的程式 `important.c` 屬性為 `-r--r--r--`，則取出版本是否鎖定？  
Sol. 已鎖定
8. 如果從 RCS 管理檔案中讀出的程式 `important.c` 屬性為 `-rw-r--r--`，則取出版本是否鎖定？  
Sol. 已鎖定

9. 將修改後的程式 `important.c` 存入 RCS 管理檔案後，檔案 `important.c` 是否存在？

Sol. 不存在。

10. 有那兩種狀況，RCS 會自動要求輸入版本的附註（`log`）？

Sol. 1. 初始 RCS 管理檔案；2. 寫入管理檔案。

11. 如何使用指令 `ci`，將程式 `important.c` 的版本附註 "test for log" 直接寫入管理檔案中，而不會再產生提示訊息？

Sol. `ci -m "test for log" important.c`

12. 如何使用指令 `rlog`，查詢程式 `important.c` 修改的摘要？

Sol. `rlog important.c`

13. 使用指令 `rlog`，查詢程式 `important.c` 修改的摘要，在版本 1.2 中，有一行行尾出現 "lines: +1 -0"，代表意義為何？

Sol. 表示增加了一行，但沒有刪除任何行。

14. 如何使用指令 `rlog`，查詢程式 `important.c` 附記的標頭（`header`）？

Sol. `rlog -h important.c`

15. 如何使用指令 `rlog`，查詢程式 `important.c` 之管理檔案之路徑與名稱？

Sol. `rlog -R important.c`

16. 如何使用指令 `rlog`，查詢程式 `important.c` 附記的標頭（`header`）+ 描述（`description`）？

Sol. `rlog -t important.c`

17. 使用 `rlog` 指令，查看檔案 `important.c` 附註時，出現 `locks: strict`，代表意義為何？

Sol. 鎖住方式為 `strict`。

18. 使用 `rlog` 指令，查看檔案 `important.c` 附註時，出現 `keyword substitution: kv`，代表意義為何？

Sol. 識別關鍵字中解釋方式為 `kv`。

19. 如果以指令 `co important.c` 讀出程式 `important.c`，修改後以指令寫入 RCS 管理檔案，會出現什麼狀況？

Sol. 沒有鎖住，故無法存入。

20. 當從 RCS 管理檔案中讀出版本 1.2 之程式 `important.c`，但未鎖住，修改後要如何寫入 RCS 管理檔案？

Sol. 先執行指令 `rcs -l1.2 important.c` 鎖住，再寫入。

21. 如何從 RCS 管理檔案中，刪除版本 1.3 之程式 important.c？

Sol. `rcs -d1.3 important.c`

22. 如何從 RCS 管理檔案中，解除版本 1.3 之程式 important.c 鎖住？

Sol. `rcs -u1.3 important.c`

23. 如果在版本 1.2 產生版本 1.3 之後，要版本 1.2 處產生分支，則分支之序號為何？

Sol. `1.2.1.1`

24. 如果從 RCS 管理檔案中，讀出版本 1.3 之程式 important.c，未經修改的情況下寫入管理檔案，是否會產生新的版本？

Sol. 不會

25. 如果從 RCS 管理檔案中，最新版本為 1.6，現在讀出版本 1.4 之程式 important.c，修改後再寫入管理檔案，產生的新版本序號為何？

Sol. `1.4.1.1`

## 16.3 識別關鍵字串

- RCS 提供識別關鍵字串 ( IdKeyword )，幫助使用者產生各版本的相關資訊，這些識別關鍵字串都是用符號 \$ 夾住以供 RCS 辨識。
- 各 IdKeyword 代表意義：

| keyword      | 相關資訊                                                                                |
|--------------|-------------------------------------------------------------------------------------|
| -----        |                                                                                     |
| \$Author\$   | 寫入該版本的作者                                                                            |
| \$Date\$     | 日期和時間( UTC )                                                                        |
| \$Header\$   | RCS 檔名(含路徑)+版本(Revision)+日期(Date)<br>+作者(Author)+狀態(State)+正 lock 該檔案者(Locker)      |
| \$Id\$       | 除 RCS 檔名不含路徑外，餘與 \$Header\$ 相同                                                      |
| \$Locker\$   | 目前 lock 住該檔案者，如未 lock，則空白                                                           |
| \$Log\$      | 關於該版本的 log 的訊息                                                                      |
| \$Name\$     | 標記名稱<br>例如：執行 <code>co -r first</code> ，則 \$Name\$ 展開為 <code>`Name: first'</code> 。 |
| \$RCSfile\$  | RCS 檔名(不含路徑)                                                                        |
| \$Revision\$ | 版本序號                                                                                |
| \$Source\$   | RCS 檔名(含路徑)                                                                         |
| \$State\$    | 狀態：Exp (for experimental), Stab (for stable),                                       |

and Rel (for released)

- IdKeyword 的使用：通常加在程式檔案的檔頭

1. 讀出並鎖定 important.c。

```
1 [dywang@dywOffice testrcs]$ co -l important.c
RCS/important.c,v --> important.c
3 revision 1.4 (locked)
done
```

2. 在 important.c 的檔頭加上 \$Id\$

```
[dywang@dywOffice testrcs]$ vi important.c
2 [dywang@dywOffice testrcs]$ cat important.c
/*
4 * Id
 This is an important file for managing the project.
6 It implements the canonical "Hello World" program.
*/
8 以下省略
```

3. 將 important.c 寫入 archive file。

```
[dywang@dywOffice testrcs]$ ci -m"test for IdKeyword" important.c
2 RCS/important.c,v <-- important.c
new revision: 1.5; previous revision: 1.4
4 done
```

4. 再將 important.c 從 archive file 讀出，則 \$Id\$ 代表的相關資訊，會被加在 \$ 符號裡面。

```
[dywang@dywOffice testrcs]$ co -l important.c
2 RCS/important.c,v --> important.c
revision 1.5 (locked)
4 done
[dywang@dywOffice testrcs]$ cat important.c
6 /*
 * $Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $
8 This is an important file for managing the project.
 It implements the canonical "Hello World" program.
10 */
 以下省略
12 # RCS 檔名(不含路徑) : important.c,v
 # 版本(Revision) : 1.5
14 # 日期與時間(Date) : 2008/03/07 03:30:31
```

```

16 # 作者(Author) : dywang
 # 狀態(State) : Exp
 # 目前 lock 住該檔案者(locker) : dywang

```

- RCS 如何來解釋各個識別關鍵字串 ( IdKeyword )

| Option | 解釋方式                | 說明                |
|--------|---------------------|-------------------|
| -kkv   | keyword+相關資訊        | default 的解釋方式     |
| -kkvl  | keyword+相關資訊+locker | 如 -kkv 再加上 locker |
| -kk    | keyword             | 只顯示 keyword 不加解釋  |
| -kv    | 相關資訊                | 只顯示相關資訊不加 keyword |

1. 以 -kk 選項登出檔案 ( unlock )

```

1 [dywang@dywOffice testrcs]$ co -kk important.c
 RCS/important.c,v --> important.c
3 revision 1.5
 done
5 [dywang@dywOffice testrcs]$ cat important.c
 /*
7 * Id
 This is an important file for managing the project.
9 It implements the canonical "Hello World" program.
 */
11 以下省略

```

2. 以 -kv 選項登出檔案 ( unlock )

```

1 [dywang@dywOffice testrcs]$ co -kv important.c
 RCS/important.c,v --> important.c
3 revision 1.5
 done
5 [dywang@dywOffice testrcs]$ cat important.c
 /*
7 * important.c,v 1.5 2008/03/07 03:30:31 dywang Exp
 This is an important file for managing the project.
9 It implements the canonical "Hello World" program.
 */
11 以下省略

```

- ident 命令：找出檔案中之 IdKeyword。

1. IdKeyword 只存在於註解中，則編譯完之目標檔 important.o 及可執行檔 important 中不會有 IdKeyword

```

1 [dywang@dywOffice testrcs]$ gcc -c important.c; gcc -o important
 important.o
[dywang@dywOffice testrcs]$ ident important.c important.o important
3 important.c:
 $Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $
5
7 important.o:
 ident warning: no id keywords in important.o
9
10 important:
 ident warning: no id keywords in important

```

2. 編輯程式 important.c，加入字串變數 rcsid[] = "\$Id\$"

```

[dywang@dywOffice testrcs]$ vi important.c
2 [dywang@dywOffice testrcs]$ cat important.c
/*
4 * $Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $
 This is an important file for managing the project.
6 It implements the canonical "Hello World" program.
*/
8
9 #include <stdlib.h>
10 #include <stdio.h>
 static char const rcsid[] = "Id";
12
13 int main()
14 {
 printf("Hello World\n");
16 printf("This is an extra line added later\n");
 printf("test for lock\n");
18 printf("test for comment\n");
 printf("%s\n", rcsid);
20 exit(EXIT_SUCCESS);
}

```

3. 寫入 important.c,v 後再讀出，得到 \$Id\$ 之相關資訊

```

1 [dywang@dywOffice testrcs]$ ci important.c
 RCS/important.c,v <-- important.c
3 new revision: 1.6; previous revision: 1.5
 enter log message, terminated with single '.' or end of file:
5 >> test ident on object files
 >> .
7 done
[dywang@dywOffice testrcs]$ co -l important.c

```



```

9 RCS/important.c,v --> important.c
revision 1.6 (locked)
11 done
[dywang@dywOffice testrcs]$ cat important.c
13 /*
 * $Id: important.c,v 1.6 2008/03/12 01:51:38 dywang Exp dywang $
15 This is an important file for managing the project.
 It implements the canonical "Hello World" program.
17 */

19 #include <stdlib.h>
#include <stdio.h>
21 static char const rcsid[] =
 "$Id: important.c,v 1.6 2008/03/12 01:51:38 dywang Exp dywang $
 ";
23
25 int main()
{
 printf("Hello World\n");
27 printf("This is an extra line added later\n");
 printf("test for lock\n");
29 printf("test for comment\n");
 printf("%s\n", rcsid);
31 exit(EXIT_SUCCESS);
}

```

4. 編譯 `important.c`，產生目標檔 `important.o` 及可執行檔 `important`。

```

[dywang@dywOffice testrcs]$ gcc -c important.c; gcc -o important
important.o

```

5. `IdKeyword` 字串已被結合到目標檔 `important.o` 及可執行檔 `important` 之中。以指令 `ident` 查看：

```

1 [dywang@dywOffice testrcs]$ ident important.c important.o important
important.c:
3 $Id: important.c,v 1.6 2008/03/12 01:51:38 dywang Exp dywang $
 $Id: important.c,v 1.6 2008/03/12 01:51:38 dywang Exp dywang $
5
important.o:
7 $Id: important.c,v 1.6 2008/03/12 01:51:38 dywang Exp dywang $
9
important:
 $Id: important.c,v 1.6 2008/03/12 01:51:38 dywang Exp dywang $

```

- `make` 配合 `RCS` 產生可執行檔

## 1. 刪除 important.c

```
[dywang@dywOffice testrcs]$ rm -f important.c
```

## 2. 若 important.c 不存在，則 make 會利用 RCS，從 important.c,v 讀出最新版本，產生 important.c，並進行編譯。

```
1 [dywang@dywOffice testrcs]$ make important
2 co RCS/important.c,v important.c
3 RCS/important.c,v --> important.c
4 revision 1.5
5 done
6 cc -c -o important.o important.c
7 cc important.o -o important
8 rm important.o important.c
```

## 3. make 在編譯產生可執行檔 important 後，再刪除 important.o 及 important.c 檔案。

```
[dywang@dywOffice testrcs]$ ll
total 20
-rw-r--r-- 1 dywang users 21 Mar 7 10:16 commentfile
-rwxr-xr-x 1 dywang users 6850 Mar 11 15:23 important*
-rw-r--r-- 1 dywang users 341 Mar 7 11:29 important.c~
drwxr-xr-x 2 dywang users 4096 Mar 11 15:23 RCS/
```

**練習題**

## 1. RCS 識別關鍵字串，使用什麼符號夾住以供 RCS 辨識？

Sol. `$`2. RCS 識別關鍵字串 `$Author$`，產生什麼資訊？

Sol. 寫入該版本的作者

3. RCS 識別關鍵字串 `$Date$`，產生什麼資訊？

Sol. 日期和時間 (UTC)

4. RCS 識別關鍵字串 `$Header$`，產生什麼資訊？

Sol. RCS 檔名 (含路徑)+ 版本 (Revision)+ 日期 (Date)+ 作者 (Author)+ 狀態 (State)+ 正鎖住該檔案者 (Locker)

5. RCS 識別關鍵字串 `$Id$`，產生什麼資訊？

Sol. RCS 檔名 (不含路徑) + 版本 (Revision) + 日期 (Date) + 作者 (Author) + 狀態 (State) + 正鎖住該檔案者 (Locker)

6. RCS 識別關鍵字串 `$Locker$`，產生什麼資訊？

Sol. 目前鎖住該檔案者，如未鎖住，則空白

7. RCS 識別關鍵字串 `$Log$`，產生什麼資訊？

Sol. 關於該版本的附註訊息

8. RCS 識別關鍵字串 `$Name$`，產生什麼資訊？

Sol. 標記名稱

9. RCS 識別關鍵字串 `$RCSfile$`，產生什麼資訊？

Sol. RCS 檔名 (不含路徑)

10. RCS 識別關鍵字串 `$Revision$`，產生什麼資訊？

Sol. 版本序號

11. RCS 識別關鍵字串 `$Source$`，產生什麼資訊？

Sol. RCS 檔名 (含路徑)

12. RCS 識別關鍵字串 `$State$`，產生什麼資訊？

Sol. 狀態 (實驗、穩定或釋出)

13. RCS 識別關鍵字串通常加在什麼地方？

Sol. 程式檔案的檔頭

14. 在程式中加入 RCS 識別關鍵字串 `$Id$`，寫入管理檔案後再讀出，出現 `$Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $`，其中 `important.c,v` 表示什麼訊息？

Sol. RCS 檔名 (不含路徑) 為 `important.c,v`

15. 在程式中加入 RCS 識別關鍵字串 `$Id$`，寫入管理檔案後再讀出，出現 `$Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $`，其中 1.5 表示什麼訊息？

Sol. 版本序號 1.5

16. 在程式中加入 RCS 識別關鍵字串 `$Id$`，寫入管理檔案後再讀出，出現 `$Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $`，其中 2008/03/07 03:30:31 表示什麼訊息？

Sol. 日期 2008/03/07，時間 03:30:31

17. 在程式中加入 RCS 識別關鍵字串 `$Id$`，寫入管理檔案後再讀出，出現 `$Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $`，其中第一次出現之 `dywang` 表示什麼訊息？

Sol. 作者 (Author) 為 `dywang`

18. 在程式中加入 RCS 識別關鍵字串 `$Id$`，寫入管理檔案後再讀出，出現 `$Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $`，其中 `Exp` 表示什麼訊息？

Sol. 版本狀態為實驗階段

19. 在程式中加入 RCS 識別關鍵字串 `$Id$`，寫入管理檔案後再讀出，出現 `$Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp dywang $`，其中第二次出現之 `dywang` 表示什麼訊息？

Sol. 目前居住該檔案者為 `dywang`

20. 從 RCS 管理檔案中讀出程式 `important.c` 時，如何以 `keyword+` 相關資訊，顯示識別關鍵字串？

Sol. `co -kvw important.c`

21. 從 RCS 管理檔案中讀出程式 `important.c` 時，預設之識別關鍵字串解釋方式為何？

Sol. `co -kvw important.c`，以 `keyword+` 相關資訊，顯示識別關鍵字串。

22. 從 RCS 管理檔案中讀出程式 `important.c` 時，如何以 `keyword+` 相關資訊 `+locker`，顯示識別關鍵字串？

Sol. `co -kvw1 important.c`

23. 從 RCS 管理檔案中讀出程式 `important.c` 時，如何只以 `keyword`，顯示識別關鍵字串？

Sol. `co -kw important.c`

24. 從 RCS 管理檔案中讀出程式 `important.c` 時，如何只以相關資訊，顯示識別關鍵字串？

Sol. `co -kv important.c`

25. 若識別關鍵字串只存在於程式 `important.c` 之註解中，則編譯完之目標檔 `important.o` 及可執行檔 `important` 中會不會有識別關鍵字串之相關資訊？

Sol. 不會

26. 若在程式 `important.c` 中加入變數 `char rcsfile[] = "$RCSfile$";`，寫入管理檔案後，再以預設解釋方式讀出、編譯，則目標檔 `important.o` 及可執行檔

important 中之變數 rcsfile[] 內容為何？

Sol. `$RCS important.c,v`

27. 若 RCS 管理檔案 `important.c,v` 存在，但程式 `important.c` 並未讀出，是否可以直接使用 `make` 指令編譯可執行檔 `important`？為什麼？

Sol. 可以。make 會從 `important.c,v` 讀出最新版本，產生 `important.c`，並進行編譯。在編譯產生可執行檔 `important` 後，再刪除 `important.c` 及 `important.c` 檔案。

## 16.4 標記符號

- 何謂標記符號 ( mark symbol )？
  1. 標記符號是指一個版本除 revision number 外賦予的符號名稱。
  2. 當版本很多時，若各個版本只以 revision number 區分，則不易找到其中特定的版本。
  3. 因為只從 revision number 很難知道該版本的內容，故 RCS 提供自訂版本標記符號之功能，以利了解各版本之關係。
- 如何產生標記符號：
  1. 產生指令 `ci -n` 加入或之後用 `rcs -n` 加入，一般建議的形式是用數字跟下底線 "\_" 來建構標記符號，例如 `sym1_0 sym1_0_1` 等。

```
2 [dywang@dywOffice testrcs]$ rcs -nmarkSymbol1_6:1.6 important.c
RCS file: RCS/important.c,v
done
```

2. 除 RCS 的特殊字元不能使用 ( 包括 \$ (錢字號) ,(逗號) ,(句點) ;(分號) :(冒號) 以及 @ 等符號) 外，不限制符號的形式。

```
1 [dywang@dywOffice testrcs]$ ci -n$markSymbol_1 important.c
ci: missing symbolic name after -n
3 [dywang@dywOffice testrcs]$ ci -nmark,Symbol_1 important.c
ci: invalid symbol `mark,Symbol_1'
5 ci aborted
[dywang@dywOffice testrcs]$ ci -nmark.Symbol_1 important.c
7 ci: invalid symbol `mark.Symbol_1'
ci aborted
9 [dywang@dywOffice testrcs]$ ci -nmark@Symbol_1 important.c
ci: invalid symbol `mark@Symbol_1'
11 ci aborted
[dywang@dywOffice testrcs]$ ci -nmark$Symbol_1 important.c
```

```

13 RCS/important.c,v <-- important.c
 file is unchanged; reverting to previous revision 1.6
15 done

```

### 3. 以 rlog 指令查詢產生的標記符號

```
[dywang@dywOffice testrcs]$ rlog -h important.c
```

```

RCS file: RCS/important.c,v
Working file: important.c
head: 1.6
branch:
locks: strict
access list:
symbolic names:
 mark: 1.6 %*<== 因使用符號 $，造成錯誤(原為 mark$Symbol_1*)
 markSymbol1_6: 1.6
keyword substitution: kv
total revisions: 7
=====

```

### 4. 可在一個版本產生多個標記符號。例如在版本 1.2 處產生標記符號 r1\_0 及 r1\_0\_head:

```

1 [dywang@dywOffice testrcs]$ rcs -nr1_0:1.2 -nr1_0_head:1.2
 important.c
 RCS file: RCS/important.c,v
3 done

```

### 5. 再以 rlog 指令查詢產生的標記符號

```

1 [dywang@dywOffice testrcs]$ rlog -h important.c
3 RCS file: RCS/important.c,v
 Working file: important.c
5 head: 1.6
 branch:
7 locks: strict
 access list:
9 symbolic names:
 r1_0_head: 1.2
11 r1_0: 1.2

```

```

13 mark: 1.6
 markSymbol1_6: 1.6
keyword substitution: kv
15 total revisions: 7
=====

```

• 如何取消標記符號？

1. 以 `rsc` 指令取消標記符號：-n 直接接要取消的標記，而不接:[revision number]

```

2 [dywang@dywOffice testrcs]$ rsc -nmark important.c
 RCS file: RCS/important.c,v
 done

```

2. 以 `rlog` 指令查詢產生的標記符號。

```

1 [dywang@dywOffice testrcs]$ rlog -h important.c
3 RCS file: RCS/important.c,v
 Working file: important.c
5 head: 1.6
 branch:
7 locks: strict
 access list:
9 symbolic names:
 r1_0_head: 1.2
11 r1_0: 1.2
 markSymbol1_6: 1.6
13 keyword substitution: kv
 total revisions: 7
15 =====

```

3. 以 `rsc` 指令一次取消多個標記符號

```

1 [dywang@dywOffice testrcs]$ rsc -nr1_0 -nr1_0_head important.c
 RCS file: RCS/important.c,v
3 done

```

4. 再以 `rlog` 指令查詢，只剩一個標記符號。

```

1 [dywang@dywOffice testrcs]$ rlog -h important.c
3 RCS file: RCS/important.c,v

```

```

Working file: important.c
5 head: 1.6
 branch:
7 locks: strict
 access list:
9 symbolic names:
 markSymbol1_6: 1.6
11 keyword substitution: kv
 total revisions: 7
13 =====

```

• 標記符號的好處：

1. 標記符號比 revision number 更能傳達該版本的特性，例如：若版本 1.3 與 1.5 相關，可分別給予同樣式的標記符號 r1\_3 與 r1\_5 標示。

```

1 [dywang@dywOffice testrcs]$ rcs -nr1_3:1.3 -nr1_5:1.5 important.c
 RCS file: RCS/important.c,v
3 done

```

2. 以 rlog 指令查詢產生的標記符號。

```

1 [dywang@dywOffice testrcs]$ rlog -h important.c
3 RCS file: RCS/important.c,v
 Working file: important.c
5 head: 1.6
 branch:
7 locks: strict
 access list:
9 symbolic names:
 r1_5: 1.5
11 r1_3: 1.3
 markSymbol1_6: 1.6
13 keyword substitution: kv
 total revisions: 7
15 =====

```

### 練習題

1. 何謂 RSC 標記符號 ( mark symbol )？

Sol. 標記符號是指一個版本除版本序號外賦予的符號名稱。

2. RCS 為什麼要使用標記符號 ( mark symbol )？

Sol. 因從版本序號很難知道該版本的內容，故 RCS 提供自訂版本標記符號之功能，以了解各版本之關係。



3. 如何使用 rcs 指令，於程式檔案 important.c 版本 1.5 中，加入標記符號 markS1\_5？

Sol. `rcs -nmarkS1_5:1.5 important.c`

4. 如何使用 rcs 指令，於程式檔案 important.c 版本 1.6 中，加入標記符號 markS1\_6？

Sol. `rcs -nmarkS1_6:1.6 important.c`

5. 如何使用 ci 指令，於程式檔案 important.c 版本 1.6 中，加入標記符號 markS1\_6？

Sol. `ci -nmarkS1_6:1.6 important.c`

6. 那些特殊字元，不可使用於 RCS 標記符號？

Sol. `$`，`;`，`:`，`@` 六個符號。

7. 指令 `ci -n$mS_1 important.c`，是否正確？若有錯，請說明為什麼？

Sol. 錯誤，標記符號不可使用 `$` 符號。

8. 指令 `ci -nm,S_1 important.c`，是否正確？若有錯，請說明為什麼？

Sol. 錯誤，標記符號不可使用 `,` 符號。

9. 指令 `ci -nm.S_1 important.c`，是否正確？若有錯，請說明為什麼？

Sol. 錯誤，標記符號不可使用 `.` 符號。

10. 指令 `ci -nm;S_1 important.c`，是否正確？若有錯，請說明為什麼？

Sol. 錯誤，標記符號不可使用 `;` 符號。

11. 指令 `ci -nm@S_1 important.c`，是否正確？若有錯，請說明為什麼？

Sol. 錯誤，標記符號不可使用 `@` 符號。

12. 如何使用 rcs 指令，於程式檔案 important.c 版本 1.6 中，同時加入兩個標記符號 markS1\_6 及 markS1\_6\_0？

Sol. `rcs -nmarkS1_6:1.6 -nmarkS1_6_0:1.6 important.c`

13. 如何使用 rcs 指令，同時於程式檔案 important.c，版本 1.3 中加入標記符號 r1\_3 及版本 1.5 中加入標記符號 r1\_5？

Sol. `rcs -nr1_3:1.3 -nr1_5:1.5 important.c`

14. 執行 `rlog -h important.c`，出現 symbolic names: r1\_0\_head: 1.2，代表意義為何？

Sol. 表示版本 1.2 有標記符號 r1\_0\_head。

15. 如何使用 `rcs` 指令，取消程式檔案 `important.c` 中之標記符號 `markS1_6`？

Sol. `rcs -markS1_6 important.c`

16. 如何使用 `rcs` 指令，取消程式檔案 `important.c` 中之標記符號 `markS1_6_0`？

Sol. `rcs -markS1_6_0 important.c`

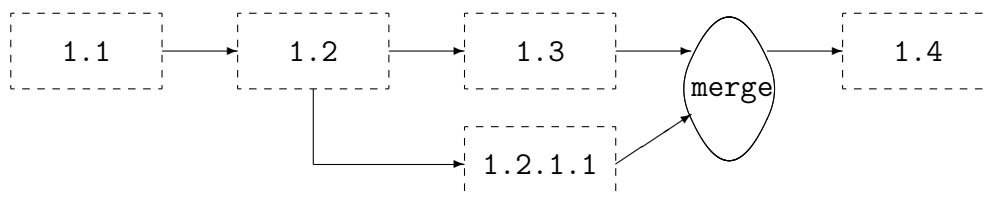
17. 如何使用 `rcs` 指令，一次取消程式檔案 `important.c` 中之標記符號 `r1_0` 及 `r1_0_head`？

Sol. `rcs -r1_0 -r1_0_head important.c`

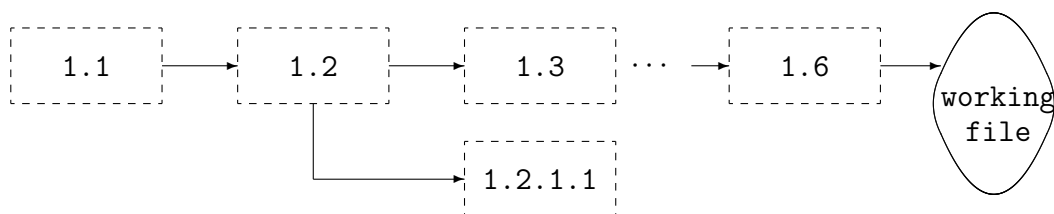
## 16.5 版本比較與整合

• 為何要比較與整合版本？例如：

1. 在版本 1.3 發現錯誤 ( bug )；
2. 回頭找尋錯誤發生的階段，發現錯誤是在版本 1.2；
3. 在除錯完成後產生了校正版本 1.2.1.1。
4. 如果版本 1.3 已經完成某個階段性的目標且棄之可惜，則可用整合 ( merge ) 的功能將版本 1.2.1.1 和版本 1.2 的不同處與版本 1.3 整合起來。



• 比較版本的不同



1. `rcsdiff` 命令：比較版本內容的差異 (包括 IDKeyword)，以瞭解兩個版本間哪些被改變。

```

1 [root@dywHome2 ~]# rcsdiff [-kkkkv1r]
 -kk 比較版本內容的差異(不包括 IDKeyword)
3 -kkv1 指定 IDKeyword 的比較方式，預設比較方式。
 -r 指定欲比較版本

```

2. 都不指定版本時，rcsdiff 會比較目前工作檔案與主幹（default branch）上最後存入的一個版本。

```

[dywang@dywOffice testrcs]$ rcsdiff important.c
=====
RCS file: RCS/important.c,v
4 retrieving revision 1.6
diff -r1.6 important.c

```

3. 目前工作檔案與任一版本的比較

```

1 [dywang@dywOffice testrcs]$ rcsdiff -r1.3 important.c
=====
3 RCS file: RCS/important.c,v
 retrieving revision 1.3
5 diff -r1.3 important.c
 1a2
7 > * $Id: important.c,v 1.6 2008/03/12 01:51:38 dywang Exp dywang $
 # 1.3 版的第 1 行後，插入了新的第 2 行。
9 7a9
 > static char const rcsid[] = "$Id: important.c,v 1.6 2008/03/12
 01:51:38
11 dywang Exp dywang $";
 # 1.3 版的第 7 行後，插入了新的第 9 行。
13 13a16,17
 > printf("test for comment\n");
15 > printf("%s\n", rcsid);
 # 1.3 版的第 7 行後，插入了新的第 16,17 行。

```

4. 比較任意兩個版本：

```

[dywang@dywOffice testrcs]$ rcsdiff -r1.3 -r1.4 important.c
=====
RCS file: RCS/important.c,v
4 retrieving revision 1.3
 retrieving revision 1.4
6 diff -r1.3 -r1.4
 13a14
8 > printf("test for comment\n");
 # 1.3 版的第 13 行後，插入了新的第 14 行。

```

## 5. 不比較 IdKeyword

```

1 [dywang@dywOffice testrcs]$ rcsdiff -r1.5 -r1.6 -kk important.c
=====
3 RCS file: RCS/important.c,v
 retrieving revision 1.5
5 retrieving revision 1.6
 diff -r1.5 -r1.6
7 8a9
 > static char const rcsid[] = "Id";
9 15a17
 > printf("%s\n", rcsid);

```

## 6. 預設比較 IdKeyword

```

2 [dywang@dywOffice testrcs]$ rcsdiff -r1.5 -r1.6 -kkvl important.c
=====
4 RCS file: RCS/important.c,v
 retrieving revision 1.5
6 retrieving revision 1.6
 diff -r1.5 -r1.6
8 2c2
 < * $Id: important.c,v 1.5 2008/03/07 03:30:31 dywang Exp $

10 > * $Id: important.c,v 1.6 2008/03/12 01:51:38 dywang Exp $
 8a9
12 > static char const rcsid[] = "$Id: important.c,v 1.6 2008/03/12
 01:51:38
 dywang Exp $";
14 15a17
 > printf("%s\n", rcsid);

```

## • 版本合併

1. rcsmerge 命令：將兩版本內容的差異整合到工作檔案。

```

1 [root@dywHome2 ~]# rcsmerge [-kkkkvlr]
 -kk 整合檔案版本(不包括 IDKeyword)
3 -kkvl 指定 IDKeyword 的整合方式
 -r 整合檔案版本(包括 IDKeyword)

```

2. 使用指令 rcsmerge：需先取出要寫入的版本

```

2 [dywang@dywOffice testrcs]$ co -l -r1.6 important.c
 RCS/important.c,v --> important.c
 revision 1.6 (locked)

```

```

4 done
[dywang@dywOffice testrcs]$ rcsmerge -r1.2 -r1.2.1.1 important.c
6 RCS file: RCS/important.c,v
 retrieving revision 1.2
8 retrieving revision 1.2.1.1
 Merging differences between 1.2 and 1.2.1.1 into important.c
10 rcsmerge: warning: conflicts during merge

```

3. 使用指令 `co -j` : 在從 archive file 中取出工作檔案時即整合之

```

[dywang@dywOffice testrcs]$ co -l -r1.6 -j1.2:1.2.1.1 important.c
2 RCS/important.c,v --> important.c
 revision 1.6 (locked)
4 revision 1.2
 revision 1.2.1.1
6 merging...
 merge: warning: conflicts during merge
8 done

```

4. 使用指令 `co -j` : 一次整合多個檔案的相異處

```

[dywang@dywOffice testrcs]$ co -l -r1.6 -j1.2:1.2.1.1,1.3:1.4
 important.c
2 RCS/important.c,v --> important.c
 revision 1.6 (locked)
4 revision 1.2
 revision 1.2.1.1
6 merging...
 merge: warning: conflicts during merge
8 revision 1.3
 revision 1.4
10 merging...
 merge: warning: conflicts during merge
12 done

```

### 練習題

1. 請舉例說明，為什麼 RCS 要比較與整合版本？

Sol. 1. 若在版本 1.3 發現 bug；2. 發現錯誤來自版本 1.2；3. 除錯完成產生校正版本 1.2.1.1；4. 比較版本 1.2.1.1 和版本 1.2 的不同，並與版本 1.3 整合。

2. 如何使用 `rcsdiff` 指令，比較目前工作檔案 `important.c` 與主幹上最後存入的一個版本？

Sol. `rcsdiff important.c` 或 `rcsdiff -kkvl important.c`

3. 如何使用 `rcsdiff` 指令，比較目前工作檔案 `important.c` 與主幹上最後存入的一個版本，且不包括識別關鍵字串？

Sol. `rcsdiff -kk important.c`

4. 如何使用 `rcsdiff` 指令，比較目前工作檔案 `important.c` 與版本 1.3？

Sol. `rcsdiff -r1.3 important.c`

5. 如何使用 `rcsdiff` 指令，比較目前工作檔案 `important.c` 與版本 1.3，且不包括識別關鍵字串？

Sol. `rcsdiff -r1.3-kk important.c`

6. 使用 `rcsdiff` 指令，出現訊息 `1a2` 代表意義為何？

Sol. 兩版本比較差異為：第 1 行後，插入了新的第 2 行。

7. 使用 `rcsdiff` 指令，出現訊息 `13a16,17` 代表意義為何？

Sol. 兩版本比較差異為：第 13 行後，插入了新的第 16 及 17 行。

8. 如何使用 `rcsdiff` 指令，比較工作檔案 `important.c` 版本 1.3 與 1.4？

Sol. `rcsdiff -r1.3-r1.4 important.c`

9. 如何使用 `rcsmerge` 指令，將檔案 `important.c` 版本 1.2 與 1.2.1.1 之差異，整合至目前工作檔案？

Sol. `rcsmerge -r1.2 -r1.2.1.1 important.c`

10. 如何使用 `rcsmerge` 指令，將檔案 `important.c` 版本 1.2 與 1.2.1.1 之差異，整合至目前工作檔案，且不包括識別關鍵字串？

Sol. `rcsmerge -r1.2 -r1.2.1.1 -kk important.c`

11. 如何使用 `co` 指令，將檔案 `important.c` 版本 1.2 與 1.2.1.1 之差異，在鎖定方式讀出時即整合至目前工作檔案？

Sol. `co -l -j1.2:1.2.1.1 important.c`

12. 如何使用 `co` 指令，一次將檔案 `important.c` 版本 1.2 與 1.2.1.1 之差異及版本 1.3 與 1.4 之差異，在鎖定方式讀出時即整合至目前工作檔案？

Sol. `co -l -j1.2:1.2.1.1 1.3:1.4 important.c`

## 16.6 存取名單

- 設定存取名單：只有記錄在存取名單中的使用者，才能存取該檔案的版本。

- 將使用者加入可存取名單

```

[dywang@dywOffice testrcs]$ rcs -adywang important.c
2 RCS file: RCS/important.c,v
 done
4 [dywang@dywOffice testrcs]$ cd ../testrcs1
[dywang@dywOffice testrcs1]$ rcs -adywtest f.c
6 RCS file: f.c,v
 done

```

## 2. 查看存取名單

```

1 [dywang@dywOffice testrcs1]$ rlog -h f.c
3 RCS file: f.c,v
 Working file: f.c
5 head: 1.1
 branch:
7 locks: strict
 access list:
9 dywtest
 symbolic names:
11 abc_1: 1.1
 keyword substitution: kv
13 total revisions: 1
 =====
15 [dywang@dywOffice testrcs]$ cd ../testrcs
[dywang@dywOffice testrcs]$ rlog -h important.c
17
18 RCS file: RCS/important.c,v
19 Working file: important.c
20 head: 1.6
21 branch:
22 locks: strict
23 access list:
24 dywang
25 symbolic names:
26 r1_5: 1.5
27 r1_3: 1.3
28 markSymbol1_6: 1.6
29 keyword substitution: kv
30 total revisions: 7
31 =====

```

- 複製存取名單：將 f.c 的存取名單複製到 important.c。

### 1. 複製存取名單

```

1 [dywang@dywOffice testrcs]$ rcs -A../testrcs1/f.c important.c
 RCS file: RCS/important.c,v
3 done

```

## 2. 查看存取名單

```
1 [dywang@dywOffice testrcs]$ rlog -h important.c
3 RCS file: RCS/important.c,v
 Working file: important.c
5 head: 1.6
 branch:
7 locks: strict
 access list:
9 dywang
 dywtest
11 symbolic names:
 r1_5: 1.5
13 r1_3: 1.3
 markSymbol1_6: 1.6
15 keyword substitution: kv
 total revisions: 7
17 =====
```

## • 移除存取名單

## 1. 將使用者 dywtest 從存取名單中移除

```
1 [dywang@dywOffice testrcs]$ rcs -edywtest important.c
 RCS file: RCS/important.c,v
3 done
```

## 2. 查看存取名單

```
1 [dywang@dywOffice testrcs]$ rlog -h important.c
3 RCS file: RCS/important.c,v
 Working file: important.c
5 head: 1.6
 branch:
7 locks: strict
 access list:
9 dywang
 symbolic names:
11 r1_5: 1.5
 r1_3: 1.3
13 markSymbol1_6: 1.6
 keyword substitution: kv
15 total revisions: 7
```



```
=====
```

### 3. 將存取名單全部移除

```
2 [dywang@dywOffice testrcs]$ rcs -e important.c
 RCS file: RCS/important.c,v
 done
```

### 4. 查看存取名單

```
1 [dywang@dywOffice testrcs]$ rlog -h important.c
3 RCS file: RCS/important.c,v
 Working file: important.c
5 head: 1.6
 branch:
7 locks: strict
 access list:
9 symbolic names:
 r1_5: 1.5
11 r1_3: 1.3
 markSymbol1_6: 1.6
13 keyword substitution: kv
 total revisions: 7
15 =====
```

## 練習題

### 1. RCS 中存取名單之功能為何？

Sol. 只有記錄在存取名單中的使用者，才能存取該檔案的版本。

### 2. 如何使用 rcs 指令，將使用者 csie 加至檔案 important.c 的存取名單？

Sol. `rcs -acsie important.c`

### 3. 如何使用 rlog 指令，查看檔案 important.c 的存取名單？

Sol. `rlog -h important.c`

### 4. 使用 rlog 指令，查看檔案 important.c 附註時，出現 access list: dywang，代表意義為何？

Sol. 存取名單中有使用者 dywang

### 5. 如何使用 rcs 指令，將 f.c 的存取名單複製到 important.c？

Sol. `rcs -Af.c important.c`

6. 如何使用 `rcs` 指令，將使用者 `dywtest` 從 `important.c` 的存取名單中移除？

Sol. `rcs -edywtest important.c`

7. 如何使用 `rcs` 指令，將所有使用者從 `important.c` 的存取名單中移除？

Sol. `rcs -e important.c`

# Chapter 17

## \* 使用 QT 設計 KDE 視窗程式

### 17.1 KDE 和 QT 介紹

- KDE (K Desktop Environment)
  1. KDE 與 GNOME 是現今 Linux 上，兩個常見的圖形化使用者介面。
  2. KDE 是一個開放原始碼的桌面環境，它建構於 Qt GUI 函式庫之上。
  3. KDE 也提供很多應用程式和工具，包含完整的辦公軟體、網頁瀏覽器、甚至設計 KDE/Qt 應用程式的 IDE 工具。
  4. 因為蘋果電腦的 Mac OS X (稱為 Safari) 選擇使用 KDE 網頁瀏覽器，才讓業界開始認識 KDE 應用程式。
  5. KDE 專案的主網頁，可以找到很多詳細資訊、也可以下載 KDE 和 KDE 應用程式、找尋文件、參與郵件討論、取得其它設計人員的資訊。
  6. KDE 工藝網頁，可以下載 KDE 樣式、按鈕、顏色、桌面圖片…等材料，來裝飾 KDE。
- GUI 開發工具：Qt
  1. Qt 是挪威公司 Trolltech 以 C++ 設計的一種 GUI 開發工具。
  2. Qt 具有跨平台的功能，支援 Linux、UNIX、Windows、Mac OS X、甚至嵌入式的版本。
  3. Qt 的商業版價格相當高，但 Trolltech 另外提供 Linux、Windows 和 MacOS 上的 Qt 免費版本。
  4. Trolltech 的網頁 提供一些 API 文件。
- QT Designer
  1. QT Designer 是一個 GUI 的工具。

2. QT Designer 利用所見既所得的方式，產生 QT 程式的 GUI 程式碼。
  3. 利用 QT Designer 可快速的設計軟體 GUI，再撰寫 GUI 相關動作之程式碼，即可產生一互動式之軟體。
- KDE 程式開發環境：KDevelop
    1. KDevelop 是一個 C 和 C++ 程式的 IDE 工具。
    2. KDevelop 是一個自由軟體，下載網站。
    3. KDevelop 包含文件的樣版、GPL 授權文字和一般的安裝說明。

- 其它環境

| 環境        | 型態                          | 產品的 URL                                                                          |
|-----------|-----------------------------|----------------------------------------------------------------------------------|
| gbuilder  | GNOME 的 IDE 開發環境            | <a href="http://gbuilder.sourceforge.net/">http:// gbuilder.sourceforge.net/</a> |
| Anjuta    | GNOME 的 IDE 開發環境            | <a href="http://anjuta.sourceforge.net/">http:// anjuta.sourceforge.net/</a>     |
| Klint     | KDE 的 IDE 開發環境              | <a href="http://klint.sourceforge.net/">http:// klint.sourceforge.net/</a>       |
| QtEZ      | KDE 的 IDE 開發環境              | <a href="http://projects.uid0.sk/qtez/">http://projects.uid0.sk/qtez/</a>        |
| RHIDE     | 文字模式的 IDE 開發環境              | <a href="http://www.rhide.com/">http://www.rhide.com/</a>                        |
| CRiSP     | 商用的程式設計編輯器                  | <a href="http://www.crisp.com/">http://www.crisp.com/</a>                        |
| SlickEdit | 商用的多程式設計編輯器                 | <a href="http://www.slickedit.com/">http://www.slickedit.com/</a>                |
| Kylix     | 商用的 C++ 和 Delphi 的 IDE 開發環境 | <a href="http://www.borland.com/kylix">http://www.borland.com/kylix</a>          |
| Eclipse   | Java 工具平台和 IDE              | <a href="http://www.eclipse.org">http://www.eclipse.org</a>                      |

### 練習題

1. 請現今 Linux 上，兩個常見的圖形化使用者介面。
 

Sol. KDE 與 GNOME
2. Linux KDE 圖形介面建構於什麼 GUI 函式庫上？
 

Sol. QT
3. Linux GNOME 圖形介面建構於什麼 GUI 函式庫上？
 

Sol. GTK
4. Qt 是挪威公司 Trolltech 以什麼程式語言設計的一種 GUI 開發工具？
 

Sol. C++
5. QT Designer 用途為何？
 

Sol. 利用所見既所得的方式，產生 QT 程式的 GUI 程式碼。

## 17.2 Qt 開發環境建立

- QT3 designer

### 1. 安裝 QT3 套件：

```
1 [root@dywHome2 ~]# urpmi libqt3-devel
To satisfy dependencies, the following packages are going to be
 installed:
3 libqt3-3.3.6-18.4mdv2007.0.i586
 libqt3-devel-3.3.6-18.4mdv2007.0.i586
5 qt3-common-3.3.6-18.4mdv2007.0.i586
Proceed with the installation of the 3 packages? (16 MB) (Y/n)
7 # urpmi 會自動將相關相依套件一併安裝
```

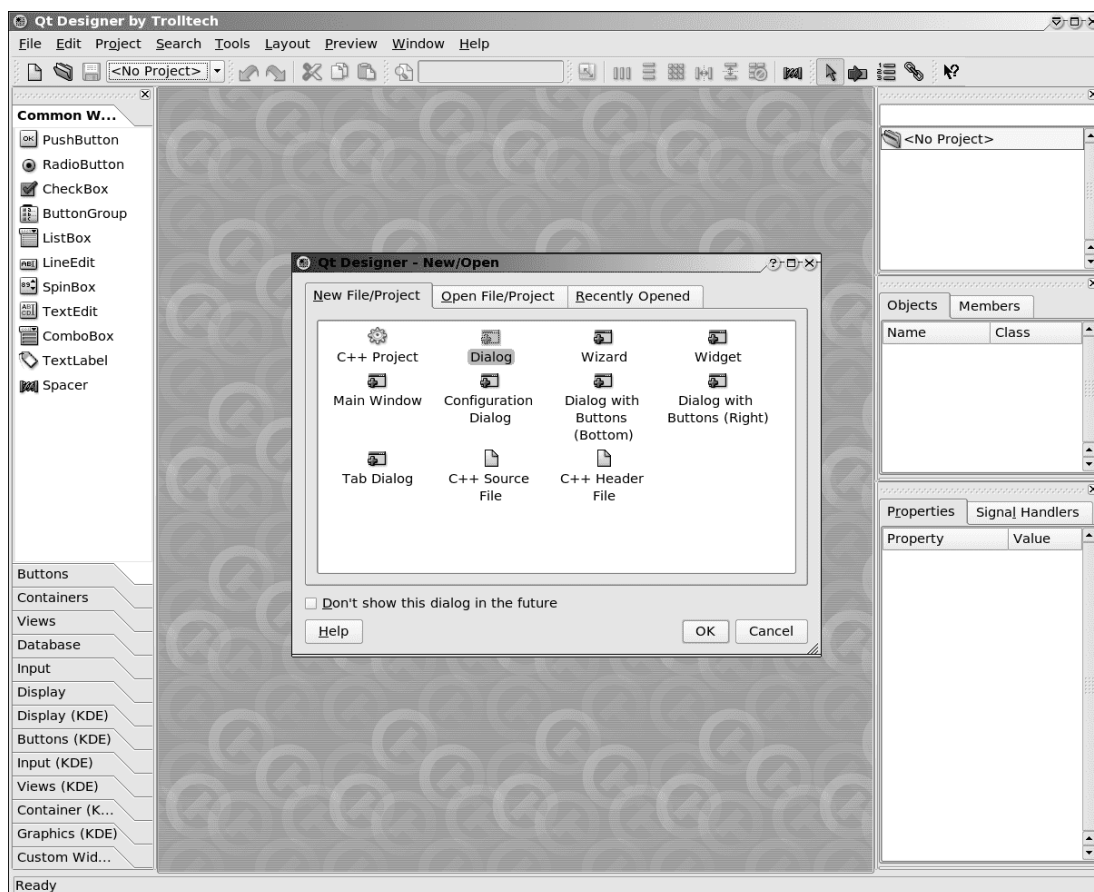
### 2. 檢查 QTDIR 環境變數是否設定到 Qt 的安裝目錄：

```
1 [root@dywHome2 ~]# echo $QTDIR
/usr/lib/qt3/
```

### 3. 執行命令：

```
1 [dywang@dywOffice ~]$ ll /usr/bin/designer-qt3
2 lrwxrwxrwx 1 root root 30 Jan 14 10:48 /usr/bin/designer-qt3 ->
 ../../usr/lib/qt3/bin/designer*
4 [dywang@dywOffice ~]$ designer-qt3
```

### 4. 執行畫面：



- 實例：Qmainwindow 程式

1. 輸入以下程式 qt1.cpp：

```

1 #include <qapplication.h>
2 #include <qmainwindow.h>
3 int main(int argc, char **argv)
4 {
5 QApplication app(argc, argv);
6 QMainWindow window();
7 app.setMainWidget(window);
8 window.show();
9 return app.exec();
10 }
```

2. 編譯時，需要引用 Qt 的 include 和 lib 目錄：

```
$ g++ -o qt1 qt1.cpp -I$QTDIR/include -L$QTDIR/lib -lqt-mt
```

3. 執行這個應用程式，看到一個 Qt 視窗。

```
1 | $./qt1
```



- QMainWindow 程式說明

1. 每個 Qt 應用程式必須有一個 QApplication 物件，且必須在進行其它工作前產生。
2. QApplication 會處理內部的 Qt 運作，例如事件處理、地區語言處理 (localization) 和控制視窗外觀。
3. QApplication 的操作函式：
  - (a) setMainWidget：設定應用程式的主要 widget；
  - (b) exec：開始事件的回圈。exec 在 QApplication::quit() 被呼叫前或是主要 widget 被關閉前，都不會回覆。
4. QMainWindow 是基礎的 Qt 視窗 widget，它支援選單、工具列和狀態列。

## 17.3 gtk+ 開發環境建立

1. 安裝 Glade 套件：

```
1 [root@dywHome2 ~]# [root@deyu ~]# yum install glade3
```

2. 執行 glade designer

```
1 圖形界面
 Applications->Programming->Glade Interface Designer
3 文字界面
 [dywang@deyu glade]$ glade-3
```

3. 相關聯結

- (a) GTK+ and Glade3 GUI Programming Tutorial
- (b) 用 Libglade 快速開發 GTK+ 視窗程式
- (c) Calculator Program using Python and Glade
- (d) Python 教學

#### 4. 下載範例檔 Calculator Program Download

#### 5. 以 glade-3 開啓 calculator.glade

```
1. table1 Rows: 5 -> 6
2. insert ok button in table1 (0,5)
3. rename ok button Label and Name to Sqrt
4. set Sqrt button clicked Signals to on_Sqrt_clicked
5. vim calculatorglade.py

6
coding: utf-8
8 import sys
import math # 匯入數學模組
10 try:
 import pygtk
12 pygtk.require('2.0')
except:
14 pass
.... omission
16
class Calculator:
18 omission
 "on_Add_clicked" : self.displayAdd,
20 "on_Sqrt_clicked" : self.displaySqrt, # 定義sqrt按鈕接受函式
.... omission
22 if operator == 'Sqrt':
 self.firstOperand = self.wTree.get_widget("displayText").
 get_text()
24 result = math.sqrt(float(self.firstOperand))
 self.wTree.get_widget("displayText").set_text(str(result))
26 omission

28 def displaySqrt(self,widget): #定義displaySqrt函式
 self.compute("Sqrt")
```

#### 6. 執行

```
1 [dywang@deyu glade]$ python calculatorglade.py
```



## 17.4 Signals 和 Slots

- 何謂 signals 和 slots?
  1. Qt 信號處理的機制。
  2. GUI 應用程式利用 signals 和 slots 來回應使用者輸入。
  3. 在 Qt 中，signals 和 slots 為巨集關鍵字。
- Signal/slot 與 widget?
  1. Widget 為選單、工具列、按鈕、輸入窗等 GUI 元件。
  2. 當使用者與 widget 互動時，widget 會發出一個 signal。
  3. 將 signal 連結到一個回呼函式 slot。
  4. 執行回呼函式 slot 指定動作。
- 類別使用 signals 和 slots 成員函式的限制：
  1. 必須繼承自 QObject 類別 (class)。
  2. 一定要使用 Q\_OBJECT 巨集 (macro)，即 Q\_OBJECT 必須出現在類別定義中。
  3. signals 和 slots 的參數不可使用函式指標。
- 類別使用 signals 和 slots 成員函式：
  1. 編輯類別定義 MyWindow.h。

```
1 # MyWindow 繼承類別 QMainWindow，提供應用程式的主要視窗功能。
2 # 若需要一個對話窗，要繼承 QDialog。
3 class MyWindow : public QMainWindow
4 {
5 Q_OBJECT
6 public:
7 MyWindow();
8 virtual ~MyWindow();
9 signals:
10 void A_Signal();
11 # signals A_Signal() 沒指定參數
12 private slots:
13 void doSomething();
14 # slots doSomething() 沒指定參數
15 }
```

2. 呼叫 emit 發出 A\_Signal() 信號：

```
1 emit A_Signal();
```

3. 透過 QObject 類別的 connect 成員函式將 slots 連結到信號。

```
1 bool QObject::connect (const QObject * sender, const char * signal,
 const QObject * receiver, const char *
 member)
3 # connect 函式要傳入擁有信號的物件（傳送者）、信號函式、擁有 slot
 的物件（接收者）及 slot 名稱。
5 connect (button, SIGNAL(clicked()), this, SLOT(doSomething()));
 # this 在此代表 MyWindow
```

4. 實作 slot

```
void MyWindow::doSomething()
2 {
 // Slot code
4 }
```

- 實例：做一個簡單的按鈕，它可以有一個標籤和位元圖示，使用者可以透過滑鼠或鍵盤來點選它。

1. 編輯 ButtonWindow.h 宣告類別：

```
#include <qmainwindow.h>
2 class ButtonWindow : public QMainWindow
 {
4 Q_OBJECT
 public:
6 ButtonWindow(QWidget *parent = 0, const char *name = 0);
 virtual ~ButtonWindow();
8 private slots:
 void Clicked();
10 };
```

2. 編輯 ButtonWindow 建構函式

```
ButtonWindow::ButtonWindow(QWidget *parent, const char *name)
2 : QMainWindow(parent, name)
 {
4 # setCaption 是 QMainWindow 的成員函式，可設定視窗標題。
 this->setCaption("This is the window Title");
```

```
6 # 產生按鈕、將按鈕的 clicked 信號連結到 Clicked() slot 中。
 QPushButton *button = new QPushButton("Click Me!", this, "Button1");
8 # 設定按鈕的幾何大小。
 button->setGeometry(50,30,70,20);
10 # 將按鈕的 clicked signal 連結到 Clicked() slot 中。
 connect (button, SIGNAL(clicked()), this, SLOT(Clicked()));
12 }
```

### 3. 編輯 ~ButtonWindow 解構函式

```
ButtonWindow::~ButtonWindow()
2 {
 # Qt 自動管理 widget 的解構工作，所以解構函式是空的。
4 }
```

### 4. 編輯 slot Clicked() :

```
void ButtonWindow::Clicked(void)
2 {
 std::cout << "clicked!\n";
4 }
```

### 5. 編輯 main 程式 ButtonWindow.cpp

```
#include "ButtonWindow.moc"
2 #include <qpushbutton.h>
 #include <qapplication.h>
4 #include <iostream>

6 int main(int argc, char **argv)
 {
8 QApplication app(argc,argv);
 # 產生一個 ButtonWindow 的物件，設定應用程式的主視窗，並將視窗顯示
 在螢幕上。
10 ButtonWindow *window = new ButtonWindow();
 # 設定應用程式的主視窗。
12 app.setMainWidget(window);
 # 將視窗顯示在螢幕上。
14 window->show();
 return app.exec();
16 }
```

### 6. 編譯前，執行前置處理器 moc :

```

2 $ moc ButtonWindow.h -o ButtonWindow.moc
4 # Qt 的 MOC(Meta-Object System)
6 ## 標準的 C++ 無法提供 signal/slot 連結所需之 meta 訊息。
 ## 標頭檔若包含 \verb|Q_OBJECT| 巨集，MOC 會解析巨集定義，
 ## 並產生 Qt meta-object 相關的 C++ 程式碼。
 ## 使用 qmake 產生 Makefile，就會包括 moc 的使用。

```

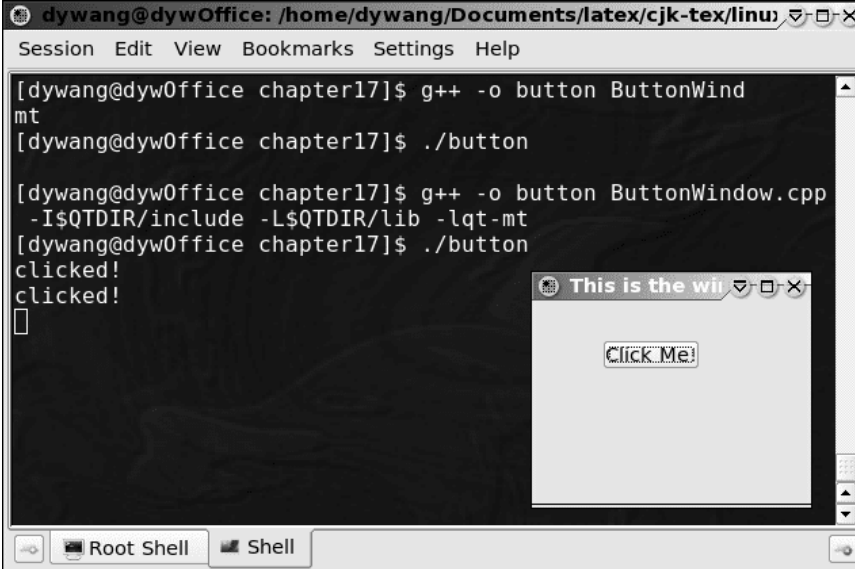
7. 編譯程式，並連結 moc 的結果：

```

1 $ g++ -o button ButtonWindow.cpp -I$QTDIR/include -L$QTDIR/lib -lqt
 -mt

```

8. 執行程式：



```

dywang@dywOffice: /home/dywang/Documents/latex/cjk-tex/linux
Session Edit View Bookmarks Settings Help

[dywang@dywOffice chapter17]$ g++ -o button ButtonWind
mt
[dywang@dywOffice chapter17]$./button

[dywang@dywOffice chapter17]$ g++ -o button ButtonWindow.cpp
-I$QTDIR/include -L$QTDIR/lib -lqt-mt
[dywang@dywOffice chapter17]$./button
clicked!
clicked!
█

```

• QPushButton 建構函式說明：

1. QPushButton 的建構函式：

```

1 QPushButton::QPushButton(const QString &text, QWidget *parent,
 const char* name=0)

```

- (a) 第一個參數是按鈕的文字標籤，
- (b) 隨後是它的父親 widget，
- (c) 最後就是 Qt 內部認定的按鈕名稱。

2. parent 參數

- (a) 在 QWidget 很常見，父親 widget 可以控制何時顯示或破壞它，包含不同的屬性。
  - (b) 如果在 parent 參數傳入 NULL，表示這個 widget 是最上層的 widget，而且產生一個空白視窗來包含它。
  - (c) 範例中用 this 來代表 ButtonWindow 物件，按鈕就會加到 ButtonWindow 的主要區域。
3. name 參數設定 Qt 內部使用的 widget 名稱。如果 Qt 遇到一個錯誤，這個 widget 名稱就會被印在錯誤訊息上，所以最好輸入適切的 widget 名稱，方便往後除錯。
  4. setGeometry 決定絕對位置，但很少用，因為它無法隨視窗自動調整大小。
- 類別 QLayout 與 box widget
    1. 利用 QLayout 或 box widget，給定區域和 widget 之間的空間之後，它會自動調整。
    2. QLayout 類別和 box widget 之間的關鍵性差異就是 layout 物件不是 widget。
    3. layout 類別從 QObject 衍伸而來。
    4. Box widget (即 QHBoxLayout 和 QVBoxLayout) 衍伸自 QWidget，可以把它當成一般的 widget 來處理。
    5. QLayout 有自動調整大小的優點，而如果 widget 要改變大小時，必須人工呼叫 QWidget::resizeEvent()。

- QVBoxLayout 建構函式說明：

1. QVBoxLayout 建構函式 (HBoxLayout 也擁有相同的 API)。

```
QLayout 的 parent 參數，可以是 widget 或其他的 QLayout。
2 QVBoxLayout::QVBoxLayout (QWidget *parent, int margin,
 int spacing, const char *name)
4 QVBoxLayout::QVBoxLayout (QLayout *parentLayout, int spacing,
 const char * name)
6 QVBoxLayout::QVBoxLayout (int spacing, const char *name)
```

2. 若沒指定 parent，只能藉由成員函式 addLayout 加到其他 QLayout 中。

```
2 QVBoxLayout::addWidget (QWidget *widget, int stretch = 0,
 int alignment = 0)
 QVBoxLayout::addLayout (QLayout *layout, int stretch = 0)
```

3. margin 和 spacing 的單位為像素 (pixel)，分別用來指定 QLayout 外圍和 widget 之間的空白空間。
- 實例：使用 QVBoxLayout 類別，設計三個 QLabel，讓視窗大小改變時，label 自動被放大或縮小來符合可用的空間。

1. 程式的標頭檔案 LayoutWindow.h。

```
1 #include <qmainwindow.h>
class LayoutWindow : public QMainWindow
3 {
 Q_OBJECT
5 public:
 LayoutWindow(QWidget *parent = 0, const char *name = 0);
7 virtual ~LayoutWindow();
};
```

2. 程式 LayoutWindow.cpp。

```
#include <qapplication.h>
2 #include <qlabel.h>
#include <qlayout.h>
4 #include "LayoutWindow.moc"
LayoutWindow::LayoutWindow(QWidget *parent, const char *name) :
6 QMainWindow(parent, name)
{
8 this->setCaption("Layouts");
10 # 因為不能直接將 QLayout 加到 QMainWindow，所以產生一個假的 QWidget
 。
 QWidget *widget = new QWidget(this);
12 setCentralWidget(widget);
 QHBoxLayout *horizontal = new QHBoxLayout(widget, 5, 10,
14 "horizontal");
 QVBoxLayout *vertical = new QVBoxLayout();
16 QLabel* label1 = new QLabel("Top", widget, "textLabel1");
 QLabel* label2 = new QLabel("Bottom", widget, "textLabel2");
18 QLabel* label3 = new QLabel("Right", widget, "textLabel3");
 vertical->addWidget(label1);
20 vertical->addWidget(label2);
 horizontal->addLayout(vertical);
22 horizontal->addWidget(label3);
 resize(150, 100);
24 }

26 LayoutWindow::~~LayoutWindow()
{
28 }

30 int main(int argc, char **argv)
{
```

```

32 | QApplication app(argc,argv);
 | LayoutWindow *window = new LayoutWindow();
34 | app.setMainWidget(window);
 | window->show();
36 | return app.exec();
 | }

```

3. 編譯前，在標頭檔案上執行 moc：

```

1 | $ moc LayoutWindow.h -o LayoutWindow.moc
 | $ g++ -o layout LayoutWindow.cpp -I$QTDIR/include -L$QTDIR/lib -lqt
 | -mt

```

4. 執行程式：



## 17.5 QT Widget

- QLineEdit：單行文字輸入的 widget。

1. 建構函式和常用成員函式：

```

#include <qlineedit.h>
2 | QLineEdit::QLineEdit (QWidget *parent, const char* name = 0)
 | QLineEdit::QLineEdit (const QString &contents, QWidget *parent,
 | const char *name = 0)
4 | QLineEdit::QLineEdit (const QString &contents,
 | const QString &inputMask,
 | QWidget *parent, const char *name = 0)
6 |
8 | void setInputMask (const QString &inputMask)
 | void insert (const QString &newText)
10 | bool isModified (void)
 | void setMaxLength (int length)
12 | void setReadOnly (bool read)
 | void setText (const QString &text)
14 | QString text (void)
 | void setEchoMode(EchoMode mode)

```

2. 屬性 EchoMode 決定文字如何顯示在 widget 上。它可以有以下三種數值：

- (a) QLineEdit::Normal：顯示輸入字元（預設值）。
- (b) QLineEdit::Password：顯示星號，取代真正的字元。
- (c) QLineEdit::NoEcho：不顯示任何東西。

3. 使用 setEchoMode 設定 EchoMode 模式：

```
1 QLineEdit->setEchoMode(QLineEdit::Password);
```

4. inputMask 是字元構成的字串，用來說明接受的字元，其與正規表示式使用相同的原則。

(a) inputMask 字元代表能不能存在某字元。

| 意義                | 必要性字元 | 選擇性字元 |
|-------------------|-------|-------|
| ASCII A-Z，a-z     | A     | a     |
| ASCII A-Z，a-z，0-9 | N     | n     |
| 任何字元              | X     | x     |
| 數值 0-9            | 9     | 0     |
| 數值 1-9            | D     | d     |

(b) inputMask 結尾可以選擇性加上分號。

(c) inputMask 進階的特殊字元：

- # 數字或 +, - 符號之選擇性字元。
- > 將隨後的字元變成大寫。
- < 將隨後的字元變成小寫。
- ! 停止轉換。
- \ 跳脫字元

5. 遮罩範例：

(a) “AAAAAA-999D”

- i. 可接受 Athens-2004，
- ii. 但不能接受 Sydney-2000 或 Atlanta-1996。

(b) “AAAAAnn-99-99;”

- i. 可接受 March-03-12，
- ii. 但不能接受 May-03-12 或 September-03-12。

(c) “000.000.000.000”：允許 IP 位址，例如 192.168.0.1。

#### • 實例：QLineEdit

1. 標頭檔案 QLineEdit.h。



```

1 #include <qmainwindow.h>
 #include <qlineedit.h>
3 #include <qstring.h>
 class QLineEdit : public QMainWindow
5 {
 Q_OBJECT
7 public:
 QLineEdit(QWidget *parent = 0, const char *name = 0);
9 QLineEdit *password_entry;
 private slots:
11 void Clicked();
 };

```

## 2. 程式檔案 QLineEdit.cpp。

```

 #include "LineEdit.moc"
2 #include <qpushbutton.h>
 #include <qapplication.h>
4 #include <qlabel.h>
 #include <qlayout.h>
6 #include <iostream>
 LineEntry::LineEntry(QWidget *parent, const char *name) :
8 QMainWindow(parent, name)
 {
10 QWidget *widget = new QWidget(this);
 setCentralWidget(widget);
12
 # 使用 QGridLayout 來安排 widget。指定行數、列數、邊界設定和間隔。
14 QGridLayout *grid = new QGridLayout(widget,3,2,10, 10,"grid");
 QLineEdit *username_entry = new QLineEdit(widget,
16 "username_entry");
 password_entry = new QLineEdit(widget, "password_entry");
18 password_entry->setEchoMode(QLineEdit::Password);
20 # 在方格中加入一個 widget，必須告知行編號、列編號，而起始值為 (0,0)
 代表左上方的格子。
 grid->addWidget(new QLabel("Username", widget, "userlabel"),
22 0, 0, 0);
 grid->addWidget(new QLabel("Password", widget, "passwordlabel"),
24 1, 0, 0);
 grid->addWidget(username_entry, 0,1, 0);
26 grid->addWidget(password_entry, 1,1, 0);
 QPushButton *button = new QPushButton ("Ok", widget, "button");
28 grid->addWidget(button, 2,1,Qt::AlignRight);
 resize(350, 200);
30 connect (button, SIGNAL(clicked()), this, SLOT(Clicked()));
 }
32 void LineEntry::Clicked(void)
 {
34 std::cout << password_entry->text() << "\n";
 }
36 int main(int argc, char **argv)

```

```

{
38 QApplication app(argc,argv);
 LineEntry *window = new LineEntry();
40 app.setMainWidget(window);
 window->show();
42 return app.exec();
}

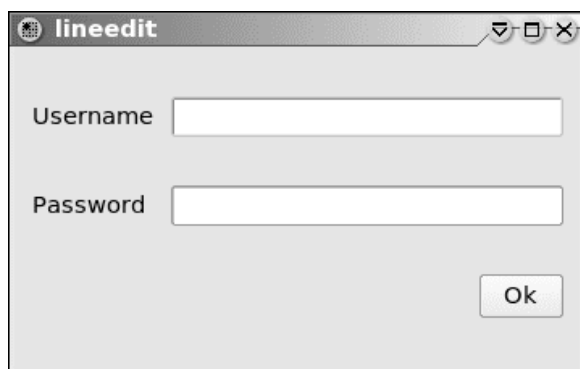
```

3. 程式執行結果：

```

1 [dywang@dywOffice chapter17]$ moc LineEdit.h -o LineEdit.moc
[dywang@dywOffice chapter17]$ g++ -o lineedit LineEdit.cpp \
3 -I$QTDIR/include -L$QTDIR/lib -lqt-mt
[dywang@dywOffice chapter17]$./lineedit

```



4. 結果說明：

- (a) 產生兩個 QLineEdit widget，其中一個設定 EchoMode 讓它變成密碼輸入的視窗；
- (b) 按下按鈕時就會印出結果。

- Qt 按鈕 (Buttons)

1. QPushButton 的成員函式：

```

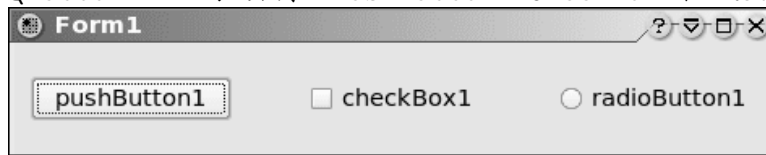
#include <qbutton.h>
2 virtual void QPushButton::setText (const QString &)
 virtual void QPushButton::setPixmap (const QPixmap &)
4 bool QPushButton::isToggleButton () const
 virtual void QPushButton::setDown (bool)
6 bool QPushButton::isDown () const
 bool QPushButton::isOn () const
8 enum QPushButton::ToggleState { Off, NoChange, On }
 ToggleState QPushButton::state () const

```

2. 函式說明：

- (a) `isToggleButton` 函式表示按鈕是否為雙態 (toggle) 的按鈕 (亦即有 on 或 off 狀態)，預設為 `FALSE`。
- (b) `isDown` 函式表示按鈕被按下就回覆 `TRUE`。
- (c) 如果按鈕拴牢 ( `toggled` )，`isOn` 函式回傳 `TRUE`。

3. `QPushButton` 三個子類別：`PushButton`、`CheckBox` 和 `RadioButton`。



- `QPushButton`：簡單的按鈕 widget，被按下時就會執行一些動作。

1. 建構函式和有用的成員函式：

```
1 #include <qpushbutton.h>
 QPushButton (QWidget *parent, const char *name = 0)
3 QPushButton (const QString &text, QWidget *parent,
 const char *name = 0)
5 QPushButton (const QIconSet &icon, const QString &text,
 QWidget *parent, const char * name = 0)
7 void QPushButton::setToggleButton (bool);
```

2. 按鈕上可以有文字或圖示。例如 `OK` 或 `Cancle`。

3. 可呼叫 `setToggleButton`，從無狀態 (stateless) 按鈕變成雙態 (toggle) 按鈕。

- `QCheckBox`

1. 建構函式和有用的成員函式：

```
1 #include <qcheckbox.h>
 QCheckBox (QWidget *parent, const char *name = 0)
3 QCheckBox (const QString &text, QWidget *parent,
 const char *name = 0)
5 bool QCheckBox::isChecked ()
 void QCheckBox::setTristate (bool y = TRUE)
7 bool QCheckBox::isTristate ()
```

2. 一般只有 on 或 off 狀態資訊；

3. 也可變成三種狀態，中間的狀態代表“沒有改變”。

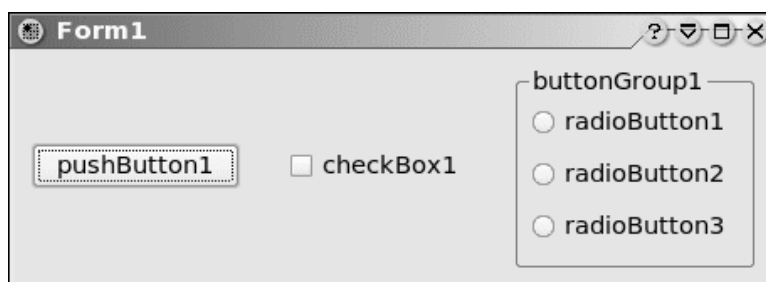
- `QRadioButton`

## 1. 建構函式和一個成員函式：

```

1 #include <qradiobutton.h>
 QRadioButton (QWidget *parent, const char *name = 0)
3 QRadioButton (const QString &text, QWidget *parent,
 const char *name = 0)
5 bool QRadioButton::isChecked ()

```



## 2. 通常是成群的，在同一個時間只能按下群組中的一個按鈕。

## 3. 還要透過 QButtonGroup 類別來控制群組和唯一選擇性。

## • QButtonGroup

## 1. 建構函式和一個成員函式：

```

1 #include <qbuttongroup.h>
 QButtonGroup (QWidget *parent = 0, const char * name = 0)
3 QButtonGroup (const QString & title, QWidget * parent = 0,
 const char * name = 0)
5 int insert (QPushButton *button, int id = -1)
void remove (QPushButton *button)
7 int id (QPushButton *button) const
int count () const
9 int selectedId () const

```

## 2. 要將按鈕加入 QButtonGroup 可以使用 insert(), 或指定 QButtonGroup 為要按鈕的父親 widget。

## 3. 在 insert() 中，還可以指定一個 id，以辨識每個按鈕。

## 4. selectedId 會回傳被選擇按鈕的 id。

## 5. 加入群組的 QRadioButton 都會自動設成排他性，不會有重複選擇的問題。

## • 實例：QButtons

## 1. 檔案 Buttons.h：

```

1 #include <qmainwindow.h>

```

```

#include <qcheckbox.h>
3 #include <qbutton.h>
#include <qradiobutton.h>
5 class Buttons : public QMainWindow
{
7 Q_OBJECT
 public:
9 Buttons(QWidget *parent = 0, const char *name = 0);

11 # 在 slot 函式中會查詢按鈕的狀態，所以在類別定義中宣告按鈕指標為私
 有的。
 private:
13 void PrintActive(QButton *button);
 QCheckBox *checkbox;
15 QRadioButton *radiobutton1, *radiobutton2;
 private slots:
17 void Clicked();
};

```

## 2. 檔案 Buttons.cpp :

```

#include "Buttons.moc"
2 #include <qbuttongroup.h>
#include <qpushbutton.h>
4 #include <qapplication.h>
#include <qlabel.h>
6 #include <qlayout.h>
#include <iostream>
8 Buttons::Buttons(QWidget *parent, const char *name) :
 QMainWindow(parent, name)
10 {
 QWidget *widget = new QWidget(this);
12 setCentralWidget(widget);
 QVBoxLayout *vbox = new QVBoxLayout(widget, 5, 10, "vbox");
14 checkbox = new QCheckBox("CheckButton", widget, "check");
 vbox->addWidget(checkbox);
16
 # 為兩個 radio 按鈕，產生一個 QButtonGroup :
18 QButtonGroup *buttongroup = new QButtonGroup(0);
 radiobutton1 = new QRadioButton("RadioButton1", widget, "radio1")
 ;
20 buttongroup->insert(radiobutton1);
 vbox->addWidget(radiobutton1);
22 radiobutton2 = new QRadioButton("RadioButton2", widget, "radio2")
 ;
 buttongroup->insert(radiobutton2);
24 vbox->addWidget(radiobutton2);
 QPushButton *button = new QPushButton ("Ok", widget, "button");
26 vbox->addWidget(button);
 resize(350, 200);
28 connect (button, SIGNAL(clicked()), this, SLOT(Clicked()));
}

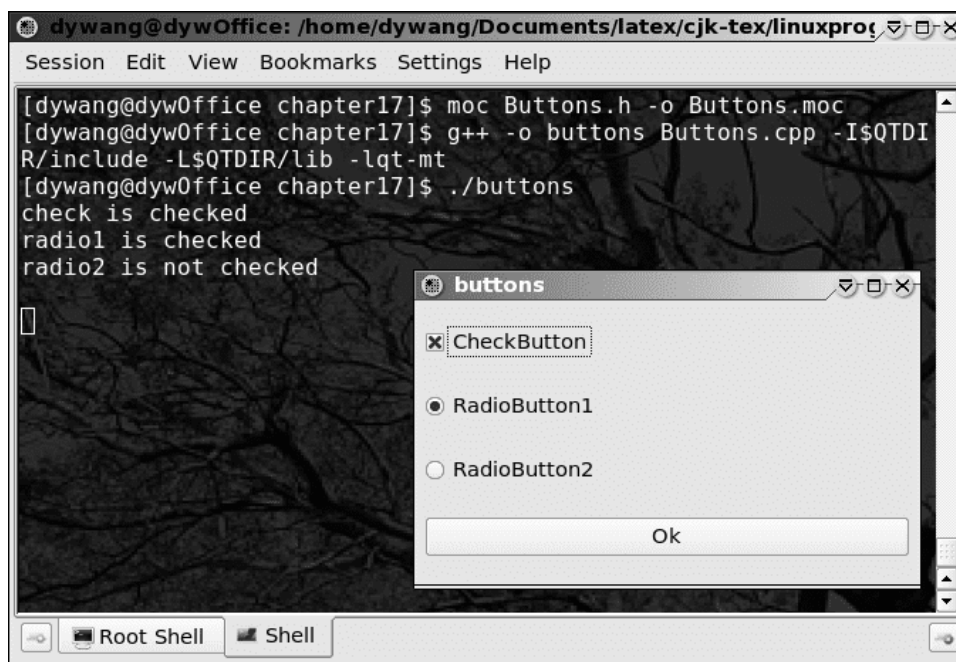
```

## 3. 印出按鈕狀態的成員函式：

```
1 void Buttons::PrintActive(QButton *button)
2 {
3 if (button->isOn())
4 std::cout << button->name() << " is checked\n";
5 else
6 std::cout << button->name() << " is not checked\n";
7 }
8 void Buttons::Clicked(void)
9 {
10 PrintActive(checkbox);
11 PrintActive(radiobutton1);
12 PrintActive(radiobutton2);
13 std::cout << "\n";
14 }
15 int main(int argc, char **argv)
16 {
17 QApplication app(argc,argv);
18 Buttons *window = new Buttons();
19 app.setMainWidget(window);
20 window->show();
21 return app.exec();
22 }
```

## 4. 程式執行結果。

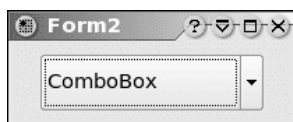
```
[dywang@dywOffice chapter17]$ moc Buttons.h -o Buttons.moc
2 [dywang@dywOffice chapter17]$ g++ -o buttons Buttons.cpp\
>-I$QTDIR/include -L$QTDIR/lib -lqt-mt
4 [dywang@dywOffice chapter17]$./buttons
check is checked
6 radio1 is checked
radio2 is not checked
```



- QComboBox

1. Radio 按鈕有六個以上的選項時，會讓視窗難以調整。
2. 建構函式

```
1 # QComboBox 建構函式中的布林參數 TRUE，指定 QComboBox 為可讀寫。
 QComboBox *combo = new QComboBox(TRUE, parent, "widgetname");
```



3. 加入選項，輸入型態可以為 QStrings 或使用傳統的 char \* 格式。

```
2 combo->insertItem(QString("An Item"), 1);
 # 數值 1 代表設定這個選項為串列中的第一個選項。
 # 如果要將加到串列尾端，只要傳入任何負的整數。
```

4. 利用 char \* 陣列加入多個選項：

```
1 char* weather[] = { "Thunder", "Lightning", "Rain", 0 };
 combo->insertStrList(weather, 3);
```

5. 呼叫 QComboBox 的 setInsertionPolicy() 成員函式：

```
combo->setInsertionPolicy(QComboBox::AtTop);
```

| 關鍵字串                     | 插入策略             |
|--------------------------|------------------|
| QComboBox::AtTop         | 將新項目插到串列的第一個選項。  |
| QComboBox::AtBottom      | 將新項目插到串列的最後一個選項。 |
| QComboBox::AtCurrent     | 取代先前選擇的選項。       |
| QComboBox::BeforeCurrent | 將新項目插到先前選擇的選項之前。 |
| QComboBox::AfterCurrent  | 將新項目插到先前選擇的選項之後。 |
| QComboBox::NoInsertion   | 不要將新選項插入串列中。     |

#### 6. QComboBox 的建構函式和其他成員函式：

```

1 #include <qcombobox.h>
 QComboBox (QWidget *parent = 0, const char *name = 0)
3 QComboBox (bool readwrite, QWidget *parent = 0, const char *name =
 0)
 # count 回覆串列選項的數量。
5 int count ()
 # QStringList 和 QStringList 是 Qt 字串集合類別，可以用來加入多個選
 項。
7 void insertStringList (const QStringList &list, int index = -1)
 void insertStrList (const QStringList &list, int index = -1)
9 void insertStrList (const QStringList *list, int index = -1)
 void insertStrList (const char **strings, int numStrings = -1,
11 int index = -1)
 # insertItem 加入選項；
13 void insertItem (const QString &t, int index = -1)
 # removeItem() 移除選項；
15 void removeItem (int index)
 # CurrentItem() 取得目前選項；
17 virtual void setCurrentItem (int index)
 # CurrentText() 取得目前選項內容；
19 QString currentText ()
 virtual void setCurrentText (const QString &)
21 # setEditable() 切換可編輯的狀態。
 void setEditable (bool)

```

- 實例：設計兩個 QComboBox widget，一個可以編輯，另一個是唯讀。

#### 1. 程式檔案 ComboBox.cpp：

```

 #include "ComboBox.moc"
2 #include <qlayout.h>
 #include <iostream>
4 ComboBox::ComboBox(QWidget *parent, const char *name) :
 QMainWindow(parent, name)
6 {

```



```

8 QWidget *widget = new QWidget(this);
 setCentralWidget(widget);
 QVBoxLayout *vbox = new QVBoxLayout(widget, 5, 10, "vbox");
10 QComboBox *editablecombo = new QComboBox(TRUE, widget, "editable"
);
 vbox->addWidget(editablecombo);
12 QComboBox *readonlycombo = new QComboBox(FALSE, widget, "readonly"
);
 vbox->addWidget(readonlycombo);
14 static const char* items[] = { "Macbeth",
 "Twelfth Night", "Othello", 0 };
16 editablecombo->insertStrList (items);
 readonlycombo->insertStrList (items);
18 # 當一個新選項被選擇之後，QComboBox 會發出 textChanged(QString &)
 信號。
 connect (editablecombo, SIGNAL(textChanged(const QString&)),
20 this, SLOT(Changed(const QString&)));
 resize(350, 200);
22 }

24 # QString 參數 s，會被信號傳送。
void ComboBox::Changed(const QString& s)
26 {
 std::cout << s << "\n";
28 }

30 int main(int argc, char **argv)
 {
32 QApplication app(argc,argv);
 ComboBox *window = new ComboBox();
34 app.setMainWidget(window);
 window->show();
36 return app.exec();
 }

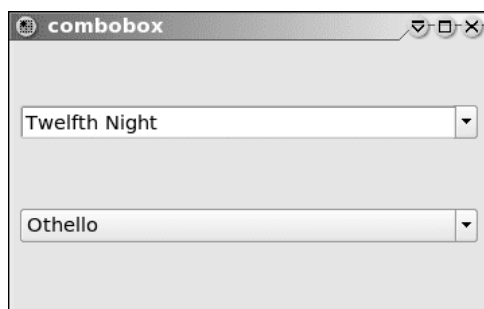
```

2. 執行：命令列看到新選擇的選項被印出來。

```

1 [dywang@dywOffice chapter17]$ moc ComboBox.h -o ComboBox.moc
 [dywang@dywOffice chapter17]$ g++ -o combobox ComboBox.cpp \
3 >-I$QTDIR/include -L$QTDIR/lib -lqt-mt
 [dywang@dywOffice chapter17]$./combobox
5 Twelfth Night

```



- QListView

1. QListView 能以目錄架構的方式顯示，按下加號和減號擴展或緊縮子元件，與檔案顯示程式一樣。
2. 建構函式：只要指定父親和 widget 名稱。

```
1 QListView *view = new QListView(parent, "name");
```

3. 設定欄位的標題，要利用 addColumn() 成員函式：

```
1 view->addColumn("Left Column", width1); // Fixed width
 view->addColumn("Right Column"); // Width autosizes
```

- QListViewItem：將列元件傳入 QListView 中：

1. 建構函式：

```
2 QListViewItem::QListViewItem (QListView * parent, QString label1,
4 QString label2 = QString::null, QString label3 = QString::null,
 QString label4 = QString::null, QString label5 = QString::null,
 QString label6 = QString::null, QString label7 = QString::null,
 QString label8 = QString::null)
6 # 可以提供八個行的標籤，若不需要可不給參數或指定 NULL
```

2. 將列元件傳入 view：

```
2 QListViewItem *toplevel = new QListViewItem(view, "Left Data",
 "Right Data");
```

3. 將列元件傳入 topLevel 子節點：

```
new QListViewItem(toplevel, "Left Data", "Right Data");
```

```
2 // A Child of toplevel
```

#### 4. QListViewItem 成員函式：

```
#include <qlistview.h>
2 virtual void insertItem (QListViewItem * newChild)
virtual void setText (int column, const QString & text)
4 virtual QString text (int column) const
QListViewItem *firstChild () const
6 QListViewItem *nextSibling () const
QListViewItem *parent () const
8 QListViewItem *itemAbove ()
QListViewItem *itemBelow ()
```

#### 5. 實例：印出上層節點的第一欄

```
1 QListViewItem *child = view->firstChild();
while(child)
3 {
 cout << myChild->text(1) << "\n";
5 myChild = myChild->nextSibling();
}
```

#### • 實例：QListView

##### 1. 跳過標頭檔案 ListView.h，直接看類別檔案 ListView.cpp：

```
#include "ListView.moc"
2 ListView::ListView(QWidget *parent, const char *name) :
QMainWindow(parent, name)
4 {
 listview = new QListView(this, "listview1");
6 listview->addColumn("Artist");
 listview->addColumn("Title");
8 listview->addColumn("Catalogue");
 listview->setRootIsDecorated(TRUE);
10 QListViewItem *toplevel = new QListViewItem(listview,
 "Avril Lavigne", "Let Go", "AVCD01");
12 new QListViewItem(toplevel, "Complicated");
 new QListViewItem(toplevel, "Sk8er Boi");
14 setCentralWidget(listview);
}
16
17 int main(int argc, char **argv)
18 {
 QApplication app(argc,argv);
20 ListView *window = new ListView();
```

```

22 app.setMainWidget(window);
 window->show();
 return app.exec();
24 }

```

## 2. 編譯和執行 ListView

```

[dywang@dywOffice chapter17]$ moc ListView.h -o ListView.moc
2 [dywang@dywOffice chapter17]$ g++ -o listview ListView.cpp \
-I$QTDIR/include -L$QTDIR/lib -lqt-mt
4 [dywang@dywOffice chapter17]$./listview

```



## 17.6 對話窗 dialog

- Modal 與 Nonmodal 對話視窗：

1. Modal 對話視窗：在使用者回應對話視窗之前，擱置其它視窗的輸入。對於立即抓取使用者的回應和顯示重要的錯誤訊息，這類型的對話視窗就很有用。
2. Nonmodal 對話視窗：非擱置視窗，與一般應用程式視窗相同。對於搜尋視窗或輸入視窗就很有用。

- QDialog：Qt 的基本對話視窗類別

1. 實例：一般會產生一個繼承自 QDialog 的對話視窗類別，並加上一些 widget 來構成對話視窗介面。

```

#include <qdialog.h>
2 MyDialog::MyDialog(QWidget *parent, const char *name) :
 QDialog(parent, name)
4 {
 QHBoxLayout *hbox = new QHBoxLayout(this);
6 hbox->addWidget(new QLabel("Enter your name"));

```

```
8 hbox->addWidget(new QLineEdit());
 hbox->addWidget(ok_pushbutton);
 hbox->addWidget(cancel_pushbutton);
10 connect (ok_pushbutton, SIGNAL(clicked()), this, SLOT(accept()));
 connect (cancel_pushbutton, SIGNAL(clicked()), this,
12 SLOT(reject()));
}
```

2. Modal 對話窗：呼叫 `exec()`，所有處理都會被擱置。

```
1 MyDialog *dialog = new MyDialog(this, "mydialog");
 if (dialog->exec() == QDialog::Accepted)
3 {
 // User clicked 'Ok'
5 doSomething();
 }
7 else
 {
9 // user clicked 'Cancel' or dialog killed
 doSomethingElse();
11 }
 delete dialog;
```

3. signal/slot 連結方式如同 modal 對話視窗。

```
MyDialog::MyDialog(QWidget *parent, const char *name) :
2 QDialog(parent, name)
 {
4 ...
 connect (ok_pushbutton, SIGNAL(clicked()), this, SLOT(OkClicked()
));
6 connect (cancel_pushbutton, SIGNAL(clicked()), this,
 SLOT(CancelClicked()));
8 }
 MyDialog::OkClicked()
10 {
 //Do some processing
12 }
 MyDialog::CancelClicked()
14 {
 //Do some other processing
16 }
```

4. NonModal 對話窗：呼叫 `show()`，顯示對話窗並立刻回覆，繼續主要的處理迴圈。

```
MyDialog *dialog = new MyDialog(this, "mydialog");
```

```
2 dialog->show();
```

- QMessageBox

1. QMessageBox 是一個 modal 對話窗，顯示一個簡單的訊息，加上一個小圖示和按鈕。QMessageBox 一般的訊息、警告訊息、其它關鍵訊息成員函式：

```
#include <qmessagebox.h>
2 int information (QWidget *parent, const QString &caption,
 const QString &text,
4 int button0, int button1=0, int button2=0)
int warning (QWidget *parent, const QString &caption,
6 const QString &text,
 int button0, int button1, int button2=0)
8 int critical (QWidget *parent, const QString &caption,
 const QString &text,
10 int button0, int button1, int button2=0)
```

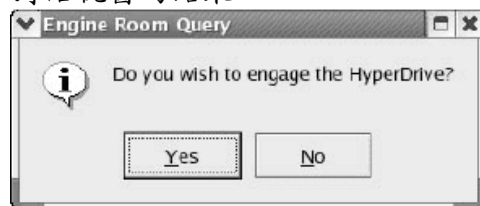
2. Each of the buttons, button0, button1 and button2 may be set to one of the following values:

```
QMessageBox::Ok
2 QMessageBox::Cancel
QMessageBox::Yes
4 QMessageBox::No
QMessageBox::Abort
6 QMessageBox::Retry
QMessageBox::Ignore
```

3. 實例：QMessageBox 的使用

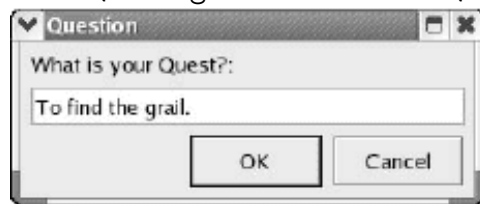
```
1 int result = QMessageBox::information(this,
 "Engine Room Query",
3 "Do you wish to engage the HyperDrive?",
 QMessageBox::Yes | QMessageBox::Default,
5 QMessageBox::No | QMessageBox::Escape);
switch (result) {
7 case QMessageBox::Yes:
 hyperdrive->engage();
9 break;
 case QMessageBox::No:
11 // do something else
 break;
13 }
```

## 4. 對話視窗的結果。



## • QDialog

## 1. 類似 QMessageBox，但多一個 QLineEdit 可讓使用者輸入。



## 2. 可輸入文字、布林、整數及浮點數，但分別使用不同的建構函式：

```

1 #include <qinputdialog.h>
 QString getText (const QString &caption, const QString &label,
3 QLineEdit::EchoMode mode=QLineEdit::Normal,
 const QString &text=QString::null, bool * ok = 0,
5 QWidget * parent = 0, const char * name = 0)
 QString getItem (const QString &caption, const QString &label,
7 const QStringList &list, int current=0,
 bool editable=TRUE,
9 bool * ok=0, QWidget *parent = 0, const char *
 name=0)
 int getInteger (const QString &caption, const QString &label,
11 int num=0,
 int from = -2147483647, int to = 2147483647,
13 int step = 1,
 bool * ok = 0, QWidget * parent = 0,
15 const char * name = 0)
 double getDouble (const QString &caption, const QString &label,
17 double num = 0,
 double from = -2147483647, double to =
21 2147483647,
 int decimals = 1, bool * ok = 0,
 QWidget * parent = 0,
 const char * name = 0)

```

## 3. 實例：輸入一行的文字：

```

1 bool result;
 QString text = QDialog::getText("Question",
3 "What is your Quest?:",
 QLineEdit::Normal,
5 QString::null, &result,
 this,

```

```

7 "input");
 if (result) {
 doSomething(text);
9 } else {
 // user pressed cancel
11 }

```

- (a) `getText` 利用一個 `QLineEdit`，可以設定 `EchoMode`，也可以指定預設文字或清成空白。
- (b) 每個 `QInputDialog` 都有 `Ok` 和 `Cancel` 按鈕，必須傳入一個 `bool` 指標，才能知道哪個按鈕被按下，如果使用者按下 `Ok`，`result` 就是 `TRUE`。

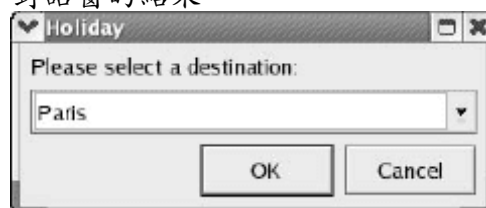
4. `getItem` 透過 `QComboBox`，提供使用者一系列的選項。

```

1 bool result;
 QStringList options;
3 options << "London" << "New York" << "Paris";
 QString city = QInputDialog::getItem("Holiday",
5 "Please select a destination:",
 "",
 options, 1, TRUE, &result,
7 this, "combo");
 if (result)
9 selectDestination(city);

```

5. 對話窗的結果。



- 使用 `qmake` 簡化 `makefile` 的設計

1. Qt 提供一個工具稱為 `qmake`，可以產生 `makefile`。
2. `qmake` 需要一個 `.pro` 的輸入檔案。這個檔案包含基本的資訊，例如：原始碼、標頭檔案、目的檔案和 KDE/Qt 函式庫位置。一般 KDE 的 `.pro` 檔案如下：

```

1 TARGET = app
 MOC_DIR = moc
3 OBJECTS_DIR = obj
 INCLUDEPATH = /usr/include/kde
5 QMAKE_LIBDIR_X11 += /usr/lib

```



```
7 QMAKE_LIBS_X11 += -lkdeui -lkdecore
 SOURCES = main.cpp window.cpp
 HEADERS = window.h
```

3. 產生 makefile。

```
$qmake file.pro -o Makefile
```

## 17.7 選單和工具列

- KAction widget

1. KAction 的建構函式：

```
1 #include <kde/kaction.h>
 KAction (const QString &text, const KShortcut &cut,
3 const QObject *receiver,
 const char *slot, QObject *parent, const char *name = 0)
5 # 必須輸入文字、快速鍵、圖示和一個 slot，slot 為 KAction 被選定時被
 呼叫。
```

2. 實例：產生一個 New 選單和工具列，當它們被按下時就會呼叫 newFile()。

```
1 KAction *new_file = new KAction("New", "filenew",
 KstdAccel::key(KstdAccel::New),
3 this, SLOT(newFile()), this,
 "newaction");
```

3. 為 KAction new\_file 加入選單、工具列：

```
new_file->plug(a_menu);
2 new_file->plug(a_toolbar);
```

4. 取消 KAction new\_file。

```
new_file->setEnabled(FALSE);
```

## 5. KDE 提供一些標準 KAction 物件

```
1 #include <kde/kaction.h>
 KAction * openNew (const QObject *recvr, const char *slot,
3 KActionCollection* parent,
 const char *name = 0)
5 KAction * save ...
 KAction * saveAs ...
7 KAction * revert ...
 KAction * close ...
9 KAction * print ...
 etc...
```

## 6. KActionCollection 物件用來管理一個視窗中的 KAction，利用 KMainWindow 的 actionCollection() 成員函式來取得目前的物件。

```
2 KAction *saveas = KStdAction::saveAs(this, SLOT(saveAs()),
 actionCollection(),"saveas");
```

## • 實例：含有選單和工具列的 KDE 應用程式

## 1. 標頭檔案 KDEMenu.h

```
1 #include <kmainwindow.h>
2 class KDEMenu : public KMainWindow
3 {
4 Q_OBJECT
5 public:
6 KDEMenu(const char * name = 0);
7 private slots:
8 void newFile();
9 void aboutApp();
10 };
```

## 2. 在 KDEMenu.cpp 中引入 widget 相關的檔案：

```
1 #include "KDEMenu.h"
2 #include <kapp.h>
3 #include <kaction.h>
4 #include <kstdaccel.h>
5 #include <kmenubar.h>
6 #include <kaboutdialog.h>
```

## 3. 在建構函式中產生三個 KAction widget。

```

KDEMenu::KDEMenu(const char *name = 0) : KMainWindow (0L, name)
2 {
 KAction *new_file = new KAction("New", "filenew",
4 KStdAccel::key(KStdAccel::New),
 this, SLOT(newFile()), this, "newaction")
 ;
6 KAction *quit_action = KStdAction::quit(
 KApplication::kApplication(),
8 SLOT(quit()), actionCollection())
 ;
 KAction *help_action = KStdAction::aboutApp(this, SLOT(aboutApp()
10),
 actionCollection()
);

```

4. 產生兩層的選單，並將它插入 KApplication 的選單列中：

```

QPopupMenu *file_menu = new QPopupMenu;
2 QPopupMenu *help_menu = new QPopupMenu;
menuBar()->insertItem("&File", file_menu);
4 menuBar()->insertItem("&Help", help_menu);

```

5. 將 action 都插入選單和工具列中，並在 new\_file 和 quit\_action 之間插入一個分隔行：

```

new_file->plug(file_menu);
2 file_menu->insertSeparator();
quit_action->plug(file_menu);
4 help_action->plug(help_menu);
new_file->plug(toolBar());
6 quit_action->plug(toolBar());
}

```

6. slot 的定義：aboutApp 產生一個 KAbout 對話窗，顯示程式的相關資訊。

```

1 void KDEMenu::newFile()
{
3 // Create new File
}
5 void KDEMenu::aboutApp()
{
7 KAboutDialog *about = new KAboutDialog(this, "dialog");
 about->setAuthor(QString("A. N. Author"), QString("an@email.net"
9),
 QString("http://url.com"), QString("work"));
 about->setVersion("1.0");

```

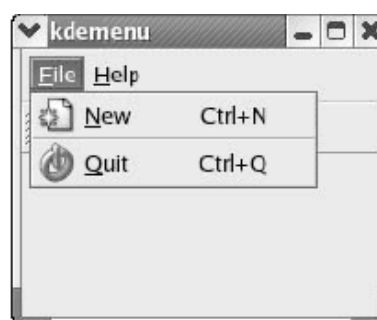
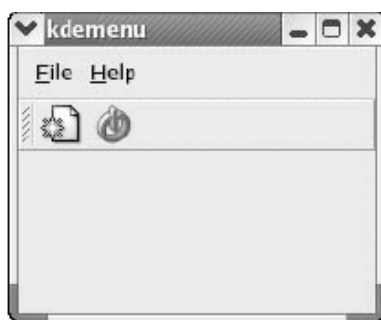
```
11 about->show();
12 }
13
14 int main(int argc, char **argv)
15 {
16 KApplication app(argc, argv, "cdapp");
17 KDEMenu *window = new KDEMenu("kdemenu");
18 app.setMainWidget(window);
19 window->show();
20 return app.exec();
21 }
```

7. 提供一個 menu.pro 檔案給予 qmake：

```
1 TARGET = kdemenu
MOC_DIR = moc
3 OBJECTS_DIR = obj
INCLUDEPATH = /usr/include/kde
5 QMAKE_LIBDIR_X11 += -L$KDEDIR/lib
QMAKE_LIBS_X11 += -lkdeui -lkdecore
7 SOURCES = KDEMenu.cpp
HEADERS = KDEMenu.h
```

8. 執行 qmake 來產生 Makefile，隨後再進行編譯、執行程式。

```
2 $ qmake menu.pro -o Makefile
$ make
$./kdemenu
```



# Chapter 18

## \*Tarball 套件發行

### 18.1 套件發行

- 程式發行時要面對：
  1. 使用者發現臭蟲；
  2. 作者想要提升功能或更新。
- 解決方法：
  1. 若發行只提供二進位的程式時，通常只要再補上新版的二進位檔案即可。
  2. 供應商直接發表新版的程式。
  3. 以原始碼方式發行軟體，可以讓使用者檢查程式的進行，並且再利用部分原始碼。
  4. patch 命令，讓您可以只散佈兩個版本之間的差異部分。
- 軟體的發行方法：
  1. 將所有檔案元件包裝成一個單獨的檔案套件（package）。
  2. 使用標準工具管理套件的版本編號。
  3. 在套件的檔案名稱中，加入版本編號，這樣使用者就可知道他們使用的版本。
  4. 在套件中使用子目錄，確保往後解開套件時，檔案都能放在個別的目錄。
- 安裝與升級套件
  1. 安裝與升級之可能原因：
    - (a) 需要新的功能，但舊有主機的舊版套件並沒有；

- (b) 舊版本的套件上面可能有安全上的顧慮；
  - (c) 舊版的套件執行效能不彰，或者執行的能力不能讓管理者滿足。
2. 使用之套件的格式：
- (a) 原始碼打包壓縮成 tarball 套件；
  - (b) 編譯好的二進位執行檔 RPM 套件；
  - (c) 含原始碼的 RPM，SRPM 套件。

### 練習題

1. 請列舉兩項，軟體發行後需要修改的原因。  
Sol. 1. 使用者發現臭蟲；2. 作者想要提升功能或更新。
2. 軟體發行後，若要除錯或更新，比較簡單的方法為何？  
Sol. 以原始碼方式發行軟體，再以 patch 命令，發佈兩個版本之間差異部分。
3. 軟體發行時，將所有檔案元件包裝成一個單獨的檔案，稱之為何？  
Sol. 套件 (package)。
4. 軟體發行後，升級之可能原因為何？  
Sol. 1. 需要新的功能，但舊版套件沒有；2. 舊版套件可能有安全的顧慮；3. 舊版套件執行效能不彰。
5. 軟體發行，安裝及升級的方式為何？  
Sol. 1. 原始碼 tarball；2. 編譯好的 RPM 套件；3. 原始碼 RPM 之 SRPM 套件。

## 18.2 Tarballs 簡介

- 何謂 Tarball 套件？
  1. Tarball：壓縮過的 TAR (tape archive) 檔案。
  2. Linux 程式和原始碼通常是以一個 tarball 檔案的方式散佈。
  3. 程式 tarball 名稱通常有版本資訊，副檔名為 .tar.gz 或 .tgz。
- Tarball 套件安裝的基本步驟
  1. 下載 Tarball；
  2. 將 Tarball 解打包壓縮會在目前目錄下產生套件目錄，目錄下通常會有：
    - (a) 原始程式碼檔案；

- (b) 偵測程式檔案 ( 可能是 `configure` 或 `config` 等檔名) ;
  - (c) 本套件的簡易說明與安裝說明 ( `INSTALL` 或 `README` ) 。
  - 3. 根據 `INSTALL/README` 的內容安裝好相依的套件 ;
  - 4. 以 `configure` 或 `config` 自動偵測作業環境，並建立 `Makefile` 檔案 ;
  - 5. 以指令 `make` 配合該目錄下的 `Makefile`，進行 `make` 動作 ;
  - 6. 以指令 `make` 並以 `Makefile` 中的 `install` 目標項目，安裝到正確的路徑。
- 使用原始碼管理套件所需要的基礎套件
    - 1. `gcc` 或 `cc` 等 C 語言編譯器 ( `compiler` ) :
    - 2. `make` 及 `autoconfig` 等套件。
      - (a) 不同的系統可能具有的基礎套件環境並不相同，必須偵測使用者的作業環境，以建立 `makefile` 檔案。
      - (b) 自行偵測程式必須藉由 `autoconfig` 相關的套件輔助。
    - 3. 需要 `kernel` 提供的 `library` 以及相關的 `include` 檔案 :
  - `Tarball` 的安裝可跨平台
    - 1. 因為 C 語言的程式碼在各個平台上面是可以共通的 ;
    - 2. 需要的編譯器可能並不相同。例如 `Linux` 上編譯器用 `gcc` 而 `Windows` 上則用相關的 C 編譯器。
  - 多用途指令 `tar`
    - 1. `tar` 指令

```
1 [root@linux ~]# tar [-cxtzjvfpN] 檔案與目錄
 選項：
3 -c : 建立一個打包檔案(create)；
 -x : 解開一個打包檔案；
5 -t : 查看 tarfile 內的檔案；(c/x/t 不可同時存在！)
 -z : 同時用 gzip 壓縮；
7 -j : 同時用 bzip2 壓縮；
 -v : 壓縮的過程中顯示檔案；
9 -f : 使用檔名，在 f 之後要立即接檔名！
 -p : 保留檔案的原來屬性（屬性不會依據使用者而變）
11 -P : 保留絕對路徑；
 -N : 比後面接的日期(yyyy/mm/dd)還要新的才會被打包；
13 --exclude FILE: 排除 FILE 。
```

2. 將檔案 txt 打包為 txt.tar

```
1 [root@dywOffice tmp]# tar -cvf txt.tar txt
```

3. 將整個 /etc 目錄下的檔案全部打包成為 /tmp/etc.tar

```
1 [root@dywOffice tmp]# tar -cvf /tmp/etc.tar /etc <== 僅打包，不壓縮
[root@dywOffice tmp]# tar -zcvf /tmp/etc.tar.gz /etc <== 打包後，以
gzip 壓縮
3 [root@dywOffice tmp]# tar -jcvf /tmp/etc.tar.bz2 /etc <== 打包後，
以 bzip2 壓縮
```

4. 查閱 /tmp/etc.tar.gz 檔案內有那些檔案？

```
1 [root@dywOffice tmp]# tar -ztvf /tmp/etc.tar.gz
```

5. 將 /tmp/etc.tar.gz 檔案解壓縮

```
1 [root@dywOffice tmp]# tar -zxvf /tmp/etc.tar.gz
```

6. 只將 /tmp/etc.tar.gz 內的 etc/passwd 解開

```
1 [root@dywOffice tmp]# tar -zxvf /tmp/etc.tar.gz etc/passwd
```

7. 將 /home/csie/testdir 內的所有檔案備份（打包壓縮），並保存其絕對路徑

```
1 [root@dywOffice tmp]# tar -zcvPf testdir.tar.gz /home/csie/testdir
```

8. 將 /etc/ 內的所有檔案備份，並保存其權限

```
1 [root@dywOffice tmp]# tar -zcvpf /tmp/etc.tar.gz /etc
```

9. 在 /home 當中，比 2007/11/30 新的檔案才備份

```
1 [root@dywOffice tmp]# tar -N '2007/11/30' -zcvf home.tar.gz /home
```



10. 備份 /home，但不要 /home/csie/tmp

```
1 [root@dywOffice tmp]# tar -zcvf home.tar.gz /home --exclude /home/
csie/tmp
```

• Tarballs 檔案處理步驟：

### 1. 打包檔案

```
1 $ tar -cvf myapp-1.0.tar main.c 2.c 3.c *.h myapp.1 Makefile5
main.c
3 2.c
3.c
5 a.h
b.h
7 c.h
myapp.1
9 Makefile5
$
```

### 2. 顯示 tar 檔案

```
$ ls -l *.tar
2 -rw-r--r-- 1 neil users 10240 2003-02-15 11:31 myapp-1.0.tar
$
```

### 3. 再利用 gzip 壓縮檔案，檔案大幅減小。

```
1 $ gzip myapp-1.0.tar
$ ls -l *.gz
3 -rw-r--r-- 1 neil users 1668 2003-02-15 11:31 myapp-1.0.tar.gz
$
```

4. Windows 對於正確的副檔名非常在意，而 Linux/UNIX 則不同。若要在 windows 執行可更改附檔名：

```
2 $ mv myapp-1.0.tar.gz myapp_v1.tgz
$ mv myapp_v1.tgz myapp-1.0.tar.gz
```

5. 取回檔案，要先解壓縮，隨後在解開 tar 檔案。

```
$ gzip -d myapp-1.0.tar.gz
2 $ tar -xvf myapp-1.0.tar
main.c
4 2.c
3.c
6 a.h
b.h
8 c.h
myapp.1
10 Makefile5
$
```

- GNU 版的 tar 可以更簡化，可以一個步驟產生壓縮檔。

#### 1. 打包並壓縮檔案

```
1 $ tar -zcvf myapp_v1.tgz main.c 2.c 3.c *.h myapp.1 Makefile5
main.c
3 2.c
3.c
5 a.h
b.h
7 c.h
myapp.1
9 Makefile5
$
```

#### 2. 也可以直接解壓縮：

```
$ tar -zxvf myapp_v1.tgz
2 main.c
2.c
4 3.c
a.h
6 b.h
c.h
8 myapp.1
Makefile5
10 $
```

- 也可利用修改 makefile，增加一個新的目標項目 dist，建立 tarball 收集檔案。

#### 1. 修改 makefile。

```
all: myapp
2 # Which compiler
 CC = gcc
4 # Where are include files kept
 INCLUDE = .
6 # Options for development
 CFLAGS = -g -Wall -ansi
8 # Options for release
 # CFLAGS = -O -Wall -ansi
10 # Local Libraries
 MYLIB = mylib.a
12 myapp: main.o $(MYLIB)
 $(CC) -o myapp main.o $(MYLIB)
14 $(MYLIB): $(MYLIB)(2.o) $(MYLIB)(3.o)
 main.o: main.c a.h
16 2.o: 2.c a.h b.h
 3.o: 3.c b.h c.h
18 clean:
 -rm main.o 2.o 3.o $(MYLIB)
20 dist: myapp-1.0.tar.gz
 myapp-1.0.tar.gz: myapp myapp.1
22 -rm -rf myapp-1.0
 mkdir myapp-1.0
24 cp *.c *.h *.1 Makefile myapp-1.0
 tar -zcvf $@ myapp-1.0
```

2. 執行以下命令，產生 tarball 檔案：

```
1 $ make dist
```

3. 查看 myapp-1.0.0.tar.gz。

```
1 $ ls myapp-1.0.tar.gz
myapp-1.0.tar.gz
```

- 一般 tarball 軟體安裝之指令下達方式：

```
2 $ tar -zxvf tarballfile.tar.gz
$./configure
$ make clean
4 $ make
$ make install
```

- Tarball 套件安裝建議事項：

1. Tarball 在 `/usr/local/src` 解壓縮；
  2. Linux distribution 釋出安裝的套件大多在 `/usr`，而使用者自行安裝的套件則建議放置在 `/usr/local`；
  3. `man` 預設會去搜尋 `/usr/local/man` 裡的說明文件；
  4. 自行安裝套件在 `/usr/local` 較易管理，例如：解除安裝。
- Linux distribution 預設安裝套件的路徑
    1. 以自由軟體多媒體播放軟體 `kplayer` 為例，安裝路徑如下：

```
1 /usr/bin <== 執行檔
 /usr/lib <== 函式庫
3 /usr/share/applnk/ <== 程式選單連結檔
 /usr/share/apps/ <== 程式內容
5 /usr/share/doc/ <== 程式文件
 /usr/share/icons/ <== 程式圖標
7 /usr/share/locale/ <== 程式支援之語系
 /usr/share/services/ <== 程式支援之服務
```

2. `/usr/local` 預設目錄：

```
2 /usr/local/bin
 /usr/local/lib
 /usr/local/share
```

3. 將 `kplayer` 安裝在 `/usr/local/kplayer` 中，路徑為：

```
1 /usr/local/kplayer/bin
 /usr/local/kplayer/lib
3 /usr/local/kplayer/share/applnk
 /usr/local/kplayer/share/apps
5 /usr/local/kplayer/share/doc
 /usr/local/kplayer/share/icons
7 /usr/local/kplayer/share/locale
 /usr/local/kplayer/share/services
```

- 套件移除：
  1. 單一套件的檔案都在同一個目錄之下，只要將該目錄移除即可視為該套件已經被移除。
  2. 實際安裝時得視該軟體的 `Makefile` 裡的 `install` 資訊才能知道安裝情況。

- 套件相依性：
  1. 套件相依性：Linux 軟體套件，經常無法單獨安裝，往往需要其他相關套件存在，才能正常工作。
  2. 套件相依有順序：例如：kplayer 必須先安裝套件 mplayer，而 mplayer 必須先安裝套件 kdelibs-common。
  3. 依套件相依性，移除順序為 kdelibs-common, mplayer, kplayer。
  4. kplyaer 相依套件不只 kdelibs-common 與 mplayer 兩套件，須視實際安裝訊息而定。

### 練習題

1. 何謂 Tarball 套件？

Sol. 壓縮過的 TAR (tape archive) 檔案。

2. 程式 tarball 名稱通常有版本資訊，副檔名為何？

Sol. `.tar.gz` 或 `.tgz`。

3. Tarball 套件解壓縮後，通常會有那些檔案？

Sol. 1. 原始程式碼檔案；2. 偵測程式檔案 `configure` 或 `config`；3. 套件簡易說明 `README` 與（或）安裝說明 `INSTALL`。

4. Tarball 套件解壓縮後，通常會有那個偵測程式偵測作業環境，建立 Makefile 檔案？

Sol. `configure` 或 `config`。

5. Tarball 套件解壓縮後，通常要先查閱那兩個檔案，以瞭解套件如何安裝？

Sol. 套件簡易說明 `README` 與安裝說明 `INSTALL`。

6. Linux 上編譯 C 語言的程式碼的主要工具為何？

Sol. `gcc`。

7. Linux 上編譯 C++ 語言的程式碼的主要工具為何？

Sol. `g++`。

8. Tarball 套件在 Linux 上安裝時，基本上需要那些基礎套件？

Sol. `tar`, `gcc`, `make` 及 `autoconfig` 等套件。

9. 程式 `make` 時，需要那個檔案提供，要產生之目標檔案及其法則？

Sol. `makefile` 或 `Makefile`。

10. 偵測使用者的作業環境，以建立 `makefile` 檔案之套件為何？

Sol. `autoconfig`。

11. Tarball 套件可以在 Windows 上安裝嗎？

Sol. 可以，使用 Windows 下相關的 C 編譯器。

12. 如何以指令 tar 將檔案 txt 打包為 txt.tar？

Sol. `tar -cvf txt.tar txt`

13. 如何以指令 tar 將整個 /etc 目錄下的檔案全部打包成為 /tmp/etc.tar？

Sol. `tar -cvf /tmp/etc.tar /etc`

14. 如何以指令 tar 將整個 /etc 目錄下的檔案全部打包，且以 gzip 壓縮成為 /tmp/etc.tar.gz？

Sol. `tar -czvf /tmp/etc.tar.gz /etc`

15. 如何以指令 tar 將整個 /home 目錄下的檔案全部打包，且以 gzip 壓縮成為 /tmp/home.tar.gz？

Sol. `tar -czvf /tmp/home.tar.gz /home`

16. 如何以指令 tar 查閱 tar 打包壓縮檔 /tmp/etc.tar.gz 內有那些檔案？

Sol. `tar -ztvf /tmp/etc.tar.gz`

17. 如何以指令 tar 將 /etc/ 內的所有檔案備份為 /tmp/etc.tar.gz 下，並且保存其權限？

Sol. `tar -czxvf /tmp/etc.tar.gz /etc`

18. 如何以指令 tar 將 /tmp/home.tar.gz 檔案解壓縮在根目錄 /？

Sol. `tar -zxvf /home/etc.tar.gz /`

19. 如何以指令 tar 將 /tmp/etc.tar.gz 檔案解壓縮在目前目錄？

Sol. `tar -zxvf /tmp/etc.tar.gz`

20. 如何以指令 tar 將備份 /home 成 myhome.tar.gz，但不要 /home/test？

Sol. `tar --exclude /home/test -czvf myhome.tar.gz /home/*`

21. 如何以指令 tar 打包檔案 main.c 2.c 3.c \*.h 成 myapp-1.0.tar？

Sol. `tar -cvf myapp-1.0.tar main.c 2.c 3.c *.h`

22. 如何將打包檔案 myapp-1.0.tar，再利用 gzip 壓縮檔案？

Sol. `gzip myapp-1.0.tar`

23. 打包檔案 myapp-1.0.tar，再利用 gzip 壓縮後之檔名為何？

Sol. `myapp-1.0.tar.gz`

24. 如何將打包且壓縮檔案 myapp-1.0.tar.gz，利用 gzip 解壓縮？

Sol. `gzip -d myapp-1.0.tar.gz`

25. 如何將打包檔案 myapp-1.0.tar，利用 tar 解打包？

Sol. `tar -xvf myapp-1.0.tar`

26. 如何以指令 tar 打包並壓縮檔案 main.c 2.c 3.c \*.h 成 myapp-1.0.tar.gz？

Sol. `tar -zcvf myapp-1.0.tar main.c 2.c 3.c *.h`

27. 如何以指令 tar 將 myapp-1.0.tar.gz 打包壓縮？

Sol. `tar -zxvf myapp-1.0.tar.gz`

28. makefile 新增一個的目標項目 dist，建立 tarball，其中一行 dist: myapp-1.0.tar.gz 代表意義為何？

Sol. 目標項目為 dist，其相依項目為 myapp-1.0.tar.gz

29. makefile 新增一個的目標項目 dist，建立 tarball，其中一行 myapp-1.0.tar.gz: myapp myapp.1 代表意義為何？

Sol. 目標項目為 myapp-1.0.tar.gz，其相依項目為 myapp 與 myapp.1

30. makefile 新增一個的目標項目 dist，建立 tarball，其中一行 tar -zcvf \$@ myapp-1.0 代表意義為何？

Sol. 將 myapp-1.0 打包壓縮，檔名為目前之目標項目。

31. 一般 tarball 軟體安裝時，請說明執行 ./configure 之目的為何？

Sol. ./ 表示在目前目錄下執行 configure，自動偵測作業環境，並產生 makefile。

32. 一般 tarball 軟體安裝時，請說明執行 make clean 之目的為何？

Sol. 執行目標項目 clean，清除目標檔。

33. 一般 tarball 軟體安裝時，請說明執行 make 之目的為何？

Sol. 編譯程式，產生二進位之執行檔。

34. 一般 tarball 軟體安裝時，請說明執行 make install 之目的為何？

Sol. 執行目標項目 install，將編譯好之可執行檔及相關檔案，複製到指定的目錄。

35. 一般 Linux distribution 釋出安裝的套件大多在那個目錄？

Sol. `/usr`

36. 一般 Linux 建議使用者自行安裝的套件放置在那個目錄？

Sol. `/usr/local`

37. Linux distribution 預設安裝套件的路徑 /usr/bin 放置什麼檔案？

Sol. 執行檔

38. Linux distribution 預設安裝套件的路徑 /usr/lib 放置什麼檔案？

Sol. 函式庫

39. Linux distribution 預設安裝套件的路徑 /usr/share/applnk 放置什麼檔案？

Sol. 程式選單連結檔

40. Linux distribution 預設安裝套件的路徑 /usr/share/apps/ 放置什麼檔案？

Sol. 程式內容

41. Linux distribution 預設安裝套件的路徑 /usr/share/doc/ 放置什麼檔案？

Sol. 程式文件

42. Linux distribution 預設安裝套件的路徑 /usr/share/icons/ 放置什麼檔案？

Sol. 程式圖標

43. Linux distribution 預設安裝套件的路徑 /usr/share/locale/ 放置什麼檔案？

Sol. 程式支援之語系

44. Linux distribution 預設安裝套件的路徑 /usr/share/services/ 放置什麼檔案？

Sol. 程式支援之服務

## 18.3 KPlayer Tarball 實例

### • 安裝前置作業

1. 自由軟體多媒體播放軟體Kplayer 官方網站下載原始碼
2. 在目前目錄（一般為 /usr/src ），解開原始碼：

```
[root@dywHome2 src]# tar -jxvf kplayer-0.5.3.tar.bz2
2 kplayer-0.5.3/
 kplayer-0.5.3/README
4 kplayer-0.5.3/AUTHORS
 kplayer-0.5.3/COPYING
6 kplayer-0.5.3/ChangeLog
 kplayer-0.5.3/INSTALL
8 kplayer-0.5.3/Makefile.am
 kplayer-0.5.3/Makefile.in
```



10 | ----以下省略----

3. 顯示解開原始碼之目錄：

```
2 [root@dywHome2 src]# ls -ld kplayer*
drwxrwxrwx 8 1002 1002 1024 Jan 9 2005 kplayer-0.5.3/
-rw-r--r-- 1 root root 3156593 Apr 3 08:50 kplayer-0.5.3.tar.bz2
```

4. 閱讀 README 與 INSTALL：

```
1 [root@dywHome2 kplayer-0.5.3]# cat INSTALL
截取 INSTALL 中之重要安裝訊息
3 Extract the tarball

5 tar -xjf kplayer-0.5.3.tar.bz2
cd kplayer-0.5.3
7
8 Create configure script
9
10 make -f Makefile.dist
11
12 Configure
13
14 ./configure --prefix=`kde-config --prefix`
15
16 Compile
17
18 make
19
20 Install
21
22 su -c 'make install'
23
24 Run
25
26 kplayer
```

• 依照上述 INSTALL 之重要安裝訊息，逐步安裝：

1. Extract the tarball，已完成。
2. Create configure script

```
2 [root@dywHome2 kplayer-0.5.3]# make -f Makefile.dist
This Makefile is only for the CVS repository
This will be deleted before making the distribution
4
```

```

6 *** Creating acinclude.m4
 *** Creating list of subdirectories
 *** Creating configure.files
8 *** Creating configure.in
 *** Creating aclocal.m4
10 *** Creating configure
 *** Creating config.h template
12 *** Creating Makefile templates
 *** Postprocessing Makefile templates
14 *** Creating date/time stamp
 *** Finished
16 Don't forget to run ./configure
 If you haven't done so in a while, run ./configure --help

```

### 3. Configure

- (a) 假設相依套件已安裝；
- (b) `./configure`：表示執行目前目錄下之自動偵測作業環境執行檔 `configure`；
- (c) `--prefix=PATH`：指定安裝目錄為 `PATH`。  
例如：`--prefix=/usr/local/kplayer`；
- (d) ``kde-config --prefix``：會先執行，以找到套件安裝目錄，在本系統為 `/usr`；
- (e) 若系統環境檢查一切正常，則出現訊息 `Good - your configure finished. Start make now`。

```

1 [root@dywHome2 kplayer-0.5.3]# ./configure --prefix=`kde-config --
 prefix`
 [root@dywHome2 kplayer-0.5.3]# ./configure --prefix=/usr/local/
 kplayer
3 checking build system type... i686-pc-linux-gnu
 checking host system type... i686-pc-linux-gnu
5 ---- 中間省略 ----
 configure: creating ./config.status
7 fast creating Makefile
 fast creating admin/Makefile
9 fast creating doc/Makefile
 fast creating doc/da/Makefile
11 fast creating doc/en/Makefile
 fast creating doc/pt/Makefile
13 fast creating doc/sv/Makefile
 fast creating icons/Makefile
15 fast creating kplayer/Makefile
 fast creating po/Makefile
17 config.pl: fast created 10 file(s).
 config.status: creating config.h
19 config.status: config.h is unchanged
 config.status: executing depfiles commands

```

```
21 | Good - your configure finished. Start make now
```

#### 4. Compile

```
2 | [root@dywHome2 kplayer-0.5.3]# make
----以上省略----
make[2]: Entering directory `/usr/src/kplayer-0.5.3'
4 | make[2]: Leaving directory `/usr/src/kplayer-0.5.3'
make[1]: Leaving directory `/usr/src/kplayer-0.5.3'
```

#### 5. Install

```
1 | [root@dywHome2 kplayer-0.5.3]# make install
----以上省略----
3 | make[2]: Leaving directory `/usr/src/kplayer-0.5.3'
make[1]: Leaving directory `/usr/src/kplayer-0.5.3'
```

#### 6. 查看安裝目錄 /usr/local/kplayer/

```
2 | [root@dywHome2 src]# ll /usr/local/kplayer/
total 3
4 | drwxr-xr-x 2 root root 1024 Apr 3 10:09 bin/
drwxr-xr-x 3 root root 1024 Apr 3 10:09 lib/
drwxr-xr-x 8 root root 1024 Apr 3 10:09 share/
```

- 進入目錄 kplayer-0.5.3 修改原始碼：

```
1 | [root@dywHome2 src]# cd kplayer-0.5.3
```

- 將 kplayer-0.5.3 整個目錄，打包壓縮成 tarball。

##### 1. 直接以 tar 指令打包壓縮

```
1 | [root@dywHome2 kplayer-0.5.3]# cd..
[root@dywHome2 src]# tar -zcvf kplayer-0.5.3.systray.tar.gz ./
kplayer-0.5.3
3 | ./kplayer-0.5.3/
./kplayer-0.5.3/README
5 | ./kplayer-0.5.3/AUTHORS
./kplayer-0.5.3/COPYING
7 | ./kplayer-0.5.3/ChangeLog
```

```

9 ./kplayer-0.5.3/INSTALL
 ---- 中間省略 ----
11 ./kplayer-0.5.3/Makefile
12 ./kplayer-0.5.3/libtool
13 ./kplayer-0.5.3/config.status
 ./kplayer-0.5.3/config.h

```

## 2. 利用 Makefile 產生 tarball

(a) 修改 Makefile 中版本序號 VERSION: 0.5.3 後面加入 .systray

```

1 [root@dywHome2 kplayer-0.5.3.systray]# vi Makefile
 VERSION = 0.5.3.systray
3 ## 所有目錄 kplayer-0.5.3 皆改成 kplayer-0.5.3.systray
 ## 在 vi 環境下指令 :1,$s/kplayer-0.5.3/kplayer-0.5.3.systray/g
5 install_sh = /usr/src/rpm/SOURCES/kplayer-0.5.3.systray/admin/
 install-sh

```

(b) 執行 make dist-gzip，產生 tarball  
kplayer-0.5.3.systray.tar.gz

```

1 [root@dywHome2 kplayer-0.5.3.systray]# make dist-gzip
 { test ! -d kplayer-0.5.3.systray || { find kplayer-0.5.3.
 systray
3 -type d ! -perm -200 -exec chmod u+w {} ';'
 && rm -fr kplayer-0.5.3.systray; }; }
5 mkdir kplayer-0.5.3.systray
 ----以下省略----

```

- 直接以 tarball 將軟體釋出

1. 將 tarball kplayer-0.5.3.systray.tar.gz 移到上一層目錄 /usr/src/rpm/SOURCES。

```

[root@dywHome2 kplayer-0.5.3.systray]# mv kplayer-0.5.3.systray.tar
.gz ../

```

2. 查看 tarball kplayer-0.5.3.systray.tar.gz。

```

1 [root@dywHome2 kplayer-0.5.3.systray]# cd..
 [root@dywHome2 SOURCES]# ll
3 total 6818
 drwxr-xr-x 9 root root 1024 Apr 5 15:17 kplayer-0.5.3.systray/

```

```
5 | -rw-r--r-- 1 root root 3793524 Apr 5 15:15 kplayer-0.5.3.systray.
 | tar.gz
 | -rw-r--r-- 1 root users 3156593 Jul 3 2006 kplayer-0.5.3.tar.bz2
```

3. 釋出 kplayer-0.5.3.systray.tar.gz

### 練習題

1. 若將套件 kplayer 安裝在 /usr/local/kplayer，則其執行檔放置在那個目錄？

Sol. /usr/local/kplayer/bin

2. 若將套件 kplayer 安裝在 /usr/local/kplayer，則其函式庫放置在那個目錄？

Sol. /usr/local/kplayer/lib

3. 若將套件 kplayer 安裝在 /usr/local/kplayer，則其程式選單連結檔放置在那個目錄？

Sol. /usr/local/kplayer/share/applications

4. 若將套件 kplayer 安裝在 /usr/local/kplayer，則其程式內容放置在那個目錄？

Sol. /usr/local/kplayer/share/apps

5. 若將套件 kplayer 安裝在 /usr/local/kplayer，則其程式文件放置在那個目錄？

Sol. /usr/local/kplayer/share/doc

6. 若將套件 kplayer 安裝在 /usr/local/kplayer，則其程式圖標放置在那個目錄？

Sol. /usr/local/kplayer/share/icons

7. 若將套件 kplayer 安裝在 /usr/local/kplayer，則其程式支援語系之檔案放置在那個目錄？

Sol. /usr/local/kplayer/share/locale

8. 若將套件 kplayer 安裝在 /usr/local/kplayer，則其程式支援服務之檔案放置在那個目錄？

Sol. /usr/local/kplayer/share/services

9. 若將套件 kplayer 安裝在 /usr/local/kplayer，如何以一個指令將其完全移除？

Sol. rm -rf /usr/local/kplayer

10. 請說明 Linux 套件相依性。

Sol. Linux 軟體套件，經常無法單獨安裝，往往需要其他相關套件存在，才能正常工作。

11. 套件 kplayer 安裝前，需要先安裝套件 mplayer，則移除套件 mplayer 後，kplayer 能否執行？

Sol. 無法執行。

12. 套件 kplayer 安裝前，必須先裝套件 mplayer，而 mplayer 必須先安裝套件 kdelibs-common，解除安裝之順序為何？

Sol. 移除順序為 kdelibs-common, mplayer, kplayer。

13. Tarball 套件 kplayer-0.5.3.tar.bz2 下載後要如何解打包壓縮？

Sol. `tar -jxvf kplayer-0.5.3.tar.bz2`

14. Tarball 套件 kplayer 解打包壓縮後，請說明 INSTALL 中重要訊息

Create configure script

make -f Makefile.dist。

Sol. 產生作業環境偵測腳本，需執行 `make -f Makefile.dist`。

15. Tarball 套件 kplayer 解打包壓縮後，請說明 INSTALL 中重要訊息。  
`configure --prefix=`kde-config --prefix``。

Sol. 先執行 `kde-config --prefix` 產生 kde 配置目錄，一般為 /usr；在目前目錄下以選項 `--prefix=/usr` 執行作業環境偵測 `configure`。

16. 安裝 tarball 套件 kplayer，執行 `make -f Makefile.dist` 最後出現 Don't forget to run `./configure`，代表意義為何？

Sol. 表示作業環境偵測腳本產生成功，不要忘了執行 `./configure`。

17. 安裝 tarball 套件 kplayer，執行 `./configure`，代表意義為何？

Sol. 表示執行目前目錄下之自動偵測作業環境執行檔 `configure`。

18. 安裝 tarball 套件 kplayer，執行 `./configure --prefix=PATH`，其中選項 `--prefix=PATH` 代表意義為何？

Sol. 指定安裝目錄為 PATH。

19. 安裝 tarball 套件 kplayer，執行 `./configure --prefix=`kde-config --prefix``，其中選項 ``kde-config --prefix`` 代表意義為何？

Sol. 換句話說，表示先執行 `kde-config --prefix`，產生 kde 配置目錄。

20. 安裝 tarball 套件 kplayer，執行 `./configure --prefix=`kde-config --prefix``，最後出現 Good - your configure finished. Start make now 代

表意義為何？

Sol. 表示自動偵測作業環境產生 `makefile` 成功，可以繼續執行 `make`。

# Chapter 19

## \*RPM 與 SRPM 套件發行

### 19.1 RPM 與 SRPM 套件

#### 1. 何謂 RPM？

- (a) RPM 全名是『RedHat Package Manager』簡稱為 RPM。
- (b) 由 Red Hat 公司發展出來，由於 RPM 使用方便，所以成了目前最熱門的套件管理程式。
- (c) RPM 是以資料庫記錄的方式來將所需要的套件安裝到 Linux 主機的一套管理程式。
- (d) RPM 將要安裝的套件先編譯過，並且打包好，安裝時 RPM 會先依照套件裡的紀錄資料查詢相依屬性套件是否滿足，若滿足則予以安裝，若不滿足則不予安裝。

#### 2. RPM 優點：

- (a) 免編譯：已編譯；
- (b) 避免安裝錯誤：安裝前先檢查系統的硬碟容量、作業系統版本等；
- (c) 瞭解套件：提供套件版本資訊、相依屬性套件名稱、套件用途說明、套件所含檔案等資訊；
- (d) 安裝、移除、更新升級及驗證套件方便：資料庫記錄參數，且一個指令即可完成。
- (e) 容易傳送：打包成一個檔案。

#### 3. RPM 缺點：

- (a) 安裝的環境必須與打包時的環境需求一致或相當；
- (b) 相依套件必須滿足；



(c) 反安裝時，下層套件不可先移除，否則可能造成整個系統的問題。

#### 4. RPM 套件的屬性相依

(a) RPM 打包套件檔案時，會同時加入套件的訊息。例如：

- i. 版本、
- ii. 打包套件者、
- iii. 相依屬性的套件、
- iv. 套件的功能說明、
- v. 套件的所有檔案與目錄紀錄。

(b) 在 Linux 系統上亦建立一個 RPM 套件資料庫。

(c) 當要安裝某個以 RPM 型態提供的套件時，如果資料庫顯示其相依套件不存在，則會顯示錯誤訊息。

(d) 屬性相依的克服方式

- i. 手動下載並安裝好所有相依套件。
- ii. 利用 `urpmi` 安裝 `rpm` 套件：自行尋找未安裝的相依套件加以安裝。

#### 5. 何謂 SRPM？

(a) SRPM 是 Source RPM，也就是 RPM 檔案裡含有原始碼（Source Code）。

(b) SRPM 除含 `tarball` 原始碼外，亦包含 `rpm` 規格檔 `.spec`，提供重建 `rpm` 套件所有資訊。例如：所須相依性套件。

(c) SRPM 可經由修改規格檔，以重建符合自己系統之 `rpm` 套件。

#### 6. SRPM 安裝與利用：

- (a) 先將該套件以 RPM 管理的方式安裝；
- (b) 安裝完成後，會有原始碼 `tarball` 及 `rpm` 規格檔 `.spec`；
- (c) 修改、重新編譯原始碼，再打包成 `tarball`；
- (d) 修改 `rpm` 規格檔，以適合自己的 Linux 環境；
- (e) 重建 RPM 套件；
- (f) 以 RPM 管理方式，將重建之 RPM 套件安裝至系統；

#### 7. RPM 與 SRPM 的格式：

|   |                                                    |                               |
|---|----------------------------------------------------|-------------------------------|
| 2 | <code>name-version-release.architecture.rpm</code> | <code>&lt;== RPM 檔名格式</code>  |
|   | <code>name-version-release.src.rpm</code>          | <code>&lt;== SRPM 檔名格式</code> |

(a) 例如檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 的意義為：

|   |                      |                |                    |                |                         |                |                   |                |                  |
|---|----------------------|----------------|--------------------|----------------|-------------------------|----------------|-------------------|----------------|------------------|
| 2 | <code>kplayer</code> | <code>-</code> | <code>0.5.3</code> | <code>-</code> | <code>5mdv2007.0</code> | <code>.</code> | <code>i586</code> | <code>.</code> | <code>rpm</code> |
|   | 套件名稱                 |                | 套件的版本資訊            |                | 釋出的次數                   |                | 適合的硬體平台           |                | 附檔名              |

(b) 釋出版本次數：5mdv2007.0。

- i. 不同的 distribution 會有不同的環境與函式庫，釋出版本次數後可能會再加上 distribution 的簡寫。
- ii. mdv2007.0 表示 Linux distribution Mandriva 2007.0。
- iii. 安裝 RPM 套件，最好選擇相同環境之檔案，即 distribution 的簡寫與自己的系統環境相同。

(c) 操作硬體平台之選擇：

i. 平台名稱說明：

| 平台名稱   | 適合平台說明                                                 |
|--------|--------------------------------------------------------|
| i386   | 幾乎適用於所有的 x86 平台。<br>i 是 Intel 相容的 CPU，386 是 CPU 的等級。   |
| i586   | 586 等級的電腦，包括 pentium 第一代 MMX CPU，AMD 的 K5, K6 系列等 CPU。 |
| i686   | pentium II 以後的 Intel 系列 CPU，及 K7 以後等級的 CPU。            |
| noarch | 沒有任何硬體等級上的限制。                                          |
| athlon | AMD Athlon 晶片。                                         |

- ii. i386 可安裝在 586 或 686 的機器；
- iii. i686 是針對 686 等級的 CPU 進行最佳化編譯，所以不一定可以使用於 386 或 586 的硬體。
- iv. 在 686 的機器上使用 i686 的檔案會比使用 i386 的檔案，效能可能比較好。
- v. 目前 Linux distribution Mandriva 2007.0 釋出之 rpm，平台名稱為 i586。

## 8. RPM 檔案系統

(a) Mandriva 2007 以 `/usr/src/rpm/` 為工作目錄

(b) `/usr/src/rpm` 以下五個目錄：

| rpm 子目錄 | 說明                       |
|---------|--------------------------|
| BUILD   | 編譯過程中，暫存資料放置目錄。          |
| RPMS    | 編譯打包成功後，完成的 rpm 檔案放置目錄。  |
| SOURCES | 放置套件的原始碼，即 tarball 檔。    |
| SPECS   | 放置套件的規格檔 spec 檔案。        |
| SRPMS   | 編譯打包成功後，完成的 srpm 檔案放置目錄。 |

(c) /usr/src/rpm/RPMS 目錄通常有一些架構的子目錄，如下：

```
$ ls RPMS
2 athlon
 i386
4 i486
 i586
6 i686
 noarch
```

## 練習題

1. RPM 全名為何？

Sol. RedHat Package Manager

2. 何謂 RPM？

Sol. 以資料庫記錄的方式來將所需要的套件安裝到 Linux 主機的一套管理程式。

3. RPM 套件是否經過編譯？

Sol. 是。

4. RPM 套件安裝前是否會檢查相依屬性套件？

Sol. 會。

5. RPM 套件安裝時將套件的資訊寫入資料庫之目的為何？

Sol. 便於查詢、驗證、升級更新及反安裝。

6. 請列舉 RPM 套件之優點。

Sol. 1. 免編譯；2. 避免安裝錯誤；3. 可瞭解套件；4. 更新升級、移除、查詢與驗證方便；5. 容易傳送。

7. 請列舉 RPM 套件之缺點。

Sol. 1. 安裝的環境要一致；2. 相依套件要滿足；3. 下層套件不可先移除。

8. RPM 打包套件檔案時，會同時加入套件的訊息。請列舉三項。

Sol. 1. 版本、2. 打包套件者、3. 相依屬性的套件、4. 套件的功能說明、5. 套件的所有檔案與目錄記錄。

9. 如何克服 RPM 安裝套件屬性相依的問題？

Sol. 1. 手動下載並安裝好所有相依套件、2. 利用 `urpmi` 安裝 rpm 套件。

10. 何謂 SRPM？

Sol. SRPM 是 Source RPM，也就是 RPM 檔案裡含有原始碼。

11. 請說明 SRPM 與 tarball 之差異？

Sol. SRPM 除含 tarball 原始碼外，亦包含 rpm 規格檔 `.spec`，提供重建 rpm 套件所有資訊。例如：所須相依性套件。

12. 請說明 SRPM 之優點？

Sol. SRPM 可經由修改規格檔，以重建符合自己系統之 rpm 套件。

13. 下載 SRPM 並以 RPM 管理的方式安裝後，會有那些檔案？

Sol. tarball 及 rpm 規格檔 `.spec`。

14. 修改 rpm 規格檔 `.spec` 的主要目的為何？

Sol. 以重建符合自己系統之 rpm 套件。

15. 請寫出 RPM 檔名格式。

Sol. `name-version-release.architecture.rpm`

16. 請寫出 SRPM 檔名格式。

Sol. `name-version-release.src.rpm`

17. 檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 之套件名稱為何？

Sol. `kplayer`

18. 檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 之套件版本資訊為何？

Sol. `0.5.3`

19. 檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 之套件釋出的次數為何？

Sol. `5mdv2007.0`

20. 檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 之套件適合的硬體平台為何？

Sol. `i586`

21. 檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 中 `mdv2007.0` 代表意義為何？

Sol. `mdv2007.0` 表示打包之環境為 Mandriva 2007.0。

22. 將檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 安裝在 Fedora 7 是否恰當？為什麼？

Sol. 不恰當，因為包環境為 Mandriva 2007.0，不見得適合合 Fedora 7。

23. 檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 中 `i586` 代表意義為何？

Sol. 適合安裝於 Intel 相容的 CPU 等級 586 的電腦。

24. 檔案 `kplayer-0.5.3-5mdv2007.0.noarch.rpm` 中 `noarch` 代表意義為何？

Sol. 安裝上沒有任何硬體等級上的限制。

25. 檔案 `kplayer-0.5.3-5mdv2007.0.athlon.rpm` 中 `athlon` 代表意義為何？

Sol. 適合安裝於 AMD Athlon 晶片的電腦。

26. 檔案 `kplayer-0.5.3-5mdv2007.0.i386.rpm` 是否可安裝在 586 等級的電腦？

Sol. 可以。

27. 檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 是否可安裝在 386 等級的電腦？

Sol. 不可以。

28. RPM 檔案系統工作目錄 `/usr/src/rpm/` 下之子目錄 `BUILD` 主要用途為何？

Sol. 編譯過程中，暫存資料放置目錄。

29. RPM 檔案系統工作目錄 `/usr/src/rpm/` 下之子目錄 `RPMS` 主要用途為何？

Sol. 編譯打包成功後，完成的 `rpm` 檔案放置目錄。

30. RPM 檔案系統工作目錄 `/usr/src/rpm/` 下之子目錄 `SOURCES` 主要用途為何？

Sol. 放置套件的原始碼，即 `tarball` 檔。

31. RPM 檔案系統工作目錄 `/usr/src/rpm/` 下之子目錄 `SPECS` 主要用途為何？

Sol. 放置套件的規格檔 `spec` 檔案。

32. RPM 檔案系統工作目錄 `/usr/src/rpm/` 下之子目錄 `SRPMS` 主要用途為何？

Sol. 編譯打包成功後，完成的 `srpm` 檔案放置目錄。

33. 以 `i586` 硬體重建 RPM 檔案，產生之 `rpm` 套件放在那個目錄？

Sol. `/usr/src/rpm/RPMS/i586`

## 19.2 建立 RPM 套件

- 步驟：

1. 收集要包裝的軟體。
2. 產生 `spec` 檔案，這個檔案說明如何建立套件。

3. 以 `rpmbuild` 命令建立套件。

- 聚集 (gather) 軟體

1. 收集應用程式原始碼、建立的檔案 (例如 `makefile`)、線上的使用手冊文件。
2. 收集軟體最簡單的方法，就是將這些檔案以 `tarball` 產生方法，裝成一個 `tarball`。
3. `tarball` 的檔案名稱，必須有應用程式名稱和版本編號，例如 `myapp-1.0.tar.gz`。
4. 將 `myapp-1.0.0.tar.gz` 複製到 RPM 的 `SOURCES` 目錄。

```
1 $ cp myapp-1.0.tar.gz /usr/src/rpm/SOURCES
```

- 建立程式 `myapp` 的 RPM spec 檔案，檔案名稱定為 `myapp.spec`。

1. 註解：以 `#` 為行首的資料。在 spec 檔中加入註解，例如：

```
1 # spec file for package myapp (Version 1.0)
```

2. 標籤 (tags)：資料定義。一般格式如下：

```
1 <something>:<something-else>
```

(a) 導文 (preamble)：設定套件名稱、版本編號和其它資訊。例如：

```
1 Vendor: Wrox Press #發展的廠商
 Distribution: Any #
3 Name: myapp #套件的名稱
 Version: 1.0 #套件的版本資訊
5 Release: 1 #版本打包的次數說明
 Packager: neil@provider.com #套件打包者
7 License: Copyright 2003 by Wrox Press #套件的授權模式
 Group: Applications/Media #套件的發展團體名稱
9 Provides: goodness #系統提供的功能
 Requires: mysql >= 3.23 #套件之相依套件
11 Buildroot: %{_tmppath}/%{name}-%{version}-root
 Source: %{name}-%{version}.tar.gz #套件的來源
13 Summary: Trivial application #主要的套件說明
```

- (b) `%description` 標籤：提供相關說明，不同於上述格式，其以 `%` 開頭，且可展開成多行。例如：

```
1 %description
 MyApp Trivial Application
3 A trivial application used to demonstrate development tools.
 This version pretends it requires MySQL at or above 3.23.
5 Authors: Neil Matthew and Richard Stones
```

- (c) 套件名稱與版本巨集（`marco`），例如：

```
1 Source: %{name}-%{version}.tar.gz
3 # %{name} 與 %{version} 為 RPM 的巨集，分別表示套件名稱及版本；
 # 在本例中 rpmbuild 命令會將 %{name} 展開成為 myapp，而 %{
 version} 展開為 1.0；
```

- (d) `Buildroot` 設定安裝目錄。

```
Buildroot: %{_tmppath}/%{name}-%{version}-root
2 # %{_tmppath} 巨集內容可以 rpm --showrc 查詢。
```

### 3. RPM 工作腳本（`scripts`）

- (a) `%prep:build` 前的準備（`prepare`）動作，主要是執行套件解打包壓縮。由於準備動作幾乎是公式化的動作，故已寫成巨集 `%setup`。例如：

```
%prep
2 %setup -q # 選項 -q 代表設定為安靜模式。
```

- (b) `%build`：建立應用程式。例如：

```
%build
2 make
```

- (c) `%install`：安裝應用程式、使用手冊和支援的檔案。例如：

```
%install
2 mkdir -p $RPM_BUILD_ROOT%{_bindir}
 mkdir -p $RPM_BUILD_ROOT%{_mandir}
4 install -m755 myapp $RPM_BUILD_ROOT%{_bindir}/myapp
```

```

6 | install -m755 myapp.1 $RPM_BUILD_ROOT%{_mandir}/myapp.1
 | # %RPM_BUILD_ROOT 環境變數紀錄 Buildroot 的位置。
 | # %{bindir} 和 %{_mandir} 巨集，分別展開為二進位執行檔案目錄和
 | 使用手冊目錄。

```

(d) %clean：清除 rpmbuild 命令產生的檔案。例如：

```

1 | %clean
 | rm -rf $RPM_BUILD_ROOT

```

4. %files：列出套件包含的檔案。例如：

```

2 | %files
 | %{_bindir}/myapp
 | %{_mandir}/myapp.1

```

5. %post：套件安裝後執行的腳本。例如：

```

1 | %post
 | mail root -s "myapp installed - please register" </dev/null

```

6. 應用程式 myapp 的完整 spec 檔案如下：

```

2 | #
 | # spec file for package myapp (Version 1.0)
 | #
4 | Vendor: Wrox Press
 | Distribution: Any
6 | Name: myapp
 | Version: 1.0
8 | Release: 1
 | Packager: neil@provider.com
10 | License: Copyright 2003 by Wrox Press
 | Group: Applications/Media
12 | Provides: goodness
 | Requires: mysql >= 3.23
14 | Buildroot: %{_tmppath}/%{name}-%{version}-root
 | source: %{name}-%{version}.tar.gz
16 | Summary: Trivial application
 | %description
18 | MyApp Trivial Application
 | A trivial application used to demonstrate development tools.
20 | This version pretends it requires MySQL at or above 3.23.
 | Authors: Neil Matthew and Richard Stones
22 | %prep

```



```

24 %setup -q
%build
make
26 %install
mkdir -p $RPM_BUILD_ROOT%{_bindir}
28 mkdir -p $RPM_BUILD_ROOT%{_mandir}
install -m755 myapp $RPM_BUILD_ROOT%{_bindir}/myapp
30 install -m755 myapp.1 $RPM_BUILD_ROOT%{_mandir}/myapp.1
%clean
32 rm -rf $RPM_BUILD_ROOT
%post
34 mail root -s "myapp installed - please register" </dev/null
%files
36 %{_bindir}/myapp
%{_mandir}/myapp.1

```

- 利用 rpmbuild 建立一個 RPM 套件。

#### 1. 語法：rpmbuild -bBuildStage spec\_file

| 選項  | bBuildStage | 意義                     |
|-----|-------------|------------------------|
| -ba |             | 建立所有，包含二進位和原始碼 RPM。    |
| -bb |             | 建立一個二進位 RPM。           |
| -bc |             | 建立（編譯）程式，但是不會建立整個 RPM。 |
| -bp |             | 準備建立一個二進位 RPM。         |
| -bi |             | 產生一個二進位 RPM，並且安裝它。     |
| -bl |             | 檢查 RPM 的檔案列表。          |
| -bs |             | 只建立一個原始碼 RPM。          |

#### 2. 建立套件輸出結果：

```

1 $ rpmbuild -ba myapp.spec
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.71108
3 + umask 022
+ cd /usr/src/redhat/BUILD
5 + LANG=C
+ export LANG
7 + cd /usr/src/redhat/BUILD
+ rm -rf myapp-1.0
9 + /usr/bin/gzip -dc /usr/src/redhat/SOURCES/myapp-1.0.tar.gz
+ tar -xf -
11 + STATUS=0
+ '[' 0 -ne 0 ']'
13 + cd myapp-1.0
++ /usr/bin/id -u
15 + '[' 0 = 0 ']'
+ /bin/chown -Rhf root .
17 ++ /usr/bin/id -u
+ '[' 0 = 0 ']'

```

```

19 + /bin/chgrp -Rh root .
 + /bin/chmod -Rf a+rX,g-w,o-w .
21 + exit 0
 Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.43788
23 + umask 022
 + cd /usr/src/redhat/BUILD
25 + cd myapp-1.0
 + LANG=C
27 + export LANG
 + make
29 gcc -g -Wall -ansi -c -o main.o main.c
 gcc -g -Wall -ansi -c -o 2.o 2.c
31 ar rv mylib.a 2.o
 a - 2.o
33 gcc -g -Wall -ansi -c -o 3.o 3.c
 ar rv mylib.a 3.o
35 a - 3.o
 gcc -o myapp main.o mylib.a
37 + exit 0
 Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.90688
39 + umask 022
 + cd /usr/src/redhat/BUILD
41 + cd myapp-1.0
 + LANG=C
43 + export LANG
 + mkdir -p /var/tmp/myapp-1.0-root/usr/bin
45 + mkdir -p /var/tmp/myapp-1.0-root/usr/share/man
 + install -m755 myapp /var/tmp/myapp-1.0-root/usr/bin/myapp
47 + install -m755 myapp.1 /var/tmp/myapp-1.0-root/usr/share/man/myapp
 .1
 + /usr/lib/rpm/find-debuginfo.sh /usr/src/redhat/BUILD/myapp-1.0
49 extracting debug info from /var/tmp/myapp-1.0-root/usr/bin/myapp
 1 block
51 + /usr/lib/rpm/redhat/brp-compress
 + /usr/lib/rpm/redhat/brp-strip /usr/bin/strip
53 + /usr/lib/rpm/redhat/brp-strip-static-archive /usr/bin/strip
 + /usr/lib/rpm/redhat/brp-strip-comment-note /usr/bin/strip
55 /usr/bin/objdump
 Processing files: myapp-1.0-1
57 Provides: goodness
 Requires(interp): /bin/sh
59 Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1
 rpmlib(PayloadFilesHavePrefix) <= 4.0-1
61 Requires(post): /bin/sh
 Requires: libc.so.6 libc.so.6(GLIBC_2.0) mysql >= 3.23
63 Processing files: myapp-debuginfo-1.0-1
 Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1
65 rpmlib(PayloadFilesHavePrefix) <= 4.0-1
 Checking for unpackaged file(s): /usr/lib/rpm/check-files
67 /var/tmp/myapp-1.0-root
 Wrote: /usr/src/redhat/SRPMS/myapp-1.0-1.src.rpm
69 Wrote: /usr/src/redhat/RPMS/i386/myapp-1.0-1.i386.rpm
 Wrote: /usr/src/redhat/RPMS/i386/myapp-debuginfo-1.0-1.i386.rpm
71 Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.17422

```

```
73 + umask 022
 + cd /usr/src/redhat/BUILD
 + cd myapp-1.0
75 + rm -rf /var/tmp/myapp-1.0-root
 + exit 0
```

- 建立程式成功後產生之檔案

1. 在 RPMS 目錄之子目錄 (例如 RPMS/i386) 的二進位 RPM 檔：

```
myapp-1.0-1.i386.rpm
```

2. 在 SRPMS 目錄的原始碼 RPM 檔：

```
1 myapp-1.0-1.src.rpm
```

## 練習題

1. 請說明建立 RPM 套件三步驟。

Sol. 1. 收集要包裝的軟體。2. 產生 spec 規格檔案。3. 以 rpmbuild 命令建立套件。

2. 建立 RPM 套件要先收集程式相關檔案，但一般而言，軟體開發所產生之檔案不只一個，最簡單的方法為何？

Sol. 將所有相關檔案打包壓縮成一個 tarball。

3. 建立 RPM 套件要先收集程式相關檔案，並將其複製到 RPM 檔案系統下那個目錄？

Sol. SOURCES

4. RPM 規格檔案，一般附檔名為何？

Sol. spec

5. 如何在 RPM 規格檔案中加入註解？

Sol. 行首以 # 開頭。

6. RPM 規格檔案中標籤定義 Vendor: Wrox Press，意義為何？

Sol. 發行的廠商 Wrox Press

7. RPM 規格檔案中標籤定義 Distribution: Any，意義為何？

Sol. 適用任何的 Linux distribution

8. RPM 規格檔案中標籤定義 Name: myapp，意義為何？

Sol. 套件的名稱 myapp

9. RPM 規格檔案中標籤定義 Version: 1.0，意義為何？

Sol. 套件的版本為 1.0

10. RPM 規格檔案中標籤定義 Release: 1，意義為何？

Sol. 版本打包的次數 1

11. RPM 規格檔案中標籤定義 Packager: dywang@cyut.edu.tw，意義為何？

Sol. 套件打包者為 dywang@cyut.edu.tw

12. RPM 規格檔案中標籤定義 License: GPL，意義為何？

Sol. 套件的授權模式 GPL

13. RPM 規格檔案中標籤定義 Group: Applications/Media，意義為何？

Sol. 套件的發展團體名為 Applications/Media

14. RPM 規格檔案中標籤定義 Provides: goodness，意義為何？

Sol. 系統提供的功能有 goodness

15. RPM 規格檔案中標籤定義 Requires: mysql >= 3.23，意義為何？

Sol. 套件之相依套件 mysql 且版本需 >= 3.23

16. RPM 規格檔案中標籤定義 Buildroot: %{\_tmppath}/%{name}-%{version}-root，意義為何？

Sol. 建立 rpm 套件的暫存工作目錄為 %{\_tmppath}/%{name}-%{version}-root

17. RPM 規格檔案中標籤定義 Source: %{name}-%{version}.tar.gz，意義為何？

Sol. 套件的來源 %{name}-%{version}.tar.gz

18. RPM 規格檔案中標籤定義 Summary: Trivial application，意義為何？

Sol. 主要的套件說明 Trivial application

19. RPM 規格檔案中標籤 %description，與其他資料定義標籤，主要不同為何？

Sol. 以 % 開頭，且可展開成多行，可提供套件相關說明。

20. RPM 規格檔案中標籤定義 Source: %{name}-%{version}.tar.gz，其中 %{name}-%{version} 意義為何？

Sol. %{name} 與 %{version} 為 RPM 的巨集，rpmbuild 命令會將其展開為 套件名稱及版本。

21. RPM 規格檔案中標籤定義 Buildroot: `%{_tmppath}/%{name}-%{version}-root`，其中 `%{_tmppath}` 意義為何？

Sol. `%{_tmppath}` 為 RPM 的巨集；`rpmbuild` 命令會將其展開為：建立 rpm 套件的暫存工作目錄。

22. RPM 規格檔案中工作腳本 `%prep`，主要工作內容為何？

Sol. 建立 rpm 套件前的準備動作，主要是執行套件解打包壓縮。

23. RPM 規格檔案中工作腳本 `%prep`，通常是執行那個巨集？為什麼？

Sol. 由於準備動作幾乎是公式化的動作，故已寫成巨集 `%setup`。

24. RPM 規格檔案中工作腳本 `%setup -q`，其中選項 `-q` 代表意義為何？

Sol. 安靜模式。

25. RPM 規格檔案中工作腳本 `%build`，主要工作內容為何？

Sol. 建立應用程式，通常是執行 `make`。

26. RPM 規格檔案之工作腳本中 `%RPM_BUILD_ROOT`，代表意義為何？

Sol. `%RPM_BUILD_ROOT` 為環境變數，記錄 Buildroot 的位置。

27. RPM 規格檔案之工作腳本中 `%{_bindir}`，代表意義為何？

Sol. `%{_bindir}` 為 RPM 的巨集；`rpmbuild` 命令會將其展開為二進位執行檔案目錄。

28. RPM 規格檔案之工作腳本中 `%{_mandir}`，代表意義為何？

Sol. `%{_mandir}` 為 RPM 的巨集；`rpmbuild` 命令會將其展開為使用手冊目錄。

29. RPM 規格檔案中工作腳本 `%clean`，主要工作內容為何？

Sol. 清除 `rpmbuild` 命令產生的檔案。

30. RPM 規格檔案中工作腳本 `%file`，主要工作內容為何？

Sol. 列出套件包含的檔案。

31. RPM 規格檔案中工作腳本 `%post`，主要工作內容為何？

Sol. 套件安裝後需執行的動作。

32. 建立 RPM 套件指令為何？

Sol. `rpmbuild`

33. 若 RPM 規格檔為 `myapp.spec`，如何同時建立 RPM 與 SRPM 套件？

Sol. `rpmbuild -ba myapp.spec`

34. 若 RPM 規格檔為 myapp.spec，如何只建立 RPM 套件？

Sol. `rpmbuild -bb myapp.spec`

35. 若 RPM 規格檔為 myapp.spec，如何建立 RPM 並安裝它？

Sol. `rpmbuild -bi myapp.spec`

36. 若 RPM 規格檔為 myapp.spec，如何只建立 SRPM 套件？

Sol. `rpmbuild -bs myapp.spec`

## 19.3 KPlayer RPM 實例

### • 安裝 rpm 原始碼

1. 下載 Mandriva 2007.0 kplayer rpm 原始碼至目錄 /usr/src/rpm/SRPMS

```
1 [root@dywHome2 SRPMS]# wget ftp://ftp.isu.edu.tw/Linux/Mandriva/
 devel/2007.0/SRPMS/contrib/release/kplayer-0.5.3-5mdv2007.0.src.rpm
```

2. 使用 urpmi 在 /usr/src/rpm/SRPMS，安裝 rpm 原始碼：

```
2 [root@dywHome2 SRPMS]# cd ..
3 [root@dywHome2 rpm]# urpmi SRPMS/kplayer-0.5.3-5mdv2007.0.src.rpm
To satisfy dependencies, the following package is going to be
 installed:
4 libnas2-devel-1.8-1.1mdv2007.0.i586
Proceed with the installation of the 1 packages? (0 MB) (Y/n) y
6
 ftp://ftp.isu.edu.tw/Linux/Mandriva/official/2007.0/i586/media/
 main
8 /updates/libnas2-devel-1.8-1.1mdv2007.0.i586.rpm
installing libnas2-devel-1.8-1.1mdv2007.0.i586.rpm from /var/cache/
 urpmi/rpms
10 Preparing... #
 #####
 1/1: libnas2-devel #
 #####
```

3. 查看 rpm 原始碼之目錄：

```
1 [root@dywHome2 rpm]# ls BUILD/ RPMS/ SOURCES/ SPECS/ SRPMS/
3 BUILD/:
 RPMS/:
5 athlon/ i386/ i486/ i586/ i686/ noarch/
```

```

7 SOURCES/:
 kplayer-0.5.3.tar.bz2
9
11 SPECS/:
 kplayer.spec
13 SRPMS/:
 kplayer-0.5.3-5mdv2007.0.src.rpm

```

• 程式修改與重新打包壓縮：

1. tarball 解打包壓縮在目錄 kplayer-0.5.3。
2. 修改目錄名稱：

```

2 [root@dywHome2 SOURCES]# mv kplayer-0.5.3 kplayer-0.5.3.systray
 [root@dywHome2 SOURCES]# ll
total 3097
4 drwxr-xr-x 9 root root 1024 Apr 5 15:17 kplayer-0.5.3.systray/
 -rw-r--r-- 1 root users 3156593 Jul 3 2006 kplayer-0.5.3.tar.bz2

```

3. 修改程式：不詳述。
4. Makefile 修改：與 tarball 相同。
5. 版本序號 0.5.3 後，加入 .systray，故打包壓縮之 tarball 名稱爲：  
kplayer-0.5.3.systray.tar.gz。

```

1 [root@dywHome2 SOURCES]# ll
total 6820
3 drwxr-xr-x 9 root root 1024 Apr 5 17:27 kplayer-0.5.3.systray/
 -rw-r--r-- 1 root root 3795740 Apr 5 17:27 kplayer-0.5.3.systray.
 tar.gz
5 -rw-r--r-- 1 root users 3156593 Jul 3 2006 kplayer-0.5.3.tar.bz2
 [root@dywHome2 SOURCES]# cd ../SPECS/

```

• 建立 rpm 檔案：

1. 修改 kplayer.spec：版本序號 0.5.3 後，加入 .systray

```

2 [root@dywHome2 rpm]# vi SPECS/kplayer.spec
 %define version 0.5.3.systray

```

## 2. 產生 kplayer-0.5.3.systray-5mdv2007.0.i586.rpm

```

[root@dywHome2 SPECS]# rpmbuild -bb kplayer.spec
2 ----以上省略----
Wrote: /usr/src/rpm/RPMS/i586/kplayer-0.5.3.systray-5mdv2007.0.i586
 .rpm
4 Wrote: /usr/src/rpm/RPMS/i586/kplayer-debug-0.5.3.systray-5mdv2007
 .0.i586.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.6408
6 + umask 022
+ cd /usr/src/rpm/BUILD
8 + cd kplayer-0.5.3.systray
+ rm -rf /var/tmp/kplayer-0.5.3.systray-buildroot
10 + exit 0

```

## 3. 查看 rpm 檔案

```

[root@dywHome2 SPECS]# ll ../RPMS/i586/
2 total 6377
-rw-r--r-- 1 root root 3122228 Apr 5 17:29
4 kplayer-0.5.3.systray-5mdv2007.0.i586.rpm
-rw-r--r-- 1 root root 3378549 Apr 5 17:29
6 kplayer-debug-0.5.3.systray-5mdv2007.0.i586.rpm

```

## • 建立二進位和原始碼 RPM 檔案：

## 1. 產生 kplayer-0.5.3.systray-5mdv2007.0.i586.src.rpm

```

[root@dywHome2 SPECS]# rpmbuild -ba kplayer.spec
2 ----以上省略----
Wrote: /usr/src/rpm/SRPMS/kplayer-0.5.3.systray-5mdv2007.0.src.rpm
4 Wrote: /usr/src/rpm/RPMS/i586/kplayer-0.5.3.systray-5mdv2007.0.i586
 .rpm
Wrote: /usr/src/rpm/RPMS/i586/kplayer-debug-0.5.3.systray-5mdv2007
 .0.i586.rpm
6 Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.26586
+ umask 022
8 + cd /usr/src/rpm/BUILD
+ cd kplayer-0.5.3.systray
10 + rm -rf /var/tmp/kplayer-0.5.3.systray-buildroot
+ exit 0

```

## 2. 查看二進位和原始碼 RPM 檔案

```

1 [root@dywHome2 SPECS]# ll ../SRPMS/ ../RPMS/i586/
 ../RPMS/i586/:
3 total 6377

```



```
5 -rw-r--r-- 1 root root 3122404 Apr 5 17:46
 kplayer-0.5.3.systray-5mdv2007.0.i586.rpm
7 -rw-r--r-- 1 root root 3378599 Apr 5 17:46
 kplayer-debug-0.5.3.systray-5mdv2007.0.i586.rpm
9
 ../SRPMS/:
 total 6845
11 -rw-r--r-- 1 root root 3168326 Apr 5 10:35
 kplayer-0.5.3-5mdv2007.0.src.rpm
13 -rw-r--r-- 1 root root 3808850 Apr 5 17:46
 kplayer-0.5.3.systray-5mdv2007.0.src.rpm
```

- 將 rpm 及原始碼 rpm 釋出

1. 釋出 kplayer-0.5.3.systray-5mdv2007.0.i586.rpm
2. 釋出 kplayer-0.5.3.systray-5mdv2007.0.src.rpm

# Chapter 20

## \* 套件修補、檢驗與管理

### 20.1 patch 程式

- 為什麼 patch？
  1. 檔案（版本）之間的差異，可以指令 diff 儲存在一個 patch 檔案。
  2. 若舊版本需要修改，只要將 patch 檔案釋出。
  3. 使用者可以指令 patch，配合 patch 檔案中記錄之新舊版差異，將舊版程式更新。
  4. 使用者若發現並修正一個程式的臭蟲，簡單、正確的方式是，寄一個 patch 檔案給作者，而不要只是說明修正的地方。
- diff 指令：比對兩個檔案之間的差異，一般是用在 ASCII 純文字檔的比對上。
  1. diff 指令用法：

```
[root@linux ~]# diff [-bBiqn] from-file to-file
2 選項：
 from-file : 檔名，作為原始比對檔案的檔名；
4 to-file : 檔名，作為目的比對檔案的檔名；
 # from-file 或 to-file 可以 - 取代，- 代表『Standard input』。
6
 -b : 忽略一行當中，多個空白的差異
 (例如 "about me" 與 "about me" 視為相同)
8 -B : 忽略空白行的差異。
10 -i : 忽略大小寫的不同。
 -q : 只列出檔案是否有差異。
12 -n : 以 RCS 格式輸出檔案之差異。
 -c (-C NUM) : 兩個檔案皆加入差異部分前後 NUM 行，以增加輸出之可讀
 性。預設 NUM=3。
14 -u (-U NUM) : 加入差異部分前後 NUM 行，以增加輸出之可讀性。預設 NUM
 =3。
```

2. 預處理：將 `/etc/passwd` 第四行刪除，第六行取代為『no six line』，新的檔案放到 `/tmp/test`。

```

1 [root@linux ~]# mkdir -p /tmp/test
2 [root@linux ~]# cat /etc/passwd | \
> sed -e '4d' -e '6c no six line' > /tmp/test/passwd
4 # sed 後面若要接超過兩個以上的動作時，每個動作前面得加 -e 。

```

3. 比對 `/tmp/test/passwd` 與 `/etc/passwd` 的差異。

```

1 [root@dywHome2 ~]# diff /etc/passwd /tmp/test/passwd
2 4d3
< adm:x:3:4:adm:/var/adm:/bin/sh
4 6c5
< sync:x:5:0:sync:/sbin:/bin/sync
6 ---
> no six line

```

4. 只列出檔案是否有差異。

```

1 [root@dywHome2 ~]# diff -q /etc/passwd /tmp/test/passwd
Files /etc/passwd and /tmp/test/passwd differ

```

5. 以 RCS 格式輸出檔案之差異。

```

1 [root@dywHome2 ~]# diff -n /etc/passwd /tmp/test/passwd
2 d4 1
d6 1
4 a6 1
no six line

```

6. 加入差異部分前後 3 行，以增加輸出之可讀性。

```

1 [root@dywHome2 tmp]# diff -u old/ new/ > test.patch
2 [root@dywHome2 tmp]# cat test.patch
3 diff -u old/passwd new/passwd
4 --- old/passwd 2008-04-16 13:14:50.000000000 -0400
5 +++ new/passwd 2008-04-16 11:15:12.000000000 -0400
6 @@ -1,9 +1,8 @@
7 root:x:0:0:root:/root:/bin/bash
8 bin:x:1:1:bin:/bin:/bin/sh
9 daemon:x:2:2:daemon:/sbin:/bin/sh
10 -adm:x:3:4:adm:/var/adm:/bin/sh
11 lp:x:4:7:lp:/var/spool/lpd:/bin/sh
 -sync:x:5:0:sync:/sbin:/bin/sync

```

```

13 | +no six line
 | shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
15 | halt:x:7:0:halt:/sbin:/sbin/halt
 | mail:x:8:12:mail:/var/spool/mail:/bin/sh

```

7. 比對 /etc 與 /tmp/test 的差異。

```

[root@linux ~]# diff /etc /tmp/test
2 |(前面省略).....
 | Only in /etc: paper.config
4 | diff /etc/passwd /tmp/test/passwd
 | 4d3
 | < adm:x:3:4:adm:/var/adm:/sbin/nologin
 | 6c5
 | < sync:x:5:0:sync:/sbin:/bin/sync
 | ---
10 | > no six line
 | Only in /etc: passwd-
12 |(後面省略).....

```

- cmp：主要利用『位元』單位去比對（diff 主要是以『行』為單位比對，cmp 則是以『位元』為單位去比對）。

1. cmp 指令用法：

```

[root@linux ~]# cmp [-bcsi] file1 file2
2 | 選項：
 | -b : 列出第一個的不同點之字元。
 | -c : 同上。
 | -i SKIP1:SKIP2 : file1 與 file2 分別忽略前 SKIP1 與 SKIP2 位元。
 | -s : 安靜模式，不顯示任何訊息。
6 |

```

2. 用 cmp 比較 /etc/passwd 與 /tmp/test/passwd

```

[root@dywHome2 ~]# cmp /etc/passwd /tmp/test/passwd
2 | /etc/passwd /tmp/test/passwd differ: byte 94, line 4
 | # 不同點在第四行，而且位元數是在第 94 個位元。

```

3. 列出第一個的不同點之字元

```

1 | [root@dywHome2 ~]# cmp -c /etc/passwd /tmp/test/passwd
 | /etc/passwd /tmp/test/passwd differ: byte 94, line 4 is 141 a 154 l
3 | # 不同點在檔案 /etc/passwd 的第一個字元為 a，在檔案 /tmp/test/
 | passwd 的第一個字元為 l。

```

```
字元 a 之八進位碼為 141，字元 l 之八進位碼為 154。
```

#### 4. 以 printf 驗證 ASCII 之字元

```
2 [root@dywHome2 ~]# printf '\x61\t\x6c\n'
a l
\xNN : NN 為十六進位
```

- patch：檔案補丁。需與 diff 配合使用。

##### 1. patch 指令用法

```
1 [root@linux ~]# patch [OPTION]... [ORIGFILE [PATCHFILE]]
選項：
3 -pNUM : 取消 NUM 層目錄。
 例如：假設檔名 /u/howard/src/blurfl/blurfl.c
5 -p0 : 代表 u/howard/src/blurfl/blurfl.c
 -p4 : 代表 blurfl/blurfl.c
7 -l : 忽略空白之差異。
-i PATCHFILE : 從 PATCHFILE 讀取補丁。
9 -o FILE : 輸出補丁到檔案 FILE。
-r FILE : 輸出錯誤到檔案 FILE。
```

##### 2. 預處理：建立兩個不同版本的檔案 /tmp/test/passwd 與 /etc/passwd。

```
2 [root@dywHome2 ~]# mkdir /tmp/old; cp /etc/passwd /tmp/old
[root@dywHome2 ~]# mkdir /tmp/new; cp /tmp/test/passwd /tmp/new
```

##### 3. 建立補丁檔案 /tmp/test.patch 記錄新舊檔案之間的差異。

```
2 [root@dywHome2 ~]# cd /tmp ; diff old/ new/ > test.patch
diff 製作檔案時，舊的檔案必須是在前面，亦即是 diff oldfile
newfile。
```

##### 4. 將舊的內容 (/tmp/old/passwd) 更新到新版 (/tmp/new/passwd) 的內容

```
2 [root@dywHome2 tmp]# cd /tmp/old
[root@dywHome2 old]# patch passwd -i /tmp/test.patch
patching file passwd
4 # 選項 -i 亦可省略
```

5. 更新內容，並指定存於 passwd1

```
2 [root@dywHome2 old]# patch passwd /tmp/test.patch -o passwd1
 patching file passwd
```

6. 內容已更新，若再做一次補丁，系統會詢問是否執行？

- (a) 預設回答 n(o)：系統會將錯誤訊息存在 passwd.rej

```
2 [root@dywHome2 old]# patch passwd -i /tmp/test.patch
 patching file passwd
 Reversed (or previously applied) patch detected! Assume -R? [n]
] n
 4 Apply anyway? [n] n
 Skipping patch.
 6 2 out of 2 hunks ignored -- saving rejects to file passwd.rej
```

- (b) 回答 y(es)：系統會將更新後的內容存在 passwd.orig

```
2 [root@dywHome2 old]# patch passwd -i /tmp/test.patch
 patching file passwd
 Reversed (or previously applied) patch detected! Assume -R? [n]
] y
```

- (c) 查詢檔案

```
1 [root@dywHome2 old]# ll passwd*
-rw-r--r-- 1 root root 1207 Apr 16 13:14 passwd
 3 -rw----- 1 root root 1156 Apr 16 13:10 passwd1
-rw-r--r-- 1 root root 1156 Apr 16 13:11 passwd.orig
 5 -rw-r--r-- 1 root root 149 Apr 16 13:12 passwd.rej
```

7. 使用選項 -pNUM 更新舊版資料

- (a) 變換目錄至 /tmp

```
1 [root@dywHome2 old]# cd ..
```

- (b) 以選項 -u 建立目錄 old/ 與 new/ 下檔案之差異檔，再回到目錄 /tmp/old。

```
1 [root@dywHome2 tmp]# diff -u old/ new/ > /tmp/test.patch
 [root@dywHome2 tmp]# cd old
```

- (c) test.patch 儲存 `diff -u old/passwd new/passwd`，故必須一層目錄，即 `-p1`。

```
[root@dywHome2 old]# patch -p1 </tmp/test.patch
2 patching file passwd
```

- (d) 若使用 `-p0`，則無法找到要更新的檔案，系統會要求輸入要更新的檔案。

```
[root@dywHome2 old]# patch -p0 </tmp/test.patch
2 can't find file to patch at input line 4
 Perhaps you used the wrong -p or --strip option?
 The text leading up to this was:

 6 |diff -u old/passwd new/passwd
 |--- old/passwd 2008-04-16 14:46:21.000000000 -0400
 8 |+++ new/passwd 2008-04-16 11:15:12.000000000 -0400
 |-----
10 File to patch:
```

- (e) 命令 `patch -pnum <patchfile>` 適用於目錄下多個檔案，較接近實際狀況。

- 範例：利用 `patch` 命令及第一版檔案、第一版和第二版的差異檔案，產生完整的第二版檔案。

1. 第一版的檔案 file1.c：

```
2 This is file one
 line 2
 line 3
 4 there is no line 4, this is line 5
 line 6
```

2. 產生第二版的檔案 file2.c：

```
1 This is file two
 line 2
 3 line 3
 line 4
 5 line 5
 line 6
 7 a new line 8
```

3. 利用 diff 命令產生差異：

```
1 $ diff file1.c file2.c > diffs
diffs 檔案如下：
3 1c1
 < This is file one
5 —
 > This is file two
7 4c4,5
 < there is no line 4, this is line 5
9 —
 > line 4
11 > line 5
 5a7
13 > a new line 8
```

4. 假設有 file1.c 和 diffs 檔案。可以利用 patch 命令更新 file1.c，使其與 file2.c 一樣。

```
1 $ patch file1.c diffs
 Hmm... Looks like a normal diff to me...
3 Patching file file1.c using Plan A...
 Hunk #1 succeeded at 1.
5 Hunk #2 succeeded at 4.
 Hunk #3 succeeded at 7.
7 done
 $
```

5. 若希望回覆 file1.c，只要再使用 patch，加上-R（反向 patch）選項，file1.c 就會回到原本的狀況。。

```
1 $ patch -R file1.c diffs
2 Hmm... Looks like a normal diff to me...
 Patching file file1.c using Plan A...
4 Hunk #1 succeeded at 1.
 Hunk #2 succeeded at 4.
6 Hunk #3 succeeded at 6.
 done
8 $
```

• 例題：完成下列工作。

1. 以 vi 在 printf.txt 最後加入一行 csie ---，並另存為 printf.new；



2. 以 `diff` 比較 `printf.txt` 及 `printf.new`，並建立其 patch file，`printf.patch`；
3. 以 `cmp` 比較 `printf.txt` 及 `printf.new`；
4. 利用 patch file，`printf.patch` 將 `printf.txt` 更新為 `printf.new`。

### 練習題

1. 使用開放原始碼之套件時，若發現並修正一個程式的臭蟲，要如何以一個簡單、正確的方式通知者？

Sol. 寄一個 patch 檔案給作者，而不要只是說明修正的地方。

2. 請以指令 `cat` 及 `sed` 配合管線處理，將 `/etc/passwd` 第四行刪除，第六行取代為『no six line』，新的檔案存為 `/tmp/passwd`。

Sol. `cat /etc/passwd | sed -e '4d' -e '6c no six line' > /tmp/passwd`

3. 如何以指令 `diff` 比對兩個檔案 `file1` 與 `file2` 之差異？

Sol. `diff file1 file2`

4. 以 `diff` 比對 `file1` 與 `file2`，出現訊息 `4d3`，代表意義為何？

Sol. `file1` 第四行被刪除

5. 以 `diff` 比對 `file1` 與 `file2`，出現訊息 `6c5`，代表意義為何？

Sol. `file1` 第六行被取取代成 `file2` 的第五行

6. 如何以指令 `diff` 比對兩個檔案 `file1` 與 `file2`，且只列出檔案是否有差異？

Sol. `diff -q file1 file2`

7. 如何以指令 `diff` 比對兩個檔案 `file1` 與 `file2`，且忽略一行當中，多個空白的差異？

Sol. `diff -b file1 file2`

8. 如何以指令 `diff` 比對兩個檔案 `file1` 與 `file2`，且忽略空白行的差異？

Sol. `diff -B file1 file2`

9. 如何以指令 `diff` 比對兩個檔案 `file1` 與 `file2`，且忽略大小寫的不同？

Sol. `diff -i file1 file2`

10. 如何以指令 `diff` 比對兩個檔案 `file1` 與 `file2`，且以 RCS 格式輸出檔案之差異？

Sol. `diff -u file1 file2`

11. 如何以 diff 比對兩目錄 /tmp/old 與 /tmp/new，並將新舊檔案之間的差異記錄在 test.patch？

Sol. `diff /tmp/old /tmp/new > test.patch`

12. 如何以 diff 比對兩目錄 /tmp/old 與 /tmp/new，並將新舊檔案之間的差異記錄在 test.patch，且加入差異部分前後 3 行，以增加輸出之可讀性？

Sol. `diff -u /tmp/old /tmp/new > test.patch`

13. 若有一檔案 test.patch 記錄兩目錄 /tmp/old 與 /tmp/new 之間的差異，如何將舊的內容 /tmp/old 更新到新版 /tmp/new 的內容？

Sol. `patch -p1 < /tmp/test.patch`

14. 如何以『位元』為單位比對兩檔案 file1 與 file2？

Sol. `cmp file1 file2`

15. 以指令 cmp 比對兩檔案 file1 與 file2，最後出現 differ: byte 94, line 4，代表意義為何？

Sol. 不同點在第四行，而且位元數是在第 94 個位元。

16. 如何以指令 cmp 比對兩檔案 file1 與 file2，且列出第一個的不同點之字元？

Sol. `cmp -v file1 file2`

17. 以指令 cmp 比對兩檔案 file1 與 file2，最後出現 differ: byte 94, line 4 is 141 a 154 l，其中 a 與 l 代表意義為何？

Sol. 不同點在檔案 file1 的第一個字元為 a，在檔案 file2 的第一個字元為 l。

18. 以指令 cmp 比對兩檔案 file1 與 file2，最後出現 differ: byte 94, line 4 is 141 a 154 l，其中 141 代表意義為何？

Sol. 字元 a 之八進位碼為 141。

19. 以指令 cmp 比對兩檔案 file1 與 file2，最後出現 differ: byte 94, line 4 is 141 a 154 l，其中 154 代表意義為何？

Sol. 字元 l 之八進位碼為 154。

20. 指定格式方式輸出 printf '\x61\t\x6c\n'，其中 \x61 代表意義為何？

Sol. 表示輸出十六進位 61（即八進位 141）之 ASCII 字元，當其輸出應為 a。

21. 指定格式方式輸出 printf '\x61\t\x6c\n'，其中 \x6c 代表意義為何？

Sol. 表示輸出十六進位 6c（即八進位 154）之 ASCII 字元，當其輸出應為 l。

22. 如何以指令 `patch`，配合 `passwd` 之補丁檔案 `/tmp/test.patch`，將 `passwd` 更新？

Sol. `patch passwd -i /tmp/test.patch`

23. 若以指令 `patch` 進行檔案更新，出現 `patching file passwd`，代表意義為何？

Sol. 表示更新檔案 `passwd` 完成。

24. 如何以指令 `patch`，配合 `passwd` 之補丁檔案 `/tmp/test.patch`，將 `passwd` 更新，且將更新內容另存於 `passwd1`？

Sol. `patch passwd /tmp/test.patch -o passwd1`

25. 若以指令 `patch` 進行檔案 `file1` 更新發生錯誤，系統會將錯誤訊息存在那個檔案？

Sol. `file1.rej`

26. 若補丁檔案在 `/tmp/test.patch`，如何以指令 `patch` 進行檔案 `file1` 更新，並指定錯誤輸出檔案為 `error.rej`？

Sol. `patch file1 /tmp/test.patch -r error.rej`

27. 若已以指令 `patch` 進行檔案 `passwd` 更新，現再次進行 `patch`，會出現什麼訊息？

Sol. 詢問是否進行。

28. 若已以指令 `patch` 進行檔案 `passwd` 更新，現再次進行 `patch` 會詢問是否進行，如果回應確定執行，會產生什麼結果？

Sol. 系統會將更新後的內容另存在 `passwd.orig`

29. 若已以指令 `patch` 進行檔案 `passwd` 更新，現再次進行 `patch` 會詢問是否進行，如果回應不繼續執行，會產生什麼結果？

Sol. 系統會將錯誤訊息存在 `passwd.rej`

30. 在目錄 `/tmp` 下，執行 `diff -u old/ new/ > /tmp/test.patch`，再進入目錄 `/tmp/old`，如何以 `patch` 進行目前目錄 `/tmp/old` 下所有檔案之更新？

Sol. `patch -p1 </tmp/test.patch`

31. 若補丁檔中記錄之檔案為 `/u/howard/src/blurfl/blurfl.cdiff`，則執行 `patch -p0 </tmp/test.patch`，其中 `-p0` 將以什麼字串取代？

Sol. `/u/howard/src/blurfl/blurfl.cdiff`

32. 若補丁檔中記錄之檔案為 `/u/howard/src/blurfl/blurfl.cdiff`，則執行 `patch -p1 </tmp/test.patch`，其中 `-p1` 將以什麼字串取代？

Sol. `u/howard/src/blurfl/blurfl.cdiff`

33. 若補丁檔中記錄之檔案為 `/u/howard/src/blurfl/blurfl.cdiff`，則執行 `patch -p2 </tmp/test.patch`，其中 `-p2` 將以什麼字串取代？

Sol. `howard/src/blurfl/blurfl.cdiff`

34. 若補丁檔中記錄之檔案為 `/u/howard/src/blurfl/blurfl.cdiff`，則執行 `patch -p3 </tmp/test.patch`，其中 `-p3` 將以什麼字串取代？

Sol. `u/howard/src/blurfl/blurfl.cdiff`

35. 若補丁檔中記錄之檔案為 `/u/howard/src/blurfl/blurfl.cdiff`，則執行 `patch -p4 </tmp/test.patch`，其中 `-p4` 將以什麼字串取代？

Sol. `blurfl/blurfl.cdiff`

36. 若補丁檔中記錄之檔案為 `/u/howard/src/blurfl/blurfl.cdiff`，則執行 `patch -p5 </tmp/test.patch`，其中 `-p5` 將以什麼字串取代？

Sol. `blurfl.cdiff`

37. 若產生補丁檔時，新舊檔案順序互換，例如 `diff new old > test.patch`，則要將舊版檔案 `old` 更新為 `new`，要如何執行 `patch`？

Sol. `patch -R old test.patch`

## 20.2 檢驗軟體正確性

- 指紋資料庫的建立

- 一般而言，每個系統裡的檔案內容大概都不相同，因此，可利用指紋分析程式 `md5sum` 可計算出檔案的指紋。

```
[root@linux ~]# md5sum filename
```

- 計算 `kplayer-0.5.3.systray.tar.gz` 的檔案指紋：

```
1 [root@dywHome2 SOURCES]# md5sum kplayer-0.5.3.systray.tar.gz
b907112f049c3c46664b4eab042f545e kplayer-0.5.3.systray.tar.gz
```

- 計算 `kplayer-0.5.3.systray.tar.gz` 的檔案指紋，並存成指紋檔：

```
[root@dywHome2 SOURCES]# md5sum kplayer-0.5.3.systray.tar.gz >
kplayer.md5
```

4. 查看指紋檔 kplayer.md5 內容：

```
1 [root@dywHome2 SOURCES]# cat kplayer.md5
b907112f049c3c46664b4eab042f545e kplayer-0.5.3.systray.tar.gz
```

5. 編輯 readme 檔：

```
2 [root@dywHome2 SOURCES]# vi readme
2 [root@dywHome2 SOURCES]# cat readme
kplayer-0.5.3.systray.tar.gz kplyaer source codes tarball
4 kplayer.md5 md5sum file
```

6. 計算 readme 的檔案指紋，並累加至指紋檔 kplayer.md5：

```
[root@dywHome2 SOURCES]# md5sum readme >> kplayer.md5
```

7. 再查看指紋檔 kplayer.md5 內容：

```
1 [root@dywHome2 SOURCES]# cat kplayer.md5
b907112f049c3c46664b4eab042f545e kplayer-0.5.3.systray.tar.gz
3 761adab1a7b0bc9fb2b60542137ee335 readme
```

• 指紋驗證機制 MD5：判斷檔案有沒有被更動過。

1. md5sum 驗證選項：

```
1 [root@linux ~]# md5sum [-bct] filename
選項：
3 -b : 使用 binary 的讀檔方式，預設為 Windows/DOS 檔案型態的讀取方
式；
-c : 檢驗 md5sum 檔案指紋；
5 -t : 以文字型態來讀取 md5sum 的檔案指紋。
```

2. 讀取 readme 之檔案指紋：

```
1 [root@dywHome2 SOURCES]# md5sum readme
761adab1a7b0bc9fb2b60542137ee335 readme
3 [root@dywHome2 SOURCES]# md5sum -b readme
761adab1a7b0bc9fb2b60542137ee335 *readme
5 [root@dywHome2 SOURCES]# md5sum -t readme
761adab1a7b0bc9fb2b60542137ee335 readme
```

## 3. 檢驗 kplayer.md5 檔案指紋：

```

2 [root@dywHome2 SOURCES]# md5sum -c kplayer.md5
kplayer-0.5.3.systray.tar.gz: OK
readme: OK

```

## 4. 修改 readme：

```

1 [root@dywHome2 SOURCES]# vi readme
[root@dywHome2 SOURCES]# cat readme
3 kplayer-0.5.3.systray.tar.gz kplyaer: source codes tarball
kplayer.md5: md5sum file

```

## 5. 再讀取 readme 之檔案指紋，已經不同於 kplayer.md5 所存內容：

```

2 [root@dywHome2 SOURCES]# md5sum -t readme
5a4d99c2a9b759dca679c2c1001cf8d4 readme

```

## 6. 再檢驗 kplayer.md5 檔案指紋：

```

2 [root@dywHome2 SOURCES]# md5sum -c kplayer.md5
kplayer-0.5.3.systray.tar.gz: OK
readme: FAILED
4 md5sum: WARNING: 1 of 2 computed checksums did NOT match

```

- 以 Mandriva 2007.0 更新目錄之 MD5SUM 為例：

## 1. 下載檔案指紋碼：

```

2 [root@dywHome2 rpm]# wget ftp://ftp.isu.edu.tw/Linux/Mandriva
/official/2007.0/i586/media/main/updates/media_info/MD5SUM
--20:57:43-- ftp://ftp.isu.edu.tw/Linux/Mandriva
4 /official/2007.0/i586/media/main/updates/media_info/MD5SUM
=> `MD5SUM'
6 Resolving ftp.isu.edu.tw... 140.127.177.15, 140.127.177.17
Connecting to ftp.isu.edu.tw|140.127.177.15|:21... connected.
8 Logging in as anonymous ... Logged in!
==> SYST ... done. ==> PWD ... done.
10 ==> TYPE I ... done. ==> CWD /Linux/Mandriva/official/2007.0/i586/
media/main/updates/media_info ... done.
==> PASV ... done. ==> RETR MD5SUM ... done.
12 Length: 98 (unauthoritative)
14 100%[=====>] 98
--.-K/s

```

```
16 20:57:49 (51.26 KB/s) - `MD5SUM' saved [98]
```

## 2. 查看 MD5SUM 內容：

```
2 [root@dywHome2 rpm]# cat MD5SUM
3 374fca8ea858a7079c3717acca208e47 hdlist.cz
4 38b6616b9514707ff8dc344eadd5468f synthesis.hdlist.cz
```

## 3. 下載檔案：

```
1 [root@dywHome2 rpm]# wget ftp://ftp.isu.edu.tw/Linux/Mandriva
2 /official/2007.0/i586/media/main/updates/media_info/hdlist.cz
3 --20:58:30-- ftp://ftp.isu.edu.tw/Linux/Mandriva
4 /official/2007.0/i586/media/main/updates/media_info/hdlist.cz
5 => `hdlist.cz'
6 Resolving ftp.isu.edu.tw... 140.127.177.15, 140.127.177.17
7 Connecting to ftp.isu.edu.tw|140.127.177.15|:21... connected.
8 Logging in as anonymous ... Logged in!
9 ==> SYST ... done. ==> PWD ... done.
10 ==> TYPE I ... done. ==> CWD /Linux/Mandriva/official/2007.0/i586/
11 media/main/updates/media_info ... done.
12 ==> PASV ... done. ==> RETR hdlist.cz ... done.
13 Length: 21,737,096 (21M) (unauthoritative)
14 100%[=====] 21,737,096
15 102.50K/s ETA 00:00
16 21:02:04 (99.67 KB/s) - `hdlist.cz' saved [21737096]
17
18 [root@dywHome2 rpm]# wget ftp://ftp.isu.edu.tw/Linux/Mandriva
19 /official/2007.0/i586/media/main/updates/media_info/synthesis.
20 hdlist.cz
21 --21:02:21-- ftp://ftp.isu.edu.tw/Linux/Mandriva
22 /official/2007.0/i586/media/main/updates/media_info/synthesis.
23 hdlist.cz
24 => `synthesis.hdlist.cz'
25 Resolving ftp.isu.edu.tw... 140.127.177.17, 140.127.177.15
26 Connecting to ftp.isu.edu.tw|140.127.177.17|:21... connected.
27 Logging in as anonymous ... Logged in!
28 ==> SYST ... done. ==> PWD ... done.
29 ==> TYPE I ... done. ==> CWD /Linux/Mandriva/official/2007.0/i586/
30 media/main/updates/media_info ... done.
31 ==> PASV ... done. ==> RETR synthesis.hdlist.cz ... done.
32 Length: 156,870 (153K) (unauthoritative)
33 100%[=====] 156,870
34 99.67K/s
35 21:02:24 (99.44 KB/s) - `synthesis.hdlist.cz' saved [156870]
```

4. 讀取 `hdlist.cz` 的 `md5sum` 檢查碼：

```
1 [root@dywHome2 rpm]# cat MD5SUM
374fca8ea858a7079c3717acca208e47 hdlist.cz
3 38b6616b9514707ff8dc344eadd5468f synthesis.hdlist.cz
[root@dywHome2 rpm]# md5sum -b hdlist.cz
5 374fca8ea858a7079c3717acca208e47 *hdlist.cz
[root@dywHome2 rpm]# md5sum -t hdlist.cz
7 374fca8ea858a7079c3717acca208e47 hdlist.cz
```

5. 檢驗 `md5sum` 檔案指紋：

```
1 [root@dywHome2 rpm]# md5sum -c MD5SUM
hdlist.cz: OK
3 synthesis.hdlist.cz: OK
```

### 練習題

1. 如何計算檔案 `file1` 的 `md5` 指紋？

Sol. `md5sum file1`

2. 如何計算檔案 `file1` 的 `md5` 指紋，並存成 `file1.md5`？

Sol. `md5sum file1 >file1.md5`

3. 如何計算檔案 `file2` 的 `md5` 指紋，並累加至 `file1.md5`？

Sol. `md5sum file2 >>file1.md5`

4. 如何讀取檔案 `file1` 的 `md5` 指紋？

Sol. `md5sum file1`

5. 如何使用二進位讀檔方式讀取檔案 `file1` 的 `md5` 指紋？

Sol. `md5sum -b file1`

6. 如何使用文字型態讀取檔案 `file1` 的 `md5` 指紋？

Sol. `md5sum -t file1`

7. 如何檢驗檔案指紋檔 `file.md5` 中檔案的 `md5` 指紋？

Sol. `md5sum -c file.md5`

8. 檢驗檔案指紋檔 `file.md5` 中檔案的 `md5` 指紋，出現 `checksums did NOT match` 訊息，代表意義為何？

Sol. `md5sum` 校驗，發現檔案 `md5` 指紋不符，表示檔案被更動過。



## 20.3 RPM 套件管理程式

- rpm 指令

1. 基本功能：套件安裝、升級、更新、移除、查詢及數位簽章驗證及檢查。

```

1 INSTALLING, UPGRADING, AND REMOVING PACKAGES:
 rpm {-i|--install} [install-options] PACKAGE_FILE ...
3 rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...
 rpm {-F|--freshen} [install-options] PACKAGE_FILE ...
5 rpm {-e|--erase} [--allmatches] [--nodeps] [--noscripts]
 [--notriggers] [--repackage] [--test] PACKAGE_NAME ...
7 QUERYING AND VERIFYING PACKAGES:
 rpm {-q|--query} [select-options] [query-options]
9 rpm {-V|--verify} [select-options] [verify-options]
 rpm --import PUBKEY ...
11 rpm {-K|--checksig} [--nosignature] [--nodigest]
 PACKAGE_FILE ...

```

2. 一般選項：適用於所有模式

```

-?, --help : 列出輔助訊息
2 --version : 列出 rpm 版本
--quiet : 僅列出錯誤訊息
4 -v : 察看更細部的安裝資訊
--vv : 列出大量除錯訊息
6 --pipe CMD : 將 rpm 的輸出以管線處理送到命令 CMD

```

- RPM 套件安裝

1. 安裝選項以 -i 開頭

```

rpm {-i|--install} [install-options] PACKAGE_FILE ...
2 install-options :
-h : 以安裝資訊列顯示安裝進度
4 --nodeps : 不要去檢查 rpm 套件的相依性。
--nomd5 : 不要檢查 rpm 套件的 MD5 資訊。
6 --noscripts : 不想讓該套件自行啓用或者自行執行某些系統指令。
--replacefiles : 直接覆蓋已存在的檔案。
8 --replacepkgs : 重新安裝某個已經安裝過的套件。
--force : --replacefiles 與 --replacepkgs 。
10 --test : 測試套件是否可以被安裝到使用者的 Linux 環境當中。
--prefix NEWPATH : 安裝到新目錄 NEWPATH

```

2. 安裝 rp-pppoe-3.1-5.i386.rpm

```

1 [root@linux ~]# rpm -ivh rp-pppoe-3.1-5.i386.rpm
Preparing... ##### [100%]
3 1:rp-pppoe ##### [100%]

```

3. 一次安裝兩個以上的套件：

```

1 [root@linux ~]# rpm -ivh a.i386.rpm b.i386.rpm *.rpm
後面直接接上許多的套件檔案。

```

4. 直接由網路 <http://ftp.isu.edu.tw> 安裝套件 logrotate：

```

2 [root@dywHome2 ~]# rpm -ivh http://ftp.isu.edu.tw/pub/Linux/
Mandriva\
>/official/2007.0/i586/media/main/release/logrotate-3.7.3-4mdk.i586
.rpm

```

5. 測試 FlashPlayer 是否可以被安裝在目前作業環境：

```

2 [root@dywHome2 ~]# rpm -ivh FlashPlayer-9.0.21.78-1.mdv2007.0.mde.
i586.rpm --test
Preparing... #
[100%]

```

#### • RPM 套件升級與更新

1. 升級選項以 `-U` 開頭，更新選項以 `-F` 開頭：

```

2 rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...
rpm {-F|--freshen} [install-options] PACKAGE_FILE ...
install-options :
4 與安裝相同

```

2. 升級更新與安裝之差異：

- (a) 升級後所有舊版套件皆被移除；
- (b) 若舊版未安裝，則無法更新，即套件不會被安裝。

3. 目前系統未安裝套件 baghira 情況下，以選項 `-U` 安裝：

```

2 [root@dywOffice rpm]# rpm -Uvh baghira-0.7-2.2006mcnl.i586.rpm
Preparing... #
[100%]

```

```
1:baghira #
[100%]
```

4. 目前系統未安裝套件 baghira 情況下，以選項 -F 安裝舊版：

```
1 [root@dywOffice rpm]# rpm -Fvh baghira-0.7-2.2006mcnl.i586.rpm
[root@dywOffice rpm]# rpm -Fvh baghira-0.7-0arn.20060mdk.i586.rpm
```

5. 目前系統已安裝套件 baghira 情況下，以選項 -F 升級：

```
2 [root@dywOffice rpm]# rpm -Fvh baghira-0.7-7arn.cvs20051211.20060
mdk.i586.rpm
Preparing... #
[100%]
1:baghira #
[100%]
```

#### • RPM 套件移除

1. 移除選項：

```
1 rpm {-e|--erase} [--allmatches] [--nodeps] [--test] PACKAGE_NAME
...
3 --nodeps：不要去檢查 rpm 套件的相依性。
--test：只是測試，不是真的移除套件。常與一般選項 -vv 一起使用來除錯
```

2. 測試是否可移除 mplayer：

```
2 [root@dywHome2 rpm]# rpm -e mplayer --test
error: Failed dependencies:
 mplayer is needed by (installed) kplayer-0.5.3-5mdv2007.0.
 i586
```

3. 測試不要檢查 rpm 套件的相依性下，是否可移除 mplayer：

```
1 [root@dywHome2 rpm]# rpm -e mplayer --test --nodeps
```

4. 移除套件 baghira：

```
1 [root@dywHome2 rpm]# rpm -e baghira
```

- RPM 查詢：選項以 -q 開頭。

#### 1. 查詢已安裝套件資訊的選項：

```
1 [root@linux ~]# rpm -qa
[root@linux ~]# rpm -q[licdR] 已安裝的套件名稱
3 [root@linux ~]# rpm -qf 存在於系統上面的某個檔案

5 -q : 僅查詢，後面接的套件名稱是否有安裝；
-q : 列出所有已經安裝在本機 Linux 系統上面的套件名稱；
7 -qi : 列出該套件的詳細資訊 (information)，包含開發商、版本與說明
 等；
-q : 列出該套件所有的檔案與目錄所在完整檔名 (list)；
9 -qc : 列出該套件的所有設定檔 (僅找出在 /etc/ 底下的檔名)
-qd : 列出該套件的所有說明檔 (僅找出與 man 有關的檔案)
11 -qR : 列出與該套件有關的相依套件所含的檔案 (Required)
-qf : 由後面接的檔案名稱，找出該檔案屬於哪一個已安裝的套件；
```

#### 2. 查詢未安裝套件資訊的選項：

```
1 [root@linux ~]# rpm -qp[licdR] 未安裝的某個檔案名稱
2 -qp[icdlR] : 後面接的所有參數以上面的說明一致。
```

#### 3. 找出是否有安裝套件 logrotate？

```
1 [root@linux ~]# rpm -q logrotate # 不必加上版本。
2 logrotate-3.7.1-10
[root@linux ~]# rpm -q logrotating
4 package logrotating is not installed
```

#### 4. 列出套件 logrotate 的所有目錄與檔案：

```
1 [root@linux ~]# rpm -ql logrotate
2 /etc/cron.daily/logrotate
 /etc/logrotate.conf
4以下省略.....
```

#### 5. 列出套件 logrotate 的相關說明資料：

```
[root@linux ~]# rpm -qi logrotate
2 Name : logrotate Relocations: (not relocatable)
Version : 3.7.1 Vendor: Red Hat, Inc.
4 Release : 10 Build Date: Fri Apr 1 03:54:42 2005
Install Date: Sat Jun 25 08:28:26 2005 Build Host: tweety.build.
redhat.com
6 Group : 系統環境/基礎 Source RPM: logrotate-3.7.1-10.src.
rpm
Size : 47825 License: GPL
8 Signature : DSA/SHA1, Sat May 21 01:34:11 2005, Key ID
b44269d04f2a6fd2
Packager : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
10 Summary : 循環、壓縮、移除以及郵寄系統紀錄檔案。
Description :
12 The logrotate utility is designed to simplify the administration of
log files on a system which generates a lot of log files. Logrotate
14 allows for the automatic rotation, compression, removal, and
mailing of
log files. Logrotate can be set to handle a log file daily, weekly,
16 monthly, or when the log file gets to a certain size. Normally,
logrotate runs as a daily cron job.
```

#### 6. 查詢 mplayer 的設定檔

```
1 [root@dywHome2 ~]# rpm -qc mplayer
/etc/mplayer/codecs.conf
3 /etc/mplayer/input.conf
/etc/mplayer/menu.conf
5 /etc/mplayer/mplayer.conf
```

#### 7. 查詢 mplayer 的說明檔

```
1 [root@dywHome2 ~]# rpm -qd mplayer
/usr/share/doc/mplayer-1.0/AUTHORS
3 /usr/share/doc/mplayer-1.0/ChangeLog
/usr/share/doc/mplayer-1.0/Copyright
5 /usr/share/doc/mplayer-1.0/HTML/cs/advaudio.html
/usr/share/doc/mplayer-1.0/HTML/cs/aspect.html
7 /usr/share/doc/mplayer-1.0/HTML/cs/audio-codecs.html
/usr/share/doc/mplayer-1.0/HTML/cs/audio-formats.html
9 ----以下省略----
```

#### 8. 查詢 logrotate 相依套件

```
1 [root@linux ~]# rpm -qR logrotate
/bin/sh
3 config(logrotate) = 3.7.1-10
```

```
5 libc.so.6
 ----以下省略----
```

9. 查詢 /bin/sh 是由那個套件提供？

```
1 [root@linux ~]# rpm -qf /bin/sh
 bash-3.0-31
3 # 參數後面接的是『檔案』，不是接套件。
```

10. 查詢套件 kplayer (未安裝) 之相依套件？

```
1 [root@dywHome2 rpm]# rpm -qpR kplayer-0.5.3-5mdv2007.0.i586.rpm
 mplayer
3 kdelibs-common
 /bin/sh
5 /bin/sh
 rpmlib(PayloadFilesHavePrefix) <= 4.0-1
7 rpmlib(CompressedFileNames) <= 3.0.4-1
 libDCOP.so.4
9 libICE.so.6
 ----以下省略----
```

11. 查詢套件 logrotate 設定檔

```
2 [root@dywHome2 ~]# rpm -qc logrotate
 /etc/logrotate.conf
 /etc/logrotate.d/syslog
```

12. 若查出 logrotate 的設定檔案已被改過，而要從網路直接重新安裝，該如何作？

```
1 [root@dywHome2 ~]# rpm -ivh http://ftp.isu.edu.tw/pub/Linux/
 Mandriva\
 >/official/2007.0/i586/media/main/release/logrotate-3.7.3-4mdk.i586
 .rpm\
3 > --replacepkgs
```

13. 如果誤刪 /etc/crontab，如何查詢它屬於那個套件，以重新安裝？

```
1 # 雖然檔案 /etc/crontab 已不存在，但 /var/lib/rpm 資料庫中有記錄：
 [root@dywHome2 ~]# rpm -qf /etc/crontab
3 crontabs-1.10-6mdk
```

14. 查詢系統中含 player 字串的套件？

```
1 [root@dywHome2 rpm]# rpm -qa | grep player
mplayer-1.0-1.pre8.13.2mdv2007.0
3 kplayer-0.5.3-5mdv2007.0
```

15. 如何知道系統中以 c 開頭的套件有幾個？

```
1 [root@dywHome2 rpm]# rpm -qa | grep ^c | wc -l
15
```

- RPM 驗證

1. 驗證使用時機：

- (a) 當資料不小心遺失；
- (b) 誤殺了某個套件的檔案；
- (c) 不知道修改到某一個套件的檔案內容。

2. 驗證方法：

- (a) rpm 套件安裝相關訊息存在 /var/lib/rpm 資料庫；
- (b) 比對目前 Linux 系統環境與 /var/lib/rpm 資料庫。

3. rpm 驗證：選項以 -V 開頭。

```
1 [root@linux ~]# rpm -Va
2 [root@linux ~]# rpm -V 已安裝的套件名稱
3 [root@linux ~]# rpm -Vp 某 rpm 套件名稱
4 [root@linux ~]# rpm -Vf 在系統上面的某個檔案
選項：
6 -V : 後接套件名稱，若該套件所含的檔案被更動過，才會列出來；
7 -Va : 列出目前系統上面所有可能被更動過的檔案；
8 -Vp : 後接套件名稱，列出該套件內可能被更動過的檔案；
9 -Vf : 列出某個檔案是否被更動過。
```

4. 列出 Linux 內的套件 logrotate 是否被更動過？

```
1 [root@dywHome2 rpm]# rpm -V logrotate
.M..... d /usr/share/man/man8/logrotate.8.bz2
```

5. 查詢 /etc/crontab 是否被更動過？

```

2 [root@dywHome2 rpm]# rpm -Vf /etc/crontab
.....T c /etc/crontab

```

6. 查詢 kplayer-0.5.3-5mdv2007.0.i586.rpm 內被更動過的檔案？

```

2 [root@dywHome2 rpm]# rpm -Vp kplayer-0.5.3-5mdv2007.0.i586.rpm
S.5....T /usr/bin/kplayer

```

7. 查詢結果第一項資訊說明：

```

2 S : file Size differs
 檔案的容量大小被改變
4 M : Mode differs (includes permissions and file type)
 檔案的類型或檔案的屬性(讀/寫/執行)已被改變
6 5 : MD5 sum differs
 MD5 加密防駭的屬性已被改變
8 D : Device major/minor number mis-match
 裝置名稱已被改變
10 L : readLink(2) path mis-match
 Link 屬性已被改變
12 U : User ownership differs
 檔案的所屬人已被改變
14 G : Group ownership differs
 檔案的所屬群組已被改變
16 T : mTime differs
 檔案的建立時間已被改變
18 #當檔案所有的資訊都被更動過會顯：
 SM5DLUGT c filename

```

8. 查詢結果第二項資訊說明：

```

1 c : 設定檔(config file)
 d : 文件資料檔(documentation)
3 g : 鬼檔案~通常是該檔案不被某個套件所包含，較少發生。(ghost file)
 l : 授權檔案(license file)
5 r : 讀我檔案(read me)

```

- RPM 數位簽章

1. RPM 也可利用數位簽證來判斷待安裝的套件檔案是否有問題。
2. 一般使用的是 GPG 的金鑰 ( public key )。



## 3. 應用方法：

- (a) 欲使用某 distribution 釋出的套件時，需將其釋出的 GPG 金鑰安裝在自己的 Linux 系統上。
- (b) 當安裝該 distribution 釋出的套件時，就會檢查兩者的 key 是否相同：如果相同就直接安裝；否則顯示未安裝 GPG 金鑰訊息。

## 4. 使用網路安裝金鑰套件 gnupg-1.4.5-1mdv2007.0：

```
1 [root@dywHome2 ~]# rpm -ivh http://ftp.isu.edu.tw/pub/Linux/
Mandriva\
>/official/2007.0/i586/media/main/release/gnupg-1.4.5-1mdv2007.0
```

## 5. 查詢金鑰檔案及其設定檔：

```
2 [root@dywHome2 ~]# find /etc /usr -name RPM-GPG-KEY*
/etc/RPM-GPG-KEYS
/usr/share/doc/rpm-4.4.6/RPM-GPG-KEY
```

## 6. 匯入金鑰：

```
1 [root@dywHome2 ~]# rpm --import /usr/share/doc/rpm-4.4.6/RPM-GPG-
KEY
```

## 7. 查詢所有金鑰：

```
1 [root@dywHome2 ~]# rpm -qa gpg-pubkey*
gpg-pubkey-78d019f5-3fd7504d
3 gpg-pubkey-db42a60e-37ea5438
gpg-pubkey-22458a98-3969e7de
5 gpg-pubkey-70771ff3-3c8f768f
```

## 8. 查詢金鑰內容：

```
1 [root@dywHome2 ~]# rpm -qi gpg-pubkey-78d019f5-3fd7504d
Name : gpg-pubkey Relocations: (not
relocatable)
3 Version : 78d019f5 Vendor: (none)
Release : 3fd7504d Build Date: Fri 28 Dec
2007 08:15:11 AM EST
5 Install Date: Fri 28 Dec 2007 08:15:11 AM EST Build Host:
localhost
Group : Public Keys Source RPM: (none)
7 Size : 0 License: pubkey
```

```

Signature : (none)
9 Summary : gpg(MandrakeContrib <cooker@linux-mandrake.com>)
Description :
11 -----BEGIN PGP PUBLIC KEY BLOCK-----
Version: rpm-4.4.6 (beecrypt-4.1.2)
13
mQGIBD/XUEORBAC/sGlzkZ7WofaPgL2A7Vmi4aMLF9xNkIUzeek2kxnZ+3
P277i2wckvIox
15 gGl130vqWUf6+20jBPWrOGz6hs+MUve7k+A7sgmg3n2P6fa999nxWTK+7m7705x+2
qXb+dxF
---- 中間省略 ----
17 41trJiEWiEkEGBECAAkFAj/XUE4CGwwACgkQRfk1+
HjQGfWi6wCeJPiCFBejOnfnBfa0Wh98
aNq3XfUAoKIIfvhoijsRwjSNmygJtZbiKOWC
19 =yJ39
-----END PGP PUBLIC KEY BLOCK-----

```

#### 9. 安裝 Fedora 7 rpm 套件 baghira :

```

2 [root@dywHome2 rpm]# rpm -ivh baghira-0.8-1.fc6.i386.rpm
warning: baghira-0.8-1.fc6.i386.rpm: Header V3 DSA signature: NOKEY
,
key ID ff6382fa error: Failed dependencies:
4 rtld(GNU_HASH) is needed by baghira-0.8-1.fc6.i386

```

#### 10. 移除金鑰 :

```

[root@dywHome2 ~]# rpm -e gpg-pubkey-78d019f5-3fd7504d

```

#### 11. 再查詢所有金鑰 :

```

1 [root@dywHome2 ~]# rpm -qa gpg-pubkey*
gpg-pubkey-db42a60e-37ea5438
3 gpg-pubkey-22458a98-3969e7de
gpg-pubkey-70771ff3-3c8f768f

```

#### • RPM 資料庫初始與重建 :

##### 1. 選項

```

2 rpm {--initdb|--rebuilddb} [-v]
--initdb : 產生一個新的資料庫
--rebuilddb : 重已安裝套件標頭 (headers)重建資料庫

```

## 2. 產生一個新的資料庫

```
1 [root@dywHome2 ~]# rpm --initdb -v
```

## 3. 重建資料庫

```
1 [root@dywHome2 ~]# rpm --rebuilddb -v
```

**練習題**

1. rpm 指令執行時，若要察看細部的安裝資訊，要使用什麼選項？

Sol. `-v`

2. rpm 指令執行時，若僅列出錯誤訊息，要使用什麼選項？

Sol. `--quiet`

3. rpm 指令執行時，若要列出大量除錯訊息，要使用什麼選項？

Sol. `--vv`

4. rpm 指令執行安裝時，要以什麼選項做為開頭？

Sol. `-i`

5. 如何以 rpm 指令安裝套件 a.i586.rpm，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm`

6. 如何以 rpm 指令同時安裝套件 a.i586.rpm 與 b.i586.rpm，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm b.i586.rpm`

7. 如何以 rpm 指令安裝網路套件 `http://ftp.isu.edu.tw/a.i586.rpm` 與 `b.i586.rpm`，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh http://ftp.isu.edu.tw/a.i586.rpm`

8. 如何以 rpm 指令測試套件 a.i586.rpm 是否可以被安裝到目前的環境，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm --test`

9. 如何以 rpm 指令，不檢查 rpm 套件的相依性下，安裝套件 a.i586.rpm，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm --nodeps`

10. 如何以 rpm 指令，不檢查 rpm 套件的 MD5 資訊下，安裝套件 a.i586.rpm，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm --nomd5`

11. 如何以 rpm 指令，直接覆蓋已存在的檔案方式，安裝套件 a.i586.rpm，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm --replacefiles`

12. 如何以 rpm 指令，重新安裝已安裝套件方式，安裝套件 a.i586.rpm，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm --replacepkgs`

13. 如何以 rpm 指令，重新安裝已安裝套件且直接覆蓋已存在檔案的方式，安裝套件 a.i586.rpm，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm --force`

14. 如何以 rpm 指令，安裝套件 a.i586.rpm 至目錄 /usr/local，並以安裝資訊列顯示安裝進度？

Sol. `rpm -ivh a.i586.rpm --prefix /usr/nomd5`

15. rpm 指令執行套件升級時，要以什麼選項做為開頭？

Sol. `-U`

16. rpm 指令執行套件更新時，要以什麼選項做為開頭？

Sol. `-F`

17. rpm 指令升級與安裝之差異為何？

Sol. 升級後所有舊版套件皆被移除。

18. rpm 指令更新與安裝之差異為何？

Sol. 若舊版未安裝，則無法更新，即套件不會被安裝。

19. 若系統未安裝套件 a.i586.rpm，則可以那兩種方式（再）安裝？

Sol. `rpm -ivh a.i586.rpm` 或 `rpm -Uvh a.i586.rpm`

20. 若系統已安裝套件 a.i586.rpm，則可以那三種方式（再）安裝？

Sol. `rpm -ivh a.i586.rpm` 或 `rpm -Uvh a.i586.rpm` 或 `rpm -Fvh a.i586.rpm`

21. 若系統未安裝套件 a.i586.rpm，則執行 `rpm -Fvh a.i586.rpm`，出現什麼狀況？

Sol. 無法安裝，沒有任何訊息顯示。

22. rpm 指令執行套件移除時，要以什麼選項做為開頭？  
Sol. `-e`
23. 如何以 rpm 指令移除套件 baghira？  
Sol. `rpm -e baghira`
24. 如何以 rpm 指令同時移除套件 mplayer 與 baghira？  
Sol. `rpm -e mplayer baghira`
25. 如何以 rpm 指令測試是否可以移除套件 mplayer？  
Sol. `rpm -e mplayer --test`
26. 如何以 rpm 指令測試不要檢查 rpm 套件的相依性下，是否可移除套件 mplayer？  
Sol. `rpm -e mplayer --test --nodeps`
27. rpm 指令執行套件查詢時，要以什麼選項做為開頭？  
Sol. `-q`
28. 如何以 rpm 指令查詢套件 mplayer 是否安裝？  
Sol. `rpm -q mplayer`
29. 如何以 rpm 指令查詢套件 mplayer 的所有目錄與檔案？  
Sol. `rpm -ql mplayer`
30. 如何以 rpm 指令查詢套件 mplayer 的相關說明資料？  
Sol. `rpm -qi mplayer`
31. 如何以 rpm 指令查詢套件 mplayer 的設定檔？  
Sol. `rpm -qc mplayer`
32. 如何以 rpm 指令查詢套件 mplayer 的說明檔？  
Sol. `rpm -qd mplayer`
33. 如何以 rpm 指令查詢套件 mplayer 的相依套件？  
Sol. `rpm -qR mplayer`
34. 如何以 rpm 指令查詢檔案 /bin/sh 是由那個套件提供？  
Sol. `rpm -qf /bin/sh`
35. 如何以 rpm 指令查詢未安裝套件 kplayer-0.5.3-5mdv2007.0.rpm 的相依套件？  
Sol. `rpm -pR kplayer-0.5.3-5mdv2007.0.rpm`

36. 如果誤刪 `/etc/crontab`，如何以 `rpm` 指令查詢它屬於那個套件，以重新安裝？

Sol. `rpm -qf /etc/crontab`

37. 如何以 `rpm` 指令查詢系統中含 `player` 字串的套件？

Sol. `rpm -qa | grep "player"`

38. 如何以 `rpm` 指令知道系統中以 `c` 開頭的套件有幾個？

Sol. `rpm -qa | grep "^c" | wc -l`

39. 請說明 RPM 驗證使用時機。

Sol. 1. 當資料不小心遺失；2. 誤殺了某個套件的檔案；3. 不知道修改到某一個套件的檔案內容。

40. RPM 套件安裝相關訊息資料庫在那個目錄？

Sol. `/var/lib/rpm`

41. RPM 驗證，主要是將目前 Linux 系統環境與什麼資料庫比對？

Sol. `/var/lib/rpm` 資料庫

42. `rpm` 指令執行套件驗證時，要以什麼選項做為開頭？

Sol. `-V`

43. 如何以 `rpm` 指令驗證套件 `logrotate` 是否被更動過？

Sol. `rpm -V logrotate`

44. 如何以 `rpm` 指令驗證檔案 `/etc/crontab` 是否被更動過？

Sol. `rpm -Vf /etc/crontab`

45. 如何以 `rpm` 指令查詢 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 內被更動過的檔案？

Sol. `rpm -Vp kplayer-0.5.3-5mdv2007.0.i586.rpm`

46. 若以 `rpm` 指令驗證某套件，驗證資訊第一項 `SM5DLUGT` 中 `S` 代表意義為何？

Sol. 檔案的檔案大小被改變

47. 若以 `rpm` 指令驗證某套件，驗證資訊第一項 `SM5DLUGT` 中 `M` 代表意義為何？

Sol. 檔案的類型或檔案的屬性（讀/寫/執行）已被改變

48. 若以 `rpm` 指令驗證某套件，驗證資訊第一項 `SM5DLUGT` 中 `5` 代表意義為何？

Sol. MD5 加密防竊的屬性已被改變

49. 若以 `rpm` 指令驗證某套件，驗證資訊第一項 `SM5DLUGT` 中 `D` 代表意義為何？

Sol. 裝置名稱已被改變

50. 若以 rpm 指令驗證某套件，驗證資訊第一項 SM5DLUGT 中 L 代表意義為何？

Sol. Link 屬性已被改變

51. 若以 rpm 指令驗證某套件，驗證資訊第一項 SM5DLUGT 中 U 代表意義為何？

Sol. 檔案的所有人已被改變

52. 若以 rpm 指令驗證某套件，驗證資訊第一項 SM5DLUGT 中 G 代表意義為何？

Sol. 檔案的所有群組已被改變

53. 若以 rpm 指令驗證某套件，驗證資訊第一項 SM5DLUGT 中 T 代表意義為何？

Sol. 檔案的建立時間已被改變

54. 若以 rpm 指令驗證某套件，驗證資訊第二項出現 c 代表檔案類型為何？

Sol. 設定檔 (config file)

55. 若以 rpm 指令驗證某套件，驗證資訊第二項出現 d 代表檔案類型為何？

Sol. 文件資料檔 (documentation)

56. 若以 rpm 指令驗證某套件，驗證資訊第二項出現 g 代表檔案類型為何？

Sol. 鬼檔案 (ghost file)

57. 若以 rpm 指令驗證某套件，驗證資訊第二項出現 l 代表檔案類型為何？

Sol. 授權檔案 (license file)

58. 若以 rpm 指令驗證某套件，驗證資訊第二項出現 r 代表檔案類型為何？

Sol. 讀我檔案 (read me)

59. RPM 數位簽章用途為何？

Sol. RPM 利用數位簽章來判斷待安裝的套件檔案是否有問題。

60. 若要使 RPM 數位簽章，必須安裝那個套件？

Sol. gnuPG

61. RPM 數位簽章金鑰檔案為何？

Sol. RPM-GPG-KEY

62. 若 RPM 數位簽章金鑰檔案為 RPM-GPG-KEY，如何將其匯入？

Sol. rpm --import RPM-GPG-KEY

63. 若已安裝 RPM 數位簽章在目前的 mandriva 系統，現安裝 Fedora 7 之 rpm 套件，會出現什麼問題？

Sol. 沒有簽章金鑰，故無法安裝。

64. 如何移除 RPM 數位簽章金鑰 gpg-pubkey?

Sol. `rpm -e gpg-pubkey`

65. 如何產生一個新的 RPM 資料庫?

Sol. `rpm --initdb`

66. 如何重建 RPM 資料庫?

Sol. `rpm --rebuilddb`