

Linux/Unix 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



March 19, 2016

- Instructor: De-Yu Wang

1. Email: dywang@csie.cyut.edu.tw
2. Homepage: <http://dywang.csie.cyut.edu.tw>
3. Phone: (04)23323000 ext 4538
4. Office: E738

- 參考資料

1. 鳥哥的 Linux 私房菜
2. 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。
3. vim 官方網站亦稱 vim 為程式開發工具。
4. 大家來學 VIM（一個歷久彌新的編輯器）
5. Bash shell
6. 中央研究院計算中心 RCS 簡介網頁
7. C++ GUI Programming with Qt 3 by Jasmin Blanchette and Mark Summerfield, Published by Prentice Hall.
8. Maximum RPM 網頁
9. Python
10. Python Tutorial
11. Glade - A User Interface Designer
12. PyGTK: GTK+ for Python
13. PyGTK tutorial
14. Python 線上程式練習

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	9
1.4	vi 實機練習題	12
1.4.1	練習一	12
1.4.2	練習二	12
1.4.3	練習三	13
2	Shell 變數	15
2.1	Shell 的變數功能	15
2.2	環境變數	19
2.3	變數鍵盤讀取與宣告	27
2.4	陣列	28
2.5	變數的變化與取代	29
2.6	變數的設定	30
2.7	實機練習題	36
2.7.1	練習一	36
3	資料導向與管線處理	37
3.1	資料流重導向	37
3.2	連續命令	40
3.3	管線命令 (pipe)	41
3.4	實機練習題	57
3.4.1	練習一	57
3.4.2	練習二	57
3.4.3	練習三	58
4	正規表示法	59
4.1	前言	59
4.2	基礎正規表示法	60
4.3	延伸正規表示法	69
4.4	格式化列印	71
4.5	實機練習題	74
4.5.1	練習一	74

4.5.2 練習二	74
5 sed 與 awk 工具	75
5.1 sed 工具	75
5.2 awk 工具	80
6 Shell Scripts – 簡介與 test 功能	85
6.1 前言	85
6.2 視 Shell 為一種程式語言	88
6.3 shell script 練習	90
6.4 善用判斷式	92
7 Shell scripts – 條件判斷與迴圈	99
7.1 條件判斷式	99
7.2 迴圈 (loop)	109
7.3 shell script 的追蹤與 debug	114
8 開發工具 – make 與 makefile	119
8.1 編譯器與可執行檔	119
8.2 make 命令和 makefile	122
8.3 makefile 的變數	131
8.4 多重目標項目 (target)	135
8.5 Makefile 其他法則	139
8.6 函式庫管理	141
8.7 撰寫使用者手冊	148
8.8 實機練習題	155
8.8.1 練習一	155
9 開發工具 – makefile 其他功能	157
9.1 多重目標項目 (target)	157
9.2 Makefile 其他法則	160
9.3 函式庫管理	163
9.4 撰寫使用者手冊	170
9.5 實機練習題	177
9.5.1 練習一	177
10 *RCS 版本控制系統	179
10.1 版本控制	179
10.2 RCS 基本功能	182
10.3 識別關鍵字串	199
10.4 標記符號	207
10.5 版本比較與整合	212
10.6 存取名單	217

11 *使用 QT 設計 KDE 視窗程式	221
11.1 KDE 和 QT 介紹	221
11.2 Qt 開發環境建立	222
11.3 gtk+ 開發環境建立	225
11.4 Signals 和 Slots	226
11.5 QT Widget	233
11.6 對話窗 dialog	246
11.7 選單和工具列	251
12 Python	257
12.1 Python 簡介	257
12.2 Python 變數	261
12.3 python 運算子	265
12.4 python 流程控制	265
12.5 python 函式與模組	265
12.6 python I/O 與 Exceptions	265
13 PyGTK	267
13.1 PyGTK 簡介與開始	267
13.2 PyGTK 除錯	268
13.3 Buttons 與 Layout	270
13.4 Signals, Events and Functions	271
13.5 Label and Entry	273
14 Glade	277
14.1 簡介	277
14.2 Calculator Example	279
15 *Tarball 套件發行	281
15.1 套件發行	281
15.2 Tarballs 簡介	282
15.3 KPlayer Tarball 實例	292
16 *RPM 與 SRPM 套件發行	299
16.1 RPM 與 SRPM 套件	299
16.2 建立 RPM 套件	304
16.3 KPlayer RPM 實例	312
17 *套件修補、檢驗與管理	317
17.1 patch 程式	317
17.2 檢驗軟體正確性	327
17.3 RPM 套件管理程式	332

Chapter 1

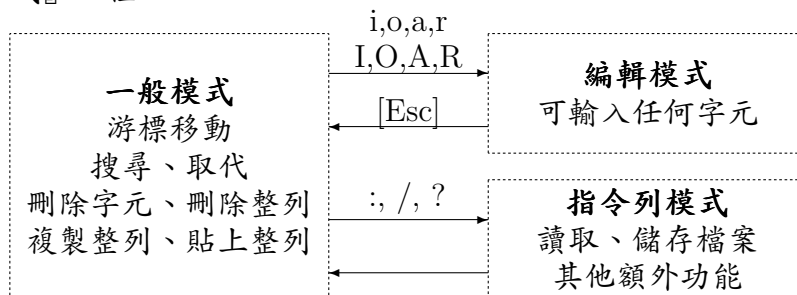
vi 編輯器

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 :x 儲存後離開 vi。

3. 常用指令列表

一般模式	移動游標
h 或向左方向鍵	游標向左移動一個字元
j 或向下方向鍵	游標向下移動一個字元
k 或向上方向鍵	游標向上移動一個字元
l 或向右方向鍵	游標向右移動一個字元
[Ctrl]+f	螢幕『向前』移動一頁
[Ctrl]+b	螢幕『向後』移動一頁
[Ctrl]+d	螢幕『向前』移動半頁
[Ctrl]+u	螢幕『向後』移動半頁
+	游標移動到非空白字元的下一列
-	游標移動到非空白字元的上一列
n<space>	按下數字後再按空白鍵，游標會向右移動這一行的 n 個字元。例如 20<space> 則游標會向後面移動 20 個字元距離。
0	這是數字『0』：移動到這一行的最前面字元處。
\$	移動到這一行的最後面字元處
H	游標移動到這個螢幕的最上方那一行
M	游標移動到這個螢幕的中央那一行
L	游標移動到這個螢幕的最下方那一行
G	游標移動到這個檔案的最後一行
nG	游標移動到這個檔案的第 n 行。例如 20G 則會移動到檔案的第 20 行(可配合 :set nu)
n<Enter>	游標向下移動 n 行
一般模式	搜尋與取代
/word	向游標之後尋找一個字串名稱爲 word 的字串。
?word	向游標之前尋找一個字串名稱爲 word 的字串。
n	重複前一個搜尋。
N	反向進行前一個搜尋動作。
:n1,n2s/word1/word2/g	在第 n1 與 n2 行之間尋找 word1 這個字串，並將該字串取代爲 word2。
:1,\$s/word1/word2/g	從第一行到最後一行尋找 word1 字串，並將該字串取代爲 word2。
:1,\$s/word1/word2/gc	從第一行到最後一行尋找 word1 字串，並將該字串取代爲 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 爲向後刪除一個字元，X 爲向前刪除一個字元
nx	向後刪除 n 個字元
dd	刪除游標所在的那一整列
ndd	刪除游標所在的向下 n 列，例如 20dd 則是刪除 20 列
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 列，例如 20yy 則是複製 20 列
y1G	複製游標所在列到第一列的所有資料
yG	複製游標所在列到最後一列的所有資料
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	將游標所在列與下一列的資料結合成同一列
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。
<hr/> 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。
<hr/> 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『filename』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。

練習題

1. vi 有那三種模式？
Sol. 一般模式、編輯模式與指令列模式
2. 在 vi 一般模式下，可以那四個字元鍵進行游標向左、下、上、右移動一個字元？
Sol. h j k l
3. 在 vi 一般模式下，如何使螢幕『向前』移動一頁？
Sol. [Ctrl]+f
4. 在 vi 一般模式下，如何使螢幕『向後』移動一頁？
Sol. [Ctrl]+b
5. 在 vi 一般模式下，如何使螢幕『向前』移動半頁？
Sol. [Ctrl]+d
6. 在 vi 一般模式下，如何使螢幕『向後』移動半頁？
Sol. [Ctrl]+u
7. 在 vi 一般模式下，如何使游標移動到非空白字元的下一列？
Sol. 按 + 鍵
8. 在 vi 一般模式下，如何使游標移動到非空白字元的上一列？
Sol. 按 - 鍵
9. 在 vi 一般模式下，如何使游標向右移動這一行的 12 個字元？
Sol. 按 12<space> 鍵
10. 在 vi 一般模式下，如何使游標移動到這一行的最前面字元處？
Sol. 按 0 鍵
11. 在 vi 一般模式下，如何使游標移動到這一行的最後面字元處？
Sol. 按 \$ 鍵
12. 在 vi 一般模式下，如何使游標移動到這個螢幕的最上方那一行？
Sol. 按 H 鍵
13. 在 vi 一般模式下，如何使游標移動到這個螢幕的中央那一行？
Sol. 按 M 鍵
14. 在 vi 一般模式下，如何使游標移動到這個螢幕的最下方那一行？
Sol. 按 L 鍵
15. 在 vi 一般模式下，如何使游標移動到這個螢幕的最後一行？
Sol. 按 G 鍵

16. 在 vi 一般模式下，如何使游標移動到這個檔案的第 12 行？
Sol. 輸入 12G
17. 在 vi 一般模式下，如何使游標游標向下移動 12 行？
Sol. 輸入 12<Enter>
18. 在 vi 一般模式下，如何向游標之後尋找一個字串名稱爲 word 的字串，接著重複搜尋？
Sol. 輸入 /word ，接著按 n 鍵
19. 在 vi 一般模式下，如何向游標之後尋找一個字串名稱爲 word 的字串，接著反向重複搜尋？
Sol. 輸入 /word ，接著按 N 鍵
20. 在 vi 一般模式下，如何向游標之前尋找一個字串名稱爲 word 的字串，接著重複搜尋？
Sol. 輸入 ?word ，接著按 n 鍵
21. 在 vi 一般模式下，如何向游標之前尋找一個字串名稱爲 word 的字串，接著反向重複搜尋？
Sol. 輸入 ?word ，接著按 N 鍵
22. 在 vi 一般模式下，如何向游標後刪除一個字元？
Sol. 按 x 鍵
23. 在 vi 一般模式下，如何向游標前刪除一個字元？
Sol. 按 X 鍵
24. 在 vi 一般模式下，如何向游標後刪除 12 個字元？
Sol. 按 12x 鍵
25. 在 vi 一般模式下，如何向游標前刪除 12 個字元？
Sol. 按 12X 鍵
26. 在 vi 一般模式下，如何刪除游標所在的那一整列？
Sol. 按 dd 鍵
27. 在 vi 一般模式下，如何刪除一個字？（不適用於中文）
Sol. 按 dw 鍵
28. 在 vi 一般模式下，如何刪除游標所在的向下 12 列？
Sol. 按 12dd 鍵
29. 在 vi 一般模式下，如何刪除游標所在到最後一行的所有資料？
Sol. 按 d1G 鍵
30. 在 vi 一般模式下，如何複製游標所在的那一行？
Sol. 按 yy 鍵

31. 在 vi 一般模式下，如何複製游標所在的向下 12 列？
Sol. `12yy`
32. 在 vi 一般模式下，如何複製游標所在列到第一列的所有資料？
Sol. `y1G`
33. 在 vi 一般模式下，如何複製游標所在列到最後一列的所有資料？
Sol. `yG`
34. 在 vi 一般模式下，如何將複製的資料在游標下一行貼上？
Sol. `p`
35. 在 vi 一般模式下，如何將複製的資料在游標上一行貼上？
Sol. `P`
36. 在 vi 一般模式下，如何將游標所在列與下一列的資料結合成同一列？
Sol. `J`
37. 我已經在 vi 內進行很多動作，但現在想要復原，如何做？
Sol. 一直重複按 `u` 即可
38. 在 vi 的一般模式中，反復原前一個動作？
Sol. `[Ctrl]+z`
39. 在 vi 的一般模式中，重做前一個動作？
Sol.
40. vi 如何由一般模式進入編輯模式(在目前游標處插入輸入之文字)？
Sol. `i`
41. vi 如何由一般模式進入編輯模式(所在行第一個非空白字元處插入輸入之文字)？
Sol. `I`
42. vi 如何由一般模式進入編輯模式(由目前游標所在的下一個字元處開始輸入)？
Sol. `a`
43. vi 如何由一般模式進入編輯模式(由目前游標所在的最後一個字元處開始輸入)？
Sol. `A`
44. vi 如何由一般模式進入編輯模式(從游標所在的下一行插入新的一行)？
Sol. `o`
45. vi 如何由一般模式進入編輯模式(從游標所在的上一行插入新的一行)？
Sol. `O`
46. vi 如何由一般模式進入編輯模式(會取代游標所在的那一個字元)？
Sol. `x`

47. vi 如何由一般模式進入編輯模式(會一直取代游標所在文字，直到退出編輯模式)？

Sol. `Esc`

48. vi 如何退出編輯模式，回到一般模式中？

Sol. `按 Esc`

49. 在 vi 的一般模式中，輸入 13 後按 Enter 與輸入 13 後按下 G，代表什麼意義？

Sol. `向下移動 13 行，以及去到第 13 行。`

50. 在 vi 的一般模式中，先以指令將游標移到第 13 行，複製 13 到 17 行，再以指令將游標移到第 20 行，最後將資料貼到 20 行以後，如何做？

Sol. `13G 5yy 20G p`

51. 在 vi 的一般模式中，先以指令將游標移到第 50 行，刪除 50 到 55 行，如何做？

Sol. `50G 5dd`

52. 在 vi 的一般模式中，如何以指令將游標移到第 5 行，再連結第 5, 6 行？

Sol. `5G J`

53. 在 vi 的一般模式，輸入 /find 代表什麼意義？另外，輸入 j 又代表什麼意義？輸入 5k 代表什麼

Sol. `找尋 find 這個字串； 向下移動； 向上移動 5 個字元。`

54. 在 vi 的一般模式中，我想要取代目前檔案第 100 到 200 行之間的 test 成為 TEST 且不詢問，要如何做？

Sol. `:100,200s/test/TEST/g`

55. 在 vi 的一般模式中，我想要取代目前檔案第 100 到 200 行之間的 test 成為 TEST 且取代前詢問，要如何做？

Sol. `:100,200s/test/TEST/gc`

56. 在 vi 的一般模式中，我想要取代目前檔案第 1 到最後一行之間的 test 成為 TEST 且不詢問，要如何做？

Sol. `:1,$s/test/TEST/g`

57. 在 vi 的一般模式中，我想要取代目前檔案第 1 到最後一行之間的 test 成為 TEST 且取代前詢問，要如何做？

Sol. `:1,$s/test/TEST/gc`

58. 在 vi 的一般模式中，如何儲存並結束檔案後離開？

Sol. `在一般模式下執行 :wq 或 :x`

59. 在 vi 的指令列模式中，『 :q! 』代表什麼意思？

Sol. `不儲存，強制離開 vi`

60. 在 vi 的一般模式當中，輸入 ZZ 代表什麼意思？
Sol. 若檔案已變動，則儲存後離開，若未做變動，則直接離開 vi。
61. 在 vi 的一般模式中，如何將編輯的資料寫入硬碟檔案中？
Sol. 輸入 :w
62. 在 vi 的一般模式中，如何將目前編輯的檔案另存為 file1.txt？
Sol. 輸入 :w file1.txt
63. 在 vi 的一般模式中，若目前編輯的檔案為唯讀檔，如何強制存回？
Sol. 輸入 :w!
64. 在 vi 的一般模式中，如何離開 vi？
Sol. 輸入 :q
65. 在 vi 的一般模式中，如何強制離開不儲存檔案？
Sol. 輸入 :q!
66. 在 vi 的一般模式中，如何儲存後離開？
Sol. 輸入 :wq 或 :x
67. 在 vi 的一般模式中，如何強制儲存後離開？
Sol. 輸入 :wq! 或 :x!
68. 在 vi 的一般模式中，如何將目前編輯的檔案還原到原始狀態？
Sol. 輸入 :e!
69. 在 vi 的一般模式中，如何設定顯示行號？
Sol. 輸入 :set nu
70. 在 vi 的一般模式中，如何取消設定顯示行號？
Sol. 輸入 :set nonu
71. 在 vi 的一般模式中，如何讀取另一檔案 file.txt？
Sol. 輸入 :r file.txt
72. 在 vi 的一般模式中，如何將 10 到 20 的內容儲存成 file1.txt 這個檔案？
Sol. 輸入 :10,20 w file1.txt
73. 在 vi 的一般模式中，如何將 10 到最後一行的內容儲存成 file1.txt 這個檔案？
Sol. 輸入 :10,\$ w file1.txt
74. 在 vi 的一般模式中，如何暫時離開，執行指令 ls -l？
Sol. 輸入 :! ls -l
75. 在 vi 編輯檔案 file.txt 中，同時會產生一暫存檔，其檔名為何？
Sol. .file.txt.swp

76. 如果要以 vi 編輯檔案 file.txt 時，出現是否回復的訊息，表示有那個暫存檔存在？

Sol. `.file.txt.swp`

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	
<code>:n</code>	編輯下一個檔案。
<code>:N</code>	編輯上一個檔案。
<code>:files</code>	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` 則兩視窗為同一檔案。

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

練習題

1. 在 vim 的一般模式中，如何將游標經過的地方反白選擇？
Sol. `<v>`
2. 在 vim 的一般模式中，如何將游標經過的反白選擇？
Sol. `<V>`
3. 在 vim 的一般模式中，如何進入區塊選擇(長方形)環境？
Sol. `[Ctrl]+v`
4. 在 vim 的一般模式中，區塊選擇後，要將反白的區塊複製，需按什麼鍵？
Sol. `y`
5. 在 vim 的一般模式中，區塊選擇後，要將反白的區塊刪除，需按什麼鍵？
Sol. `d`
6. vim 在多檔編輯時，如何列出開啓的所有檔案？
Sol. `:files`
7. vim 如何將檔案 filename 的內容在游標處插入？
Sol. `r filename`
8. 如何在啓動 vim 時，同時編輯 file1 與 file2 兩個檔案？
Sol. `vim file1 file2`
9. vim 在多檔案編輯時，如何編輯下一個檔案？
Sol. `<v>`
10. vim 在多檔案編輯時，如何編輯上一個檔案？
Sol. `<v>`
11. vim 如何開啓同一檔案於新視窗？
Sol. `:sp`
12. vim 如何開啓檔案 filename 於新視窗？
Sol. `:sp filename`
13. vim 在多視窗下，如何將檔游標移動到上方的視窗？
Sol. 先按 `[Ctrl]` 不放，再按 `w` 後，放開所有按鍵，再按下 `k`。
14. vim 在多視窗下，如何將檔游標移動到下方的視窗？
Sol. 先按 `[Ctrl]` 不放，再按 `w` 後，放開所有按鍵，再按下 `j`。
15. vim 個人動作記錄檔案為何(請寫出目錄/檔名)？
Sol. `~/.viminfo`
16. 整體 vim 的設定檔案為何(請寫出目錄/檔名)？
Sol. `/etc/vimrc`
17. 若要更改 vim 設定，可以在自己的家目錄建立那個檔案？
Sol. `~/.vimrc`

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，並重新命名為vi1.txt
4. 使用 vi 或 vim 編輯 vi1.txt，執行以下動作：
 - (a) 先到第 10 行。
 - (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，並重新命名為vi2.txt
4. 使用 vi 或 vim 編輯 vi2.txt，執行以下動作：
 - (a) 先到第 5 行。
 - (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`，並重新命名為 `vi3.txt`
4. 使用 `vi` 或 `vim` 編輯 `vi3.txt`，執行以下動作：
 - (a) 先到第 35 行。
 - (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) 再到第 10 行，在這行的最後加入文字 '`ABCDEabcde`'。
 - (e) 換行再加入 '`1234567890`'。
 - (f) 從第 30 行至第 55 行搜尋字串 "`man`"，並將其取代為 "`MAN`"。
 - (g) 存檔後退出。

Chapter 2

Shell 變數

2.1 Shell 的變數功能

- 何謂變數：

1. 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。
2. 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@dywOffice ~]# cat /etc/profile
3 LOGNAME=$USER
  MAIL="/var/spool/mail/$USER"
```

- 變數取用

```
2 [root@dywOffice ~]# echo $variable
  [root@dywOffice ~]# echo ${variable}
4 [root@dywOffice ~]# echo $PATH
  /sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/usr/
  local/sbin
6 [root@dywOffice ~]# echo ${PATH}
  /sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/usr/
  local/sbin
```

- 變數設定

```
1 [root@linux ~]# echo $myname
  <==這裡並沒有任何資料～因為這個變數尚未被設定。是空的。
3 [root@linux ~]# myname=csie
  [root@linux ~]# echo $myname
```

5 `csie` `<==`出現了。因為這個變數已經被設定了。

- 變數設定規則

1. 變數與變數內容以等號『=』來連結；
2. 等號兩邊不能直接接空白字元；
3. 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；

```

1 #範例一：設定一變數 name，且內容為 csie。
  [root@linux ~]# 12name=csie
3 -bash: 12name=csie: command not found <==不能以數字開頭。
  [root@linux ~]# name = csie <==不能有空白。
5 [root@linux ~]# name=csie <==OK。

```

4. 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，
 - (b) 單引號內的特殊字元則僅為一般字元；

```

1 #範例二：單引號與雙引號之差異
  [root@linux ~]# name=csie
3 [root@linux ~]# echo $name
  csie
5 [root@linux ~]# myname="$name its me"
  [root@linux ~]# echo $myname
7 csie its me
  [root@linux ~]# myname='$name its me'
9 [root@linux ~]# echo $myname
  $name its me

```

5. 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；

```

#範例三：承上題，若變數內容為 csie's name ?
2 [root@linux ~]# name=csie's name
  # 單引號將特殊字符 Enter 取消且不完全，另與要達到的功能也不同。
4 [root@linux ~]# name="csie's name" <==OK。
  [root@linux ~]# name=csie\'s\ name
6 # 利用反斜線 (\) 跳脫特殊字元，例如單引號與空白鍵，也。 OK

```

6. 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 ' 』；(' 是鍵盤上方的數字鍵 1 左邊那個按鍵，而不是單引號。)
- (b) 使用 \$(command)。

```

2  #範例四：先以指令 locate 找到 crontab 目錄，再顯示目錄內容
   [root@linux ~]# ls -l 'locate crontab'

4  #範例五：進入目前核心的模組目錄？
   [root@linux ~]# cd /lib/modules/$(uname -r)/kernel

6  # 利用 uname -r 指令先取得版本資訊。

```

7. 若該變數為擴增變數內容時，則需以雙引號及 \$變數名稱 如：『"\$PATH":/home』繼續累加內容；

```

2  #範例六：在 PATH 變數中『累加』目錄/home/csie/bin
   [root@linux ~]# PATH=$PATH:/home/csie/bin
   [root@linux ~]# PATH="$PATH":/home/csie/bin

4  #上面這兩種格式在 PATH 裡頭的設定都是 OK。但是底下的例子就不見得。

6  #範例七：承範例三，將 name 的內容多出 "yes" ?
   [root@linux ~]# name=$nameyes

8  # name 的內容是 $nameyes 這個變數。但沒有設定變數，應該如下： nameyes
   [root@linux ~]# name="$name"yes

10 [root@linux ~]# name=${name}yes

```

8. 通常大寫字元為系統預設變數，自行設定變數可以使用小寫字元，方便判斷；
9. 取消變數的方法為：『unset 變數名稱』。

```

2  #範例九：取消設定的 name 變數內容
   [root@linux ~]# unset name

```

● 變數的用途

1. 簡化路徑名稱

```

2  [dywang@dywOffice ~]$ linuxpro=\
   /home/dywang/Documents/latex/cjk-tex/linuxprogram/
   [dywang@dywOffice ~]$ cd linuxpro

4  bash: cd: linuxpro: No such file or directory
   [dywang@dywOffice ~]$ cd $linuxpro

```

```
6 [dywang@dywOffice linuxprogram]$ pwd
/home/dywang/Documents/latex/cjk-tex/linuxprogram
```

2. 方便程式修改

```
1 #define QGL_VERSION      450
2 #define QGL_VERSION_STR "4.5"
3 QM_EXPORT_OPENGL inline const char *qGLVersion() {
4     qObsolete( 0, "qGLVersion", "qVersion" );
5     return QGL_VERSION_STR;
```

練習題

1. 在 Bash shell 環境下，如何取出變數 VAR1 的內容？
Sol. `$VAR1` 或 `${VAR1}`
2. 在 Bash shell 環境下，如何將變數 VAR1 的內容顯示在螢幕上？
Sol. `echo $VAR1` 或 `echo ${VAR1}`
3. 在 Bash shell 環境下，以 `23name=csie` 設定變數是否正確，若不正確請說明其原因？
Sol. 變數不能以數字開頭
4. 在 Bash shell 環境下，以 `name = csie` 設定變數是否正確，若不正確請說明其原因？
Sol. 等號兩邊不能直接接空白字元
5. 在 Bash shell 環境下，以 `na$me=csie` 設定變數是否正確，若不正確請說明其原因？
Sol. 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
6. 在 Bash shell 環境下，若變數 `name=csie`，則 `myname="$name its me"`，`myname` 內容為何？
Sol. `csie its me`
7. 在 Bash shell 環境下，若變數 `name=csie`，則 `myname='$name its me'`，`myname` 內容為何？
Sol. `$name its me`
8. 在 Bash shell 環境下，若變數 `name=csie`，則 `myname=CYUT\ CSIE`，`myname` 內容為何？
Sol. `CYUT CSIE`
9. 在 Bash shell 環境下，要在 `PATH` 變數中『累加』目錄 `/home/csie/bin` 如何處理？
Sol. `PATH=$PATH:/home/csie/bin` 或 `PATH="$PATH":/home/csie/bin`

10. 在 Bash shell 環境下，要將 name 的內容多出 "yes" 如何處理？
Sol. `name="$name"yes & name=${name}yes`
11. 在 Bash shell 環境下，若變數 name="csie's name"，則 myname=name，myname 內容為何？
Sol. `name`
12. 在 Bash shell 環境下，若變數 name="csie's name"，則 myname=\$name，myname 內容為何？
Sol. `csie's name`
13. 在 Bash shell 環境下，若變數 name=csie\'s\ name，則 myname=\$name，myname 內容為何？
Sol. `csie's name`
14. 在 Bash shell 環境下，若要先以指令 locate 找到 crontab 目錄，再顯示目錄，如何以一行指令完成？
Sol. `ls -l $(locate crontab)` (不是鍵盤上方的數字鍵，而是那個按鈕，而不是單引號。)或 `ls -l ${locate crontab}`
15. 在 Bash shell 環境下，若要進入目前核心的模組目錄，如何以一行指令完成？
Sol. `cd /lib/modules/$(uname -r)/kernel` (不是鍵盤上方的數字鍵，而是那個按鈕，而不是單引號。)或 `cd /lib/modules/${uname -r}/kernel`
16. 在 Bash shell 環境下，若要先指令 foo 時，需要先執行命令 bar 以提供資訊，除了以 quote 'bar' 括起來外，還可如何執行？
Sol. `$(bar)`

2.2 環境變數

- 變數的有效範圍

1. shell 的父、子程序：被導出 (export) 後的變數，可以被子程序所引用，其他自訂變數內容則不會存在於子程序中。
2. 腳本(scripts)的編寫：軟體若使用到 2 個 scripts，scripts1.sh 及 scripts2.sh，scripts2.sh 要引用 scripts1.sh 的變數，則 scripts1.sh 中設定的變數必須以 export 設定。
3. 環境變數可以讓子程序繼續引用的原因：
 - (a) 啟動一個 shell 時，作業系統分配一記憶區塊給 shell 使用，此區域之變數可以讓子程序存取；
 - (b) 利用 export 功能，可以讓變數的內容寫到上述的記憶區塊當中(環境變數)；
 - (c) 當載入另一個 shell 時 (亦即啟動子程序，而離開原本的父程序了)，子 shell 可以將父 shell 的環境變數所在的記憶區塊導入自己的環境變數區塊當中。

- `export`、`env`、`set` 不加選項參數，列出之變數：
 1. `export` 列出所有設定為導出（`export`）的變數：
 2. `env` 列出所有的環境變數：
 3. `set` 列出所有的自訂及環境變數，也包括設定為空的變數。
- `export`、`env`、`set` 使用之不同：
 1. 使用 `export` 設定變數為導出，給子程序或後來呼叫的程式使用；
 2. 使用 `env` 設定環境變數給後接命令使用，只限於命令執行時使用；
 3. 使用 `set` 設定環境變數，等同於直接設定，如 `F00=settest`。
- 自訂變數轉成環境變數：`export`
 1. 僅下達 `export` 而沒有接變數時，會列出所有設定為導出（`export`）的變數。

```

1 [root@dywHome2 ~]# export
  declare -x DISPLAY=":0"
3 declare -x ENV="/root/.bashrc"
  declare -x GCONF_TMPDIR="/tmp"
5 declare -x G_FILENAME_ENCODING="@locale"
  declare -x HISTCONTROL="ignoredups"
7 declare -x HISTSIZE="1000"
  declare -x HOME="/root"
9 declare -x HOSTNAME="dywHome2"
  declare -x INPUTRC="/etc/inputrc"
11 declare -x LANG="en_US.UTF-8"中間省略

13 declare -x USER="root"
  declare -x USERNAME="root"
15 declare -x XAUTHORITY="/home/dywang/.Xauthority"

```

2. 進入子程序後，在父程序中的自訂變數設定將不再繼續的存在。會存在子程序中的，僅有『環境變數』。
3. 如果能將自訂變數變成導出，則可讓該變數值繼續存在於子程序。
4. 想讓變數內容繼續在子程序中使用：`export` 變數。
5. 若該變數需要在其他子程序執行，則需要以 `export` 來導出變數，例如『`export PATH`』；

```

1 #範例：如何讓 name=csie 用在下個 shell 的程序？
  [root@dywOffice ~]# echo $SHLVL <==列出目前 shell 的 level
3 1
  [root@dywOffice ~]# name=csie
5 [root@dywOffice ~]# bash <==進入到所謂的子程序

```

```

7 [root@dywOffice ~]# echo $SHLVL <==列出目前 shell 的 level
2
[root@dywOffice ~]# echo $name <==並沒有剛剛設定的內容。
9 [root@dywOffice ~]# exit <==離開剛剛的子程序
[root@dywOffice ~]# echo $SHLVL <==列出目前 shell 的 level
11 1
[root@dywOffice ~]# export name
13 [root@dywOffice ~]# bash <==進入到子程序
[root@dywOffice ~]# echo $SHLVL <==列出目前 shell 的 level
15 2
[root@dywOffice ~]# echo $name <==出現設定值了。
17 [root@dywOffice ~]# exit <==離開剛剛的子程序
# 一般狀態下，父程序的自訂變數無法在子程序內使用的。
19 # 透過 export 將變數變成環境變數後，才能夠在子程序下應用

```

- env 列出及設定環境變數

1. 列出所有被導出的環境變數：env

```

1 [root@linux ~]# env
HOSTNAME=linux.csie.tw <== 這部主機的主機名稱
3 SHELL=/bin/bash <== 目前這個環境下，使用的 Shell 是
  哪一個程式？
TERM=xterm <== 這個終端機使用的環境類型
5 HISTSIZE=1000 <== 『記錄指令的筆數』 預設可記錄
  1000 筆
USER=root <== 使用者的名稱。
7 LS_COLORS=no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd
  =40;33;
01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=00;32:*.cmd
  =00;32:*.
9 exe=00;32:*.com=00;32:*.btm=00;32:*.bat=00;32:*.sh=00;32:*.
  csh=00;
32:*.tar=00;31:*.tgz=00;31:*.arj=00;31:*.taz=00;31:*.lzh
  =00;31:*.zip=00;
11 31:*.z=00;31:*.Z=00;31:*.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz
  =00;
31:*.rpm=00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp
  =00;35:*.xbm=00;
13 35:*.xpm=00;35:*.png=00;35:*.tif=00;35: <== 一些顏色顯示
ENV=/root/.bashrc <== 使用者 root 的個人環境設定檔
15 MAIL=/var/spool/mail/root <== 這個使用者所取用的 mailbox 位置
/usr/local/bin:/usr/local/sbin:/root/bin <== 執行檔指令搜尋路
  徑
INPUTRC=/etc/inputrc <== 與鍵盤按鍵功能有關。可以設定特
  殊按鍵。
19 PWD=/root <== 目前使用者所在的工作目錄（利用
  pwd 取出。）

```

21	LANG=en_US.UTF-8	<== 使用語系
	HOME=/root	<== 使用者的家目錄
	_=/bin/env	<== 上一次使用的指令的最後一個參數(或指令本身)

2. 使用 env 設定只執行一次之環境變數

```

[root@dywHome2 ~]# name=CSIE
2 [root@dywHome2 ~]# echo $name
CSIE
4 [root@dywHome2 ~]# env name=CYUT 前面省略
6 name=CYUT
[root@dywHome2 ~]# echo $name
8 CSIE

```

● 列出目前 shell 的所有變數：set

```

[root@linux ~]# set
2 BASH=/bin/bash <== bash 的主程式放置路徑
  BASH_VERSION=([0]="3" [1]="00" [2]="16" [3]="1" [4]="release"
4 [5]="i386-redhat-linux-gnu") <== bash 的版本
  BASH_VERSION='3.00.16(1)-release' <== bash 的版本
6 COLORS=/etc/DIR_COLORS.xterm <== 使用的顏色紀錄檔案
  COLUMNS=115 <== 終端機環境的欄位之字元長度
8 HISTFILE=/root/.bash_history <== 歷史命令記錄的放置檔案，隱藏檔
  HISTFILESIZE=1000 <== 存起來(與上個變數有關)的檔案之指令的最大紀錄筆數。
10 HISTSIZE=1000 <== 目前環境下，可記錄的歷史命令最大筆數。
  HOSTTYPE=i386 <== 主機安裝的軟體主要類型。
12 IFS=$' \t\n' <== 預設的分隔符號
  LINES=35 <== 終端機的最大行數
14 MACHTYPE=i386-redhat-linux-gnu <== 安裝的機器類型
  MAILCHECK=60 <== 每 60 秒掃描一次信箱有無新信
16 OLDPWD=/home <== 上個工作目錄。我們可以用 cd - 來取用這個變數。
  OSTYPE=linux-gnu <== 作業系統的類型。
18 PPID=20046 <== 父程序的 PID
  PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME%.*}:${PWD/#$HOME/~}\007"'
20 <== 命令提示字元。與底下也有關。
  PS1='[\u@\h \W]\$ ' <== 第一層命令提示字元的設定值
22 RANDOM=13586 <== 亂數
  SUPPORTED=zh_TW.UTF-8:zh_TW:zh:en_US.UTF-8 <== 本系統所支援的語系
24 name=csie <== 剛剛設定的自訂變數
  $ <== 目前這個 shell 的 PID

```

```
26 || ?      <== 剛剛執行完指令的回傳值。
    || 0      <== shell 執行命令的名稱
```

1. 顯示目前 shell 執行命令的名稱

```
1 [dywang@dywOffice linuxprogram]$ echo $0
/bin/bash
```

2. \$: 『目前這個 Shell 的執行緒代號』，亦即是所謂的 PID (Process ID)。

```
2 [dywang@dywOffice ~ 10:51 #17]$ echo $$
5021
```

3. ?: 『上個執行的指令所回傳的值』。一般來說，如果成功的執行該指令，則會回傳一個 0 值，如果執行過程發生錯誤，就會回傳『錯誤代碼』。

```
2 [root@linux ~]# echo $SHELL
/bin/bash
4 [root@linux ~]# echo $?
0
# 因為上個指令執行過程中，並沒有錯誤，為成功的執行完畢，所以回傳 0。
6 [root@linux ~]# 12name=csie
-bash: 12name=csie: command not found
8 [root@linux ~]# echo $?
127
10 # 發生錯誤。所以 echo $? 時，會出現錯誤的代碼，可以利用代碼來搜尋錯誤的原因。
12 [root@linux ~]# echo $?
0
# 又正確了？因為 "?" 只與『上一個執行指令』有關。
```

4. 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。

```
1 [root@linux ~]# declare -i number=$RANDOM*10/32767 ; echo $number
8 <== 此時會隨機取出 0 9 之間的數值。
```

5. 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 linux.csie.tw。
- (c) \h：僅取主機名稱的第一個名字。上例為 linux，.csie.tw 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 \$。

```
[dywang@dywOffice ~]$ PS1='[\u@\h \w \A #\#]\$ ',  
2 [dywang@dywOffice ~ 10:51 #16]$ echo $PS1  
[\u@\h \w \A #\#]\$  
4 [dywang@dywOffice ~ 10:51 #17]$
```

練習題

1. 在 Bash shell 環境下，如何讓變數 name=csie 用在下一層 shell 的程序？
Sol. `export name`
2. 在 Bash shell 環境下，如何取消變數 name=csie 的設定？
Sol. `unset name`
3. 在 Bash shell 環境下，設定變數 linux=/home/csie/Documents/linux/，則執行 `cd linux`，結果為何？
Sol. `No such file or directory`
4. 在 Bash shell 環境下，設定變數 linux=/home/csie/Documents/linux/，則執行 `cd $linux`，結果為何？
Sol. `改變目錄至 /home/csie/Documents/linux/`
5. 在目前 shell 環境下自訂變數 name=csie，進入子 shell 後執行 `echo $name` 會出現什麼結果？
Sol. `空的内容`，因為自訂變數內容不會存在於子程序中。

6. 在目前 shell 環境下有一環境變數 name=csie，則進入子 shell 後執行 echo \$name 會出現什麼結果？
Sol. csie
7. 某一體使用到 2 個腳本 (scripts)：scripts1.sh 及 scripts2.sh，scripts2.sh 要引用 scripts1.sh 的變數，則 scripts1.sh 中設定的變數必須如何設定？
Sol. 以 export 設定。
8. 在 Bash shell 環境下，export、env、set 不加選項參數，列出之變數有何不同？
Sol. export 及 env 列出所有導出的環境變數；set 列出所有的自訂及環境變數，也包括設定為空的變數。
9. 在 Bash shell 環境下，如何查看目前是在第幾層 shell？
Sol. echo \$SHLVL
10. 在 Bash shell 環境下，如何列出目前環境下的所有環境變數與其內容？
Sol. env
11. Bash shell 的環境變數 HOSTNAME=linux.csie.tw，代表意義為何？
Sol. 主機名稱為 linux.csie.tw
12. Bash shell 的環境變數 HISTSIZE=1000，代表意義為何？
Sol. 記錄指令的次數，總數可記錄 1000 次
13. Bash shell 的環境變數 ENV=/root/.bashrc，代表意義為何？
Sol. 使用者 root 的個人環境設定檔
14. Bash shell 的環境變數 PWD=/root，代表意義為何？
Sol. 目前使用者所在的文件目錄為 /root
15. Bash shell 的環境變數 HOME=/root，代表意義為何？
Sol. 使用者的家目錄為 /root
16. 在 Bash shell 中有一變數 name=CSIE，則執行 echo \$name 與 env name=CYUT，產生的變數 name 內容有何不同？
Sol. echo \$name 的結果為 CSIE，env name=CYUT env 的結果為 CYUT。
17. 列出 Bash shell 的所有變數，要使用那個指令？
Sol. set
18. Bash shell 的變數 0，代表意義為何？
Sol. 目前 shell 的執行命令名稱
19. Bash shell 的變數 \$，代表意義為何？
Sol. 目前 shell 的 PID
20. Bash shell 的變數 ?，代表意義為何？
Sol. 上個執行的指令所回傳的值

21. 一般而言，上個指令執行成功，則 `echo $?` 結果為何？

Sol. 0

22. Bash shell 的變數 `RANDOM`，代表意義為何？

Sol. 產生介於 0~32767 之間的一個整數

23. 在 Bash shell 下，隨機產生 0~9 之間的數值，如何做？

Sol. `declare -i number=$((RANDOM*10/32767)); echo $number`

24. Bash shell 設定 `PS1='\d`，則的提示字元如何顯示？

Sol. 代表日期，格式為 `Weekday Month Date`，例如 `"Mon Aug 1"`

25. Bash shell 設定 `PS1='\H`，則的提示字元如何顯示？

Sol. 完整的主機名稱，例如 `linux.csie.tw`。

26. Bash shell 設定 `PS1='\h`，則的提示字元如何顯示？

Sol. 僅取主機名稱的第一個名字，例如 `linux.csie.tw`，則只取 `linux`。

27. Bash shell 設定 `PS1='\t`，則的提示字元如何顯示？

Sol. 顯示時間，為 24 小時格式，如 `HH:MM:SS`。

28. Bash shell 設定 `PS1='\T`，則的提示字元如何顯示？

Sol. 顯示時間，12 小時的時間格式

29. Bash shell 設定 `PS1='\A`，則的提示字元如何顯示？

Sol. 顯示時間，24 小時格式，`HH:MM`。

30. Bash shell 設定 `PS1='\u`，則的提示字元如何顯示？

Sol. 目前使用者的帳號名稱。

31. Bash shell 設定 `PS1='\v`，則的提示字元如何顯示？

Sol. BASH 的版本資訊。

32. Bash shell 設定 `PS1='\w`，則的提示字元如何顯示？

Sol. 完整的工作目錄名稱，家目錄會以 `~` 取代。

33. Bash shell 設定 `PS1='\W`，則的提示字元如何顯示？

Sol. 利用 `basename` 取得工作目錄名稱，所以僅會列出最後一個目錄名。

34. Bash shell 設定 `PS1='\#`，則的提示字元如何顯示？

Sol. 下述的系統指令。

35. Bash shell 設定 `PS1='\$`，則的提示字元如何顯示？

Sol. 提示字元，如果是 `root` 時，提示字元為 `#`，否則就是 `$`。

36. Bash shell 的提示字元為 `[dywang@dywOffice ~ 10:51 #16]$`，則變數 `PS1` 應如何設定？

Sol. `PS1='\u@\h \w \A #\#]\$ '`

2.3 變數鍵盤讀取與宣告

- 讀取來自鍵盤輸入的變數：read

1. read 指令

```
[root@linux ~]# read [-pt] variable 選項：
2
-p : 後面可以接提示字元。
4 -t : 後面可以接等待的『秒數』，不會一直等待使用者。
```

2. 讓使用者由鍵盤輸入一內容，將該內容變成 atest 變數

```
[root@linux ~]# read atest
2 This is a test
[root@linux ~]# echo $atest
4 This is a test
```

3. 提示使用者 30 秒內輸入自己的大名，將該輸入字串做成 name 變數

```
[root@linux ~]# read -p "Please keyin your name: " -t 30 name
2 Please keyin your name: csie
[root@linux ~]# echo $name
4 csie
```

- 宣告變數的屬性：

1. declare / typeset 指令

```
[root@linux ~]# declare [-aixr] variable 選項：
2
-a : 將後面的 variable 定義成為陣列 (array)，大小無限
    制，index 從 0 開始
4 -i : 將後面接的 variable 定義成為整數數字 (integer)
-x : 用法與 export 一樣，就是將後面的 variable 變成環境變數；
6 -r : 將一個 variable 的變數設定成為 readonly，該變數不可被更
    改內容，也不能 unset
```

2. 讓變數 sum 進行 100+300+50 的加總結果

```
[root@linux ~]# sum=100+300+50
2 [root@linux ~]# echo $sum
100+300+50 <==沒有計算加總？因為這是文字型態的變數屬性。
```

```
4 [root@linux ~]# declare -i sum=100+300+50
  [root@linux ~]# echo $sum
6 450
```

3. 將 sum 變成環境變數

```
[root@linux ~]# declare -x sum
```

4. 讓 sum 變成唯讀屬性，不可更動。

```
1 [root@linux ~]# declare -r sum
  [root@linux ~]# sum=tesgting
3 -bash: sum: readonly variable <==不能改這個變數了。
```

2.4 陣列

1. 宣告

```
1 [root@dywOffice ~]# declare -a sum
  [root@dywOffice ~]# sum=(11 22 33)
3 [root@dywOffice ~]# echo ${sum[0]} ${sum[1]} ${sum[2]}
  11 22 33
5 [root@dywOffice ~]# sum[2]=44
  [root@dywOffice ~]# echo ${sum[0]} ${sum[1]} ${sum[2]}
7 11 22 44
9 [root@dywOffice ~]# sum=(11 22 33)
```

2. 所有元素與元素個數

```
1 [root@dywOffice ~]# echo ${#sum[@]}
  3
3 [root@dywOffice ~]# echo ${sum[@]}
  11 22 33
5 [root@dywOffice ~]# sum=(${sum[@]} 44)
  [root@dywOffice ~]# echo ${#sum[@]}
7 4
  [root@dywOffice ~]# echo ${sum[@]}
9 11 22 33 44
  [dywang@deyu ~]$ sum=(11 22 33); echo ${sum[*]}
11 11 22 33
```

```

13 [dywang@deyu ~]$ sum=(11 22 33); echo ${sum[@]}
    11 22 33

```

3. 陣列元素的變化

```

1 [dywang@deyu ~]$ arr=(1 22 333 4444); arr=(${arr[@]} 55555); echo
    ${arr[@]}
    1 22 333 4444 55555
3 #從第個元素取到最後一個元素#2
[dywang@deyu ~]$ arr=(${arr[@]:2}); echo ${arr[@]}
5 333 4444 55555
#從最前面加入元素#
7 [dywang@deyu ~]$ arr=(0 ${arr[@]}); echo ${arr[@]}
    0 333 4444 55555
9 #從第個元素取到倒數第個元素#01
[dywang@deyu ~]$ arr=(${arr[@]:0:${#arr[@]}-1}); echo ${arr[
    @]}
11 0 333 4444

```

2.5 變數的變化與取代

1. 兩種變數取用的方法：

```

1 [root@deyu ~]# echo $HOME
[root@deyu ~]# echo ${HOME}

```

2. 完整變數的內容：

```

2 [dywang@dywH ~]$ blkid /dev/sda1
/dev/sda1: UUID="716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4"
"
[dywang@dywH ~]$ uuid=$(blkid /dev/sda1);echo $uuid
4 /dev/sda1: UUID="716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4"
"

```

3. 從最前面開始比對，刪除符合的字元。

```

2 [dywang@dywH ~]$ uuid=${uuid#*="}; echo $uuid
    716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4"
[dywang@dywH ~]$ uuid=$(blkid /dev/sda1);echo $uuid

```

```

4 | /dev/sda1: UUID="716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4
   | "
   | [dywang@dywH ~]$ uuid=${uuid##*=\}; echo $uuid
6 | ext4"
   | # 變數名稱後面如果接了兩個 ## ，表示在 ## 後面的字串取『最長的』那一
   | 段；
8 | # 如果僅有一個 # ，表示取『最小的那一段』。

```

4. 從後面開始比對並刪除：

```

2 | [dywang@dywH ~]$ uuid=$(blkid /dev/sda1);echo $uuid
   | /dev/sda1: UUID="716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4
   | "
   | [dywang@dywH ~]$ uuid=${uuid%*\}; echo $uuid
4 | /dev/sda1: UUID=
   | # %% 刪除最長的匹配，% 則是最短的匹配。

```

5. 取代變數部分字串

```

1 | [dywang@dywH ~]$ uuid=$(blkid /dev/sda1);echo $uuid
   | /dev/sda1: UUID="716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4
   | "
3 | [dywang@dywH ~]$ uuid=${uuid%\}; echo $uuid
   | /dev/sda1: UUID="716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4
5 | [dywang@dywH ~]$ uuid=$(blkid /dev/sda1);echo $uuid
   | /dev/sda1: UUID="716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4
   | "
7 | [dywang@dywH ~]$ uuid=$(blkid /dev/sda1);echo ${uuid%/*}
   | /dev/sda1: UUID:"716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE="ext4
   | "
9 | [dywang@dywH ~]$ uuid=$(blkid /dev/sda1);echo ${uuid%/*/*}
   | /dev/sda1: UUID:"716a2ac8-517e-41e4-a139-c2bc12b3dd5f" TYPE:"ext4
   | "
11 | # 變數後面接 / 時，表示進行『取代』的工作，且僅取代『第一個』
    | # 如果是 // ，則表示全部的字串都取代。

```

6. 例題：請取出uuid編號，不含雙引號。

2.6 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 已設定非為空字串
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 var=expr	str 不變 var=	str 不變 var=\$str
var=\${str:=expr}	str=expr var=expr	str=expr var=expr	str 不變 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"

```

2 [root@linux ~]# unset str; var=${str:-newvar}
2 [root@linux ~]# echo var="$var", str="$str"
var=newvar, str=          <==因為 str 不存在，所以 var 為 newvar
4 [root@linux ~]# str="oldvar"; var=${str:-newvar}
4 [root@linux ~]# echo var="$var", str="$str"
6 var=oldvar, str=oldvar <==因為 str 存在，所以 var 等於 str 的內容

```

3. 若 str 不存在，則 var 與 str 均設定為 newvar，否則僅 var 為 newvar

```

2 [root@linux ~]# unset str; var=${str=newvar}
2 [root@linux ~]# echo var="$var", str="$str"
var=newvar, str=newvar <==因為 str 不存在，所以 var/str 均為
newvar
4 [root@linux ~]# str="oldvar"; var=${str=newvar}
4 [root@linux ~]# echo var="$var", str="$str"
6 var=oldvar, str=oldvar <==因為 str 存在，所以 var 等於 str 的內容

```

4. 若 str 這個變數存在，則 var 等於 str，否則輸出 "novar"

```

2 [root@linux ~]# unset str; var=${str?novar}
2 -bash: str: novar <==因為 str 不存在，所以輸出錯誤訊息
4 [root@linux ~]# str="oldvar"; var=${str?novar}
4 [root@linux ~]# echo var="$var", str="$str"
var=oldvar, str=oldvar <==因為 str 存在，所以 var 等於 str 的內容

```

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。

練習題

1. 在 Bash shell 環境下，如何讓使用者由鍵盤輸入『A test』，並將其變成 atest 變數？
Sol. `執行 read atest，接著輸入 A test，最後按 [Ctrl]+d 結束輸入。`
2. 在 Bash shell 環境下，如何提示『keyin』讓使用者由鍵盤輸入『A test』，並將其變成 atest 變數？
Sol. `執行 read -p "keyin" atest，接著輸入 A test，最後按 [Ctrl]+d 結束輸入。`
3. 在 Bash shell 環境下，如何等待 30 秒內，讓使用者由鍵盤輸入『A test』，並將其變成 atest 變數？
Sol. `執行 read -t 30 atest，接著輸入 A test，最後按 [Ctrl]+d 結束輸入。`
4. 在 Bash shell 環境下，執行 `sum=10+30+5; echo $sum`，結果為何？
Sol. `10+30+5`
5. 在 Bash shell 環境下，執行 `declare -i sum=10+30+5; echo $sum`，結果為何？
Sol. `45`
6. 在 Bash shell 環境下，如何以指令 `declare` 將變數 `sum` 變成環境變數？
Sol. `declare -x sum`
7. 在 Bash shell 環境下，如何以指令 `declare` 將變數 `sum` 變成唯讀屬性？
Sol. `declare -r sum`
8. 在 Bash shell 環境下，如何以指令 `declare` 將變數 `sum` 變成陣列？
Sol. `declare -a sum`
9. 在 Bash shell 環境下，如何將陣列變數 `sum` 的內容設定為 `sum[0]=1`，`sum[1]=2` 及 `sum[2]=3`，並列出？
Sol. `sum=(1 2 3); echo ${sum[0]} ${sum[1]} ${sum[2]}`
10. 在 Bash shell 環境下，如何將陣列變數 `sum` 的內容全部列出？
Sol. `echo ${sum[@]}`
11. 在 Bash shell 環境下，如何將陣列變數 `sum` 的元素個數列出？
Sol. `echo ${#sum[@]}`
12. 在 Bash shell 環境下，如何列出檔案系統及程序的所有限制？
Sol. `ulimit -a`
13. 在 Bash shell 環境下，如何限制單一使用者可以使用的最大程序(process)數量為 1024？
Sol. `ulimit -u 1024`

14. 在 Bash shell 環境下，如何限制使用者僅能建立 1MBytes 以下的容量的檔案？

Sol. `ulimit -f 1024`

15. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie##*/}`，結果為何？

Sol. `testing.x.sh`

16. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie#*/}`，結果為何？

Sol. `csie/testing/testing.x.sh`

17. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie##/*}`，結果為何？

Sol. `(空串，或則空)`

18. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie#/*}`，結果為何？

Sol. `/home/csie/testing/testing.x.sh`

19. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie%/*}`，結果為何？

Sol. `/home/csie/testing/testing.x.sh`

20. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie%/*}`，結果為何？

Sol. `(空串，或則空)`

21. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie%/*}`，結果為何？

Sol. `/home/csie/testing/testing.x.sh`

22. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie%/*}`，結果為何？

Sol. `/home/csie/testing`

23. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie/testing/TEST}`，結果為何？

Sol. `/home/csie/TEST/testing.x.sh`

24. 在 Bash shell 環境下，若變數 `csie="/home/csie/testing/testing.x.sh"`，則執行 `echo ${csie//testing/TEST}`，結果為何？

Sol. `/home/csie/TEST/TEST.x.sh`

25. 在 Bash shell 環境下，如果變數 `str` 已設定或為空字串，則變數 `var=$str`；否則 `var=new` 字串，如何設定？

Sol. `var=${str-new}`

26. 在 Bash shell 環境下，如果變數 `str` 已設定為非空字串，則變數 `var=$str`；否則 `var=new` 字串，如何設定？

Sol. `var=${str:-new}`

27. 在 Bash shell 環境下，如果變數 `str` 已設定或為空字串，則變數 `var=new` 字串；否則 `var=` 空字串，如何設定？

Sol. `var=${str+new}`

28. 在 Bash shell 環境下，如果變數 `str` 已設定為非空字串，則變數 `var=new` 字串；否則 `var=` 空字串，如何設定？

Sol. `var=${str:+new}`

29. 在 Bash shell 環境下，執行 `unset str; var=${str-new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=new, str=`

30. 在 Bash shell 環境下，執行 `str=""; var=${str-new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=, str=`

31. 在 Bash shell 環境下，執行 `str="old"; var=${str-new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=old, str=old`

32. 在 Bash shell 環境下，執行 `unset str; var=${str:-new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=new, str=`

33. 在 Bash shell 環境下，執行 `str=""; var=${str:-new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=new, str=`

34. 在 Bash shell 環境下，執行 `str="old"; var=${str:-new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=old, str=old`

35. 在 Bash shell 環境下，執行 `unset str; var=${str+new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=, str=`

36. 在 Bash shell 環境下，執行 `str=""; var=${str+new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=new, str=`

37. 在 Bash shell 環境下，執行 `str="old"; var=${str+new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=new, str=old`

38. 在 Bash shell 環境下，執行 `unset str; var=${str:+new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=, str=`
39. 在 Bash shell 環境下，執行 `str=""; var=${str:+new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=, str=`
40. 在 Bash shell 環境下，執行 `str="old"; var=${str:+new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=new, str=old`
41. 在 Bash shell 環境下，執行 `unset str; var=${str=new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=new, str=new`
42. 在 Bash shell 環境下，執行 `str=""; var=${str=new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=, str=`
43. 在 Bash shell 環境下，執行 `str="old"; var=${str=new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=old, str=old`
44. 在 Bash shell 環境下，執行 `unset str; var=${str:=new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=new, str=new`
45. 在 Bash shell 環境下，執行 `str=""; var=${str:=new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=new, str=new`
46. 在 Bash shell 環境下，執行 `str="old"; var=${str:=new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=old, str=old`
47. 在 Bash shell 環境下，執行 `unset str; var=${str?new}; echo var="$var", str="$str"` 的結果為何？
Sol. `-bash: str: new (No such file)`
48. 在 Bash shell 環境下，執行 `str=""; var=${str?new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=, str=`
49. 在 Bash shell 環境下，執行 `str="old"; var=${str?new}; echo var="$var", str="$str"` 的結果為何？
Sol. `var=old, str=old`

50. 在 Bash shell 環境下，執行 `unset str; var=${str:?new}; echo var="$var", str="$str"` 的結果為何？

Sol. `-bash: str: new (錯誤訊息)`

51. 在 Bash shell 環境下，執行 `str=""; var=${str:?new}; echo var="$var", str="$str"` 的結果為何？

Sol. `-bash: str: new (錯誤訊息)`

52. 在 Bash shell 環境下，執行 `str="old"; var=${str:?new}; echo var="$var", str="$str"` 的結果為何？

Sol. `var=old, str=old`

2.7 實機練習題

2.7.1 練習一

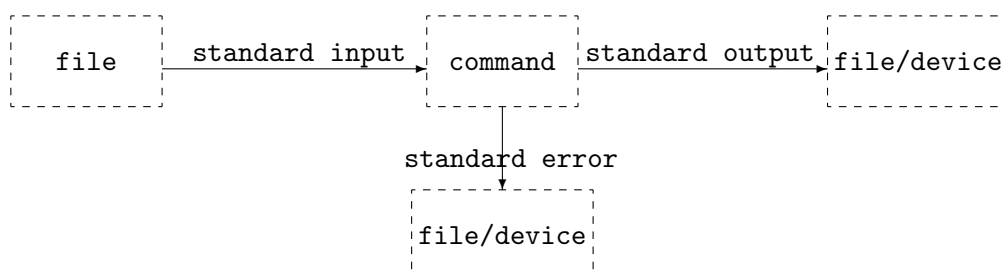
1. 在家目錄下建立 `zzz` 目錄。
2. 切換工作目錄到 `zzz`。
3. 在此目錄下寫一腳本 `a1.sh`，使用 `read` 從鍵盤輸入讀入一數字，並將其乘 2 後輸出到螢幕。

Chapter 3

資料導向與管線處理

3.1 資料流重導向

- 為何要使用命令輸出重導向？
 1. 儲存螢幕輸出的資訊；
 2. 背景執行中的程式，不希望他干擾螢幕正常的輸出結果時；
 3. 儲存系統例行命令（例如寫在 `/etc/crontab` 中的檔案）的執行結果；
 4. 將已知的可能錯誤訊息丟掉『`2> /dev/null`』；
 5. 錯誤訊息與正確訊息需要分別輸出時。
- 指令執行過程之資料傳輸



1. 例題：執行 `cat /etc/crontab /etc/dywang`，standard input 與 standard output 同時出現。

```

1 [dywang@dywHome ~]$ cat /etc/crontab /etc/dywang
2 # /etc/crontab 存在，螢幕出現以下 standard output
3 SHELL=/bin/bash
4
5 MAILTO=root
6 HOME=/
7
8 # run-parts
9 01 * * * * root nice -n 19 run-parts --report /etc/cron.
   hourly

```

```

11 02 4 * * * root nice -n 19 run-parts --report /etc/cron.daily
    22 4 * * 0 root nice -n 19 run-parts --report /etc/cron.
        weekly
    42 4 1 * * root nice -n 19 run-parts --report /etc/cron.
        monthly
13 # /etc/dywang 不存在，螢幕出現以下 standard error
    cat: /etc/dywang: No such file or directory

```

2. 傳送指令

- (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

3. 檔案的建立方式

```

2 [root@linux ~]# ls -l / > ~/rootfile
  # 將 ls -l / 要列出到螢幕上的資料『重新導向』到 ~/rootfile 檔
    案。

```

- (a) 該檔案(本例中是 ~/rootfile)若不存在，系統會自動建立；
- (b) 檔案存在時，系統會先將檔案內容清空再將資料寫入；
- (c) 以 > 輸出到一個既存檔案中，該檔案會被覆蓋。

4. 下達 > 與 >>

```

2 # 將目前目錄下的檔案資訊儲存到 list.txt
  [root@linux ~]# ls -al > list.txt
4 # 將根目錄下的資料累加到 list.txt
  [root@linux ~]# ls -al / >> list.txt

```

5. 下達 2>

```

1 [csie@linux ~]$ find /home -name testing
  find: /home/test1: Permission denied <== Starndard error
3 find: /home/root: Permission denied <== Starndard error
  find: /home/masda: Permission denied <== Starndard error

```

```

5 | /home/csie/testing                <== Starndard output
7 | [csie@linux ~]$ find /home -name testing > list_right 2>
   | list_error
   | [csie@linux ~]$ find /home -name testing > list_right 2> /dev
   | /null

```

6. 下達 `2>&1`：標準輸出與標準錯誤輸出同時寫入同一個檔案

```

[csie@linux ~]$ find /home -name testing > list 2> list
<==錯誤寫
法
2 ## 習題：請問上面命令的執行結果為何？
[csie@linux ~]$ find /home -name testing > list 2>&1
<==正確寫
法

```

7. 將螢幕輸入結果存入檔案

```

1 [root@linux ~]# cat > catfile
  testing
3 cat file test
  <== 按下 [ctrl]+d 結束輸入來離開。
5
  [root@linux ~]# cat > catfile <<eof
7 > This is a test testing
  > OK now stop
9 > eof <==輸入 eof 時，該次輸入就結束輸了。

```

8. 習題：如上述命令，請將 `catfile` 第一行內容改為學號，且輸入 `end` 時結束輸入。

9. 先編輯 `somefile`，然後再以上述的指令來將資料輸出到 `catfile`

```

1 [root@linux ~]# cat > catfile < somefile

```

練習題

1. 要將 `ls -al /` 的標準輸出重新導向，存到(覆蓋或建立) `~/list`，如何下指令？

Sol. `ls -al / > ~/list` 或 `ls -al / 1> ~/list`

2. 要將 `ls -al /` 的標準輸出重新導向，存到(建立或累加) `~/list`，如何下指令？

Sol. `ls -al / >> ~/list` 或 `ls -al / 1>> ~/list`

3. 要將 `ls -al /csie` 的錯誤輸出重新導向，存到(覆蓋或建立) `~/listerr`，如何下指令？

Sol. `ls -al /csie 2> ~/listerr`

4. 要將 `ls -al / /csie` 的標準輸出及錯誤輸出分別重新導向，存到(覆蓋或建立) `~/list` 及 `~/listerr`，如何下指令？

Sol. `ls -al / /csie > ~/list 2> ~/listerr` 或
`ls -al / /csie 1> ~/list 2> ~/listerr`

5. 要將 `ls -al / /csie` 的標準輸出及錯誤輸出分別重新導向，存到(建立或累加) `~/list` 及 `~/listerr`，如何下指令？

Sol. `ls -al / /csie >> ~/list 2>> ~/listerr` 或
`ls -al / /csie 1>> ~/list 2>> ~/listerr`

6. 要將 `ls -al / /csie` 的標準輸出及錯誤輸出同時重新導向，存到(覆蓋或建立) `~/list`，如何下指令？

Sol. `ls -al / /csie > ~/list 2>&1`

7. 要將 `ls -al / /csie` 的標準輸出及錯誤輸出同時重新導向，存到(建立或累加) `~/list`，如何下指令？

Sol. `ls -al / /csie >> ~/list 2>&1`

3.2 連續命令

1. 指令間以分號(;)隔開：連續執行指令。

```
1 [root@linux ~]# sync; sync; shutdown -h now
```

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。

```
1 [root@linux ~]# ls /tmp && touch /tmp/testingagin
## 目錄/存在，所以tmp/tmp/會被建立。testingagin
3 [root@linux ~]# ls /csie && touch /csie/test
## 目錄/不存在，所以csietouch /csie/不會被執行。test
```

3. 指令間以 `||` 隔開：前一個指令有錯誤時，後面的指令才被執行。

```
[root@linux ~]# ls /tmp/csieing || touch /tmp/csieing
```

4. 例題：以 `ls` 測試 `/tmp/csie` 是否存在？若存在則顯示 `"exist"`，若不存在，則顯示 `"not exist"`。

```
1 | ls /tmp/csie && echo "exist" || echo "not exist" <== 正確
   | ls /tmp/csie || echo "not exist" && echo "exist" <== 錯誤
```

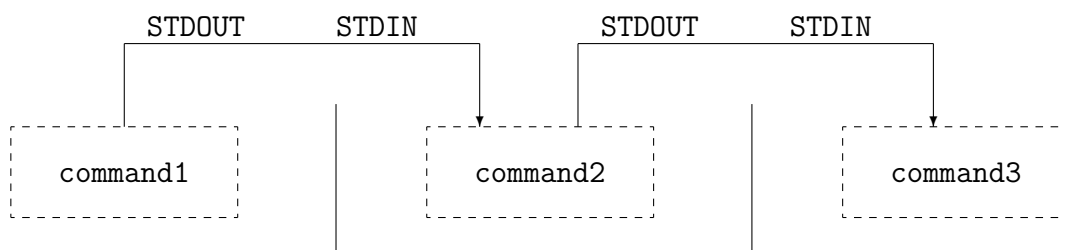
5. 習題：以 `ls` 測試目錄 `/tmp/s9427???` 是否存在，若存在則進入目錄，若不存在，則建立目錄。

練習題

- 請說明指令 `ls /tmp && touch /tmp/testing` 中『&&』之意義？
Sol. 前面指令 `ls /tmp` 執行結果正確，就接著執行後續的指令 `touch /tmp/testing`，否則就略過。
- 請說明指令 `ls /tmp/testing || touch /tmp/testing` 中『||』之意義？
Sol. 前面指令 `ls /tmp/testing` 執行有錯誤時，才執行後續的指令 `touch /tmp/testing`。
- 請以 `ls` 測試 `/tmp/csie` 是否存在？若存在則顯示 `"exist"`，若不存在，則顯示 `"not exist"`。
Sol. `ls /tmp/csie && echo "exist" || echo "not exist"`

3.3 管線命令 (pipe)

● 管線命令的處理



1. 例題：使用 `ls` 指令輸出後的內容，被 `less` 讀取，並且利用 `less` 的功能前後翻動相關的資訊。

```
[root@linux ~]# ls -al /etc | less
```

2. 習題：使用 `ls` 指令輸出 `/home/csie` 內容後，其被 `more` 讀取。
- 擷取命令 `cut`：可以將一段訊息的某一段『切』出來，處理的訊息以『行』為單位。

1. cut 指令：

```

1 [root@linux ~]# cut -d分隔字元',' -f fields
2 [root@linux ~]# cut -c 字元區間選項：
3
4 -d :後面接分隔字元。與 -f 一起使用；
5 -f :依據 -d 的分隔字元將一段訊息分割成為數段，用 -f 取出第幾
   段；
6 -c :以字元 (characters) 的單位取出固定字元區間；

```

2. 將 PATH 變數取出，找出第三個路徑。

```

1 [root@linux ~]# echo $PATH
2 /bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:/usr/X11R6/bin:/
   usr/games:
3 [root@linux ~]# echo $PATH | cut -d ':' -f 5
4 # 以 : 作為分隔符號，第五個是 /usr/local/bin
5 # 若要列出第 3 與第 5 個？
6 [root@linux ~]# echo $PATH | cut -d ':' -f 3,5
## 習題：取出 PATH 變數之第 2 與第 4 個路徑。

```

3. 將 export 輸出的訊息，取得第 12 字元以後的所有字串

```

1 [root@linux ~]# export
2 declare -x HISTSIZE="1000"
3 declare -x INPUTRC="/etc/inputrc"
4 declare -x KDEDIR="/usr"
5 declare -x LANG="zh_TW.big5"其他省略
6 .....
7 [root@linux ~]# export | cut -c 12-
8 HISTSIZE="1000"
9 INPUTRC="/etc/inputrc"
10 KDEDIR="/usr"
11 LANG="zh_TW.big5"其他省略
12 .....
13 # 用 -c 可以處理比較具有格式的輸出資料。
14 # 還可以指定某個範圍的值，例如第 12-20 的字元
   為 cut -c 12-20 等。
15 ## 習題：取出 /etc/issue 之第 1 至 13 個字元。

```

4. 用 last 將本月登入者的資訊中，僅留下使用者大名

```

1 [root@linux ~]# last
2 csie tty1 192.168.1.28 Mon Aug 15 11:55 - 17:48 (05:53)
3 csie tty1 192.168.1.28 Mon Aug 15 10:17 - 11:54 (01:37)

```



```

1 [root@linux ~]# last | cut -d ' ' -f 1
5 # 用 last 可以取得最近一個月登入主機的使用者資訊，
# 利用空白字元的間隔，取出第一個資訊，就是使用者帳號。
7 # 但因 csie tty1 之間空格並非僅有一個，cut -d ' ' -f 的輸出不
  是1,2 。tty1

```

- 擷取命令 `grep`：分析一行訊息，若當中有需要的資訊，就將該行取出。

1. `grep` 指令：

```

1 [root@linux ~]# grep [-acinv] 搜尋字串'' filename選項：
3 -a : 將 binary 檔案以 text 檔案的方式搜尋資料
-c : 計算找到搜尋字串 '' 的次數
5 -i : 忽略大小寫的不同
-n : 輸出行號
7 -v : 反向選擇，亦即顯示出沒有搜尋字串內容的那一行。''

```

2. 取出 `last` 中出現 `root` 的那一行

```

1 [root@linux ~]# last | grep 'root'
## 習題：請找出 /etc/issue 下出現 Mandrake 的那一行。

```

3. 與上例相反，只要沒有 `root` 的就取出。

```

2 [root@linux ~]# last | grep -v 'root'
## 習題：請找出 /etc/issue 下沒出現 Mandrake 的那一行。

```

4. 在 `last` 的輸出訊息中，只要有 `root` 就取出，並且僅取第一欄

```

2 [root@linux ~]# last | grep 'root' | cut -d ' ' -f1
## 習題：在 cat /etc/group 的輸出訊息中，只要有 sshd 就取出，並
  且僅取第一欄及第三欄。

```

- 排序命令 `sort`：可依據不同的資料型態來排序。

1. `sort` 指令：

```

2 [root@linux ~]# sort [-fbMnrtuk] [file or stdin]選項：

```

```

4  -f : 忽略大小寫的差異，例如 A 與 a 視為編碼相同；
    -b : 忽略最前面的空白字元部分；
    -M : 以月份的名字來排序，例如 JAN, DEC 等排序方法；
6  -n : 使用『純數字』進行排序預設是以文字型態排序 ()；
    -r : 反向排序；
8  -u : , 相同的資料中僅出現一行； uniq
    -t : 分隔符號，預設是 tab 鍵；
10 -k : 以第幾個區間 (field) 進行排序，

```

2. 個人帳號都記錄在 `/etc/passwd` 下，請將帳號進行排序。

```

[root@linux ~]# cat /etc/passwd | sort
2 adm:x:3:4:adm:/var/adm:/sbin/nologin
  apache:x:48:48:Apache:/var/www:/sbin/nologin
4 bin:x:1:1:bin:/bin:/sbin/nologin
  daemon:x:2:2:daemon:/sbin:/sbin/nologin
6 # sort 預設『以第一個』資料排序，且以『文字』型態排序。

```

3. `/etc/passwd` 內容以 `:` 分隔，如何以第三欄來排序？

```

[root@linux ~]# cat /etc/passwd | sort -t ':' -k 3
2 root:x:0:0:root:/root:/bin/bash
  iiimd:x:100:101:IIIMF server:/usr/lib/iiim:/sbin/nologin
4 uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
  operator:x:11:0:operator:/root:/sbin/nologin
6 bin:x:1:1:bin:/bin:/sbin/nologin
  games:x:12:100:games:/usr/games:/sbin/nologin

```

4. `sort -n` 使用數字排序。

```

1 [root@dywhd2 ~]# cat /etc/passwd | sort -t ':' -k 3 -n
   root:x:0:0:root:/root:/bin/bash
3 bin:x:1:1:bin:/bin:/sbin/nologin
  daemon:x:2:2:daemon:/sbin:/sbin/nologin
5 adm:x:3:4:adm:/var/adm:/sbin/nologin
  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
7 sync:x:5:0:sync:/sbin:/bin/sync
  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
9 halt:x:7:0:halt:/sbin:/sbin/halt
  mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
11 .....

```

5. `sort` 選項間可以不使用空格，`-t` 可以直接接上分隔字元，不必使用單引號括起來，也不用空格隔開。

```

1 [root@dywhd2 ~]# sort -t: -nk3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
3 bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
5 adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
7 sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
9 halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
11 .....

```

6. 利用 last 將輸出的資料僅取帳號並加以排序

```

1 [root@linux ~]# last | cut -d ' ' -f1 | sort
## 習題：請以 cat 顯示 /etc/fstab 檔案內容，切下其第二欄，並排
序。

```

• uniq：將重複的資料僅列出一個顯示

1. uniq 指令：

```

2 [root@linux ~]# uniq [-ic] 選項：
-i : 忽略大小寫字元的不同；
4 -c : 進行計數

```

2. 使用 last 將帳號列出，僅取出帳號欄，進行排序後僅取出一位；

```

2 [root@linux ~]# last | cut -d ' ' -f1 | sort -u
2 [root@linux ~]# last | cut -d ' ' -f1 | sort | uniq

```

3. 承上題，如何知道每個人的登入總次數？

```

2 [root@linux ~]# last | cut -d ' ' -f1 | sort | uniq -c
##(1) 先將所有的資料列出；(2) 再將人名獨立出來；(3) 經過排
序；(4) 只顯示一個。

```

• wc：計算輸出的訊息的行、字及字元數等

1. wc 指令：

```

1 [root@linux ~]# wc [-lwm] 選項：
2
3 -l  : 僅列出行；
4 -w  : 僅列出多少字英文單字 ( )；
5 -m  : 多少字元；

```

2. 計算 `/etc/man.config` 有多少字、行、字元數？

```

1 [root@linux ~]# cat /etc/man.config | wc
2      138      709     4506
3 # 輸出的三個數字分別代表：『行、字數、字元數』
4 ## 習題：請列出 /etc/issue 有多少字、行、字元數？

```

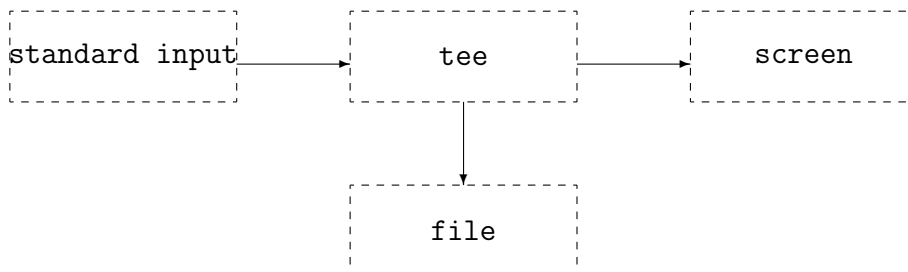
3. 如何以 `last` 指令串取得登入系統的總人次？(`last` 會輸出空白行與 `wtmp` 字樣在最底下兩行，其並非帳號內容)

```

1 [root@linux ~]# last | grep [a-zA-Z] | grep -v 'wtmp' | wc -l
2 # 利用 grep 取出非空白行以及去除 wtmp 那一行，再計算行數。
3 ## 習題：請取得以 a- 開頭登入系統的總人次。f

```

● 雙向重導向：tee 工作流程



1. tee 指令：

```

1 [root@linux ~]# tee [-a] file 選項：
2
3 -a  : 以累加 (append) 的方式，將資料加入 file 當中。

```

2. 將 `last` 的輸出存一份到 `last.list` 檔案中

```

1 [root@linux ~]# last | tee last.list | cut -d " " -f1

```

3. 將 `ls` 的資料存一份到 `~/homefile`，同時螢幕也有輸出訊息。

```
1 [root@linux ~]# ls -l /home | tee ~/homefile | more
```

4. 將 `ls` 的資料累加到 `~/homefile`，同時螢幕也有輸出訊息。

```
1 [root@linux ~]# ls -l / | tee -a ~/homefile | more
```

- 字元轉換命令 `tr`：可以用來刪除一段訊息當中的文字，或者是進行文字訊息的替換。

1. `tr` 指令：

```
1 [root@linux ~]# tr [-ds] SET1 ...選項：
3 -d : 刪除訊息當中的 SET1 這個字串；
  -s : 取代重複的字元。
```

2. 將 `last` 輸出的訊息中，所有的小寫變成大寫字元：

```
[root@linux ~]# last | tr '[a-z]' '[A-Z]'
```

3. 將 `/etc/passwd` 輸出的訊息中，將冒號 (:) 刪除

```
1 [root@linux ~]# cat /etc/passwd | tr -d ':'
```

4. 將檔案 `/home/csie/txtfile` 中的重複字元 `p` 取代成單一個 `q` 字元：

```
1 [root@linux ~]# cat /home/csie/txtfile | tr -s 'p' 'q'
```

- 字元轉換命令 `col`：濾除標準輸入的反向換行符號。

1. `col` 指令：

```
1 [root@linux ~]# col [-bfp] [-l num]選項
:
3 -b 不輸出任何的退格(backspaces)，只列出行末的符號
```

```

    -f 允許不濾除()正向半換行符("fine" 模式). 通常, 位在半行分界線
        的字元會列印到下一行。
5    -p 讓無法判別的控制字元通過並列印到輸出。
    -x 以輸出連續空格取替換跳格(tab)
7    -lnum 設定緩衝記憶體, 以"行"為單位。預設值為 128 行。

```

2. 轉存 man 指令的輸出資料。

```

1  [root@dywHome2 ~]# man ls |cat -v | sed -n '5,10p'
N^HNA^HAM^HME^HE                                <== 夾雜有相當多的控制字元
3      ls - list directory contents
5  S^HSY^HYN^HNO^HOP^HPS^HSI^HIS^HS
    l^Hls^Hs [_^HO_ ^HP_ ^HT_ ^HI_ ^HO_ ^HN]... [_^HF_ ^HI_ ^HL_ ^
    HE]...
7
9  [root@dywHome2 ~]# man ls |col -b | cat -v | sed -n '5,10p'
NAME
    ls - list directory contents
11
13 SYNOPSIS
    ls [OPTION]... [FILE]...

```

3. [tab] 按鍵被取代成空白。

```

[root@dywHome2 ~]# cat -A /etc/man.config | sed -n '43,47p'
MANPATH^I/usr/share/man$      %*<== ^I 符號, 就是 [tab]*)
MANPATH^I/usr/X11R6/man$
MANPATH^I/usr/local/man$
MANPATH^I/usr/kerberos/man$
MANPATH^I/usr/man$
[root@dywHome2 ~]# cat /etc/man.config | col -x |cat -A | sed -n '43,47p'
MANPATH /usr/share/man$      %*<== [tab] 按鍵被取代成空白鍵*)
MANPATH /usr/X11R6/man$
MANPATH /usr/local/man$
MANPATH /usr/kerberos/man$
MANPATH /usr/man$

```

- join：可以將『兩個檔案當中，有"相同資料"的那一行，將他加在一起』

1. join 指令：

```

1  [root@linux ~]# join [-ti12] file1 file2選項：

```

```

3 || -t : join 預設以空白字元分隔資料，並且比對『第一個欄位』的資
      料，如果兩個檔案相同，則將兩筆資料聯成一行，且第一個欄位放在
      第一個。
5 ||
      -i : 忽略大小寫的差異；
      -1 : 代表『第一個檔案要用那個欄位來分析』；
7 || -2 : 代表『第二個檔案要用那個欄位來分析』。

```

2. 用 root 的身份，將 /etc/passwd 與 /etc/shadow 相關資料整合成一欄

```

1 [root@linux ~]# join -t ':' /etc/passwd /etc/shadow
  bin:x:1:1:bin:/bin:/sbin/nologin:*:12959:0:99999:7:::
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin:*:12959:0:99999:7:::
  adm:x:3:4:adm:/var/adm:/sbin/nologin:*:12959:0:99999:7:::
5 # /etc/shadow 必須以 root 權限才能動作。
  # /etc/passwd 與 /etc/shadow 都是以 : 分隔欄位，必須使
    用 -t ':' 規範欄位分隔字元。
7 # /etc/shadow 與 /etc/passwd 都是以第一個欄位為帳號名稱。

```

3. /etc/passwd 第四個欄位是 GID，其記錄在 /etc/group 中的第三個欄位，如何將兩個檔案整合？

```

1 [root@linux ~]# join -t ':' -1 4 /etc/passwd -2 3 /etc/group
  0:root:x:0:root:/root:/bin/bash:root:x:
3 1:bin:x:1:bin:/bin:/sbin/nologin:bin:x:root,bin,daemon
  2:daemon:x:2:daemon:/sbin:/sbin/nologin:daemon:x:root,bin,
    daemon
5 4:adm:x:3:adm:/var/adm:/sbin/nologin:adm:x:root,adm,daemon
  # /etc/passwd 的第一行內容是 root:x:0:0:root:/root:/bin/bash
7 # /etc/group 第一行內容是 root:x:0:
  # 將第一個檔案的第四欄與第二個檔案的第三欄取出，放置到輸出的最前
    方，剩下的資料加在一起。

```

- paste：可將兩行貼在一起，且中間以 [tab] 鍵隔開

1. paste 指令：

```

2 [root@linux ~]# paste [-d] file1 file2 選項：
  -d : 後面可以接分隔字元。預設是以 [tab] 來分隔的。
4 - : 如果 file 部分寫成 -，表示來自 standard input 的資料。

```

2. 將 `/etc/passwd` 與 `/etc/shadow` 同一行貼在一起

```

1 [root@linux ~]# paste /etc/passwd /etc/shadow
2 bin:x:1:1:bin:/bin:/sbin/nologin      bin
   *:12959:0:99999:7:::
   daemon:x:2:2:daemon:/sbin:/sbin/nologin daemon
   *:12959:0:99999:7:::
4  adm:x:3:4:adm:/var/adm:/sbin/nologin  adm
   *:12959:0:99999:7:::
# 同一行中間是以 [tab] 按鍵隔開的。

```

3. 先將 `/etc/group` 讀出(用 `cat`)，然後與範例一貼上一起，且僅取出前三行。

```

1 [root@linux ~]# cat /etc/group|paste /etc/passwd /etc/shadow
   -|head -n 3
# - 代表。 stdin

```

- `expand`：將 `[tab]` 按鍵轉成空白鍵

1. `expand` 指令：

```

2 [root@linux ~]# expand [-t] file選項：
   -t : 後面可以接數字。一般，一個 tab 按鍵可以用 8 個空白鍵取
      代。也可以自行定義一個
4     [tab] 按鍵代表多少個字元。

```

2. 將 `/etc/man.config` 內行首為 `MANPATH` 的字樣取出；僅取前三行；

```

2 [root@linux ~]# grep '^MANPATH' /etc/man.config | head -n 3
MANPATH /usr/man
MANPATH /usr/share/man
4 MANPATH /usr/local/man
# 行首的代表標誌為 ^。

```

3. 承上題，將所有的符號都列出來？(用 `cat`)

```

1 [root@linux ~]# grep '^MANPATH' /etc/man.config | head -n 3 |
   cat -A
MANPATH^I/usr/man$
3 MANPATH^I/usr/share/man$
MANPATH^I/usr/local/man$

```



```
5 | # [tab] 按鍵可以被 cat -A 顯示成爲 ^I
```

4. 承上題，將 [tab] 按鍵設定成 6 個字元？

```
1 | [root@linux ~]# grep '^MANPATH' /etc/man.config | head -n 3 |
  | \
  | > expand -t 6 - | cat -A
3 | MANPATH      /usr/man$
  | MANPATH      /usr/share/man$
5 | MANPATH      /usr/local/man$
  | 123456123456123456.....
7 | # 因爲 [tab] 的長度爲 6 個字元，所以，MAN... 到 /usr 之間
  |   隔 12 兩個( [tab]) 個字元。
  | # 如果 tab 改成 9 個字元？
```

- 分割命令：split 可將檔案分割

1. split 指令：

```
2 | [root@linux ~]# split [-bl] file PREFIX選項：
  |
  | -b  : 後面接欲分割成的檔案大小，可加單位，例如 b, k, m 等；
4 | -l  : 以行數來進行分割。
```

2. /etc/termcap 有七百多K，如何分成 300K 一個檔案？

```
2 | [root@linux ~]# cd /tmp; split -b 300k /etc/termcap termcap
  | [root@linux tmp]# ls -l termcap*
  | -rw-rw-r-- 1 root root 307200 月 8 17 00:25 termcapaa
4 | -rw-rw-r-- 1 root root 307200 月 8 17 00:25 termcapab
  | -rw-rw-r-- 1 root root 184848 月 8 17 00:25 termcapac
6 | # 只要寫上前導文字，xxxsplit 就會以 xxxaa, xxxab, xxxac 建立小
  |   檔案。
```

3. 將上面的三個小檔案合成一個檔案，檔名爲 termcapback

```
[root@linux tmp]# cat termcap* >> termcapback
```

4. 使用 ls -al / 輸出的資訊中，每十行記錄成一個檔案

```
1 [root@linux tmp]# ls -al / | split -l 10 - lsroot
# - 會被當成 stdin 或 stdout
```

- 參數代換：xargs 可以讀入 stdin 的資料，並且以空白字元或斷行字元作為分辨，將 stdin 的資料分隔成為 arguments。

1. xargs 指令：使用於非管線處理命令。

```
2 [root@linux ~]# xargs [-Oepn] command選項：
-0 : 如果輸入的 stdin 含有特殊字元，例如 '，\，空白鍵等等字元
    時，這個 -0 參數可以將他還原成一般字元。
4 -e : EOF (end of file) 。當 xargs 分析到其後接的字串時，會停
    止繼續工作。
6 -p : 在執行每個指令的 argument 時，都會詢問使用者；
-n : 後面接次數，每次 command 指令執行時，要使用幾個參數。看範
    例三。當
8 xargs 後面沒有接任何的指令時，預設是以 echo 來進行輸出。
```

2. 將 /etc/passwd 內的第一欄取出，僅取三行，使用 finger 指令將每個帳號內容秀出來

```
[root@linux ~]# cut -d':' -f1 < /etc/passwd | head -n 3 | xargs
finger
2 Login: root                               Name: root
  Directory: /root                         Shell: /bin/bash
4 Never logged in.
  No mail.
6 No Plan.底下省略
  .....
8 # 利用 cut 取出帳號名稱，用 head 取出三個帳號。
# 由 xargs 將三個帳號的名稱變成 finger 後面需要的參數。
10 # 上例為 finger root 後的結果。
```

3. 同上題，每次執行 finger 時，都要詢問使用者是否動作？

```
[root@linux ~]# cut -d':' -f1 < /etc/passwd | head -n 3 | xargs
-p finger
2 finger root bin daemon ?...y底下省略
  .....
```

4. 將所有的 `/etc/passwd` 內的帳號都以 `finger` 查閱，但一次僅查閱五個帳號

```
1 [root@linux ~]# cut -d':' -f1 < /etc/passwd | xargs -p -n 5
  finger
finger root bin daemon adm lp ?...y底下省略
3 .....
# 某些指令後面可以接的 arguments 是有限制的，不能無限的累加，
5 # 此時可以利用 -n 將參數分成數個部分，每個部分分別再以指令來執行。
```

5. 同上題，但當分析到 `lp` 就結束這串指令？

```
1 [root@linux ~]# cut -d':' -f1 < /etc/passwd | xargs -p -e'lp'
  finger
finger root bin daemon adm ?...
3 # -e'lp' 中間沒有空白鍵。
# 上例中第五個參數是 lp，
5 # 分析到字串 lp 時，後面的其他 stdin 的內容會被 xargs 捨棄掉。
```

- 關於減號 `-` 的用途：代替 `stdin` 與 `stdout`

```
1 [root@linux ~]# tar -cvf - /home | tar -xvf -
# 將 /home 裡面的檔案打包，打包的資料傳送到； stdout
3 # 利用管線處理，將 tar -cvf - /home 傳送給後面的 tar -xvf -。
# 後面的 - 則是取用前一個指令的。 stdout
```

練習題

1. 如何使用管線命令，以 `ls` 指令輸出 `/home/csie` 內容後，其被 `more` 讀取？

Sol. `ls /home/csie | more`

2. 如何使用管線命令，以 `echo $PATH` 指令輸出 `PATH` 路徑後，再以指令 `cut` 切出 其第三個路徑？

Sol. `echo $PATH | cut -d ':' -f 3`

3. 如何使用管線命令，以 `echo $PATH` 指令輸出 `PATH` 路徑後，再以指令 `cut` 切出 其第二及五個路徑？

Sol. `echo $PATH | cut -d ':' -f 2,5`

4. 如何使用管線命令，將 `export` 指令輸出，用 `cut` 切出每行的第 12 字元以後的所有字串？

Sol. `export | cut -c 12-`

5. 如何使用管線命令，將 `ls -al` 指令輸出，用 `cut` 切出每行的第 10 至 22 字元？
Sol. `ls -al | cut -c 10-22`
6. 如何使用管線命令，將 `last` 指令輸出，用 `grep` 取出出現 `root` 的那一行？
Sol. `last | grep 'root'` (`root`, `"root"`亦可)
7. 如何使用管線命令，將 `last` 指令輸出，用 `grep` 取出沒有出現 `root` 的那一行？
Sol. `last | grep -v 'root'` (`root`, `"root"`亦可)
8. 如何使用管線命令，將 `last` 指令輸出，用 `grep` 取出出現 `root` 的那一行並計算找到的次數？
Sol. `last | grep -c 'root'` (`root`, `"root"`亦可)
9. 如何使用管線命令，將 `last` 指令輸出，用 `grep` 取出出現 `root` (忽略大小寫的不同) 的那一行？
Sol. `last | grep -i 'root'` (`root`, `"root"`亦可)
10. 如何使用管線命令，將 `last` 指令輸出，用 `grep` 取出出現 `root` 的那一行並輸出行號？
Sol. `last | grep -n 'root'` (`root`, `"root"`亦可)
11. 如何使用管線命令，將 `last` 指令輸出，用 `grep` 取出出現 `root` 的那一行並且用 `cut` 僅取第一欄？
Sol. `last | grep 'root' | cut -d ' ' -f1` (`root` 不加引號及加雙引號亦可)
12. 如何使用管線命令，找出 `/etc/issue` 下出現 `Mandrake` 的那一行？
Sol. `cat /etc/issue | grep 'Mandrake'` (`Mandrake` 不加引號及加雙引號亦可)
13. 如何使用管線命令，找出 `/etc/issue` 下沒有出現 `Mandrake` 的那一行？
Sol. `cat /etc/issue | grep -v 'Mandrake'` (`Mandrake` 不加引號及加雙引號亦可)
14. 如何使用管線命令，在 `cat /etc/group` 的輸出訊息中，只要有 `sshd` 就取出，並且僅取第一欄及第三欄？
Sol. `cat /etc/group | grep 'sshd' | cut -d ':' -f1,3` (`sshd` 不加引號及加雙引號亦可)
15. 如何使用管線命令，在 `cat /etc/passwd` 的輸出訊息中，以第三欄進行帳號排序？
Sol. `cat /etc/passwd | sort -t ':' -k 3`
16. 如何使用管線命令，在 `cat /etc/passwd` 的輸出訊息中，以第三欄且以數字進行帳號排序？
Sol. `cat /etc/passwd | sort -t ':' -k 3 -n`

17. 如何使用管線命令，將 `last` 指令輸出，僅取帳號並加以排序？
Sol. `last | cut -d ':' -f1 | sort`
18. 如何使用管線命令，將指令 `cat /etc/fstab` 輸出，切下其第二欄並排序？
Sol. `cat /etc/fstab | cut -d ' ' -f2 | sort`
19. 如何使用管線命令，將 `last` 指令輸出，僅取帳號並加以排序，且相同的帳號僅出現一行？
Sol. `last | cut -d ':' -f1 | sort -u` 或 `last | cut -d ':' -f1 | sort | uniq`
20. 如何使用管線命令，將 `last` 指令輸出，僅取帳號並加以排序，且相同的帳號僅出現一行並計算每個帳號的登入總次數？
Sol. `last | cut -d ':' -f1 | sort | uniq -c`
21. 如何使用管線命令，計算 `/etc/man.config` 有多少字、行、字元數？
Sol. `cat /etc/man.config | wc`
22. 如何使用管線命令，以一行指令串取得登入系統的總人次？
Sol. `last | grep '[a-z@]' | grep -v '/bin/' | wc -l`
23. 如何使用雙向重導向命令，將 `last` 指令輸出存一份到 `last.list` 檔案中，而螢幕上僅秀出帳號？
Sol. `last | tee last.list | cut -d ':' -f1`
24. 如何使用雙向重導向命令，將 `ls /home` 的資料累加到 `/homefile`，同時螢幕也以 `more` 輸出訊息？
Sol. `ls /home | tee -a /homefile | more`
25. 如何使用管線命令，將 `last` 輸出的訊息中，所有的小寫變成大寫字元？
Sol. `last | tr '[a-z]' '[A-Z]`
26. 如何使用管線命令，將 `cat /etc/passwd` 輸出的訊息中，所有的冒號 (:) 刪除？
Sol. `cat /etc/passwd | tr -d ':'`
27. 如何使用管線命令，將 `cat /etc/passwd` 輸出的訊息中，所有的重複字元 `p` 取代成單一個 `q` 字元？
Sol. `cat /home/csie/txfile | tr -s 'p' 'q'`
28. 線上說明文件 `man page` 常夾雜許多控制符號，如何將 `man ls` 轉存為一般不含控制符號的文字檔 `ls.txt`？
Sol. `man ls | col -b > ls.txt`
29. 如何將設定檔 `/etc/man.config` 中的 `[tab]` 改成空白，且轉存文字檔 `man.txt`？
Sol. `cat /etc/man.config | col -x > man.txt`

30. 如何用 root 的身份，將 /etc/group 與 /etc/gshadow 相關資料整合成一欄？

Sol. `join -t ':' /etc/group /etc/gshadow`

31. 如何用 root 的身份，將 /etc/passwd 與 /etc/shadow 相關資料整合成一欄？

Sol. `join -t ':' /etc/passwd /etc/shadow`

32. /etc/passwd 第四個欄位是 GID，其記錄在 /etc/group 中的第三個欄位，如何用 root 的身份，將兩個檔案整合？

Sol. `join -t ':' -1 4 /etc/passwd -2 3 /etc/group`

33. 如何用 root 的身份，將 /etc/fstab 與 /etc/mtab 相關資料整合成一欄？

Sol. `join /etc/fstab /etc/mtab`

34. 如何用 root 的身份，將 /etc/passwd 與 /etc/shadow 同一行貼在一起，且中間是以 [tab] 按鍵隔開？

Sol. `paste /etc/passwd /etc/shadow`

35. 如何用 root 的身份，將 /etc/passwd 與 /etc/shadow 同一行貼在一起，且中間是以冒號：隔開？

Sol. `paste -d ':' /etc/passwd /etc/shadow`

36. 如何將 /etc/man.config 內行首為 MANPATH 的字樣取出，且僅取前三行？

Sol. `grep '^MANPATH' /etc/man.config | head -n 3`

37. 如何將 /etc/man.config 內行首為 MANPATH 的字樣取出，且僅取前三行並用 cat 將所有的符號都列出來？

Sol. `grep '^MANPATH' /etc/man.config | head -n 3 | cat -A`

38. 如何將 /etc/man.config 內行首為 MANPATH 的字樣取出，且僅取前三行，並將 [tab] 按鍵設定成 9 個字元，最後用 cat 將所有的符號都列出來？

Sol. `grep '^MANPATH' /etc/man.config | head -n 3 | expand -t 9 -
cat -A`

39. /etc/termcap 有七百多 Mbytes，如何分成 300M 一個檔案？

Sol. `split -b 300m /etc/termcap termcap`

40. /etc/termcap 有七百多 Kbytes，如何分成 300K 一個檔案？

Sol. `split -b 300k /etc/termcap termcap`

41. /etc/termcap 有七百多 Kbytes，如何分成 300K 一個檔案，會產生那幾個檔案？

Sol. `termcapaa, termcapab & termcapac`

42. 有一檔案以 split 分成多個小檔案 termcap??，如何將其合成一個檔案名為 termcap？

Sol. `cat termcap?? >> termcap`

43. 如何將 `ls -al /` 輸出的資訊，每十行記錄成一個檔案 `lsroot`？

Sol. `ls -al / | split -l 10 - lsroot`

44. 如何將 `/etc/passwd` 內的第一欄取出，僅取三行，使用指令 `finger` 將每個帳號內容秀出來？

Sol. `cut -d ':' -f1 < /etc/passwd | head -n 3 | xargs finger`

45. 如何將 `/etc/passwd` 內的第一欄取出，僅取三行，使用指令 `finger` 將每個帳號內容秀出來，且每次執行 `finger` 時，都要詢問使用者是否動作？

Sol. `cut -d ':' -f1 < /etc/passwd | head -n 3 | xargs -p finger`

46. 如何將所有的 `/etc/passwd` 內的第一欄取出，使用指令 `finger` 查閱帳號內容，但一次僅查閱五個帳號，且每次執行 `finger` 時，都要詢問使用者是否動作？

Sol. `cut -d ':' -f1 < /etc/passwd | xargs -p -n 5 finger`

47. 如何將所有的 `/etc/passwd` 內的第一欄取出，使用指令 `finger` 查閱帳號內容，但當分析到 `lp` 就結束這串指令？

Sol. `cut -d ':' -f1 < /etc/passwd | xargs -p -s'lp' finger`

48. 如何將 `/home` 裡面的檔案打包，打包的資料傳送到 `stdout`，將 `/home` 裡面的檔案打包，打包的資料傳送到 `stdout`；再利用管線處理，將 `stdout` 做為後續 `tar -xvf` 的 `stdin` 解壓縮？

Sol. `tar -cvf - /home | tar -xvf -`

3.4 實機練習題

3.4.1 練習一

1. 在家目錄下建立 `zzz` 目錄。
2. 切換工作目錄到 `zzz`。
3. 在此目錄下寫一腳本 `pipe1.sh`，使用 `read` 從鍵盤輸入讀入檔名或目錄。
4. 以 `ls` 命令列出檔名或目錄，標準錯誤導向到 `perror`，標準輸出導向到 `pright`。

3.4.2 練習二

1. 在家目錄下建立 `zzz` 目錄。
2. 切換工作目錄到 `zzz`。
3. 在此目錄下寫一腳本 `pipe2.sh`，使用 `read` 從鍵盤輸入讀入檔名或目錄。
4. 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 `'exist'`，否則輸出 `'not exist'`。

3.4.3 練習三

1. 在家目錄下建立 `zzz` 目錄。
2. 切換工作目錄到 `zzz`。
3. 在此目錄下寫一腳本 `pipe3.sh`，使用 `read` 從鍵盤輸入讀入一指定字串。
4. 執行 `ifconfig` 命令。
5. 管線處理以 `grep` 命令過濾出指定的字串。
6. 管線處理以 `sort` 命令指定分隔符號為 `','`，再以第二欄位排序。
7. 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 4

正規表示法

4.1 前言

- 什麼是正規表示法

1. 處理字串的方法，以行為單位進行字串的處理行為。
2. 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。
3. 延伸的正規表示法：可以作群組的字串『或(or)』及『且(and)』的處理。
4. 正規表示法與萬用字元不一樣，不可混淆。

- 正規表示法用途

1. 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。
2. 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。
3. 軟體中輸入欄位的格式限制，例如：只能為數字。

- vi, grep, awk ,sed 等工具皆支援正規表示法。

練習題

1. 何謂正規表示法？

Sol. 處理字串的方法

2. 正規表示法以什麼為單位進行字串處理？

Sol. 以行為單位

3. 正規表示法與萬用字元是否一樣？

Sol. 不一樣

4.2 基礎正規表示法

● 重要特殊字元(characters)

|| 字符

```

2 | RE 意義與範例
   | ^word 待搜尋的字串 (word)在行首。
   | #範例：grep -n '^#' re.txt搜尋行首為
   | # 開始的那一行。
6 | word$ 待搜尋的字串 (word)在行尾。
   | #範例：grep -n '!' re.txt將行尾為
   | ! 的那一行列印出來。
   | . 代表『任意一個』字符，一定是一個任意字符。
10 | #範例：grep -n 'e.e' re.txt搜尋的字串可以是
   | (eve) (eae) (eee) (e e)，但不能僅有 (ee)。亦即
12 | e 與 e 中間『一定』僅有一個字元，而空白字元也是字元。
   | \ 跳脫字符，將特殊符號的特殊意義去除。
14 | #範例：grep -n \" re.txt搜尋含有單引號
   | ' 的那一行。
16 | * 重複零個或多個的前一個 RE 字符
   | #範例：grep -n 'ess*' re.txt找出含有
18 | (es) (ess) (esss) 等等的字串。
   | \{n,m\} 連續 n 到 m 個的『前一個 RE 字符』若為
20 | \{n\} 則是連續 n 個的前一個 RE 字符，若是
   | \{n,\} 則是連續 n 個以上的前一個 RE 字符。
22 | #範例：grep -n 'go\{2,3\}g' re.txt在
   | g 與 g 之間有 2 個到 3 個的 o 存在的字串，亦
   | 即 (goog)(gooog)
24 | [ ] 在 [ ] 當中『謹代表一個待搜尋的字元』
   | #範例：grep -n 'g[ld]' re.txt搜尋含有
26 | (gl) 或 (gd) 的那一行
   | #範例：grep -n '[0-9]' re.txt搜尋含有任意數字的那一行。在
   | 字元集合
28 |
   | [ ] 中的減號 - 是代表兩個字元之間的所有連續字元。：
30 | [^]~ 在 [ ] 內時，代表的意義是『反向選擇』
   | #範例：grep -n 'oo[^t]' re.txt搜尋的字串可以是
32 | (oog) (ood) 但不能是 (oot)。

```

● 以 grep 擷取字串

1. 編輯 re.txt 純文字檔

```

1 | [root@test root]# vi re.txt
   | "Open Source" is a good mechanism to develop programs.
3 | apple is my favorite food.
   | Football game is not use feet only.
5 | this dress doesn't fit me.

```

```
7  However, this dress is about $ 3183 dollars.
   GNU is free air not free beer.
   Her hair is very beauty.
9  I 'cant finish the test.
   Oh! The soup taste good.
11 motorcycle is cheap than car.
   This window is clear.
13 the symbol '*' is represented as start.
   Oh! My god!
15 The gd software is a library for drafting programs.
   You are the best is mean you are the no. 1.
17 The world is the same with "glad".
   I like dog.
19 google is the best tools for search keyword.
   gooooooogle yes!
21 go! go! Let's go.
   # I am csie
```

2. 搜尋特定字串：從檔案 re.txt 中取得特定字串 the

```
[root@test root]# grep -n 'the' re.txt
2  8:I can't finish the test.
   12:the symbol '*' is represented as start.
4  15:You are the best is mean you are the no. 1.
   16:The world is the same with "glad".
6  18:google is the best tools for search keyword.
```

3. 該行沒有字串 the 時，才顯示在螢幕上：

```
[root@test root]# grep -vn 'the' re.txt
2  # 螢幕上出現的行列為除了 8,12,15,16,18 五行之外的其他行列。
```

4. 取得不論大小寫的字串 the：

```
[root@test root]# grep -in 'the' re.txt
2  8:I can't finish the test.
   9:Oh! The soup taste good.
4  12:the symbol '*' is represented as start.
   14:The gd software is a library for drafting programs.
6  15:You are the best is mean you are the no. 1.
   16:The world is the same with "glad".
8  18:google is the best tools for search keyword.
```

5. 利用 [] 來搜尋集合字元

```
[root@test root]# grep -n 't[ae]st' re.txt
2 8:I can't finish the test.
9:Oh! The soup taste good.
4 # [ ] 裡面不論有幾個字元，都謹代表某『一個』字元。
```

6. 搜尋有 oo 的字元：

```
[root@test root]# grep -n 'oo' re.txt
2 1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
4 3:Football game is not use feet only.
9:Oh! The soup taste good.
6 18:google is the best tools for search keyword.
19:gooooooogle yes!
```

7. 不想要 oo 前面有 g，可以利用在集合字元的反向選擇 [^]

```
1 [root@test root]# grep -n '[^g]oo' re.txt
2:apple is my favorite food.
3 3:Football game is not use feet only.
18:google is the best tools for search keyword.
5 19:gooooooogle yes!
```

8. oo 前面不想要有小寫字元：

```
1 [root@test root]# grep -n '[^a-z]oo' re.txt
3:Football game is not use feet only.
```

9. 要求字串是數字與英文：[a-zA-Z0-9]

```
[root@test root]# grep -n '[0-9]' re.txt
2 5:However, this dress is about $ 3183 dollars.
15:You are the best is mean you are the no. 1.
```

10. 行首與行尾字元 ^ \$：

```
1 [root@test root]# grep -n '^the' re.txt
12:the symbol '*' is represented as start.
3 [root@test root]# grep -n '$[a-z]' re.txt
```

```

5 | 2:apple is my favorite food.
   4:this dress doesn't fit me.
7 | 10:motorcycle is cheap than car.
   12:the symbol '*' is represented as start.
9 | 18:google is the best tools for search keyword.
   19:gooooooogle yes!

```

11. 不要開頭是英文字母：

```

[root@test root]# grep -n '^([a-zA-Z])' re.txt
2 | 1:"Open Source" is a good mechanism to develop programs.
   20:# I am csie

```

12. 行尾結束為小數點 (.) 的那一行：

```

1 | [root@test root]# grep -n '\.$' re.txt
   1:"Open Source" is a good mechanism to develop programs.
3 | 2:apple is my favorite food.
   3:Football game is not use feet only.
5 | 4:this dress doesn't fit me.
   10:motorcycle is cheap than car.
7 | 11:This window is clear.
   12:the symbol '*' is represented as start.
9 | 15:You are the best is mean you are the no. 1.
   16:The world is the same with "glad".
11 | 17:I like dog.
   18:google is the best tools for search keyword.
13 | # 小數點具有其他意義，必須要使用跳脫字元(\)來加以解除其特殊意
    | 義。
    | # 第 5~9 行最後面也是？ .

```

13. Windows 的 notepad 會主動加上 ^M 作為斷行的判斷。

```

[root@test root]# cat -A re.txt
2 | However, this dress is about $ 3183 dollars.^M$

```

14. 找出哪一行是『空白行』？

```

[root@test root]# grep -n '^$' re.txt
2 | 21:

4 | [root@test root]# cat /etc/syslog.conf
   [root@test root]# grep -v '^$' /etc/syslog.conf | grep -v
    '^#'

```

15. 任意一個字元 `.` 與重複字元 `*`

```

1 # 『』代表『絕對有一個任意字元』。
  [root@test root]# grep -n 'g..d' re.txt
3 1:"Open Source" is a good mechanism to develop programs.
  9:Oh! The soup taste good.
5 16:The world is the same with "glad".
  # 『』代表的是『重複* 0 個或多個前面的 RE 字符』
7 [root@test root]# grep -n 'ooo*' re.txt
  1:"Open Source" is a good mechanism to develop programs.
9 2:apple is my favorite food.
  3:Football game is not use feet only.
11 9:Oh! The soup taste good.
  18:google is the best tools for search keyword.
13 19:gooooooogle yes!

```

16. 字串開頭與結尾都是 `g`，但是兩個 `g` 之間僅能存在至少一個 `o`

```

1 # 亦即 gog, goog, gooog.... 等。
  [root@test root]# grep -n 'goo*g' re.txt
3 18:google is the best tools for search keyword.
  19:gooooooogle yes!

```

17. 找出 `g` 開頭與 `g` 結尾的字串，當中的字元可有可無。(錯誤找法)

```

  [root@test root]# grep -n 'g*g' re.txt
2 1:"Open Source" is a good mechanism to develop programs.
  3:Football game is not use feet only.
4 9:Oh! The soup taste good.
  13:Oh! My god!
6 14:The gd software is a library for drafting programs.
  16:The world is the same with "glad".
8 17:I like dog.
  18:google is the best tools for search keyword.
10 19:gooooooogle yes!
  # g*g 的內容是 g, gg, ggg, °gggg

```

18. 找出 `g` 開頭與 `g` 結尾的字串，當中的字元可有可無。

```

1 [root@test root]# grep -n 'g.*g' re.txt
  1:"Open Source" is a good mechanism to develop programs.

```

```
3 14:The gd software is a library for drafting programs.  
18:google is the best tools for search keyword.  
5 19:gooooooogle yes!
```

19. 要找出『任意數字』的行列：

```
1 [root@test root]# grep -n '[0-9][0-9]*' re.txt  
5:However, this dress is about $ 3183 dollars.  
3 15:You are the best is mean you are the no. 1.  
# 使用 grep -n '[0-9]' re.txt 也可以得到相同的結果。
```

20. 限定連續 RE 字符範圍 {}：找到兩個 o 的字串。

```
[root@test root]# grep -n 'o\{2\}' re.txt  
2 1:"Open Source" is a good mechanism to develop programs.  
2:apple is my favorite food.  
4 3:Football game is not use feet only.  
9:Oh! The soup taste good.  
6 18:google is the best tools for search keyword.  
19:gooooooogle yes!
```

21. 找出 g 後面接 2 到 5 個 o，然後再接一個 g 的字串：

```
1 [root@test root]# grep -n 'go\{2,5\}g' re.txt  
18:google is the best tools for search keyword.
```

22. 找出 2 個 o 以上的 goooo....g? 除了可以是 gooo*g，也可以是：

```
[root@test root]# grep -n 'go\{2,\}g' re.txt  
2 18:google is the best tools for search keyword.  
19:gooooooogle yes!
```

23. 找出 2 個 o 以上的 goooo....g，當找到一筆即停止。

```
1 [root@test root]# grep -m 1 'go\{2,\}g' re.txt  
18:google is the best tools for search keyword.  
3 19:gooooooogle yes!
```

- 『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]

1 || 例題：不支援正規表示法的

```
ls 工具中『
3 ls -l * 』：代表列出任意檔名的檔案；『
ls -l a* 』：代表列出以 a 為開頭的任何檔名的檔案。在正規表示法中要找
到含有以
5 a 為開頭的檔案，必須搭配支援正規表示法的工具： grep
ls | grep -n '^a.*'
```

- 例題：以正規表示法顯示行號方式找出目錄 /etc 以下符合下列條件之檔案。

1. 包含 boot 或 root 字串的檔案。
2. 包含 b 開頭 t 結尾之字串的檔案。
3. b 開頭 t 結尾且中間 1 個 o 以上之字串的檔案。
4. 包含 b 開頭 t 結尾且中間 2 至 4 個 o 之字串的檔案。
5. 包含行首為 root 之字串的檔案。
6. 包含行尾為 root 之字串的檔案。

- 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)：

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

練習題

1. 如何以 grep 配合正規表示法，搜尋檔案 re.txt 中，行首為 # 的行，並列出行號？

Sol. `grep -n '^#' re.txt`

2. 如何以 grep 配合正規表示法，搜尋檔案 re.txt 中，行尾為 ! 的行，並列出行號？

Sol. `grep -n '!$' re.txt`

3. 如何以 grep 配合正規表示法，搜尋檔案 re.txt 中，e 與 e 中間『一定』僅有一個字元(含空白字元)的行，並列出行號？

Sol. `grep -n 'e.e' re.txt`

4. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有單引號，的行，並列出行號？
Sol. `grep -n \' re.txt`
5. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `es`, `ess`, `esss`,... 等字串的行，並列出行號？
Sol. `grep -n 'ess*' re.txt`
6. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，在 `g` 與 `g` 之間有 2 個到 3 個 `o` 字串的行，並列出行號？
Sol. `grep -n 'go\{2,3\}g' re.txt`
7. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `gl` 或 `gd` 的行，並列出行號？
Sol. `grep -n 'g[ld]' re.txt`
8. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有任意數字的行，並列出行號？
Sol. `grep -n '[0-9]* re.txt`
9. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `oo` 字串，但不能是 `oot` 的行，並列出行號？
Sol. `grep -n 'ool?*' re.txt`
10. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `the` 字串的行，並列出行號？
Sol. `grep -n 'the' re.txt`
11. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，沒含有 `the` 字串的行，並列出行號？
Sol. `grep -v 'the' re.txt`
12. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有不論大小寫 `the` 字串的行，並列出行號？
Sol. `grep -in 'the' re.txt`
13. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `tast` 或 `test` 字串的行，並列出行號？
Sol. `grep -n 't[ae]st' re.txt`
14. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `oo` 字串，但其前面不是 `g` 的行，並列出行號？
Sol. `grep -n '[^g]oo' re.txt`
15. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `oo` 字串，但其前面不是小寫字元的行，並列出行號？
Sol. `grep -n '[^a-z]oo' re.txt`

16. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有數字或英文字串，但其前面不是小寫字元的行，並列出行號？

Sol. `grep -n '[a-z][a-zA-Z0-9]*' re.txt`

17. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，行首為 `the` 的行，並列出行號？

Sol. `grep -n '^the' re.txt`

18. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，行首開頭為英文字母的行，並列出行號？

Sol. `grep -n '[a-zA-Z]' re.txt`

19. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，行首開頭不要是英文字母的行，並列出行號？

Sol. `grep -n '[^a-zA-Z]' re.txt`

20. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，行尾結束為小數點 `.` 的行，並列出行號？

Sol. `grep -n '\.$' re.txt`

21. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中空白行，並列出行號？

Sol. `grep -n '^$' re.txt`

22. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有以 `g` 開頭 `d` 結尾，且中間剛好兩個字元之字串的行，並列出行號？

Sol. `grep -n 'g..d' re.txt`

23. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有以 `g` 開頭 `d` 結尾之字串，但 `g` 與 `d` 之間僅能存在至少一個 `o` 的行，並列出行號？

Sol. `grep -n 'g[oo]d' re.txt`

24. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有『任意數字』的行，並列出行號？

Sol. `grep -n '[0-9][0-9]*' re.txt`

25. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 2 個 `o` 字串的行，並列出行號？

Sol. `grep -n 'o{1,2}' re.txt`

26. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `g` 後面接 2 到 5 個 `o`，然後再接一個 `g` 之字串的行，並列出行號？

Sol. `grep -n 'go{2,5}g' re.txt`

27. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `g` 後面接 2 個以上 `o`，然後再接一個 `g` 之字串的行，並列出行號？

Sol. `grep -n 'go{2,}g' re.txt`

28. 如何以 `grep` 配合正規表示法，搜尋檔案 `re.txt` 中，含有 `g` 後面接 2 個以上 `o`，然後再接一個 `g` 之字串的行，且找到第二筆即停止搜尋？

Sol. `grep -m 2 'go\{2,\}g' re.txt`

29. `*` 在『萬用字元』與『正規表示法』中有何差異？

Sol. 在萬用字元中為 0 至無限多個字元，而在正規表示法中為重複 0 到多個的前一個 `re` 字符

30. 在『萬用字元』與『正規表示法』中反向選擇之用法有何差異？

Sol. 在萬用字元中為 `[!range]`，而在正規表示法中為 `[^range]`

31. 如何以 `grep` 配合正規表示法，搜尋目錄 `/etc` 中，含有 `boot` 或 `root` 字串的檔案，並列出行號？

Sol. `grep -n '(boot|root)' /etc/*`

32. 如何以 `grep` 配合正規表示法，搜尋目錄 `/etc` 中，含有 `b` 開頭 `t` 結尾之字串的檔案，並列出行號？

Sol. `grep -n 'b.*t' /etc/*`

33. 如何以 `grep` 配合正規表示法，搜尋目錄 `/etc` 中，含有 `b` 開頭 `t` 結尾且中間 1 個 `o` 以上之字串的檔案，並列出行號？

Sol. `grep -n 'bo\{1,\}t' /etc/* && grep -n 'boot*' /etc/*`

34. 如何以 `grep` 配合正規表示法，搜尋目錄 `/etc` 中，含有 `b` 開頭 `t` 結尾且中間 2 至 4 個 `o` 之字串的檔案，並列出行號？

Sol. `grep -n 'bo\{2,4\}t' /etc/*`

35. 如何以 `grep` 配合正規表示法，搜尋目錄 `/etc` 中，含有行首為 `root` 之字串的檔案，並列出行號？

Sol. `grep -n '^root' /etc/*`

36. 如何以 `grep` 配合正規表示法，搜尋目錄 `/etc` 中，含有行尾為 `root` 之字串的檔案，並列出行號？

Sol. `grep -n 'root$' /etc/*`

37. 如何以 `grep` 配合正規表示法，搜尋目錄 `/etc` 中，含有 `!` 與 `>` 的字元的檔案，並列出行號？

Sol. `grep -n '[!>]' /etc/*`

4.3 延伸正規表示法

- `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。
- 延伸型正規表示法之特殊符號

```

1 RE 字符意義與範例
  + 重複『一個或一個以上』的前一個      RE 字符
3     #範例：egrep -n 'go+d' re.txt 搜尋
      (god) (good) (goood)... 等等的字串。
5     ? 『零個或一個』的前一個      RE 字符
      #範例：egrep -n 'go?d' re.txt 搜尋
7     (gd) (god) 這兩個字串。
  | 用或      ( or )的方式找出數個字串
9     #範例：egrep -n 'gd|good|dog' re.txt 搜尋
      gd 、 good 或 dog 這三個字串。
11    ( ) 找出『群組』字串或作為『多個重複群組』的判別
      #範例：egrep -n 'g(la|oo)d' 搜尋
13    (glad) 或 (good) 這兩個字串。
      #範例：egrep -n 'A(xyz)+C' re.txt 找開頭是
15    A 結尾是 C ，中間有一個以上的 "xyz" 字串。

```

- 例題：去除空白行與行首為 # 的行列

```

1 #基礎型的正規表示法需要使用到管線命令來搜尋兩次
  grep -v '^$' re.txt | grep -v '^#'
3 #支援延伸型正規表示法可以簡化為：
  egrep -v '^$|^#' re.txt

```

- 例題：以延伸型正規表示法顯示行號方式，找出目錄 /etc 以下符合下列條件之檔案。

1. 包含 boot 或 root 字串的檔案。
2. 包含 b 開頭 t 結尾且中間 0 個或 1 個 o 之字串的檔案。
3. b 開頭 t 結尾且中間 1 個 o 以上之字串的檔案。

練習題

1. grep 支援的是基礎型的正規表示法，如何使 grep 支援延伸正規表示法？其命令別名為何？

Sol. `grep -E, egrep`

2. 如何以 egrep 配合延伸正規表示法，搜尋檔案 re.txt 中，含有 g 後面接 1 個以上 o，然後再接一個 g 之字串的行，並列出行號？

Sol. `egrep -n 'go/g' re.txt`

3. 如何以 egrep 配合延伸正規表示法，搜尋檔案 re.txt 中，含有 g 開頭 d 結尾且中間 0 至 1 個 o 之字串的行，並列出行號？

Sol. `egrep -n 'go?g' re.txt`

4. 如何以 `egrep` 配合延伸正規表示法，搜尋檔案 `re.txt` 中，含有 `gd`、`good` 或 `dog` 這三個字串的行，並列出行號？
Sol. `egrep -n 'gd|good|dog' re.txt`
5. 如何以 `egrep` 配合延伸正規表示法，搜尋檔案 `re.txt` 中，含有 `glad` 或 `good` 之字串的行，並列出行號？
Sol. `egrep -n 'g(la|oo)d' re.txt`
6. 如何以 `egrep` 配合延伸正規表示法，搜尋檔案 `re.txt` 中，含有開頭是 `A` 結尾是 `C`，中間有一個以上的 `"xyz"` 之字串的行，並列出行號？
Sol. `egrep -n 'A(xyz)+C' re.txt`
7. 如何以 `egrep` 配合延伸正規表示法，搜尋檔案 `re.txt` 中，含有開頭是 `A` 結尾是 `C`，中間 0 至一個 `"xyz"` 之字串的行，並列出行號？
Sol. `egrep -n 'A(xyz)?C' re.txt`
8. 如何以 `egrep` 配合延伸正規表示法，搜尋檔案 `re.txt` 中，不含空白行與行首為 `#` 的行列，並列出行號？
Sol. `egrep -n '^[^#]' re.txt`
9. 如何以 `egrep` 配合延伸正規表示法，搜尋目錄 `/etc` 中，含有 `boot` 或 `root` 之字串的檔案，並列出行號？
Sol. `egrep -n '(b|s)oot' /etc/*`
10. 如何以 `egrep` 配合延伸正規表示法，搜尋目錄 `/etc` 中，含有開頭是 `b` 結尾是 `t`，中間 0 至一個 `o` 之字串的檔案，並列出行號？
Sol. `egrep -n 'bo?t' /etc/*`
11. 如何以 `egrep` 配合延伸正規表示法，搜尋目錄 `/etc` 中，含有開頭是 `b` 結尾是 `t`，中間一個以上 `o` 之字串的檔案，並列出行號？
Sol. `egrep -n 'bott' /etc/*`

4.4 格式化列印

- 列印格式管理員 `printf`

1. 指令用法：

```

1 [root@linux ~]# printf 列印格式，實際內容選項：關於格式方面的
2   幾個特殊樣式：
3
4   \a 警告聲音輸出
5   \b 倒退鍵 (backspace)
6   \f 清除螢幕 (form feed)
7   \n 輸出新的一行
8   \r 亦即 Enter 按鍵
9   \t 水平的 [tab] 按鍵

```

```

10 | \v 垂直的 [tab] 按鍵
    | \xNN NN 為兩位數的數字，可以轉換數字成為字元。關於
12 | C 程式語言內，常見的變數格式
    | %ns 那個 n 是數字，s 代表 string，亦即多少個字元；
14 | %ni 那個 n 是數字，i 代表 integer，亦即多少整數位
    | 數；
    | %N.nf 那個 n 與 N 都是數字，f 代表 floating 浮點()，如
    | 果有小數位數，假設我共要十個位數，但小數點有兩位，即
16 | 為 %10.2f。

```

2. 如何將 printf.txt 內容，以如下漂亮的版面配置輸出？

Name	Chinese	English	Math	Average
Dmdyw	80	60	92	77.33
csie	75	55	80	70.00
Ken	60	90	70	73.33

3. 每個單字中間以 [tab] 按鍵隔開。

```

[root@linux ~]# printf '%s\t %s\t %s\t %s\t %s\t \n' 'cat
printf.txt'
2 Name      Chinese      English      Math      Average
  Dmdyw    80      60      92      77.33
4 csie     75      55      80      70.00
  Ken      60      90      70      73.33
6 # %s 表示以字串 (string) 顯示內容，每個內容以 \t 即 [tab] 隔
  開。
  # 第一行因為某些單字長度較長，所以無法對齊。

```

4. 第二行以後，分別以字串、整數、小數點來顯示

```

1 [root@linux ~]# printf '%10s %5i %5i %5i %8.2f \n' 'cat
  printf.txt |\
> grep -v Name'
3      Dmdyw    80    60    92    77.33
      csie     75    55    80    70.00
5      Ken      60    90    70    73.33
  # %8.2f 針對不同的小數位數來進行格式輸出，
7  # 例題：printf '%10s %5i %5i %5i %8.1f \n' 'cat printf.txt |
    grep -v Name'

```

5. 列出數值 45 代表的字元為何？

```
1 [root@linux ~]# printf '\x45\n'
E
```

- 檔案列印：pr

```
2 # 列印 /etc/man.config
2 [root@linux ~]# pr /etc/man.config
4 2003-02-10 23:20          /etc/man.config
4          Page 1
6 #
6 # Generated automatically from man.conf.in by the
8 # configure script.以下省略
8 .....
10 # 第一行的標題中『檔案時間』、『檔案檔名』及『頁碼』為 pr 處理的結果。
```

練習題

1. 如何以 printf 將檔案 printf.txt 中的資料，以每行為 [tab] 按鍵後接字串之方式列出？
Sol. `printf '\t%s\n' `cat printf.txt``
2. 如何以 printf 將檔案 printf.txt 中的資料，以每行為 [tab] 按鍵後接 10 個字元字串之方式列出？
Sol. `printf '\t%10s\n' `cat printf.txt``
3. 如何以 printf 將檔案 printf.txt 中的資料，以每行為 [tab] 按鍵後接整數之方式列出？
Sol. `printf '\t%i\n' `cat printf.txt``
4. 如何以 printf 將檔案 printf.txt 中的資料，以每行為 [tab] 按鍵後接 7 個位數整數之方式列出？
Sol. `printf '\t%7i\n' `cat printf.txt``
5. 如何以 printf 將檔案 printf.txt 中的資料，以每行為 [tab] 按鍵後接浮點數之方式列出？
Sol. `printf '\t%f\n' `cat printf.txt``
6. 如何以 printf 將檔案 printf.txt 中的資料，以每行為 [tab] 按鍵後接，十個位數，但小數點有兩位的浮點數之方式列出？
Sol. `printf '\t%10.2f\n' `cat printf.txt``
7. 如何以 printf 列出數值 42 代表的字元？
Sol. `printf '\x42\n'`

4.5 實機練習題

4.5.1 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 `re1.txt`
4. 使用 `grep` 對 `re1.txt` 執行以下搜尋動作，不需要列印行號，所有結果依序累加到檔案 `reresult1.txt`，不要做任何的更動。
 - (a) `you` 大小寫不分。
 - (b) `tast` 或 `test`。
 - (c) `oo` 前不是 `g`，也不是 `t`。
 - (d) 剛好三個阿拉伯數字。
 - (e) 兩個以上阿拉伯數字。
 - (f) 行首是大寫英文字母。
 - (g) 行尾不是 `,` `.` 句點。

4.5.2 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 `re2.txt`
4. 使用 `grep` 對 `re2.txt` 執行以下搜尋動作，不需要列印行號，所有結果依序累加到檔案 `reresult2.txt`，不要做任何的更動。
 - (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。
 - (d) `boot` 或 `babt`。
 - (e) 開頭 `b` 結尾 是 `t`，中間 `pqr` 重複 1 次以上。

Chapter 5

sed 與 awk 工具

5.1 sed 工具

- 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)。

1. sed 指令

```

1 [root@linux ~]# sed [-nefr] 動作[]選項：
3 -n : 使用安靜 (silent) 模式。在一般 sed 的用法中，所有來自 STDIN 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。
5 -e : 直接在指令列模式上進行 sed 的動作編輯；
7 -f : -f filename 可以執行 filename 內的 sed 動作；
   -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()
9 -i : 直接修改讀取的檔案內容，而不是由螢幕輸出。動作說明：
11 [n1[,n2]]function
   n1, n2 : 選擇進行動作的行數，例如，『動作行為10,20[]』：
13 function
15 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。
   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。
17 d : 刪除，    d 後面通常不接任何字串；
   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；
19 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。
   s : 取代，    s 的動作可以搭配正規表示法。例如 1,20s/old/new/g。
```

2. 列出 /etc/passwd 的內容，並且列印行號，同時將第 2 5 行刪除。

```

2 [root@linux ~]# nl /etc/passwd | sed '2,5d'
    1 root:x:0:0:root:/root:/bin/bash
    6 sync:x:5:0:sync:/sbin:/bin/sync
    7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown後面省略
    .....().....
6 # sed 後面接的動作，務必以 ' ' 兩個單引號括住。
# 如果只要刪除第 2 行，可以使用 nl /etc/passwd | sed '2d',
8 # 刪除第 3 到最後一行，則是 nl /etc/passwd | sed '3,$d'。

```

3. 在第二行後加上『drink tea』字樣。

```

2 [root@linux ~]# nl /etc/passwd | sed '2a drink tea'
    1 root:x:0:0:root:/root:/bin/bash
    2 bin:x:1:1:bin:/bin:/sbin/nologin
4 drink tea
    3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
6 # 如果要在第二行前? nl /etc/passwd | sed '2i drink tea。'

```

4. 在第二行後面加入兩行字，例如『Drink tea or』『drink beer?』

```

2 [root@linux ~]# nl /etc/passwd | sed '2a Drink tea or .....\'
> drink beer ?'
    1 root:x:0:0:root:/root:/bin/bash
    2 bin:x:1:1:bin:/bin:/sbin/nologin
4 Drink tea or .....
6 drink beer ?
    3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
8 # 可以新增好幾行，但每一行之間都必須要以反斜線 \ 來進行新行的增加。

```

5. 將第 2-5 行的內容取代成為『No 2-5 number』？

```

2 [root@linux ~]# nl /etc/passwd | sed '2,5c No 2-5 number'
    1 root:x:0:0:root:/root:/bin/bash
No 2-5 number
4    6 sync:x:5:0:sync:/sbin:/bin/sync

```

6. 僅列出第 5-7 行

```

2 [root@linux ~]# nl /etc/passwd | sed -n '5,7p'
    5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

```

```

4 | 6 sync:x:5:0:sync:/sbin:/bin/sync
   | 7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
   | # 沒有加 -n 參數時 sed '5,7p'，行會重複輸出。5-7

```

7. 使用 ifconfig 列出 IP，若僅要 eth0 的 IP？

```

1 | [root@linux ~]# ifconfig eth0
   | eth0      Link encap:Ethernet  HWaddr 00:51:FD:52:9A:CA
3 |           inet addr:192.168.1.12  Bcast:192.168.1.255  Mask
   |           :255.255.255.0
   |           inet6 addr: fe80::250:fcff:fe22:9acb/64 Scope:Link
5 |           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
   |           以下省
   |           略
   | .....().....
7 | # 只是 inet addr 那一行，利用... grep 與 sed 來取
   | [root@linux ~]# ifconfig eth0 | grep 'inet ' | sed 's/^.*addr
   | ://g' | \
9 | > sed 's/Bcast.*$//g'

```

8. 將 /etc/man.config 檔案的內容中，有 MAN 的設定取出，且刪除註解說明。

```

1 | [root@linux ~]# cat /etc/man.config | grep 'MAN' | sed 's/^.*$
   | //g' | \
   | > sed '/^$/d'
3 | # 註解以( # 開頭)不一定寫在第一個字元，所以必須使用正規表示
   | 法 #.*$。

```

9. 利用 sed 直接在 ~/.bashrc 最後一行加入 # This is a test

```

1 | [root@linux ~]# sed -i '$a # This is a test' ~/.bashrc
   | # -i 參數可以讓 sed 直接去修改後面接的檔案內容，而不是由螢幕輸
   | 出。
3 | # $a 則代表最後一行才新增。

```

10. 利用 sed 刪除檔案 /tmp/man.config 第10行至最後一行，且將字符串 man 改成 MAN。

```

1 | [root@dywHome2 tmp]# sed -e '10,$d' -e 's/man/MAN/g' man.
   | config
   | #
3 | # Generated automatically from MAN.conf.in by the

```

```

# configure script.
5 #
# MAN.conf from MAN-1.5m
7 #
# For more information about this file, see the MAN pages MAN
# (1)
9 # and MAN.conf(5).
#
11 # 如果不加參數 -，則無法同時完成上述刪除與取代命令。e

```

11. 若將上例刪除與取代命令編輯至檔案 ds.src，如何利用 sed 及 檔案 ds.src，執行同樣的命令？

```

[root@dywHome2 tmp]# vi ds.src
2 [root@dywHome2 tmp]# cat ds.src
10,$d
4 s/man/MAN/g
[root@dywHome2 tmp]# sed -f ds.src man.config
6 #
# Generated automatically from MAN.conf.in by the
8 # configure script.
#
10 # MAN.conf from MAN-1.5m
#
12 # For more information about this file, see the MAN pages MAN
# (1)
# and MAN.conf(5).
14 #

```

12. 關鍵符號 + - * / 的取代

```

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

```

練習題

1. sed 之功能為何？

Sol. 分析 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 STDOUT。

2. 如何以 `nl` 列出 `/etc/passwd` 的內容，同時將第 2 至 5 行刪除？
Sol. `nl /etc/passwd | sed '2,5d'`
3. 如何以 `nl` 列出 `/etc/passwd` 的內容，同時將第 2 行刪除？
Sol. `nl /etc/passwd | sed '2d'`
4. 如何以 `nl` 列出 `/etc/passwd` 的內容，同時將第 5 至 最後一行刪除？
Sol. `nl /etc/passwd | sed '5,$d'`
5. 如何以 `nl` 列出 `/etc/passwd` 的內容，同時在第五行後加上『new line』字樣？
Sol. `nl /etc/passwd | sed '5a new line'`
6. 如何以 `nl` 列出 `/etc/passwd` 的內容，同時在第五行前加上『new line』字樣？
Sol. `nl /etc/passwd | sed '5i new line'`
7. 若要以 `nl` 列出 `/etc/passwd` 的內容，同時在第五行後加上多行字，則行間必須如何區隔？
Sol. 以 `\\` 來進行新行的增加。
8. 如何以 `nl` 列出 `/etc/passwd` 的內容，同時將第 2-5 行的內容取代成為『No 2-5 number』？
Sol. `nl /etc/passwd | sed '2,5c No 2-5 number'`
9. 如何以 `nl` 列出 `/etc/passwd` 的內容，同時僅列出第 5-7 行？
Sol. `nl /etc/passwd | sed -n '5,7p'`
10. 若下指令 `nl /etc/passwd | sed '5,7p'`，則第 5-7 行會如何顯示？
Sol. `sed` 沒有加 `-n` 參數時，5-7 行會重複輸出。
11. 如何使用 `ifconfig` 列出 IP，同時以 `grep` 取出含有 `inet` 的行，再以 `sed` 將其他字串取代為空白字元，而僅留下 `eth0` 的 IP？
Sol. `ifconfig eth0 | grep 'inet ' | sed 's/.*addr://g' | sed 's/bcast.*$/g'`
12. 如何將 `/etc/man.config` 檔案的內容中，有 `MAN` 的設定取出，且刪除註解說明？
Sol. `cat /etc/man.config | grep 'MAN'| sed 's/#.*$/g' | sed '/~/d'`
13. 如何利用 `sed` 直接在 `~/.bashrc` 最後一行加入『# This is a test』？
Sol. `sed -i '$a # This is a test' ~/.bashrc`
14. 利用 `sed` 刪除檔案 `/tmp/man.config` 第10行至最後一行，且將字串 `man` 改成 `MAN`。
Sol. `sed -e '10,$d' -e 's/man/MAN/g' man.config`

15. 檔案 `ds.src` 中有 `sed` 編輯命令，如何利用 `ds.src` 對檔案 `/tmp/man.config`，執行 `sed` 編輯？

Sol. `sed -f ds.src man.config`

5.2 awk 工具

- `awk` 工具：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』。

1. `awk` 指令

```
[root@linux ~]# awk 條件類型動作 '1{1} 條件類型動作2{2} ...' filename
```

2. 取出帳號與登入者的 IP：欄位的分隔符號為空白鍵或 `[tab]` 鍵

```
1 [root@linux ~]# last
csie pts/0 192.168.1.12 Mon Aug 22 09:40 still
logged in
3 root tty1 Mon Aug 15 11:38 -
11:39 (00:01)
reboot system boot 2.6.11 Sun Aug 14 18:18
(7+15:41)
5 csie pts/0 192.168.1.12 Fri Aug 12 12:07 - 12:08
(00:01)
```

3. 取出帳號與登入者的 IP，且帳號與 IP 之間以 `[tab]` 隔開：

```
1 [root@linux ~]# last | awk '{print $1 "\t" $3}'
csie 192.168.1.12
3 root Mon
reboot boot
5 csie 192.168.1.12
# 變數 $0 代表『一整列資料』，第一行的 $0 代表的就是
『csie pts/0....』。
```

- `awk` 的內建變數

1. 變數

變數名稱	代表意義
NF	每一行 (\$0) 擁有的欄位總數
NR	目前 <code>awk</code> 所處理的是『第幾行』資料
FS	目前的分隔字元，預設是空白鍵

2. 承上例：列出每一行的帳號，並列出目前處理的行數及該行有多少欄位。

```
[root@linux ~]# last | awk '{print $1 "\t lines: " NR "\t
      colums: " NF}'
2 csie      lines: 1      colums: 10
root      lines: 2      colums: 9
4 reboot   lines: 3      colums: 9
csie      lines: 4      colums: 10
```

• awk 的邏輯運算

1. 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

2. /etc/passwd 以冒號 ":" 作為欄位的分隔，查閱第三欄小於 10 以下的數據，並且僅列出帳號與第三欄：

```
1 [root@linux ~]# cat /etc/passwd | \
> awk '{FS=":"} $3 < 10 {print $1 "\t " $3}'
3 root:x:0:0:root:/root:/bin/bash
bin      1
5 daemon  2以下省略
.....().....
7 # 因讀入第一行時，變數 $1, $2... 預設還是以空白鍵為分隔。因此，
    第一行沒有正確的顯示。
```

3. 承上例，利用關鍵字 BEGIN 讓第一行正確顯示。

```
1 [root@linux ~]# cat /etc/passwd | \
> awk 'BEGIN {FS=":"} $3 < 10 {print $1 "\t " $3}'
3 root      0
bin      1
5 daemon    2以下省略
.....().....
```

4. 計算每個人薪資總額並且格式化輸出，薪資資料表 pay.txt 如下：

```

2 Name      1st      2nd      3th
  csie    23000    24000    25000
  Wang   21000    20000    23000
4  Lin     43000    42000    41000

6
8 [root@linux ~]# cat pay.txt | \
  > awk 'NR==1{printf "%10s %10s %10s %10s\n", $1,$2,$3,$4,
    "Total" }
10 NR>=2{total = $2 + $3 + $4
    printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total
    }'
12      Name      1st      2nd      3th      Total
    csie    23000    24000    25000    72000.00
    Wang   21000    20000    23000    64000.00
14     Lin     43000    42000    41000    126000.00

16 # 在 {} 內的有多個指令時，可利用分號『;』，或; [Enter] 按鍵來隔
    開指令。
    # 與 bash shell 的變數不同，在 awk 當中，變數可以直接使用，不需
    加上 $ 符號。

```

5. awk 的動作內 { } 支援 if (條件)：

```

1 [root@linux ~]# cat pay.txt | \
  > awk '{if(NR==1) printf "%10s %10s %10s %10s\n", $1,$2,
    $3,$4,"Total"}
3 NR>=2{total = $2 + $3 + $4
  printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total
  }'

```

6. 例題：將如下 printf.txt 檔案內容以 awk 指令輸出，格式如下 printf.txt 輸出格式，其中每個欄位長度為 9 個字元。

```

2 ## printf.txt 內容
  Name Chinese English Math
  Dmdyw 80 60 92
4  csie 75 55 80
  Ken 60 90 70
6 ##printf.輸出格式txt
      Name      Chinese      English      Math      Total
8      Dmdyw      80      60      92      232.00
      csie      75      55      80      210.00
10     Ken      60      90      70      220.00

```


練習題

1. 如何利用 awk 取出帳號與登入者的 IP，且帳號與 IP 之間以 [tab] 隔開？

Sol. `last | awk '{print $1 "\t" $3}'`

2. 如何利用 awk 取出帳號，並列出目前處理的行數及該行有多少欄位，且之間以預設的分隔字元隔開？

Sol. `last | awk '{print $1 FS NR FS NF}'`

3. /etc/passwd 以冒號 ":" 作為欄位的分隔，如何查閱第三欄小於等於 50 以下的數據，並且僅列出帳號與第三欄？

Sol. `cat /etc/passwd | awk 'BEGIN {FS=":"}
$3 <= 50 {print $1 "\t" $3}'`

4. 將如下 printf.txt 檔案內容以 awk 指令輸出，格式如下 printf.txt 輸出格式，其中每個欄位長度為 8 個字元。

printf.txt 內容

Name Chinese English Math

CSIE 80 60 92

CSIE1 75 55 80

Ken 60 90 70

printf.txt 輸出格式

Name	Chinese	English	Math	Total
CSIE	80	60	92	232.00
CSIE1	75	55	80	210.00
Ken	60	90	70	220.00

Sol. `cat printf.txt | awk 'NR==1{printf "%8s %8s %8s %8s
%8s\n", $1, $2, $3, $4, "Total"} NR>=2{total=$2+$3+$4; printf "%8s
%8d %8d %8d %8.2f\n", $1, $2, $3, $4, total}'`

5. 如何輸出 /etc/man.config 至印表機？

Sol. `pr /etc/man.config`

Chapter 6

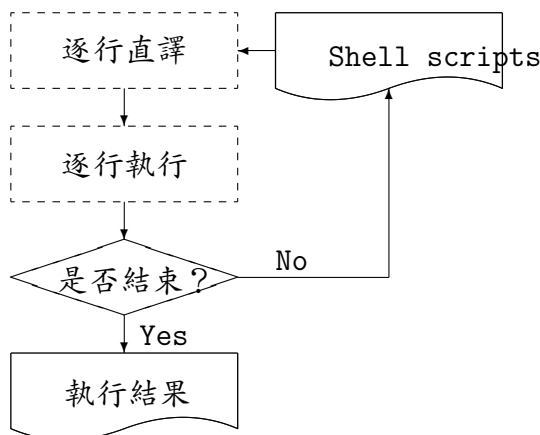
Shell Scripts – 簡介與 test 功能

6.1 前言

- 何謂 Shell Script

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

- Shell scripts 執行流程：



- 為什麼要學習 shell scripts?

1. 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。
2. 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。

3. 簡單入侵偵測功能：主動分析系統登錄檔。
4. 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。
5. 簡易的資料處理：處理數據資料的比對，文字資料的處理等。
6. 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。
7. 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。

- Scripts 撰寫注意事項：

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

- 如何執行檔案 shell.sh ？

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

- 撰寫第一支 script

```
1 [root@linux ~]# mkdir scripts; cd scripts
2 [root@linux scripts]# vi sh01.sh
3 #!/bin/bash
4 # Program:
5 #       This program is used to show "Hello World !" in screen.
6 # History:
7 # 2005/08/23      csie      First release
8
9 echo -e "Hello World ! \a \n"
10 exit 0
```

1. 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』。
2. 主要環境變數的宣告：例如 PATH。
3. 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。
4. 執行成果告知：

- (a) 可以利用指令 `exit` 讓程式中斷，並且回傳一個數值給系統。
- (b) 在此例中使用 `exit 0`，代表離開 `script`，並且回傳 0 給系統。
- (c) 指令執行成功與否，可下達 `echo $?` 觀察回傳值。
- (d) 利用 `exit n` 的功能，可以自訂錯誤訊息。

5. 執行結果

```
[root@linux scripts]# sh sh01.sh  
2 Hello World !
```

- 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`：
 - 1. `script` 的功能；
 - 2. `script` 的版本資訊；
 - 3. `script` 的作者與聯絡方式；
 - 4. `script` 的版權宣告方式；
 - 5. `script` 的 History (歷史紀錄)；
 - 6. `script` 內較特殊的指令，使用絕對路徑的方式來下達；
 - 7. `script` 運作時需要的環境變數預先宣告與設定。

練習題

1. Shell scripts 是二進位檔？
Sol. 不是，是純文字檔
2. Shell scripts 可提供陣列、迴圈、條件與邏輯判斷等重要功能？
Sol. 是
3. Linux 系統的服務是二進位執行檔？
Sol. 不是，是純文字，shell scripts 檔
4. 只要是 Unix-like 系統，都可以執行 Shell scripts？
Sol. 是
5. Shell scripts 中註解的符號為何？
Sol. #
6. 一般 shell scripts 的副檔名為何？
Sol. .sh
7. shell script 中第一行開頭，經常出現 `#!`，代表註解？
Sol. 不是，是用來宣告用什麼程式執行 script

8. shell script 中退出指令為何？

Sol. `exit`

9. shell script 中 `exit 0`，執行結果為何？

Sol. 離開 script，並且回傳 0 給系統。

10. 如何觀察 shell script 執行回傳值？

Sol. `if [$? -eq 0]`

6.2 視 Shell 為一種程式語言

- 撰寫 shell 程式的方法：

1. 互動式 (interactive) 程式：輸入一連串的命令，並允許 shell 立即執行。

```

$ for file in *
2 > do
  > if grep -l POSIX $file
4 > then
  > more $file
6 > fi
  > done
8 posix
  This is a file with POSIX in it - treat it well
10 $
  
```

- (a) 當輸入 shell 命令後，shell 命令提示符號 `$` 就變成 `>`。
 - (b) 可以不斷輸入直到結尾，再讓 shell 立刻執行這個 script。
 - (c) 迴圈變數可以使用 `i`，但 `file` 是比較有意義的變數名稱。
 - (d) 萬用字元 `"*"`，用來代表任何吻合的字串。
 - (e) `grep -l POSIX`：表示若檔案中包含字串 `POSIX` 則顯示檔名，而不顯示內容。
 - (f) 最後兩行為執行的結果。
2. 將命令存成程式檔案。使用文字編輯器產生一個 script，檔名 `sh02.sh`：

```

#!/bin/sh
2 # first
  # This file looks through all the files in the current
4 # directory for the string POSIX, and then prints the names
  of
  # those files to the standard output.
6 for file in *
  do
  
```

```

8   if grep -q POSIX $file
   then
10      echo $file
   fi
12 done
   exit 0

```

1. 可以使用 `.sh` 或隨意加上延伸檔名，`shell` 不會在乎。
2. 可以利用 `file` 命令來判斷是否為 `script`，例如 `file sh02.sh` 或 `file /bin/bash`。

```

1 [csie@localhost csie]$ file sh02.sh
sh02.sh: Bourne shell script text executable
3 [csie@localhost csie]$ file /bin/bash
/bin/bash: ELF 32-bit LSB executable, Intel 80386,
5 version 1 (SYSV), for GNU/Linux 2.2.5,
dynamically linked (uses shared libs), stripped

```

● 執行 script

1. 呼叫 `shell`，將 `script` 當成參數 (parameter)：

```
$ /bin/sh sh02.sh
```

2. 輸入 `script` 名稱直接執行

```

1 $ chmod +x sh02.sh
  $ sh02.sh
3  bash: sh02.sh: command not found
  $ ./sh02.sh

```

練習題

1. 請利用第二至四章之 `shell` 功能，撰寫一 `shell script` (每一指令或功能皆註明其在講義之頁碼)？
至少包含註解(姓名、學號、程式功能、日期、版本)、`sed`、`awk`、變數使用、管線處理、正規表示法。

Sol.

2. `shell script` 中，讀到什麼符號，就開始執行該行命令？

Sol. `[Enter]` 符號

3. shell script 中，一行的內容太多，要如何延伸至下一行？

Sol. 使用 \[Enter]

4. shell script 中，如何加入註解？

Sol. # 後面的字，全部被視為註解

5. 如何執行 shell script 檔案 shell.sh ？

Sol. 1. 將 shell.sh 加上可讀與執行 (rx) 的權限，就能夠以 ./shell.sh 執行； 2. 直接以 sh shell.sh 方式執行。

6. shell script 中，第一行出現 #!/bin/bash ，代表意義為何？

Sol. 宣告使用 /bin/bash 執行這個 script

7. shell script 中，最後出現 exit 0 ，代表意義為何？

Sol. 離開 script，並且回傳 0 給系統

8. 撰寫 shell script 有那兩種方式？

Sol. 1. 輸入一連串的命令後立即執行；2. 將命令存在一個檔案中，隨後當成一個程式來執行。

6.3 shell script 練習

- 變數內容由使用者決定：請以 read 指令，撰寫一個 script 讓使用者輸入：1. first name 與 2. last name，最後在螢幕上顯示：『Your full name is: 』的內容：

```
[root@linux scripts]# vi sh03.sh
2 #!/bin/bash
  # Program:
4 #     Let user keyin their first and last name, and show their
    full name.
  # History:
6 # 2005/08/23    csie    First release

8 read -p "Please input your first name: " firstname
read -p "Please input your last name: " lastname
10 # read -p 後面接提示字元
  echo -e "\nYour full name is: $firstname $lastname"
12 # echo -e enable interpretation of backslash escapes解釋( \n 為 new line)
```

- 利用 date 進行檔案的建立：建立三個空的檔案，檔名最開頭由使用者輸入決定，後接前天、昨天、今天日期。例如：若使用者輸入 filename，今天日期是 2005/08/23，則建立的檔案為 filename20050821, filename20050822, filename20050823。


```

[root@linux scripts]# vi sh04.sh
2  #!/bin/bash
   # Program:
4  #       User can keyin filename to touch 3 new files.
   # History:
6  # 2005/08/23    csie    First release

8  # 1. 讓使用者輸入檔案名稱，並取得 fileuser 這個變數；
   echo -e "I will use 'touch' command to create 3 files."
10 read -p "Please input the filename what you want: " fileuser

12 # 2. 爲了避免使用者隨意按 Enter，利用變數功能分析檔名是否有設定？
   filename=${fileuser:-"filename"}
14
16 # 3. 開始利用 date 指令來取得所需要的檔名了；
   date1='date --date='2 days ago' +%Y%m%d'
   date2='date --date='1 days ago' +%Y%m%d'
18 date3='date +%Y%m%d'
   file1="$filename"$date1
20 file2="$filename"$date2
   file3="$filename"$date3
22
24 # 4. 將檔名建立
   touch $file1
   touch $file2
26 touch $file3

```

- 數值運算的方法：使用者輸入兩個變數，將兩個變數的內容相乘，並輸出相乘的結果。

```

[root@linux scripts]# vi sh05.sh
2  #!/bin/bash
   # Program:
4  #       User can input 2 integer to cross by!
   # History:
6  # 2005/08/23    csie    First release

8  echo -e "You SHOULD input 2 number, I will cross they! \n"
   read -p "first number: " firstnu
10 read -p "second number: " secnu
   total=$((firstnu*secnu))
12 echo -e "\nThe number $firstnu x $secnu is ==> $total"

```

- 數值運算有：+，-，*，/，% 等。其中 % 是取餘數 13 對 3 取餘數，例如：

```
2 [root@linux scripts]# nu=$((13%3)); echo $nu
1
```

練習題

1. 撰寫 shell script，如何以 read 指令，讓使用者輸入變數 var 的內容，且提示 Please input var: ?
Sol. `read -p "Please input var: " var`
2. 撰寫 shell script，如何以 echo 指令，先換行後讓變數 var 的內容顯示在螢幕上，且提示 var= ?
Sol. `echo -e "\nvar= " var`
3. 撰寫 shell script，如何讓變數 date1 的內容為昨日，且格式為 yyymmdd ?
Sol. `date1='date --date='1 days ago' +%Y/%m/%d'`
4. 撰寫 shell script，如何讓變數 nu 的內容為變數 nu1 與 nu2 相乘的結果？
Sol. `nu=$((nu1*nu2))`
5. 撰寫 shell script，如何讓變數 nu 的內容為變數 nu1 除以 nu2 的結果？
Sol. `nu=$((nu1/nu2))`
6. 撰寫 shell script，如何讓變數 nu 的內容為變數 nu1 除 nu2 取餘數的結果？
Sol. `nu=$((nu1%nu2))`

6.4 善用判斷式

- 利用 test 指令的測試功能

1. 指令 test 測試的標誌

測試的標誌	代表意義
1.	關於某個檔名的『類型』偵測(存在與否)，如 <code>test -e filename</code>
-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？

2.	關於檔案的權限偵測，如 <code>test -r filename</code>
-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』？
3.	兩個檔案之間的比較，如： <code>test file1 -nt file2</code>
-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。
4.	關於兩個整數之間的判定，例如 <code>test n1 -eq n2</code>
-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)
5.	判定字串的資料
<code>test -z string</code>	判定字串是否為 0？若 string 為空字串，則為 true
<code>test -n string</code>	判定字串是否非為 0？若 string 為空字串，則為 false。
<code>test str1 = str2</code>	判定 str1 是否等於 str2，若相等回傳 true
<code>test str1 != str2</code>	判定 str1 是否不等於 str2，若相等回傳 false
6.	多重條件判定，例如： <code>test -r filename -a -x filename</code>
-a	(and)兩狀況同時成立。例如 <code>test -r file -a -x file</code> ，則 file 同時具有 r 與 x 權限時，才回傳 true。
-o	(or)兩狀況任何一個成立。例如 <code>test -r file -o -x file</code> ，則 file 具有 r 或 x 權限時，就可回傳 true。
!	反相狀態，如 <code>test ! -x file</code> ，當 file 不具有 x 時，回傳 true

2. 例題：檢查 /csie 是否存在：

```
[root@linux ~]# test -e /csie && echo "exist" || echo "Not exist"
```

3. 例題：讓使用者輸入一個檔名，並判斷其是否存在：

- 若不存在則給予一個『Filename does not exist』的訊息，並中斷程式；
- 若存在，則判斷是檔案或目錄，並果輸出『Filename is regular file』或『Filename is directory』；
- 判斷執行者對該檔案或目錄擁有的權限，並輸出權限資料。

```

1 [root@linux scripts]# vi sh06.sh
2 #!/bin/bash
3 # Program:
4 #     Let user input a filename, the program will search
5 #     the filename
6 #     1.) exist? 2.) file/directory? 3.) file permissions
7 # History:
8 # 2005/08/25    csie    First release
9
10 # 1. 讓使用者輸入檔名，並且判斷使用者是否真的有輸入字串？
11 echo -e "The program will show you that filename
12 is exist which input by you.\n\n"
13 read -p "Input a filename : " filename
14 test -z $filename && echo "You MUST input a filename." &&
15     exit 0
16 # 2. 判斷檔案是否存在？
17 test ! -e $filename && echo "The filename $filename DO NOT
18     exist" && exit 0
19 # 3. 開始判斷檔案類型與屬性
20 test -f $filename && filetype="regular file"
21 test -d $filename && filetype="directory"
22 test -r $filename && perm="readable"
23 test -w $filename && perm="$perm writable"
24 test -x $filename && perm="$perm executable"
25 # 4. 開始輸出資訊
26 echo "The filename: $filename is a $filetype"
27 echo "And the permission are : $perm"

```

- 利用判斷符號 []

1. 判斷變數 \$HOME 是否為空的：

```
[root@linux ~]# [ -z "$HOME" ]
```

2. 在中括號 [] 內的每個元件都需要有空白鍵來分隔；

```

1 # 假設空白鍵使用『□』來表示，則：
2 [ "$HOME" == "$MAIL" ]□
3 ["$HOME"□□=="$MAIL"□]↑↑↑↑

```

3. 在中括號內，變數要以雙引號來設定；常數要以單或雙引號來設定。

```

#設定 name="csie dyw" :
2 [root@linux ~]# name="csie dyw"
[root@linux ~]# [ $name == "csie" ]

```

```

4 | bash: [: too many arguments
6 | # 因為 $name 沒有使用雙引號括起來，判定式會變成：
   | [ csie dyw == "csie" ]
8 | # 而不是：
   | [ "csie dyw" == "csie" ]

```

4. 例題：使用中括號 []、&& 與 || 讓使用者選擇 Y 或 N ，
- (a) 如果使用者輸入 Y 或 y 時，就顯示『 OK, continue 』
 - (b) 如果使用者輸入 n 或 N 時，就顯示『 Oh, interrupt 。』
 - (c) 如果不是 Y/y/N/n 之內的其他字元，就顯示『I don't know what is your choice』

```

1 | [root@linux scripts]# vi sh07.sh
   | #!/bin/bash
3 | # Program:
   | #     This program will show the user's choice
5 | # History:
   | # 2005/08/25    csie    First release
7 |
   | read -p "Please input (Y/N): " yn
9 | [ "$yn" == "Y" -o "$yn" == "y" ] && echo "OK, continue" &&
   |     exit 0
   | [ "$yn" == "N" -o "$yn" == "n" ] && echo "Oh, interrupt!" &&
   |     exit 0
11 | echo "I don't know what is your choice" && exit 0

```

- Shell script 的預設變數 (\$0, \$1, ...)

1. 變數的對應：

```

1 | /path/to/scriptname  opt1  opt2  opt3  opt4  ...
   |                   $0      $1      $2      $3      $4      ...
3 |
   | [root@linux ~]# /etc/init.d/crond restart
5 | ## $0 為 /etc/init.d/，crond $1 為 restart

```

2. 例題：執行一個 script ，自動列出自己的檔名，後面接前三個參數：

```

1 | [root@linux scripts]# vi sh08.sh
   | #!/bin/bash
3 | # Program:
   | #     The program will show it's name and first 3
   |     parameters.

```

```
5 | # History:
   | # 2005/08/25    csie    First release
7 |
   | echo "The script name is ==> $0"
9 | [ -n "$1" ] && echo "The 1st paramter is ==> $1" || exit 0
   | [ -n "$2" ] && echo "The 2nd paramter is ==> $2" || exit 0
11| [ -n "$3" ] && echo "The 3th paramter is ==> $3" || exit 0
   |
13| # 執行結果：
   | [root@linux scripts]# sh sh08.sh theone haha quot
15| The script name is ==> sh08.sh
   | The 1st paramter is ==> theone
17| The 2nd paramter is ==> haha
   | The 3th paramter is ==> quot
```

練習題

1. 如何利用 test 指令檢查檔名 filename 是否存在？
Sol. `test -e filename`
2. 如何利用 test 指令檢查檔名 filename 是否為檔案？
Sol. `test -f filename`
3. 如何利用 test 指令檢查檔名 filename 是否為目錄？
Sol. `test -d filename`
4. 如何利用 test 指令檢查檔名 filename 是否為連結檔？
Sol. `test -L filename`
5. 如何利用 test 指令檢查檔名 filename 是否具有『可讀』的屬性？
Sol. `test -r filename`
6. 如何利用 test 指令檢查檔名 filename 是否具有『可寫』的屬性？
Sol. `test -w filename`
7. 如何利用 test 指令檢查檔名 filename 是否具有『可執行』的屬性？
Sol. `test -x filename`
8. 如何利用 test 指令比較檔案 file1 是否比 file2 新？
Sol. `test file1 -nt file2`
9. 如何利用 test 指令比較檔案 file1 是否比 file2 舊？
Sol. `test file1 -ot file2`
10. 如何利用 test 指令判斷 file1 與 file2 是否為同一檔案？
Sol. `test file1 -of file2`
11. 如何利用 test 指令比較兩個整數 n1 與 n2 是否相等？
Sol. `test n1 -eq n2`

12. 如何利用 `test` 指令比較兩個整數 `n1` 與 `n2` 是否不相等？
Sol. `test n1 -ne n2`
13. 如何利用 `test` 指令比較整數 `n1` 是否大於 `n2`？
Sol. `test n1 -gt n2`
14. 如何利用 `test` 指令比較整數 `n1` 是否小於 `n2`？
Sol. `test n1 -lt n2`
15. 如何利用 `test` 指令比較整數 `n1` 是否大於等於 `n2`？
Sol. `test n1 -ge n2`
16. 如何利用 `test` 指令判定字串 `str1` 是否為空字串？
Sol. `test -z str1`
17. 如何利用 `test` 指令判定字串 `str1` 是否非為空字串？
Sol. `test -n str1 && test str1`
18. 如何利用 `test` 指令判定字串 `str1` 是否等於 `str2`？
Sol. `test str1 = str2`
19. 如何利用 `test` 指令判定字串 `str1` 是否不等於 `str2`？
Sol. `test str1 != str2`
20. 如何利用 `test` 指令判斷檔案 `file1` 是否為具有『可讀』且『可執行』的屬性？
Sol. `test -r file1 -a -x file1`
21. 如何利用 `test` 指令判斷檔案 `file1` 是否為具有『可讀』或『可執行』的屬性？
Sol. `test -r file1 -o -x file1`
22. 如何利用 `test` 指令判斷檔案 `file1` 是否為不具有『可執行』的屬性？
Sol. `test ! -x file1`
23. 如何利用判斷符號 `[]`，判斷變數 `$HOME` 是否為空的？
Sol. `[-z "$HOME"]`
24. 指令 `["$HOME"=="$MAIL"]`，判斷變數 `$HOME` 是否等於變數 `$MAIL`，是否有錯，若有，請說明。
Sol. 有錯，在中括號 `[]` 內的每個元件都需要有空白鍵來分隔。
25. 指令 `[$name == "csie"]`，判斷變數 `$name` 是否等於字串 `csie`，是否有錯，若有，請說明。
Sol. 有錯，在中括號 `[]` 內的 `$name` 沒有使用雙引號引起來。
26. 執行 shell script 『sh sh07.sh theone haha quot』，則預設變數 `$0`，`$1`，`$2`，`$3` 分別為何？
Sol. `$0="sh07.sh"`，`$1="theone"`，`$2="haha"`，`$3="quot"`

Chapter 7

Shell scripts – 條件判斷與迴圈

7.1 條件判斷式

- `if then`：當符合條件判斷時，就進行某項工作。

1. 語法一：

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

2. 條件判斷式的判斷方法：

```
1  && 代表； AND
   || 代表； OR
3
5  [root@linux scripts]# vi sh06-2.sh
6  #!/bin/bash
7  # Program:
8  #       This program will show the user's choice
9  # History:
10 # 2005/08/25   csie   First release
11 read -p "Please input (Y/N): " yn
12
13 if [ "$yn" == "Y" ] || [ "$yn" == "y" ]; then
14     echo "OK, continue"
15     exit 0
16 fi
17 if [ "$yn" == "N" ] || [ "$yn" == "n" ]; then
18     echo "Oh, interrupt!"
19     exit 0
20 fi
21 echo "I don't know what is your choise" && exit 0
```

3. 語法二：

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

4. 語法三：

```
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
```

5. 例題：改寫 sh06-2.sh：

```
1 | [root@linux scripts]# vi sh06-3.sh  
   | #!/bin/bash  
3 | # Program:  
   | #     This program will show the user's choice  
5 | # History:  
   | # 2005/08/25    csie    First release  
7 |  
   | read -p "Please input (Y/N): " yn  
9 |  
11 | if [ "$yn" == "Y" ] || [ "$yn" == "y" ]; then  
   |     echo "OK, continue"  
13 | elif [ "$yn" == "N" ] || [ "$yn" == "n" ]; then  
   |     echo "Oh, interrupt!"  
   | else  
15 |     echo "I don't know what is your choise"  
   | fi
```

6. 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。

```

[root@linux scripts]# vi sh08.sh
2 #!/bin/bash
  # Program:
4  #       Show "Hello" from $1....
  # History:
6  # 2005/08/28   csie   First release

8  if [ "$1" == "hello" ]; then
      echo "Hello, how are you ?"
10 elif [ "$1" == "" ]; then
      echo "You MUST input parameters, ex> $0 someword"
12 else
      echo "The only parameter is 'hello'"
14 fi

```

7. 利用『`netstat -tuln`』可取得目前主機啟動的服務，其中 Local Address 冒號(：)後為 port。

```

[root@linux ~]# netstat -tuln
2 Active Internet connections (only servers)
  Proto Recv-Q Send-Q Local Address   Foreign Address  State
4  tcp        0      0 0.0.0.0:199      0.0.0.0:*        LISTEN
  tcp        0      0 :::80           :::*             LISTEN
6  tcp        0      0 :::22           :::*             LISTEN
  tcp        0      0 :::25           :::*             LISTEN
8
  # 常見的 port 與相關網路服務的關係：
10 80: WWW
   22: ssh
12 21: ftp
   25: mail
14
  # netstat [address_family_options] [--tcp|-t] [--udp|-u]
16 [--raw|-w]      [--listening|-l] [--all|-a] [--numeric|-n]
   [--numeric-hosts] [--numeric-ports] [--numeric-ports]
18 [--symbolic|-N]  [--extend|-e] [--extend|-e]  [--timers|-o]
   [--program|-p]  [--verbose|-v]  [--continuous|-c] [delay]

```

8. 例題：如何透過 `netstat` 偵測主機是否有開啓 80: WWW, 22: ssh, 21: ftp, 25: mail 四個主要的網路服務埠口？

```

1 [root@linux scripts]# vi sh09.sh

```

```

3  #!/bin/bash
# Program:
#     Using netstat and grep to detect WWW,SSH,FTP and Mail
#     services.
5  # History:
# 2005/08/28    csie    First release
7
# 1. 先作一些告知的動作
9  echo "Now, the services of your Linux system will be detect!"
echo -e "The www, ftp, ssh, and mail will be detect! \n"
11
# 2. 開始進行一些測試的工作，並且也輸出一些資訊。
13 testing='netstat -tuln | grep ":80 "'
if [ "$testing" != "" ]; then
15     echo "WWW is running in your system."
fi
17 testing='netstat -tuln | grep ":22 "'
if [ "$testing" != "" ]; then
19     echo "SSH is running in your system."
fi
21 testing='netstat -tuln | grep ":21 "'
if [ "$testing" != "" ]; then
23     echo "FTP is running in your system."
fi
25 testing='netstat -tuln | grep ":25 "'
if [ "$testing" != "" ]; then
27     echo "Mail is running in your system."
fi

```

9. 例題：計算畢業日期還剩幾天？

- (a) 先讓使用者輸入他們的畢業日期；
- (b) 再由現在日期比對畢業日期；
- (c) 由兩個日期的比較來顯示『還需要幾天』才能夠畢業的字樣。

```

[root@linux scripts]# vi sh10.sh
2  #!/bin/bash
# Program:
4  #     Tring to calculate your graduation date at how many
#     days
#     later...
6  # History:
# 2005/08/29    csie    First release
8
# 1. 告知使用者這支程式的用途，並且告知應該如何輸入日期格式？
10 echo "This program will try to calculate :"
echo "How many days about your graduation date..."
12 read -p "Please input your graduation date (YYYYMMDD ex
    >20050401): " date2

```

```

14 # 2. 測試輸入的內容是否正確？利用正規表示法
    date_d='echo $date2 |grep '[0-9]\{8\}''
16 if [ "$date_d" == "" ]; then
    echo "You input the wrong format of date..."
18     exit 1
    fi
20
    # 3. 開始計算日期
22 declare -i date_dem='date --date="$date2" +%s'
    declare -i date_now='date +%s'
24 declare -i date_total_s=$((date_dem-date_now))
    declare -i date_d=$((date_total_s/60/60/24))
26 if [ "$date_total_s" -lt "0" ]; then
    echo "You had been graduation before: " $((-1*$date_d
    )) " ago"
28 else
    declare -i date_h=$((($date_total_s-$date_d
    *60*60*24)/60/60))
30     echo "You will be demobilized after $date_d days and
    $date_h hours."
    fi

```

- case esac 判斷

1. 語法：

```

1 case 變數名稱$ in
    "第一個變數內容")程式段
3
    ;; #兩個分號## (;;) 來代表該程式段落的結束
5 "第二個變數內容")程式段
7
    ;;
    *)不包含第一個變數內容與第二個變數內容的其他程式執行段
9
    exit 1
11    ;;
    esac

```

2. 改寫 sh08.sh

```

[root@linux scripts]# vi sh08-2.sh
2 #!/bin/bash
  # Program:
4 #       Show "Hello" from $1.... by using case .... esac
  # History:

```

```
6 # 2005/08/29      csie   First release
8 case $1 in
9     "hello")
10         echo "Hello, how are you ?"
11         ;;
12     "")
13         echo "You MUST input parameters, ex> $0 someword"
14         ;;
15     *)
16         echo "Usage $0 {hello}"
17         ;;
18 esac
```

3. 例題：讓使用者能夠輸入 one, two, three，並且將使用者的變數顯示到螢幕上，如果不是 one, two, three 時，就告知使用者僅有這三種選擇。

```
[root@linux scripts]# vi sh11.sh
2 #!/bin/bash
3 # Program:
4 #     Let user input one, two, three and show in screen.
5 # History:
6 # 2005/08/29      csie   First release
8 echo "This program will print your selection !"
9 # read -p "Input your choice: " choice
10 # case $choice in
11 case $1 in
12     "one")
13         echo "Your choice is ONE"
14         ;;
15     "two")
16         echo "Your choice is TWO"
17         ;;
18     "three")
19         echo "Your choice is THREE"
20         ;;
21     *)
22         echo "Usage {one|two|three}"
23         ;;
24 esac
```

- 利用 function 功能

1. 語法：

```
function fname() {程式段  
2  
}
```

2. 在 shell script 中，function 的設定一定要在程式的最前面，才能在執行時被找到可用的程式。改寫 sh11.sh：

```
1 [root@linux scripts]# vi sh11-2.sh  
#!/bin/bash  
3 # Program:  
# Let user input one, two, three and show in screen.  
5 # History:  
# 2005/08/29 csie First release  
7  
function printit(){  
9     echo -n "Your choice is "  
}  
11  
echo "This program will print your selection !"  
13 case $1 in  
    "one")  
15     printit; echo $1 | tr 'a-z' 'A-Z'  
    ;;  
17     "two")  
    printit; echo $1 | tr 'a-z' 'A-Z'  
19     ;;  
    "three")  
21     printit; echo $1 | tr 'a-z' 'A-Z'  
    ;;  
23     *)  
    echo "Usage {one|two|three}"  
25     ;;  
esac
```

3. function 也擁有內建變數：\$0 代表函式名稱，後續接的變數以 \$1, \$2,... 取代。改寫上例：

```
[root@linux scripts]# vi sh11-3.sh  
2 #!/bin/bash  
# Program:  
4 # Let user input one, two, three and show in screen.  
# History:  
6 # 2005/08/29 csie First release  
8 function printit(){  
    echo "Your choice is $1"
```

```
10 }  
12 echo "This program will print your selection !"  
13 case $1 in  
14     "one")  
15         printit 1  
16         ;;  
17     "two")  
18         printit 2  
19         ;;  
20     "three")  
21         printit 3  
22         ;;  
23     *)  
24         echo "Usage {one|two|three}"  
25         ;;  
26 esac
```

練習題

1. 使用 if 語法，撰寫一 shell script，偵測輸入的參數是否為 hello。如果是，顯示 "Hello, how are you ?"；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；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。

Sol.

```
if [ "$1" == "hello" ]; then  
    echo "Hello, how are you ?"  
elif [ "$1" == "" ]; then  
    echo "You MUST input parameters, ex> $0 someword"  
else  
    echo "The only parameter is 'hello'"  
fi
```


2. 使用 case 語法，撰寫一 shell script，偵測輸入的參數是否為 hello。如果是，顯示 "Hello, how are you ?"；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；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。

Sol.

```
case $1 in
    "hello")
        echo "Hello, how are you ?"
        ;;
    "")
        echo "You MUST input parameters, ex> $0 someword"
        ;;
    *)
        echo "Usage $0 {hello}"
        ;;
esac
```

3. 撰寫一 shell script，透過 netstat 偵測主機是否有開啓 ftp 網路服務，其埠口為 21。

Sol.

```
testing='netstat -tuln | grep ":21 "'
if [ "$testing" != "" ]; then
    echo "FTP is running in your system."
fi
```

4. 撰寫一 shell script，讓使用者輸入畢業日期；再由現在日期比對畢業日期；由兩個日期的比較來顯示『還需要幾天』才能夠畢業的字樣。

Sol.

```
read -p "Please input your graduation date (YYYYMMDD ex>20050401): " date2

date_d='echo $date2 |grep '[0-9]\{8\}''
if [ "$date_d" == "" ]; then
    echo "You input the wrong format of date...."
    exit 1
fi

declare -i date_dem='date --date="$date2" +%s'
declare -i date_now='date +%s'
declare -i date_total_s=$((date_dem-date_now))
declare -i date_d=$((date_total_s/60/60/24))
if [ "$date_total_s" -lt "0" ]; then
    echo "You had been graduation before: " $((-1*$date_d)) " ago"
else
    declare -i date_h=$((($date_total_s-$date_d*60*60*24)/60/60)
    echo "You will be demobilized after $date_d days and $date_h hours."
fi
```

5. 使用 `case` 語法，撰寫一 shell script，讓使用者能夠輸入 `one`, `two`, `three`，並且將使用者的變數顯示到螢幕上，如果不是 `one`, `two`, `three` 時，就告知使用者僅有這三種選擇。

Sol.

```
case $1 in
    "one")
        echo "Your choice is ONE"
        ;;
    "two")
        echo "Your choice is TWO"
        ;;
    "three")
        echo "Your choice is THREE"
        ;;
    *)
        echo "Usage {one|two|three}"
        ;;
esac
```

6. 使用 `function` 功能及 `case` 語法，撰寫一 shell script，讓使用者能夠輸入 `one`, `two`, `three`，並且將使用者的變數顯示到螢幕上，如果不是 `one`, `two`, `three` 時，就告知使用者僅有這三種選擇。

Sol.

```
function printit(){
    echo -n "Your choice is "
}
case $1 in
    "one")
        printit; echo $1 | tr 'a-z' 'A-Z'
        ;;
    "two")
        printit; echo $1 | tr 'a-z' 'A-Z'
        ;;
    "three")
        printit; echo $1 | tr 'a-z' 'A-Z'
        ;;
    *)
        echo "Usage {one|two|three}"
        ;;
esac
```

7. 使用 `function` 與 變數 `$1` 功能及 `case` 語法，撰寫一 `shell script`，讓使用者能夠輸入 `one`, `two`, `three`，並且將使用者的變數顯示到螢幕上，如果不是 `one`, `two`, `three` 時，就告知使用者僅有這三種選擇。

Sol.

```
function printit(){
    echo "Your choice is $1"
}
case $1 in
    "one")
        printit 1
        ;;
    "two")
        printit 2
        ;;
    "three")
        printit 3
        ;;
    *)
        echo "Usage {one|two|three}"
        ;;
esac
```

7.2 迴圈 (loop)

- `while....do....done`, `until....do....done`

1. 語法一：當 `condition` 條件成立時，進行迴圈，直到 `condition` 條件不成立才停止。

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

2. 語法二：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
```

3. 例題：使用 `while` 語法，讓使用者輸入 `yes` 或者是 `YES` 才結束程式的執行，否則就一直進行告知使用者輸入字串。

```
[root@linux scripts]# vi sh12.sh
2  #!/bin/bash
   # Program:
4  #     Use loop to try find your input.
   # History:
6  # 2005/08/29   csie   First release

8  while [ "$yn" != "yes" ] && [ "$yn" != "YES" ]
do
10         read -p "Please input yes/YES to stop this program: "
           yn
done
```

4. 例題：使用 `until` 語法，讓使用者輸入 `yes` 或者是 `YES` 才結束程式的執行，否則就一直進行告知使用者輸入字串。

```
1  [root@linux scripts]# vi sh12-2.sh
   #!/bin/bash
3  # Program:
   #     Use loop to try find your input.
5  # History:
   # 2005/08/29   csie   First release

7  until [ "$yn" == "yes" ] || [ "$yn" == "YES" ]
9  do
           read -p "Please input yes/YES to stop this program: "
           yn
11 done
```

5. 例題：計算 $1+2+3+\dots+100$ ：

```
1  [root@linux scripts]# vi sh13.sh
   #!/bin/bash
3  # Program:
   #     Try to use loop to calculate the result
   #     "1+2+3...+100"
5  # History:
   # 2005/08/29   csie   First release

7
s=0
9 i=0
while [ "$i" != "100" ]
11 do
           i=$((i+1))
```

```

13         s=$((s+i))
14     done
15     echo "The result of '1+2+3+...+100' is ==> $s"

```

- 已知進行迴圈次數：for...do...done

1. 語法：

```

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

```

2. for 括號內的三串內容意義為：

- 初始值：某個變數在迴圈當中的起始值，直接以類似 `i=1` 設定；
- 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<=100`；
- 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：1 累加到 100 的迴圈

```

[root@linux scripts]# vi sh14.sh
2  #!/bin/bash
3  # Program:
4  #     Try do calculate 1+2+....+100
5  # History:
6  # 2005/08/29   csie   First release
7
8  s=0
9  for (( i=1; i<=100; i=i+1 ))
10 do
11     s=$((s+i))
12 done
13 echo "The result of '1+2+3+...+100' is ==> $s"

```

4. 非數字的迴圈

```

1  for var in con1 con2 con3 ...
2  do 程式段
3
4  done 第一次迴圈時，
5
6     $var 的內容為 con1 ；第二次迴圈時，
7     $var 的內容為 con2 ；第三次迴圈時，

```

```
9  $var 的內容為 con3  ;  
   ....
```

5. 例題：假設有三種動物，分別是 dog, cat, elephant，想在每一行都輸出『There are dogs...』之類的字樣：

```
1  [root@linux scripts]# vi sh15.sh  
   #!/bin/bash  
3  # Program:  
   #      Using for .... loop to print 3 animal  
5  # History:  
   # 2005/08/29   csie   First release  
7  
   for animal in dog cat elephant  
9  do  
       echo "There are "$animal"s...."  
11 done
```

6. 例題：讓使用者輸入某個目錄，然後找出該目錄內的所有檔名的權限：

```
1  [root@linux scripts]# vi sh16.sh  
   #!/bin/bash  
3  # Program:  
   #      let user input a directory and find the whole file's  
       permission.  
5  # History:  
   # 2005/08/29   csie   First release  
7  
   # 1. 先看目錄是否存在？  
9  read -p "Please input a directory: " dir  
   if [ "$dir" == "" ] || [ ! -d "$dir" ]; then  
11     echo "The $dir is NOT exist in your system."  
       exit 1  
13 fi  
  
15 # 2. 開始測試檔案  
   filelist='ls $dir'  
17 for filename in $filelist  
   do  
19     perm=""  
       test -r "$dir/$filename" && perm="$perm readable"  
21     test -w "$dir/$filename" && perm="$perm writable"  
       test -x "$dir/$filename" && perm="$perm executable"  
23     echo "The file $dir/$filename's permission is $perm "  
   done
```

練習題

1. 使用 while 語法，撰寫一 shell script，讓使用者輸入 yes 或者是 YES 才結束程式的執行，否則就一直進行告知使用者輸入字串。

Sol.

```
while [ "$yn" != "yes" ] && [ "$yn" != "YES" ]
do
    read -p "Please input yes/YES to stop this program: " yn
done
```

2. 使用 until 語法，撰寫一 shell script，讓使用者輸入 yes 或者是 YES 才結束程式的執行，否則就一直進行告知使用者輸入字串。

Sol.

```
until [ "$yn" == "yes" ] || [ "$yn" == "YES" ]
do
    read -p "Please input yes/YES to stop this program: " yn
done
```

3. 使用 while 語法，撰寫一 shell script，計算 1 累加到 100。

Sol.

```
s=0
i=0
while [ "$i" != "100" ]
do
    i=$((i+1))
    s=$((s+i))
done
echo "The result of '1+2+3+...+100' is ==> $s"
```

4. 使用 for 語法，撰寫一 shell script，計算 1 累加到 100。

Sol.

```
s=0
for (( i=1; i<=100; i=i+1 ))
do
    s=$((s+i))
done
echo "The result of '1+2+3+...+100' is ==> $s"
```

5. 使用 for 語法，撰寫一 shell script，輸出三種動物，dog, cat, elephant，且每一行都輸出：『There are dogs...』之類的字樣。

Sol.

```
for animal in dog cat elephant
do
    echo "There are "$animal"s..."
done
```

6. 使用 for 語法，撰寫一 shell script，讓使用者輸入某個目錄，然後找出該目錄內的所有檔名的權限。

Sol.

```
read -p "Please input a directory: " dir
if [ "$dir" == "" ] || [ ! -d "$dir" ]; then
    echo "The $dir is NOT exist in your system."
    exit 1
fi

filelist='ls $dir'
for filename in $filelist
do
    perm=""
    test -r "$dir/$filename" && perm="$perm readable"
    test -w "$dir/$filename" && perm="$perm writable"
    test -x "$dir/$filename" && perm="$perm executable"
    echo "The file $dir/$filename's permission is $perm "
done
```

7.3 shell script 的追蹤與 debug

- 以 bash 的相關參數來進行判斷

1. sh 指令

```
[root@linux ~]# sh [-nvx] scripts.sh 選項：
2
-n : 不要執行，僅查詢語法的問題； script
4
-v : 執行 sccript 前，先將 script 的內容輸出到螢幕上；
-x : 將使用到的 script 內容顯示到螢幕上。
```

2. 測試 sh16.sh 有無語法的問題？

```
1 [root@linux ~]# sh -n sh16.sh
# 若語法沒有問題，則不會顯示任何資訊。
3
[dywang@dywOffice ~]$ vi sh015.sh
5 1 #!/bin/bash
  2 # Program:
7 3 # Using for .... loop to print 3 animal
  4 # History:
9 5 # 2005/08/29 csie First release
  6 PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/
  local/sbin:~/bin
```



```

11 7 export PATH
12 8
13 9 for animal in dog cat elephant
14 10 do
15 11     echo "There are "$animal"s....
16 12 done
17 [dywang@dywOffice ~]$ sh -n sh015.sh
sh015.sh: line 11: unexpected EOF while looking for matching
    ' ',
19 sh015.sh: line 13: syntax error: unexpected end of file

```

3. 執行 sh15.sh 前，先將 scripts 的內容輸出到螢幕上。

```

1 [dywang@dywOffice ~]$ sh -v sh015.sh
#!/bin/bash
3 # Program:
#     Using for .... loop to print 3 animal
5 # History:
# 2005/08/29    csie    First release
7
for animal in dog cat elephant
9 do
    echo "There are "$animal"s...."
11 done
There are dogs....
13 There are cats....
There are elephants....

```

4. 將 sh15.sh 的執行過程全部列出來

```

2 [root@linux ~]# sh -x sh15.sh
+ PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/
  local/sbin:/home/csie/bin
+ export PATH
4 + for animal in dog cat elephant
+ echo 'There are dogs.... '
6 There are dogs....
+ for animal in dog cat elephant
8 + echo 'There are cats.... '
There are cats....
10 + for animal in dog cat elephant
+ echo 'There are elephants.... '
12 There are elephants....
14 # 最前面會加上 + 字號表示程式段落。

```

- Linux 系統有很多啟動的 script，想要知道每個 script 的功能，可以直接以 cat, more, less 或 vi 進入該 script 查閱。例如：查閱 script /etc/init.d/xfs：

```

[root@dywHome ~]# cat /etc/init.d/xfs
2  #!/bin/sh
   #
4  # xfs:          Starts the X Font Server
   #
6  # Version:      @(#) /etc/rc.d/init.d/xfs 1.4
   #
8  # chkconfig: 2345 20 10
   # description: Starts and stops the X Font Server at boot time
   #               and shutdown.
10 #
   # processname: xfs
12 # config: /etc/X11/fs/config
   # hide: true
14 #
   ### BEGIN INIT INFO
16 # Provides: xfs
   # Required-Start: $network
18 # Required-Stop: $network
   # Default-Start: 2 3 4 5
20 # Short-Description: X Font Server
   # Description: X Font Server
22 ### END INIT INFO

24 # Source function library.
   . /etc/rc.d/init.d/functions
26
   RETVAL=0
28
   # See how we were called.
30 case "$1" in
   start)
32     gprintf "Starting X Font Server: "
       mkdir -p /tmp
34     chmod a+w,++t /tmp
   # sticky(++t) 僅有該檔案目錄建立者與:/ root 能刪除它。
36     rm -fr /tmp/.font-unix
       daemon --check xfs xfs -port -1 -daemon -droppriv -user
       xfs
38     touch /var/lock/subsys/xfs
       echo
40     ;;
   stop)
42     gprintf "Shutting down X Font Server: "
       killproc xfs
44     rm -f /var/lock/subsys/xfs
       echo

```

```

46         ;;
47     status)
48         status xfs
49         RETVAL=$?
50         ;;
51     restart)
52         gprintf "Restarting X Font Server. "
53         if [ -f /var/lock/subsys/xfs -a ! -z "`pidof xfs`" ];
54             then
55             killproc xfs -USR1
56         else
57             rm -fr /tmp/.font-unix
58             daemon --check xfs xfs -port -1 -daemon -droppriv -
59                 user xfs
60             touch /var/lock/subsys/xfs
61         fi
62         echo
63         ;;
64     *)
65         gprintf "*** Usage: xfs {start|stop|status|restart}\n"
66         exit 1
67 esac
68 exit $RETVAL

```

練習題

1. 如何僅測試 shell script testsh.sh 有無語法的問題，而不要執行 script？

Sol. `sh -n testsh.sh`

2. 如何在執行 shell script testsh.sh 前，先將 script 的內容輸出到螢幕上？

Sol. `sh -v testsh.sh`

3. 如何在執行 shell script testsh.sh 時，將 script 的執行過程全部列出來？

Sol. `sh -x testsh.sh`

4. 如何查閱系統 script /etc/init.d/network 的內容？

Sol. `cat /etc/init.d/network` (cat, more, less 或 vi 皆可)

Chapter 8

開發工具 – make 與 makefile

8.1 編譯器與可執行檔

- 為何編譯？

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`。
2. 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。
3. 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。
4. 以 `file` 命令查看檔案型態：

```

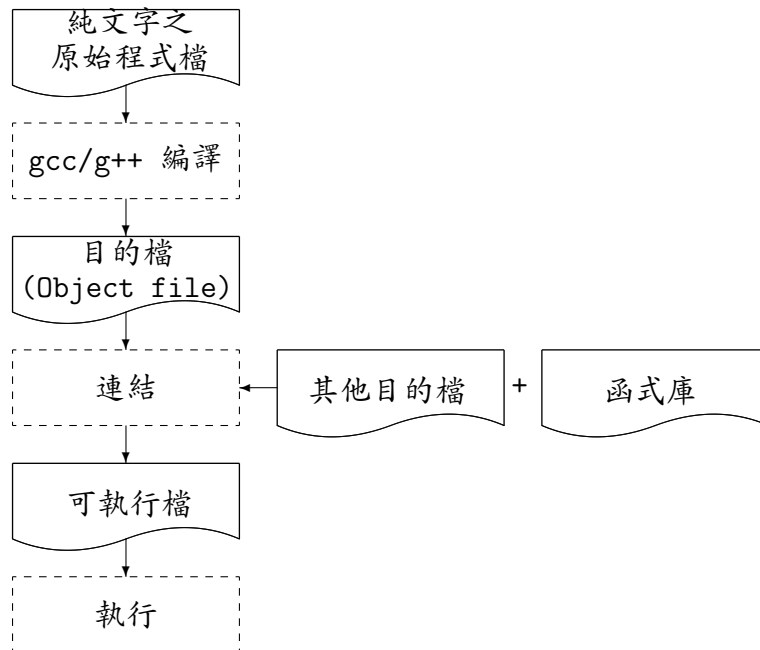
1  #binary file 之可執行檔，顯示執行檔類
   別 ( ELF 32-bit LSB executable )，
   #同時說明是否使用動態函式庫( shared libs )
3  ## ELF: Extensible Linking Format
   ## LSB: Linux Standard Base
5
6  [root@test root]# file /bin/bash
7  /bin/bash: ELF 32-bit LSB executable, Intel 80386, version 1
   (SYSV),
   for GNU/Linux 2.2.5, dynamically linked (uses shared libs),
   stripped
9
10 # shell script test01-hello.sh 檔案
11 [root@test root]# file test01-hello.sh
   test01-hello.sh: Bourne-Again shell script text executable

```

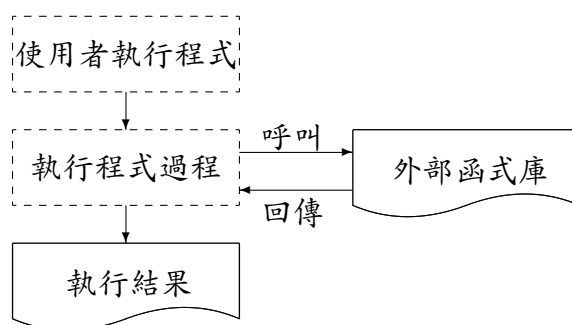
- 編譯步驟：

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. 程式執行引用函式庫流程



● 編譯器 gcc 的用法

1. 以 vi 編輯程式檔 hello.c

```
[dywang@dywOffice test]$ vi hello.c
2  #include <stdio.h>
   main()
4  {
```

```
6     printf("Hello!\n");  
    }
```

2. 僅將原始碼編譯成爲目標檔，並不製作連結等功能：

```
2 [guest@test guest]# gcc -c hello.c  
# 會自動的產生 hello.o 這個檔案，但是並不會產生 binary 執行檔。
```

3. 在編譯的時候，依據作業環境給予最佳化執行速度：

```
2 [guest@test guest]# gcc -O hello.c -c  
# 會自動的產生 hello.o 這個檔案，並且進行最佳化。
```

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

```
2 [guest@test guest]# gcc sin.c -lm -L/usr/lib -I/usr/include  
# 這個指令較常下達在最終連結成 binary file 的時候，  
# -lm 指的是函式庫檔案 libm.so 或 libm.a；  
4 # -L 後面接函式庫的搜尋目錄路徑；  
# -I 後面接原始碼內的 include 檔案之所在目錄。
```

5. 將編譯的結果輸出成某個特定檔名：

```
1 [guest@test guest]# gcc -o hello hello.c  
# -o 後接的是要輸出的 binary file 檔名
```

6. 在編譯的時候，輸出較多的訊息說明：

```
2 [guest@test guest]# gcc -o hello hello.c -Wall  
# 加入 -Wall，程式的編譯時會顯示警告訊息。  
# -Wall 或 -O 等參數爲旗標( FLAGS )，簡稱這些旗標爲 CCFLAGS。
```

練習題

1. Linux 系統上機器認識的可執行檔，格式爲何？

Sol. 二進位檔案 (binary file)

2. 程式編譯目的爲何？

Sol. 將原始碼 (source code) 翻譯成機器可以執行的程式碼。

3. 如何查詢檔案 `/bin/touch` 之型態？
Sol. `file /bin/touch`
4. 執行 `file /bin/bash`，出現 ELF 32-bit LSB executable 訊息，則 `/bin/bash` 檔案型態為何？
Sol. 一般皆可執行檔
5. 執行 `file /etc/init.d/xfs`，出現 Bourne shell script text executable 訊息，則 `/etc/init.d/xfs` 檔案型態為何？
Sol. Bourne shell script text executable
6. 請簡述程式編譯執行步驟？
Sol. 1. 編譯原始碼；2. 將原始碼編譯成目標檔；3. 將目標檔和函式庫中的程式連結成可執行檔；4. 執行。
7. 如何僅將原始碼 `hello.c` 編譯成為目標檔，並不製作連結等功能？
Sol. `gcc -c hello.c`
8. 如何在編譯 `hello.c` 時，依據作業環境給予最佳化執行速度？
Sol. `gcc -O hello.c -c`
9. 如何在編譯 `hello.c` 成可執行檔時，連結函式庫檔案 `libm.so` 或 `libm.a`？
Sol. `gcc hello.c -lm`
10. 如何在編譯 `hello.c` 成可執行檔時，指定函式庫的搜尋目錄為 `/usr/lib`？
Sol. `gcc hello.c -l/usr/lib`
11. 如何在編譯 `hello.c` 成可執行檔時，指定原始碼內的 `include` 檔案所在目錄為 `/usr/include`？
Sol. `gcc hello.c -I/usr/include`
12. 如何在編譯 `hello.c` 時，指定可執行檔名稱為 `hello`？
Sol. `gcc -o hello hello.c`
13. 如何在編譯 `hello.c` 時，指定可執行檔名稱為 `hello`，且於程式的編譯時會顯示警告訊息？
Sol. `gcc -o hello hello.c -Wall`

8.2 make 命令和 makefile

- 何謂 makefile？

1. `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。
2. `makefile` 與專案的原始碼檔案，通常放在同一個目錄中。

3. 可以同時有很多不同的 makefile 管理專案的不同部分。
 4. 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。
- make 使用 makefile 的好處：
 1. 簡化編譯時所需要下達的指令；
 2. 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；
 3. 最後可以依照相依性來更新(update)執行檔。
 - make 與 configure
 1. 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。
 2. 偵測程式 configure
 - (a) 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；
 - (b) 偵測完畢後，會主動的建立 Makefile 的規則檔案。
 - (c) 偵測程式的檔名為 configure 或者是 config。
 - 為什麼要用 make ？
 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。Source codes download

```
1  /* main.c */
   #include "a.h"
3  ...
   /* 2.c */
5  #include "a.h"
   #include "b.h"
7  ...
   /* 3.c */
9  #include "b.h"
   #include "c.h"
11 ...
```

- (a) 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。
 - (b) 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. 假設執行檔包含了四個原始碼檔案，分別是 main.c haha.c sin_value.c cos_value.c，如何讓這個程式可以執行？

(a) 不使用 makefile

```

1 # 1. 先製作出四個目標檔：
[guest@test guest]# gcc -c main.c
3 [guest@test guest]# gcc -c haha.c
[guest@test guest]# gcc -c sin_value.c
5 [guest@test guest]# gcc -c cos_value.c

7 # 2. 製作執行檔 main：
[guest@test guest]# gcc -o main main.o haha.o sin_value.o
\
9 > cos_value.o -lm -L/usr/lib -L/lib

11 # 3. 執行：
[guest@test guest]# ./main
13 HaHa! I'm the King of the world
0.706825
15 0.707388

```

(b) 使用 makefile ，以一個步驟完成所有動作：

```

1 # 1. 先建立編譯的規則
[root@linux ~]# vi makefile
3 main: main.o haha.o sin_value.o cos_value.o
      gcc -o main main.o haha.o sin_value.o cos_value.o
      -lm
5 # 第二行的 gcc 之前是 <tab> 按鍵產生的空格

7 # 2. make 會主動讀取 makefile 內容，並編譯相關的執行檔
[root@linux ~]# rm -f main *.o    <==先將之前的目標檔去除
9 [root@linux ~]# make
cc      -c -o main.o main.c
11 cc      -c -o haha.o haha.c
cc      -c -o sin_value.o sin_value.c
13 cc      -c -o cos_value.o cos_value.c
gcc -o main main.o haha.o sin_value.o cos_value.o -lm
15

# 3. 再執行一次 make
17 [root@linux ~]# make
make: 'main' is up to date.

```

• makefile 的語法 (syntax)

|| 標的

```

2 || (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

```

1. 標的 (target) 與相依檔案(就是目標檔)之間需以『:』隔開。
2. <tab> 需要在命令行的第一個字元；
3. makefile 語法中之 <tab> 與空白：
 - (a) 所有的法則必須在同一行，而且行首必須為 <tab>；不能為空白。
 - (b) 在 makefile 中，行尾如果有一個空白，會造成 make 命令執行錯誤。
4. makefile 的註解 (comment)：
 - (a) 如同 C 原始碼檔案一般，在 makefile 中，以 # 為行首的文字都是註解。
 - (b) makefile 中的註解只是協助作者和其它人，了解 makefile 的內容。

- makefile 的語法規則：

1. makefile 是由很多相依性項目 (dependencies) 和法則 (rules) 所組成。
2. 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。
3. 法則是說明如何根據相依性檔案，來建立目標項目。
4. 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。

- 相依性項目 (dependency)

1. 例題：目標項目 myapp 與 main.o、2.o、3.o 相關，main.o 與 main.c、a.h 相關，以此類推。

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

- (a) makefile 中依序為目標項目、冒號、空白或 tab、隨後就是以空白或 tab 區隔的相關檔案。
 - (b) 改變 b.h，需要重建 2.o 和 3.o，因為 2.o 和 3.o 被改變，又需要重建 myapp。
2. 以 gcc 的 -MM 選項以 makefile 的格式，輸出相依性項目。

```

$ gcc -MM main.c 2.c 3.c
2 main.o: main.c a.h
  2.o: 2.c a.h b.h
4 3.o: 3.c b.h c.h

```

- (a) 輸出結果可插入 makefile 中，成為相依性的法則。
 - (b) makedepend 工具，類似 -MM 選項，但會將相依性項目附加到 makefile 之後。
3. 利用假造的目標項目 all，建立多個目標檔案。例如：產生二進位執行檔 myapp 和使用手冊文件 myapp.1。

```
all: myapp myapp.1
```

- (a) 通常設定 all 為 makefile 的第一個目標項目，隨後再一一列出 all 的相依性項目。
 - (b) 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。
- 法則 (rule)：說明如何產生目標項目。

```

1 2.o: 2.c a.h b.h
   gcc -c 2.c
3 #，以法則 gcc -c 2.c 產生目標項目 2.o

```

- make [-f makefile] [options] ... [targets] ...，常用的選項及參數：
1. -j N：讓 make 在同一個時間執行 N 個命令，以加速編譯的時間。
 2. -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。
 3. -n：告訴 make 只印出將會進行的工作，而不真正去編譯。
 4. -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。
 5. 直接於 make 時，指定目標項目名稱

```
1 [root@dywOffice ~]# make targetfile
```

- 實作一簡單的 makefile 範例

1. 產生 makefile—檔名 Makefile1:

```

1  myapp: main.o 2.o 3.o
    gcc -o myapp main.o 2.o 3.o
3  main.o: main.c a.h
    gcc -c main.c
5  2.o: 2.c a.h b.h
    gcc -c 2.c
7  3.o: 3.c b.h c.h
    gcc -c 3.c

```

2. 執行 make 命令，因為程式碼不存在，故得到以下訊息：

```

$ make -f Makefile1
2  make: *** No rule to make target 'main.c', needed by 'main.o'
    '. Stop.
$

```

3. 利用 touch 產生空白的標頭檔案。

```

1  $ touch a.h
    $ touch b.h
3  $ touch c.h

```

4. 編輯程式 main.c, 2.c 和 3.c。

```

1  /* main.c */
   #include <stdlib.h>
3  /* 引用的標頭檔案，在 makefile 中，會有相依性關係。
   #include "a.h"
5  extern void function_two();
   extern void function_three();
7  int main()
   {
9
   /* 呼叫 function_two 和。 function_three */
11     function_two();
       function_three();
13     exit (EXIT_SUCCESS);
   }
15
   /* 2.c 定義 function_two 函式 */
17 #include "a.h"
   #include "b.h"
19 void function_two() {

```

```
21 }  
22  
23 /* 3.c 定義 function_three 函式 */  
24 #include "b.h"  
25 #include "c.h"  
26 void function_three() {  
27     }  
28 }
```

5. 再次測試 make 成功：

```
$ make -f Makefile1  
2 gcc -c main.c  
3 gcc -c 2.c  
4 gcc -c 3.c  
5 gcc -o myapp main.o 2.o 3.o  
6 $
```

6. b.h 修改後再 make：

```
$ touch b.h  
2 $ make -f Makefile1  
3 gcc -c 2.c  
4 gcc -c 3.c  
5 gcc -o myapp main.o 2.o 3.o  
6 $
```

7. 刪除目的檔案後再 make：

```
$ rm 2.o  
2 $ make -f Makefile1  
3 gcc -c 2.c  
4 gcc -o myapp main.o 2.o 3.o  
5 $
```

練習題

1. 使用 make 時，會在當時的目錄下搜尋那個文字檔，以得知如何建立應用程式？

Sol. `makefile` 或 `Makefile`

2. 軟體開發者會寫一支偵測程式來建立 makefile 的規則檔案，其檔名為何？

Sol. `configure` 或 `config` (較少用)

3. 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。

```
/* main.c */
#include "a.h"
...
/* 2.c */
#include "a.h"
#include "b.h"
...
/* 3.c */
#include "b.h"
#include "c.h"
...
```

如果程式設計人員改變 a.h，則那些檔案需要重新編譯？那些檔案則不需要被重新編譯？

Sol. main.c 及 2.c 需重新編譯；3.c 不需重新編譯。

4. 假設執行檔包含了三個原始碼檔案，分別是 main.c ha1.c ha2.c，不使用 makefile 情況下，如何建立可執行檔 main？

Sol. 逐步執行以下動作：gcc -c main.c; gcc -c ha1.c; gcc -c ha2.c; gcc -o main main.o ha1.o ha2.o

5. 假設執行檔包含了三個原始碼檔案，分別是 main.c ha1.c ha2.c，使用 makefile 情況下，如何建立可執行檔 main？

Sol.

```
%(+ 1. 先建立編譯的規則%)
$ vi makefile
main: main.o ha1.o ha2.o
    ^- gcc -o main main.o ha1.o ha2.o
%(+ 2. make 會主動讀取 makefile 內容，並編譯相關的執行檔%)
$ make
```

6. 請寫出 makefile 的語法。

Sol.

```
標的: 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
```

7. 某一 makefile 中，有如下語法，其中 □ 代表空白字元，請問是否有錯？若有請說明原因。

```
2.o:□2.c□a.h□b.h
□□□□gcc□-c□2.c
```

Sol. 有錯 *gcc 前為 <tab>，不可為空白。

8. 某一 makefile 中，有如下語法，其中 `□` 代表空白字元，`→` 代表 `<tab>`。請問是否有錯？若有請說明原因。
- ```
2.o:□2.c□a.h□b.h
→ gcc□-c□2.c□
```
- Sol. 有錯。行尾不可為空白。
9. 某一 makefile 中，有如下語法，其中 `□` 代表空白字元，`→` 代表 `<tab>`。請問是否有錯？若有請說明原因。
- ```
2.o:□2.c□a.h□b.h□
→ gcc□-c□2.c
```
- Sol. 有錯。行尾不可為空白。
10. 在 makefile 中，註解要如何處理？
- Sol. 行首以 `#` 開頭。
11. makefile 主要由那兩部分組成？
- Sol. 相依性項目 (dependencies) 和法則 (rules) 所組成。
12. 請說明 makefile 中的相依性項目？
- Sol. 描述目標項目 (target，要產生的檔案) → 產生該檔案之相關的原始檔檔案。
13. 如何以 gcc 指令，以 makefile 格式輸出 main.c, 2.c 及 3.c 的相依性項目？
- Sol. `gcc-MM main.c 2.c 3.c`
14. 請說明 makefile 中的法則？
- Sol. 說明如何根據相依性檔案，來建立目標項目。
15. 請說明 make 指令如何使用 makefile？
- Sol. 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。
16. 若要讓 make 在同一個時間執行 4 個命令，以加速編譯的時間。如何下指令？
- Sol. `make -j 4`
17. 若要讓 make 在遇到錯誤之時，仍然繼續運行，而不會停止在第一個問題點。如何下指令？
- Sol. `make -k`
18. 若要讓 make 只印出將會進行的工作，而不會真正去編譯。如何下指令？
- Sol. `make -n`
19. 若要指定 makefile1 來進行 make。如何下指令？
- Sol. `make -f makefile1`
20. 在沒指定 makefile 情況下，make 會依序搜尋那些檔名的 makefile？
- Sol. `GNUmakefile, makefile, Makefile`

21. 請撰寫一 makefile，目標檔及產生之可執行檔，檔名為 myapp，其相依性項目有 main.o 及 2.o，而目標項目 main.o 及 2.o 之相依性項目則分別為 main.h, a.h 及 2.c, a.h, b.h。

Sol.

```
myapp: main.o 2.o
→ gcc -o myapp main.o 2.o
main.o: main.c a.h
→ gcc -c main.c
2.o: 2.c a.h b.h
→ gcc -c 2.c
```

22. 執行 make -f Makefile1 時出現訊息『make: *** No rule to make target 'main.c', needed by 'main.o'. Stop』，代表意義為何？

Sol. 產生目標檔 main.o 需要檔案 main.c 不存在，故停止。

23. 若程式 2.c 中有兩行 #include "a.h" 及 #include "b.h"，則 makefile 中目標項目 2.o 要如何撰寫？

Sol.

```
2.o: 2.c a.h b.h
→ gcc -c 2.c
```

24. 若程式 3.c 中有兩行 #include "b.h" 及 #include "c.h"，則 makefile 中目標項目 3.o 要如何撰寫？

Sol.

```
3.o: 3.c b.h c.h
→ gcc -c 3.c
```

8.3 makefile 的變數

- 變數用途：

1. 簡化 makefile
2. 開發過程與最終版本使用之編譯參數不同
 - (a) 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；
 - (b) 最終版本則應是一個最小的二進位檔案，且不含除錯資訊。
3. 讓 makefile 也適用不同的編譯器。

- 變數定義：MACRONAME=value

1. 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)
2. 變數左邊不可以有 <tab> ；
3. 變數與變數內容在『=』兩邊不能具有『:』 ；

4. 習慣上，變數最好是以『大寫字母』為主；
5. 等號後面的 value 變成空白時，代表將變數清成空白。

- 變數存取

1. `$(MACRONAME)`
2. `${MACRONAME}`
3. 有些 make 版本也接受 `$MACRONAME`

- 以『變數』簡化 makefile：

```
1 [guest@test guest]# vi makefile
LIBS = -lm
3 OBJS = main.o haha.o sin_value.o cos_value.o
main: ${OBJS}
5     gcc -o main ${OBJS} ${LIBS}
clean:
7     rm -f main ${OBJS}
```

- 定義變數，以環境變數 CFLAGS 為例：

1. make 指令中直接定義

```
1 [guest@test guest]# make clean main "CFLAGS=-Wall"
```

2. makefile 檔案內直接指定環境變數

```
1 [guest@test guest]# vi makefile
LIBS = -lm
3 OBJS = main.o haha.o sin_value.o cos_value.o
CFLAGS = -Wall
5 main: ${OBJS}
    gcc -o main ${OBJS} ${LIBS}
7 clean:
    rm -f main ${OBJS}
```

- 環境變數取用規則：

1. make 指令列後面加上的環境變數為優先；
2. makefile 裡面指定的環境變數第二；
3. shell 原本具有的環境變數第三。

- 實作－含有變數的 makefile

1. Makefile1 的修改版本，加入一些變數之後，變成 Makefile2：

```
# 目標項目 all 只會產生 myapp。執行 make 預設建立目標項目 myapp。
2 all: myapp
  # Which compiler
4 CC = gcc
  # Where are include files kept
6 INCLUDE = .
  # Options for development (-g Produce debugging information)
8 CFLAGS = -g -Wall -ansi
  # Options for release
10 CFLAGS = -O -Wall -ansi
myapp: main.o 2.o 3.o
12     $(CC) -o myapp main.o 2.o 3.o
main.o: main.c a.h
14     $(CC) -I$(INCLUDE) $(CFLAGS) -c main.c
2.o: 2.c a.h b.h
16     $(CC) -I$(INCLUDE) $(CFLAGS) -c 2.c
3.o: 3.c b.h c.h
18     $(CC) -I$(INCLUDE) $(CFLAGS) -c 3.c
```

2. 執行結果：

```
$ rm *.o myapp
2 $ make -f Makefile2
gcc -I. -g -Wall -ansi -c main.c
4 gcc -I. -g -Wall -ansi -c 2.c
gcc -I. -g -Wall -ansi -c 3.c
6 gcc -o myapp main.o 2.o 3.o
$
```

3. make 程式取代 \$(CC)、\$(CFLAGS) 和 \$(INCLUDE) 的部分。故若要改變編譯器命令，只需改變這些變數。

- make 常用的變數

- \$? 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。

1. 例題：以 \$@ 代表目前的目標（target）項目。

```
[guest@test guest]# vi makefile
LIBS = -lm
OBJS = main.o haha.o sin_value.o cos_value.o
CFLAGS = -Wall
main: ${OBJS}
    gcc -o $@ ${OBJS} ${LIBS}  %*<== 在
    此 $@ 即 main 。 *)
clean:
    rm -f main ${OBJS}
```

- makefile 中兩個特別字元，可以加在要執行的命令之前：

1. - :make 會忽略命令的錯誤。

- (a) 如果希望產生一個目錄，但希望忽略錯誤，可能是因為該目錄已經存在。

```
1 -mkdir /usr/local/repository
```

- (b) 如果希望清除目標檔案，但希望忽略錯誤，可能是因為該檔案不存在。

```
1 clean:
    -rm main.o 2.o 3.o
```

2. @ :make 不會在標準輸出上，顯示要執行的命令。

- (a) 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。

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

練習題

1. 請列舉兩項 makefile 中使用變數的好處。

Sol. 1. 開發過程與最終版本使用之編譯參數不同；2. 讓 makefile 也適用不同的編譯器。

2. 在 makefile 中，定義變數 『MACRONAME=value』，有無錯誤，若有，則說明原因。

Sol. 正確

3. 在 makefile 中，定義變數『MACRONAME□=□value』，有無錯誤，若有，則說明原因，其中 □ 表示空白。
Sol. 正確。
4. 在 makefile 中，定義變數『→MACRONAME□=□value』，有無錯誤，若有，則說明原因，其中 → 表示 <tab>，□ 表示空白。
Sol. 錯誤，變數左邊不可以有 <tab>。
5. 在 makefile 中，定義變數『MACRONAME:□=□value』，有無錯誤，若有，則說明原因，其中 □ 表示空白。
Sol. 錯誤，變數與變數內容在『=』兩邊不能具有『:』。
6. 在 makefile 中，定義變數『MACRONAME□=□』，代表意義為何？其中 □ 表示空白。
Sol. 將變數清成空白。
7. 請列舉兩項，在 makefile 中取出變數 MACRONAME 的方法。
Sol. `$(MACRONAME)` 或 `${MACRONAME}`。
8. 請說明 make 取用環境變數的規則。
Sol. 1.make 指令列成如，2.makefile 裡面指定；3.shell 原已具有。
9. 若有一 makefile 中，目標項目 myapp 的編譯法則為 `gcc -o $@ main.o ha.o`，則產生之可執行檔為何？
Sol. myapp。
10. 若有一 makefile 中，目標項目 myapp 的編譯法則為 `gcc -o $@ main.o ha.o`，其中 \$@ 代表？
Sol. 變數 \$@ 代表目前的目標項目 myapp。
11. 若有一 makefile 中，有一法則為 `-mkdir /usr/local/repository`，其中 - 代表意義為何？
Sol. make 會忽略命令的錯誤。
12. 若有一 makefile 中，有一法則為 `-rm main.o 2.o 3.o`，其中 - 代表意義為何？
Sol. make 會忽略命令的錯誤。
13. 若有一 makefile 中，有一法則為 `@rm main.o 2.o 3.o`，其中 @ 代表意義為何？
Sol. make 不會在標準輸出上，顯示要執行的命令。

8.4 多重目標項目 (target)

- 可在 makefile 中建立多個目標項目，並於 make 時指定目標項目。
- 例題：沿上例，加入 clean 目標項目，以移除不想要的目的檔 (object)。

1. 建立編譯的規則

```

1 [guest@test guest]# vi makefile
main: main.o haha.o sin_value.o cos_value.o
3     gcc -o main main.o haha.o sin_value.o cos_value.o -lm
clean:
5     rm -f main main.o haha.o sin_value.o cos_value.o
# clean 冒號之後是空白。目標項目永遠會被認為過期，所以它的法則永
    遠會被執行。

```

2. 測試標的 clean :

```

2 [guest@test guest]# make clean
rm -f main main.o haha.o sin_value.o cos_value.o

```

3. 先清除目標檔再編譯程式 main :

```

2 [guest@test guest]# make clean main
rm -f main main.o haha.o sin_value.o cos_value.o
cc -c -o main.o main.c
4 cc -c -o haha.o haha.c
cc -c -o sin_value.o sin_value.c
6 cc -c -o cos_value.o cos_value.c
gcc -o main main.o haha.o sin_value.o cos_value.o -lm

```

- 沿上例，再加入 install 目標項目，將完成的應用程式安裝到不同的目錄。

1. makefile 新的版本 Makefile3 :

```

1 all: myapp
  # Which compiler
3 CC = gcc
  # Where to install
5 INSTDIR = /usr/local/bin
  # Where are include files kept
7 INCLUDE = .
  # Options for development
9 CFLAGS = -g -Wall -ansi
  # Options for release
11 # CFLAGS = -O -Wall -ansi
myapp: main.o 2.o 3.o
13     $(CC) -o myapp main.o 2.o 3.o
main.o: main.c a.h
15     $(CC) -I$(INCLUDE) $(CFLAGS) -c main.c
2.o: 2.c a.h b.h

```

```

17 | $(CC) -I$(INCLUDE) $(CFLAGS) -c 2.c
   | 3.o: 3.c b.h c.h
19 | $(CC) -I$(INCLUDE) $(CFLAGS) -c 3.c
   | clean:
21 | -rm main.o 2.o 3.o
   | install: myapp
23 | # 每個命令都會啟動一個新的，所以必須加上反斜線 shell(\)，讓所
   |   有 script 命令在同一行，且在同一個 shell 中執行。
   | @if [ -d $(INSTDIR) ]; \
25 |   then \
   |     cp myapp $(INSTDIR);\
27 |     chmod a+x $(INSTDIR)/myapp;\
   |     chmod og-w $(INSTDIR)/myapp;\
29 |     echo "Installed in $(INSTDIR)";\
   |   else \
31 |     echo "Sorry, $(INSTDIR) does not exist";\
   |   fi

```

2. &&：前個命令成功，才執行下個命令

```

   | @if [ -d $(INSTDIR) ]; \
2 |   then \
   |     cp myapp $(INSTDIR) &&\
4 |     chmod a+x $(INSTDIR)/myapp && \
   |     chmod og-w $(INSTDIR)/myapp && \
6 |     echo "Installed in $(INSTDIR)" ;\
   |   else \
8 |     echo "Sorry, $(INSTDIR) does not exist" ; false ; \
   |   fi

```

3. 執行

```

1 | $ rm *.o myapp
   | $ make -f Makefile3
3 | gcc -I. -g -Wall -ansi -c main.c
   | gcc -I. -g -Wall -ansi -c 2.c
5 | gcc -I. -g -Wall -ansi -c 3.c
   | gcc -o myapp main.o 2.o 3.o
7 | $ make -f Makefile3
   | make: Nothing to be done for 'all'.
9 | $ rm myapp
   | $ make -f Makefile3 install
11 | gcc -o myapp main.o 2.o 3.o
   | Installed in /usr/local/bin
13 | $ make -f Makefile3 clean
   | rm main.o 2.o 3.o
15 | $

```

練習題

1. makefile 中，是否可以有多個目標項目？
Sol. 可以。
2. 若 makefile 中有一目標項目 clean，則如何於執行 make 時指定此目標項目？
Sol. `make clean`
3. 若有一 makefile 中，目標項目 clean 冒號之後是空白，代表意義為何？
Sol. 目標項目永遠會被認為過期，所以它的法則永遠會被執行。
4. 若 makefile 中，多個目標項目 main, clean 與 install，要如何指定目標項目 install 執行 make？
Sol. `make install`
5. 若 makefile 中，多個目標項目 main, clean 與 install，make 如何"一次"執行 clean 後接著 install？
Sol. `make clean install`
6. 請解釋如下 makefile 片段，假設變數 INSTDIR=mydir。

```
install: myapp
    @if [ -d $(INSTDIR) ]; \
    then \
        cp myapp $(INSTDIR);\
        chmod a+x $(INSTDIR)/myapp;\
        chmod og-w $(INSTDIR)/myapp;\
        echo "Installed in $(INSTDIR)";\
    else \
        echo "Sorry, $(INSTDIR) does not exist";\
    fi
```

Sol. 1. 目標項目 install 的相依性項目有 myapp; 2. @ 表示不會在標準輸出上，顯示要執行的命令; 3. 如果 mydir 是一個目錄，則執行複製 myapp 到目錄 mydir、改變 mydir/myapp 的屬性，所有使用者加上可執行屬性、其他使用者及同群組使用者刪除可寫屬性；螢幕輸出 Installed in mydir; 4. 其他(即 mydir 不是一個目錄，則螢幕輸出 Sorry, mydir does not exist; 5. 因為每個命令都會啟動一個新的 shell，所以必須加上反斜線(\)，讓所有 script 命令在同一行，且在同一個 shell 中執行。

7. 如下 makefile，&& \ 代表意義為何？

```
chmod a+x $(INSTDIR)/myapp && \
chmod og-w $(INSTDIR)/myapp && \
```

Sol. && 表示前個命令成功，才執行下個命令，\ 讓 script 命令在同一行，且在同一個 shell 中執行。

8.5 Makefile 其他法則

- 檔尾 (suffix) 和符號 (pattern) 法則：

```
1 # 將 .old_suffix 的檔案變成 .new_suffix 的檔案
3 # 符號法則 1
  .<old_suffix>.<new_suffix>:
5
  # 符號法則 2
7  %.<old\_suffix>: %.<new\_suffix>:
```

- 例題：將 .cpp 檔案變成 .o 檔案：

- 符號法則 1：

```
1 .SUFFIXES:      .cpp
  .cpp.o:
3   $(CC) -xc++ $(CFLAGS) -I$(INCLUDE) -c $<
  # gcc 的 -xc++ 旗標告訴編譯器，它是一個 C++ 原始碼檔案。
5  # 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）。
```

- 符號法則 2：

```
1 %.o: %.cpp
   $(CC) -xc++ $(CFLAGS) -I$(INCLUDE) -c $<
```

- 內建的法則：借助檔尾 (suffix)，make 即知使用什麼法則。

- 可以利用 -p 選項，要求 make 印出內建法則，截取部分如下：

```
[dywang@dywOffice ~]$ make -p | vi -
2 OUTPUT_OPTION = -o $@
  CXX = g++ ###(gcc)
4 COMPILE.cc = $(CXX) $(CXXFLAGS) $(CPPFLAGS) $(TARGET_ARCH) -c
  COMPILE.C = $(COMPILE.cc)
6 COMPILE.cpp = $(COMPILE.cc)
8
8 %.o: %.c
  # commands to execute (built-in):
10   $(COMPILE.cc) $(OUTPUT_OPTION) $<
```

2. 例題：將 `.c` 檔尾的檔案變成 `.o` 檔尾的檔案。

(a) 編輯 `foo.c`，(傳統的 Hello World 程式)。

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 int main()
4 {
5     printf("Hello World\n");
6     exit(EXIT_SUCCESS);
7 }
```

(b) 藉由 `make` 編譯程式，但沒有指定 `makefile`。

```
1 $ make foo
cc      foo.c      -o foo
3 $
```

(c) 若不使用內建法則，可以於 `make` 時，直接加入參數

```
1 $ rm foo
$ make CC=gcc CFLAGS="-Wall -g" foo
3 gcc -Wall -g      foo.c      -o foo
$
```

3. 可以借用內建法則，簡化 `Makefile3` 中之編譯法則，儲存為 `Makefile4`：

```
1 main.o: main.c a.h
2 # $(CC) -I$(INCLUDE) $(CFLAGS) -c main.c
3 2.o: 2.c a.h b.h
4 # $(CC) -I$(INCLUDE) $(CFLAGS) -c 2.c
5 3.o: 3.c b.h c.h
6 # $(CC) -I$(INCLUDE) $(CFLAGS) -c 3.c
# 註解部分為原 Makefile3 之編譯法則
```

練習題

1. 請使用兩種方式寫出，將 `.cpp` 的檔案變成 `.o` 的檔案法則。

Sol. `.cpp.o: %.cpp`

2. `makefile` 中有一 `.cpp.o` 之編譯法則 `$(CC) -xc++ -c $<`，其中 `-xc++` 代表意義為何？

Sol. `gcc` 的 `-xc++` 旗標告訴編譯器，它是一個 C++ 原始碼檔案。

3. makefile 中有一 .cpp.o 之編譯法則 \$(CC) -xc++ -c \$<，其中 \$< 代表意義為何？

Sol. 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）。

4. 如何要求 make 印出其內建法則？

Sol. `make -p`

5. 何謂 make 的內建法則？

Sol. 若 makefile 中沒指定法則，則 make 會以內建法則執行，因此可簡化 makefile。

6. make 的內建法則，編譯器使用 cc，若要改用 gcc 編譯 foo，但 makefile 中沒指定，需如何做？

Sol. 直接於 make 時加入參數：`make CC=gcc foo`

7. 若 makefile 中有目標項目 main.o: main.c a.h，但未指定其編譯法則，則 make 會怎麼執行？

Sol. 使用內建法則，產生目標項目 main.o

8.6 函式庫管理

- 函式庫：目標檔庫

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。

- Linux 放置函式庫之目錄：

```

1 /usr/lib
  /lib
3 # kernel 的函式庫放在 /lib/modules

```

- 如何判斷某個可執行的 binary 檔案含有什麼動態函式庫？

1. ldd 指令

```

1 [root@linux ~]# ldd [-vdr] [filename]選項：
3 --version : 列印 ldd 的版本序號
  -v : 列出所有內容資訊；
5 --help : 指令用法資訊

```

2. 找出檔案 /usr/bin/passwd 的函式庫資料。

```

1 [root@linux ~]# ldd /usr/bin/passwd
   linux-gate.so.1 => (0x00d19000)中間省略
3 .....
   libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x00bd6000
   )中間省
   略
5 .....

```

3. 找出函式 /lib/libc.so.6 的相關其他函式庫。

```

1 [root@linux ~]# ldd /lib/libc.so.6
   /lib/ld-linux.so.2 (0x00bf1000)
3   linux-gate.so.1 => (0x00632000)

5 [root@linux ~]# ldd -v /lib/libc.so.6
   /lib/ld-linux.so.2 (0x00bf1000)
7   linux-gate.so.1 => (0x00111000)

9   Version information:
   /lib/libc.so.6:
11       ld-linux.so.2 (GLIBC_2.1) => /lib/ld-linux.so
       .2
       ld-linux.so.2 (GLIBC_2.3) => /lib/ld-linux.so
       .2
13       ld-linux.so.2 (GLIBC_PRIVATE) => /lib/ld-
       linux.so.2

```

- 如何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. 在 /etc/ld.so.conf 寫入想要讀入快取記憶體的動態函式庫所在的目錄；
2. 利用執行檔 ldconfig 將 /etc/ld.so.conf 的資料讀入快取中；
3. 同時也將資料記錄一份在檔案 /etc/ld.so.cache 中。

```

1 [root@test root]# ldconfig [-f conf] [-C cache] [-p] 參數說明：
3 -f conf : conf 預設為 /etc/ld.so.conf
  -C : cachecache 預設為 /etc/ld.so.cache
5 -       : 列出目前在p cache 內的資料

```

4. 例題：將 xorg 相關函式庫加到快取記憶體中：

```

1 [root@dywHome2 ~]# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
3 /usr/X11R6/lib
  /usr/lib/qt3/lib
5 [root@dywHome2 ~]# ldconfig -p | grep "libafb"
  [root@dywHome2 ~]# vi /etc/ld.so.conf
7 [root@dywHome2 ~]# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
9 /usr/X11R6/lib
  /usr/lib/qt3/lib
11 /usr/lib/xorg/modules # 加入此行
  [root@dywHome2 ~]# ldconfig
13 [root@dywHome2 ~]# ldconfig -p | grep "libafb"
      libafb.so (libc6) => /usr/lib/xorg/modules/libafb.so

```

- 建立函式庫：將多個目標檔案組合成一個函式庫檔案

1. 函式庫建立指令 ar。例如：

```

$ cc -c f1.c           # 產生 f1.o
2 $ cc -c f2.c           # 產生 f2.o
  $ ar rv mlib.a f1.o f2.o # 將目標檔 f1.o 及 f2.o 插入到函式
    庫 mlib.a
4
  # 選項 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。
6 # 選項 v : 顯示執行操作選項的附加信息。

```

2. nm：查看目標檔庫檔案 mlib.a 的內容。

```
$ nm mlib.a
```

3. make 內建函式庫管理法則，例如：將 .c 檔案變成 .a 函式庫

```
1 .c.a:
   $(CC) -c $(CFLAGS) $<
3   $(AR) $(ARFLAGS) $@ $*.o
5 # 變數 (AR)(ARFLAGS)，一般分別為命令 ar 和選項 rv。
```

4. 若有一函式庫 fud，包含 bas.o 檔案，則上例法則中的變數為

- (a) \$< 代表目前的相依性項目，就是 bas.c。
- (b) \$@ 代表目前的目標項目，就是 fud.a 函式庫。
- (c) \$* 代表目前的相依性項目，不過不含副檔名，就是 bas。

- 實作—管理一個函式庫

1. 將 2.o 和 3.o 插入 mylib.a 函式庫中。新的 Makefile5 如下：

```
1 all: myapp
   # Which compiler
3 CC = gcc
   # Where to install
5 INSDIR = /usr/local/bin
   # Where are include files kept
7 INCLUDE = .
   # Options for development
9 CFLAGS = -g -Wall -ansi
   # Options for release
11 # CFLAGS = -O -Wall -ansi
   # Local Libraries
13 MYLIB = mylib.a
   myapp: main.o $(MYLIB)
15   $(CC) -o myapp main.o $(MYLIB)
   $(MYLIB): $(MYLIB)(2.o) $(MYLIB)(3.o)
17 main.o: main.c a.h
   2.o: 2.c a.h b.h
19 3.o: 3.c b.h c.h
   clean:
21   -rm main.o 2.o 3.o $(MYLIB)
   install: myapp
23   @if [ -d $(INSDIR) ]; \
       then \
25       cp myapp $(INSDIR);\
```

```
27     chmod a+x $(INSTDIR)/myapp;\
    chmod og-w $(INSTDIR)/myapp;\
    echo "Installed in $(INSTDIR)";\
29 else \
    echo "Sorry, $(INSTDIR) does not exist";\
31 fi
```

2. 執行

```
1 $ rm -f myapp *.o mylib.a
  $ make -f Makefile5
3 gcc -g -Wall -ansi -c -o main.o main.c
  gcc -g -Wall -ansi -c -o 2.o 2.c
5 ar rv mylib.a 2.o
  a - 2.o
7 gcc -g -Wall -ansi -c -o 3.o 3.c
  ar rv mylib.a 3.o
9 a - 3.o
  gcc -o myapp main.o mylib.a
```

3. 試驗 3.o 的相依性項目

```
2 $ touch c.h
  $ make -f Makefile5
  gcc -g -Wall -ansi -c -o 3.o 3.c
4 ar rv mylib.a 3.o
  r - 3.o
6 gcc -o myapp main.o mylib.a
  $
```

● 以子目錄來管理函式庫

1. 方法一：由主要的 makefile 呼叫子目錄的 makefile。

```
1 mylib.a:
  (cd mylibdirectory;$(MAKE))
```

- (a) 要先製作 mylib.a。
- (b) 當 make 依據這個法則建立函式庫時，它會進入子目錄 mylibdirectory；
- (c) 隨後再喚起 make 命令來管理函式庫。這會喚起一個新 shell。
- (d) 括弧是為確保，所有的處理動作都是在一個 shell 中。

2. 方法二：在 makefile 中使用變數，目錄加上 D，檔案加上 F。

```
2 .c.o:
    $(CC) $(CFLAGS) -c $(@D)/$(<F) -o $(@D)/$(@F)
    # 在子目錄中編譯檔案，並將編譯完的目的檔留在子目錄中。
```

(a) 若目標項目所在目錄為 mydir，程式 2.c，目標項目 2.o 檔案，則上例法則中的變數為

- i. \$(@D) 代表目前的目標項目所在目錄，就是 mydir。
- ii. \$(<F) 代表目前的相依性項目的檔案名稱，就是 2.c。
- iii. \$(@F) 代表目前的目標項目的檔案名稱，就是 2.o。

(b) 隨後利用相依性項目，更新現在目錄的函式庫，法則如下：

```
1 mylib.a: mydir/2.o mydir/3.o
    ar rv mylib.a $?
3 # $? 代表需要重建的相依性項目，就是 mydir/2.o 及(或)
    mydir/3.o
```

練習題

1. 何謂函式庫？

Sol. 集合目標檔的目標檔庫

2. 函式庫分為那兩種？

Sol. 靜態及動態函式庫

3. 何謂靜態函式庫？

Sol. 1. 附檔名通常為 libxxx.a 的類型；2. 整個函式庫的資料被整合到執行檔中，故可執行檔可以獨立執行，但升級時，連執行檔也需要重新編譯過一次。

4. 何謂動態函式庫？

Sol. 1. 附檔名通常為 libxxx.so 的類型；2. 編譯時執行檔中僅具有指向動態函式庫所在的指標，故可執行檔不能被獨立執行，而函式庫升級後，執行檔不需要進行重新編譯；3. 目前的 Linux distribution 比較傾向使用動態函式庫。

5. 通常 Linux 放置函式庫之目錄為何？

Sol. /usr/lib 及 /lib

6. 通常 Linux kernel 的函式庫放置目錄為何？

Sol. /lib/modules

7. 如何找出檔案 /usr/bin/passwd 的函式庫資料？

Sol. ldd /usr/bin/passwd

8. 如何找出函式 `/lib/libc.so.6` 的相關其他函式庫？
Sol. `ldd /lib/libc.so.6`
9. 如何找出函式 `/lib/libc.so.6` 的相關其他函式庫，並列出所有內容資訊？
Sol. `ldd -v /lib/libc.so.6`
10. 要將某目錄下之動態函式庫，載入快取記憶體，必須於那個檔案中加入此目錄？
Sol. `/etc/ld.so.conf`
11. 如何將 `/etc/ld.so.conf` 的資料讀入快取記憶體中？
Sol. 執行 `ldconfig`
12. 如何將 `/etc/my.so.conf` 的資料讀入快取記憶體中？
Sol. 執行 `ldconfig -f /etc/my.so.conf`
13. 如何將 `/etc/ld.so.conf` 的資料讀入快取記憶體 `/etc/my.so.cache` 中？
Sol. 執行 `ldconfig -C /etc/my.so.cache`
14. 如何列出目前在 `cache` 內的資料？
Sol. 執行 `ldconfig -p`
15. 如何查看動態函式庫 `libafb.so`，是否在快取記憶體中？
Sol. 執行 `ldconfig -p | grep "libafb"`
16. 若目錄 `/usr/lib/xorg/modules` 包含動態函式庫 `libafb.so`，如何將其載入快取記憶體，以增進動態函式庫的讀取速度？請詳列步驟。
Sol. 1. 以 `vi` 為編輯器，讀 `/etc/ld.so.conf` 加入一行 `/usr/lib/xorg/modules`，命令為 `vi /etc/ld.so.conf`；2. 執行 `ldconfig`；3. 查看動態函式庫 `libafb.so`，是否在快取記憶體中，`ldconfig -p | grep "libafb"`
17. 如何將目標檔 `f1.o` 及 `f2.o` 插入到函式庫 `mlib.a`？
Sol. `ar r mlib.a f1.o f2.o`
18. 如何將目標檔 `f1.o` 及 `f2.o` 插入到函式庫 `mlib.a`，並顯示訊息？
Sol. `ar rv mlib.a f1.o f2.o`
19. 如何查看目標檔庫檔案 `mlib.a` 的內容？
Sol. `nm mlib.a`
20. 若有一函式庫 `fud`，包含 `bas.o` 檔案，則下列 `makefile` 法則中的變數 `$<`，`$@`，`$*` 分別為何？

```
.c.o:
    $(CC) -c $(CFLAGS) $<
    $(AR) $(ARFLAGS) $@ $*.o
#%* 變數 $(AR) 和 $(ARFLAGS)，一般分別為命令 ar 和選項 rv。*)
```

Sol. 1. `$<` 代表目前的相依性項目，就是 `bas.c`。2. `$(F)` 代表目前的目標項目，就是 `fud.a` 函式庫。3. `$(*)` 代表目前的相依性項目，不過不含副檔名，就是 `bas`。

21. 請列舉兩種以子目錄來管理函式庫之 `makefile` 寫法。

Sol. 1. 先進入子目錄，並在子目錄中產生函式庫；2. 在 `makefile` 中使用變數，目錄加上 `D`，檔案加上 `F`。

22. 請說明 `makefile` 法則 `(cd mydir;$(MAKE))`。

Sol. 1. 先進入子目錄 `mydir`，並在子目錄中執行變數 `$(MAKE)` 的內容；2. 括弧是為確保，所有的處理動作都是在一個 `shell` 中。

23. 若目標項目所在目錄為 `mydir`，程式 `2.c`，目標項目 `2.o` 檔案，則 `makefile` 法則 `$(CC) $(CFLAGS) -c $(@D)/$(<F) -o $(@D)/$(@F)` 中的變數 `$(@D)`，`$(<F)`，`$(@F)` 分別為何？

Sol. 1. `$(@D)` 代表目前的目標項目所在目錄，就是 `mydir`。2. `$(<F)` 代表目前的相依性項目的檔案名稱，就是 `2.c`。3. `$(@F)` 代表目前的目標項目的檔案名稱，就是 `2.o`。

24. 若有一 `makefile` 中，目標項目 `mylib.a: 2.o 3.o` 的法則為 `ar rv mylib.a $?`，其中 `$?` 代表意義為何？

Sol. 代表需要重建（被修改）的相依性項目，就是 `2.o` 及（或）`3.o`。

25. 若有一 `makefile` 中，目標項目 `mylib.a: mydir/2.o mydir/3.o` 的法則為 `ar rv mylib.a $?`，其中 `$?` 代表意義為何？

Sol. 代表需要重建（被修改）的相依性項目，就是 `mydir/2.o` 及（或）`mydir/3.o`。

8.7 撰寫使用者手冊

- 大部分使用手冊的設計都有如下樣式：

1. Header
2. Name
3. Synopsis
4. Description
5. Options
6. Files
7. See also
8. Bugs

- 簡單的範本：應用程式使用手冊之原始碼 `myapp.1`。

```
1 .TH MYAPP 1
  .SH NAME
3 Myapp \- A simple demonstration application that does very little
  .
  .SH SYNOPSIS
5 .B myapp
  [\-option ...]
7 .SH DESCRIPTION
  .PP
9 \fImyapp\fP is a complete application that does nothing useful.
  .PP
11 It was written for demonstration purposes.
  .SH OPTIONS
13 .PP
  It doesn't have any, but let's pretend, to make this template
15 complete:
  .TP
17 .BI \-option
  If there was an option, it would not be -option.
19 .SH RESOURCES
  .PP
21 myapp uses almost no resources.
  .SH DIAGNOSTICS
23 The program shouldn't output anything, so if you find it doing so
  there's probably something wrong. The return value is zero.
25 .SH SEE ALSO
  The only other program we know with this this little
  functionality is
27 the ubiquitous hello world application.
  .SH COPYRIGHT
29 myapp is Copyright (c) 2003 Wrox Press.
  This program is free software; you can redistribute it and/or
  modify
31 it under the terms of the GNU General Public License as published
  by
  the Free Software Foundation; either version 2 of the License, or
33 (at your option) any later version.
  This program is distributed in the hope that it will be useful,
35 but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
37 GNU General Public License for more details.
  You should have received a copy of the GNU General Public License
39 along with this program; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
  02111-1307
41 USA.
  .SH BUGS
43 There probably are some, but we don't know what they are yet.
  .SH AUTHORS
45 Neil Matthew and Rick Stones
```

1. 巨集都以(.)開頭，而且都很簡短。
2. 上例第一行的 1 表示使用手冊歸屬於第一類(使用者可以操作的指令或可執行檔)。
3. 巨集格式可以 `man 7 man` 查詢

```
1 [root@dywOffice ~]# 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@dywOffice ~]# man 7 man | cat -
7 MAN(7)                Linux Programmer's Manual
                        MAN(7)
9 NAME
      man - macros to format man pages
11 SYNOPSIS
13      groff -Tascii -man file ...
      groff -Tps -man file ...
15      man [section] title
17 DESCRIPTION
19 PREAMBLE
      The first command in a man page should be
21      .TH title section date source manual,
23 SECTIONS
      Sections are started with .SH followed by the heading
      name.
25      .SH NAME
      chess \- the game of chess
27 FONTS
      The commands to select the type face are:
29      .B Bold
      .BI Bold alternating with italics
31      .BR Bold alternating with Roman
      .I Italics
33      .IB Italics alternating with bold
      .IR Italics alternating with Roman
35      .RB Roman alternating with bold
      .RI Roman alternating with italics
37      .SB Small alternating with bold
      .SM Small (useful for acronyms)
```

```

39 OTHER MACROS AND STRINGS
41   Normal Paragraphs
42     .LP      Same as .PP (begin a new paragraph).
43     .P       Same as .PP (begin a new paragraph).
44     .PP      Begin a new paragraph and reset prevailing
45               indent.
46   Relative Margin Indent
47     .RS i    Start relative margin indent:
48               moves the left margin i to the right.
49     .RE      End relative margin indent and restores
50               the previous value of the prevailing indent.
51   Indented Paragraph Macros
52     .HP i    Begin paragraph with a hanging indent.
53     .IP x i   Indented paragraph with optional hanging tag
54               x.
55     .TP i    Begin paragraph with hanging tag.
56   Hypertext Link Macros
57   Miscellaneous Macros
58     .DT      Reset tabs to default tab values (every 0.5
59               inches)..
60     .PD d    Set inter-paragraph vertical distance to d (
61               if omitted,
62               d=0.4v).
63     .SS t    Subheading t (like .SH, but used for a
64               subsection).
65   Predefined Strings
66
67   SAFE SUBSET
68     Font changes (ft and the \f escape sequence) should
69     only have the
70     values 1, 2, 3, 4, R, I, B, P, or CW.
71
72   NOTES
73
74   FILES
75     /usr/share/groff/[*/]tmac/tmac.an
76     /usr/man/whatis
77
78   BUGS
79
80   AUTHORS
81
82   SEE ALSO
83     apropos(1), groff(1), man(1), man2html(1), mdoc(7),
84     mdoc.samples(7),
85     groff_man(7), groff_www(7), whatis(1)
86
87   Linux
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

- 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`。
- 利用指令 `groff` 處理使用手冊的原始碼。

1. 指令 `groff`

```
$ groff [-Tm] file
2 -：產生Tascii ASCII 文字。
-：產生Tps  PostScript
4 -：告訴man groff 輸入一個使用手冊。
```

2. 將使用手冊 `myapp.1` 以 ASCII 文字產生：

```
[dywang@dywHome2 chapter09]$ groff -Tascii -man myapp.1
2 MYAPP(1)

    MYAPP(1)

4

6 NAME
    Myapp - A simple demonstration application that does
        very little.

8

10 SYNOPSIS
    myapp [-option ...]

12

14 DESCRIPTION
    myapp is a complete application that does nothing
        useful.

16

18    It was written for demonstration purposes.

20

22 OPTIONS
    It doesn't have any, but lets pretend, to make this
        template complete:

24
    -option
26        If there was an option, it would not be -option
        .

28
RESOURCES
```

```
30      myapp uses almost no resources.
32
33  DIAGNOSTICS
34      The program shouldn't output anything, so if you
           find it doing so
35      there's probably something wrong. The return value is
           zero.
36
37  SEE ALSO
38      The only other program we know with this this little
           functionality is
39      the ubiquitous hello world application.
40
41
42  COPYRIGHT
43      myapp is Copyright (c) 2003 Wrox Press.
44
45      This program is free software; you can redistribute it
           and/or modify it
46      under the terms of the GNU General Public License as
           published by the
47      Free Software Foundation; either version 2 of the
           License, or (at your
48      option) any later version.
49
50      This program is distributed in the hope that it will
           be useful, but
51      WITHOUT ANY WARRANTY; without even the implied
           warranty of MER-
52      CHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
           the GNU General
53      Public License for more details.
54
55      You should have received a copy of the GNU General
           Public License along
56      with this program; if not, write to the Free Software
           Foundation, Inc.,
57      675 Mass Ave, Cambridge, MA 02139, USA.
58
59
60  BUGS
61      There probably are some, but we don't know what they
           are yet.
62
63
64  AUTHORS
65      Neil Matthew and Rick Stones
66
67
68
```

```

70 ACKNOWLEDGEMENT
71     Thanks to Wrox press for publishing this book.
72
73
74

```

MYAPP
(1)

- 使用 `man` 命令查詢使用手冊

1. 先以指令 `bzip2` 將使用手冊 `myapp.1` 壓縮；

```

2 [dywang@dywHome2 chapter09]$ bzip2 -z myapp.1
  [dywang@dywHome2 chapter09]$ ls myapp.*
  myapp.1.bz2  myapp.spec

```

2. 將壓縮之使用手冊 `myapp.1.bz2` 放到 `/usr/share/man/man1` 目錄下。

```

1 [dywang@dywHome2 chapter09] cp myapp.1.bz2 /usr/share/man/
  man1/

```

練習題

1. UNIX 使用手冊處理工具為何？

Sol. `unroll`

2. Linux 使用手冊處理工具為何？

Sol. `gettext`

3. 請說明 `.TH MYAPP 1` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 指令名稱顯示在 `man page` 的開頭第一行；1 代表使用手冊歸屬於第一類；顯示結果 `MYAPP(1)`

4. 請說明 `.SH NAME` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 段落的抬頭為 `NAME`；顯示結果 `NAME`

5. 請說明 `.B myapp [\-option ...]` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 字型 `myapp [\-option ...]` 為粗體字；`\-` 為凱脫特殊字元 -

6. 請說明 `.PP \fI myapp \fP` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 開始新的一行且內縮；顯示 `myapp`；`\f` 為字型改變指令；字型改變為有下線 `I`，後再改變回無下線之正常字體 `P`

7. 請說明 `.TP .BI \-option` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 開始新的一行且外凸為樣籤；顯示 `-option` 字型為粗斜體

8. 如何以指令 `groff`，將使用手冊 `myapp.1` 以 ASCII 文字產生？

Sol. `groff -tascii -man myapp.1`

9. 要讓系統可以使用 `man` 命令查詢使用手冊 `myapp.1`，必須做那兩個動作？

Sol. 1. 壓縮 `bzip2 -z myapp.1` 2. 複製 `cp myapp.1.bz2 /usr/share/man/man1/`

8.8 實機練習題

8.8.1 練習一

1. 在家目錄下建立 `zzz` 目錄。
2. 切換工作目錄到 `zzz`
3. 下載程式碼
4. 解壓縮 `544977.tgz`。
5. 再切換工作目錄到 `544977-blp3e/chapter09/`
6. 修改 `2.c` 列出自己的學號(要換行)
7. 修改 `3.c` 列出自己的姓名(要換行)
8. 修改 `Makefile3` 的 `install`:
 - (a) 建立目錄 `/tmp/local/bin`
 - (b) 安裝至此目錄

Chapter 9

開發工具 – makefile 其他功能

9.1 多重目標項目 (target)

- 可在 makefile 中建立多個目標項目，並於 make 時指定目標項目。
- 例題：沿上例，加入 clean 目標項目，以移除不想要的目的檔 (object)。

1. 建立編譯的規則

```

1 [guest@test guest]# vi makefile
main: main.o haha.o sin_value.o cos_value.o
3     gcc -o main main.o haha.o sin_value.o cos_value.o -lm
clean:
5     rm -f main main.o haha.o sin_value.o cos_value.o
# clean 冒號之後是空白。目標項目永遠會被認為過期，所以它的法則永
    遠會被執行。

```

2. 測試標的 clean：

```

[guest@test guest]# make clean
2 rm -f main main.o haha.o sin_value.o cos_value.o

```

3. 先清除目標檔再編譯程式 main：

```

[guest@test guest]# make clean main
2 rm -f main main.o haha.o sin_value.o cos_value.o
cc -c -o main.o main.c
4 cc -c -o haha.o haha.c
cc -c -o sin_value.o sin_value.c
6 cc -c -o cos_value.o cos_value.c
gcc -o main main.o haha.o sin_value.o cos_value.o -lm

```

- 沿上例，再加入 `install` 目標項目，將完成的應用程式安裝到不同的目錄。

1. `makefile` 新的版本 `Makefile3`：

```

1 all: myapp
  # Which compiler
3 CC = gcc
  # Where to install
5 INSTDIR = /usr/local/bin
  # Where are include files kept
7 INCLUDE = .
  # Options for development
9 CFLAGS = -g -Wall -ansi
  # Options for release
11 # CFLAGS = -O -Wall -ansi
myapp: main.o 2.o 3.o
13     $(CC) -o myapp main.o 2.o 3.o
main.o: main.c a.h
15     $(CC) -I$(INCLUDE) $(CFLAGS) -c main.c
2.o: 2.c a.h b.h
17     $(CC) -I$(INCLUDE) $(CFLAGS) -c 2.c
3.o: 3.c b.h c.h
19     $(CC) -I$(INCLUDE) $(CFLAGS) -c 3.c
clean:
21     -rm main.o 2.o 3.o
install: myapp
23 # 每個命令都會啟動一個新的，所以必須加上反斜線 shell(\)，讓所有
   script 命令在同一行，且在同一個 shell 中執行。
   @if [ -d $(INSTDIR) ]; \
25     then \
       cp myapp $(INSTDIR);\
27     chmod a+x $(INSTDIR)/myapp;\
       chmod og-w $(INSTDIR)/myapp;\
29     echo "Installed in $(INSTDIR)";\
   else \
31     echo "Sorry, $(INSTDIR) does not exist";\
   fi

```

2. `&&`：前個命令成功，才執行下個命令

```

@if [ -d $(INSTDIR) ]; \
2  then \
    cp myapp $(INSTDIR) &&\
4    chmod a+x $(INSTDIR)/myapp && \
    chmod og-w $(INSTDIR)/myapp && \
6    echo "Installed in $(INSTDIR)" ;\
else \
8    echo "Sorry, $(INSTDIR) does not exist" ; false ; \
fi

```

3. 執行

```

1  $ rm *.o myapp
   $ make -f Makefile3
3  gcc -I. -g -Wall -ansi -c main.c
   gcc -I. -g -Wall -ansi -c 2.c
5  gcc -I. -g -Wall -ansi -c 3.c
   gcc -o myapp main.o 2.o 3.o
7  $ make -f Makefile3
   make: Nothing to be done for 'all'.
9  $ rm myapp
   $ make -f Makefile3 install
11 gcc -o myapp main.o 2.o 3.o
   Installed in /usr/local/bin
13 $ make -f Makefile3 clean
   rm main.o 2.o 3.o
15 $

```

練習題

1. makefile 中，是否可以有多個目標項目？
Sol. 可以。
2. 若 makefile 中有一目標項目 clean，則如何於執行 make 時指定此目標項目？
Sol. make clean
3. 若有一 makefile 中，目標項目 clean 冒號之後是空白，代表意義為何？
Sol. 目標項目永遠會被認為過期，所以它的法則永遠會被執行。
4. 若 makefile 中，多個目標項目 main, clean 與 install，要如何指定目標項目 install 執行 make？
Sol. make install
5. 若 makefile 中，多個目標項目 main, clean 與 install，make 如何"一次"執行 clean 後接著 install？
Sol. make clean install
6. 請解釋如下 makefile 片段，假設變數 INSTDIR=mydir。

```

install: myapp
    @if [ -d $(INSTDIR) ]; \
    then \
        cp myapp $(INSTDIR); \

```

```

    chmod a+x $(INSTDIR)/myapp;\
    chmod og-w $(INSTDIR)/myapp;\
    echo "Installed in $(INSTDIR)";\
else \
    echo "Sorry, $(INSTDIR) does not exist";\
fi

```

Sol. 1. 目標項目 `install` 的相依性項目有 `myapp`; 2. `@` 表示不會在標準輸出上，顯示要執行的命令; 3. 如果 `mydir` 是一個目錄，則執行複製 `myapp` 到目錄 `mydir`、改變 `mydir/myapp` 的屬性，所有使用者加上可執行屬性，其他使用者及同群組使用者刪除可寫屬性；螢幕輸出 `Installed in mydir`; 4. 其他(即 `mydir` 不是一個目錄，則螢幕輸出 `Sorry, mydir does not exist`; 5. 因為每個命令都會啟動一個新的 `shell`，所以必須加上反斜線(`\`)，讓所有 `script` 命令在同一行，且在同一個 `shell` 中執行。

7. 如下 `makefile`，`&& \` 代表意義為何？

```

chmod a+x $(INSTDIR)/myapp && \
chmod og-w $(INSTDIR)/myapp && \

```

Sol. `&&` 表示前個命令成功，才執行下個命令，`\` 讓 `script` 命令在同一行，且在同一個 `shell` 中執行。

9.2 Makefile 其他法則

- 檔尾 (suffix) 和符號 (pattern) 法則：

```

1 # 將 .old_suffix 的檔案變成 .new_suffix 的檔案
3 # 符號法則 1
  .<old_suffix>.<new_suffix>:
5
  # 符號法則 2
7 %.<old_suffix>: %.<new_suffix>:

```

- 例題：將 `.cpp` 檔案變成 `.o` 檔案：

1. 符號法則 1：

```

1 .SUFFIXES:      .cpp
  .cpp.o:
3   $(CC) -xc++ $(CFLAGS) -I$(INCLUDE) -c $<
  # gcc 的 -xc++ 旗標告訴編譯器，它是一個 C++ 原始碼檔案。
5 # 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）。

```

2. 符號法則 2:

```
1 %.o: %.cpp
    $(CC) -xc++ $(CFLAGS) -I$(INCLUDE) -c $<
```

- 內建的法則：借助檔尾 (suffix)，make 即知使用什麼法則。

1. 可以利用 -p 選項，要求 make 印出內建法則，截取部分如下：

```
[dywang@dywOffice ~]$ make -p | vi -
2 OUTPUT_OPTION = -o $@
  CXX = g++ ###(gcc)
4 COMPILE.cc = $(CXX) $(CXXFLAGS) $(CPPFLAGS) $(TARGET_ARCH) -c
  COMPILE.C = $(COMPILE.cc)
6 COMPILE.cpp = $(COMPILE.cc)

8 %.o: %.c
  # commands to execute (built-in):
10    $(COMPILE.cc) $(OUTPUT_OPTION) $<
```

2. 例題：將 .c 檔尾的檔案變成 .o 檔尾的檔案。

- (a) 編輯 foo.c，(傳統的 Hello World 程式)。

```
1 #include <stdlib.h>
2 #include <stdio.h>
  int main()
4 {
    printf("Hello World\n");
6    exit(EXIT_SUCCESS);
  }
```

- (b) 藉由 make 編譯程式，但沒有指定 makefile。

```
1 $ make foo
  cc      foo.c      -o foo
3 $
```

- (c) 若不使用內建法則，可以於 make 時，直接加入參數

```
1 $ rm foo
  $ make CC=gcc CFLAGS="-Wall -g" foo
3 gcc -Wall -g      foo.c      -o foo
  $
```

3. 可以借用內建法則，簡化 Makefile3 中之編譯法則，儲存為 Makefile4：

```
main.o: main.c a.h
2 # $(CC) -I$(INCLUDE) $(CFLAGS) -c main.c
2.o: 2.c a.h b.h
4 # $(CC) -I$(INCLUDE) $(CFLAGS) -c 2.c
3.o: 3.c b.h c.h
6 # $(CC) -I$(INCLUDE) $(CFLAGS) -c 3.c
# 註解部分為原 Makefile3 之編譯法則
```

練習題

1. 請使用兩種方式寫出，將 .cpp 的檔案變成 .o 的檔案法則。
Sol. `main.o: %.cpp` 或 `main.o: %.cpp`
2. makefile 中有一 .cpp.o 之編譯法則 `$(CC) -xc++ -c $<`，其中 `-xc++` 代表意義為何？
Sol. `gcc` 的 `-xc++` 旗標告訴編譯器，它是一個 C++ 原始碼檔案。
3. makefile 中有一 .cpp.o 之編譯法則 `$(CC) -xc++ -c $<`，其中 `$<` 代表意義為何？
Sol. 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）。
4. 如何要求 make 印出其內建法則？
Sol. `make -p`
5. 何謂 make 的內建法則？
Sol. 若 makefile 中沒指定法則，則 make 會以內建法則執行，因此可簡化 makefile。
6. make 的內建法則，編譯器使用 `cc`，若要改用 `gcc` 編譯 `foo`，但 makefile 中沒指定，需如何做？
Sol. 直接於 make 時加入參數：`make CC=gcc foo`
7. 若 makefile 中有目標項目 `main.o: main.c a.h`，但未指定其編譯法則，則 make 會怎麼執行？
Sol. 使用內建法則，產生目標項目 `main.o`

9.3 函式庫管理

- 函式庫：目標檔庫

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。

- Linux 放置函式庫之目錄：

```
1 /usr/lib64
2 /usr/lib
3 /lib64
4 /lib
5 # kernel 的函式庫放在 /lib/modules
```

- 如何判斷某個可執行的 binary 檔案含有什麼動態函式庫？

1. ldd 指令

```
1 [root@linux ~]# ldd [-vdr] [filename] 選項：
3 --version : 列印 ldd 的版本序號
4 -v : 列出所有內容資訊；
5 --help : 指令用法資訊
```

2. 找出檔案 `/usr/bin/passwd` 的函式庫資料。

```

1 [root@linux ~]# ldd /usr/bin/passwd
    linux-gate.so.1 => (0x00d19000)中間省略
3 .....
    libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x00bd6000
        )中間省
        略
5 .....

```

3. 找出函式 `/lib/libc.so.6` 的相關其他函式庫。

```

1 [root@linux ~]# ldd /lib/libc.so.6
    /lib/ld-linux.so.2 (0x00bf1000)
3    linux-gate.so.1 => (0x00632000)

5 [root@linux ~]# ldd -v /lib/libc.so.6
    /lib/ld-linux.so.2 (0x00bf1000)
7    linux-gate.so.1 => (0x00111000)

9    Version information:
    /lib/libc.so.6:
11        ld-linux.so.2 (GLIBC_2.1) => /lib/ld-linux.so
        .2
        ld-linux.so.2 (GLIBC_2.3) => /lib/ld-linux.so
        .2
13        ld-linux.so.2 (GLIBC_PRIVATE) => /lib/ld-
        linux.so.2

```

- 如何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. 在 `/etc/ld.so.conf` 寫入想要讀入快取記憶體的動態函式庫所在的目錄；
2. 利用執行檔 `ldconfig` 將 `/etc/ld.so.conf` 的資料讀入快取中；
3. 同時也將資料記錄一份在檔案 `/etc/ld.so.cache` 中。

```

1 [root@test root]# ldconfig [-f conf] [-C cache] [-p] 參數說
    明：
3 -f conf : conf 預設為 /etc/ld.so.conf
  -C : cachecache 預設為 /etc/ld.so.cache
5 -      : 列出目前在p cache 內的資料

```

4. 例題：將 `xorg` 相關函式庫加到快取記憶體中：

```

1 [root@dywHome2 ~]# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
3 /usr/X11R6/lib
/usr/lib/qt3/lib
5 [root@dywHome2 ~]# ldconfig -p | grep "libafb"
[root@dywHome2 ~]# vi /etc/ld.so.conf
7 [root@dywHome2 ~]# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
9 /usr/X11R6/lib
/usr/lib/qt3/lib
11 /usr/lib/xorg/modules # 加入此行
[root@dywHome2 ~]# ldconfig
13 [root@dywHome2 ~]# ldconfig -p | grep "libafb"
libafb.so (libc6) => /usr/lib/xorg/modules/libafb.so

```

- 建立函式庫：將多個目標檔案組合成一個函式庫檔案

1. 函式庫建立指令 `ar`。例如：

```

2 $ cc -c f1.c # 產生 f1.o
$ cc -c f2.c # 產生 f2.o
$ ar rv mlib.a f1.o f2.o # 將目標檔 f1.o 及 f2.o 插入到函式
    庫 mlib.a
4 # 選項 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。
6 # 選項 v：顯示執行操作選項的附加信息。

```

2. `nm`：查看目標檔庫檔案 `mlib.a` 的內容。

```
$ nm mlib.a
```

3. `make` 內建函式庫管理法則，例如：將 `.c` 檔案變成 `.a` 函式庫

```

1 .c.a:
    $(CC) -c $(CFLAGS) $<
3     $(AR) $(ARFLAGS) $@ $*.o
5 # 變數 $(AR) 和 $(ARFLAGS)，一般分別為命令 ar 和選項。 rv

```

4. 若有一函式庫 `fud`，包含 `bas.o` 檔案，則上例法則中的變數為

(a) `$<` 代表目前的相依性項目，就是 `bas.c`。

(b) `$@` 代表目前的目標項目，就是 `fud.a` 函式庫。

(c) `$*` 代表目前的相依性項目，不過不含副檔名，就是 `bas`。

- 實作—管理一個函式庫

1. 將 2.o 和 3.o 插入 `mylib.a` 函式庫中。新的 `Makefile5` 如下：

```
1 all: myapp
  # Which compiler
3 CC = gcc
  # Where to install
5 INSTDIR = /usr/local/bin
  # Where are include files kept
7 INCLUDE = .
  # Options for development
9 CFLAGS = -g -Wall -ansi
  # Options for release
11 # CFLAGS = -O -Wall -ansi
  # Local Libraries
13 MYLIB = mylib.a
  myapp: main.o $(MYLIB)
15 $(CC) -o myapp main.o $(MYLIB)
  $(MYLIB): $(MYLIB)(2.o) $(MYLIB)(3.o)
17 main.o: main.c a.h
  2.o: 2.c a.h b.h
19 3.o: 3.c b.h c.h
  clean:
21 -rm main.o 2.o 3.o $(MYLIB)
  install: myapp
23 @if [ -d $(INSTDIR) ]; \
    then \
25     cp myapp $(INSTDIR);\
    chmod a+x $(INSTDIR)/myapp;\
27     chmod og-w $(INSTDIR)/myapp;\
    echo "Installed in $(INSTDIR)";\
29 else \
    echo "Sorry, $(INSTDIR) does not exist";\
31 fi
```

2. 執行

```
1 $ rm -f myapp *.o mylib.a
  $ make -f Makefile5
3 gcc -g -Wall -ansi -c -o main.o main.c
  gcc -g -Wall -ansi -c -o 2.o 2.c
5 ar rv mylib.a 2.o
  a - 2.o
7 gcc -g -Wall -ansi -c -o 3.o 3.c
  ar rv mylib.a 3.o
```

```
9 | a - 3.o
   | gcc -o myapp main.o mylib.a
```

3. 試驗 3.o 的相依性項目

```
2 | $ touch c.h
   | $ make -f Makefile5
   | gcc -g -Wall -ansi -c -o 3.o 3.c
   | ar rv mylib.a 3.o
   | r - 3.o
   | gcc -o myapp main.o mylib.a
   | $
```

● 以子目錄來管理函式庫

1. 方法一：由主要的 makefile 呼叫子目錄的 makefile。

```
1 | mylib.a:
   | (cd mylibdirectory;$(MAKE))
```

- (a) 要先製作 mylib.a。
- (b) 當 make 依據這個法則建立函式庫時，它會進入子目錄 mylibdirectory；
- (c) 隨後再喚起 make 命令來管理函式庫。這會喚起一個新 shell。
- (d) 括弧是爲了確保，所有的處理動作都是在一個 shell 中。

2. 方法二：在 makefile 中使用變數，目錄加上 D，檔案加上 F。

```
2 | .c.o:
   | $(CC) $(CFLAGS) -c $(@D)/$(<F) -o $(@D)/$(@F)
   | # 在子目錄中編譯檔案，並將編譯完的目的檔留在子目錄中。
```

- (a) 若目標項目所在目錄爲 mydir，程式 2.c，目標項目 2.o 檔案，則上例法則中的變數爲
 - i. \$(@D) 代表目前的目標項目所在目錄，就是 mydir。
 - ii. \$(<F) 代表目前的相依性項目的檔案名稱，就是 2.c。
 - iii. \$(@F) 代表目前的目標項目的檔案名稱，就是 2.o。
- (b) 隨後利用相依性項目，更新現在目錄的函式庫，法則如下：

```
1 | mylib.a: mydir/2.o mydir/3.o
```

```
3 || ar rv mylib.a $?  
  || # $? 代表需要重建的相依性項目，就是 mydir/2.o 及  
  || 或() mydir/3.o
```

練習題

1. 何謂函式庫？

Sol. 集合目標檔的目標檔庫

2. 函式庫分為那兩種？

Sol. 靜態及動態函式庫

3. 何謂靜態函式庫？

Sol. 1. 附檔名通常為 `libxxx.a` 的類型；2. 整個函式庫的資料被整合到執行檔中，故可執行檔可以獨立執行，但升級時，連執行檔也需要重新編譯過一次。

4. 何謂動態函式庫？

Sol. 1. 附檔名通常為 `libxxx.so` 的類型；2. 編譯時執行檔中僅具有指向動態函式庫所在的指標，故可執行檔不能被獨立執行，而函式庫升級後，執行檔不需要進行重新編譯；3. 目前的 Linux distribution 比較傾向使用動態函式庫。

5. 通常 Linux 放置函式庫之目錄為何？

Sol. `/usr/lib` 及 `/lib`

6. 通常 Linux kernel 的函式庫放置目錄為何？

Sol. `/lib/modules`

7. 如何找出檔案 `/usr/bin/passwd` 的函式庫資料？

Sol. `ldd /usr/bin/passwd`

8. 如何找出函式 `/lib/libc.so.6` 的相關其他函式庫？

Sol. `ldd /lib/libc.so.6`

9. 如何找出函式 `/lib/libc.so.6` 的相關其他函式庫，並列出所有內容資訊？

Sol. `ldd -v /lib/libc.so.6`

10. 要將某目錄下之動態函式庫，載入快取記憶體，必須於那個檔案中加入此目錄？

Sol. `/etc/ld.so.conf`

11. 如何將 `/etc/ld.so.conf` 的資料讀入快取記憶體中？

Sol. 執行 `ldconfig`

12. 如何將 `/etc/my.so.conf` 的資料讀入快取記憶體中？

Sol. 執行 `ldconfig -f /etc/my.so.conf`

13. 如何將 `/etc/ld.so.conf` 的資料讀入快取記憶體 `/etc/my.so.cache` 中？

Sol. 執行 `ldconfig -C /etc/my.so.cache`

14. 如何列出目前在 `cache` 內的資料？

Sol. 執行 `ldconfig -p`

15. 如何查看動態函式庫 `libafb.so`，是否在快取記憶體中？

Sol. 執行 `ldconfig -p | grep "libafb"`

16. 若目錄 `/usr/lib/xorg/modules` 包含動態函式庫 `libafb.so`，如何將其載入快取記憶體，以增進動態函式庫的讀取速度？請詳列步驟。

Sol. 1. 以 `vi` 編輯檔案 `/etc/ld.so.conf` 加入一行 `/usr/lib/xorg/modules`，命令為 `vi /etc/ld.so.conf`；2. 執行 `ldconfig`；3. 查看動態函式庫 `libafb.so`，是否在快取記憶體中，`ldconfig -p | grep "libafb"`

17. 如何將目標檔 `f1.o` 及 `f2.o` 插入到函式庫 `m1ib.a`？

Sol. `ar r m1ib.a f1.o f2.o`

18. 如何將目標檔 `f1.o` 及 `f2.o` 插入到函式庫 `m1ib.a`，並顯示訊息？

Sol. `ar rv m1ib.a f1.o f2.o`

19. 如何查看目標檔庫檔案 `m1ib.a` 的內容？

Sol. `nm m1ib.a`

20. 若有一函式庫 `fud`，包含 `bas.o` 檔案，則下列 `makefile` 法則中的變數 `$(<)`，`$(@)`，`$(*)` 分別為何？

```
.c.a:
    $(CC) -c $(CFLAGS) $(<
    $(AR) $(ARFLAGS) $(@) $(*.o)
# 變數 $(AR) 和 $(ARFLAGS)，一般分別為命令 ar 和選項 rv。
```

Sol. 1. `$(<)` 代表目前的相依性項目，就是 `bas.c`。2. `$(@)` 代表目前的目標項目，就是 `fud.a` 函式庫。3. `$(*)` 代表目前的相依性項目，不過不含副檔名，就是 `bas`。

21. 請列舉兩種以子目錄來管理函式庫之 `makefile` 寫法。

Sol. 1. 先進入子目錄，並在子目錄中產生函式庫；2. 在 `makefile` 中使用變數，目錄加上 `D`，檔案加上 `F`。

22. 請說明 `makefile` 法則 `(cd mydir;$(MAKE))`。

Sol. 1. 先進入子目錄 `mydir`，並在子目錄中執行變數 `$(MAKE)` 的內容；2. 括弧是為了確保，所有的處理動作都是在一個 `shell` 中。

23. 若目標項目所在目錄為 `mydir`，程式 `2.c`，目標項目 `2.o` 檔案，則 `makefile` 法則 `$(CC) $(CFLAGS) -c $(@D)/$(<F) -o $(@D)/$(@F)` 中的變數 `$(@D)`，`$(<F)`，`$(@F)` 分別為何？

Sol. 1. `$(@D)` 代表目前的目標項目所在目錄，就是 `mydir`。2. `$(<F)` 代表目前的相依性項目的檔案名稱，就是 `2.o`。3. `$(@F)` 代表目前的目標項目的檔案名稱，就是 `2.o`。

24. 若有一 makefile 中，目標項目 `mylib.a: 2.o 3.o` 的法則為 `ar rv mylib.a $?`，其中 `$?` 代表意義為何？

Sol. 代表需要重建（被修改）的相依性項目，就是 `2.o` 及（或） `3.o`。

25. 若有一 makefile 中，目標項目 `mylib.a: mydir/2.o mydir/3.o` 的法則為 `ar rv mylib.a $?`，其中 `$?` 代表意義為何？

Sol. 代表需要重建（被修改）的相依性項目，就是 `mydir/2.o` 及（或） `mydir/3.o`。

9.4 撰寫使用者手冊

- 大部分使用手冊的設計都有如下樣式：

1. Header
2. Name
3. Synopsis
4. Description
5. Options
6. Files
7. See also
8. Bugs

- 簡單的範本：應用程式使用手冊之原始碼 `myapp.1`。

```

1  .TH MYAPP 1
   .SH NAME
3  Myapp \- A simple demonstration application that does very little
   .
   .SH SYNOPSIS
5  .B myapp
   [\-option ...]
7  .SH DESCRIPTION
   .PP
9  \fImyapp\fP is a complete application that does nothing useful.
   .PP
11 It was written for demonstration purposes.
   .SH OPTIONS
13 .PP
   It doesn't have any, but let's pretend, to make this template
15 complete:
   .TP

```



```

17 .BI \-option
   If there was an option, it would not be -option.
19 .SH RESOURCES
   .PP
21 myapp uses almost no resources.
   .SH DIAGNOSTICS
23 The program shouldn't output anything, so if you find it doing so
   there's probably something wrong. The return value is zero.
25 .SH SEE ALSO
   The only other program we know with this this little
       functionality is
27 the ubiquitous hello world application.
   .SH COPYRIGHT
29 myapp is Copyright (c) 2003 Wrox Press.
   This program is free software; you can redistribute it and/or
       modify
31 it under the terms of the GNU General Public License as published
       by
       the Free Software Foundation; either version 2 of the License, or
33 (at your option) any later version.
   This program is distributed in the hope that it will be useful,
35 but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
37 GNU General Public License for more details.
   You should have received a copy of the GNU General Public License
39 along with this program; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
       02111-1307
41 USA.
   .SH BUGS
43 There probably are some, but we don't know what they are yet.
   .SH AUTHORS
45 Neil Matthew and Rick Stones

```

1. 巨集都以(.)開頭，而且都很簡短。
2. 上例第一行的 1 表示使用手冊歸屬於第一類(使用者可以操作的指令或可執行檔)。
3. 巨集格式可以 `man 7 man` 查詢

```

1 [root@dywOffice ~]# 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@dywOffice ~]# man 7 man | cat -
7 MAN(7)                  Linux Programmer's Manual
                           MAN(7)

```

```

9  NAME
    man - macros to format man pages
11
12  SYNOPSIS
13      groff -Tascii -man file ...
14      groff -Tps -man file ...
15      man [section] title
16
17  DESCRIPTION
18
19  PREAMBLE
    The first command in a man page should be
21      .TH title section date source manual,
22
23  SECTIONS
    Sections are started with .SH followed by the heading
    name.
25      .SH NAME
        chess \- the game of chess
26
27  FONTS
    The commands to select the type face are:
29      .B Bold
    .BI Bold alternating with italics
31      .BR Bold alternating with Roman
    .I Italics
33      .IB Italics alternating with bold
    .IR Italics alternating with Roman
35      .RB Roman alternating with bold
    .RI Roman alternating with italics
37      .SB Small alternating with bold
    .SM Small (useful for acronyms)
39
40  OTHER MACROS AND STRINGS
41      Normal Paragraphs
        .LP Same as .PP (begin a new paragraph).
43      .P Same as .PP (begin a new paragraph).
        .PP Begin a new paragraph and reset prevailing
            indent.
45      Relative Margin Indent
        .RS i Start relative margin indent:
47            moves the left margin i to the right.
        .RE End relative margin indent and restores
49            the previous value of the prevailing indent.
50      Indented Paragraph Macros
        .HP i Begin paragraph with a hanging indent.
51      .IP x i Indented paragraph with optional hanging tag
            x.
53      .TP i Begin paragraph with hanging tag.
54      Hypertext Link Macros
55      Miscellaneous Macros

```

```

.DT      Reset tabs to default tab values (every 0.5
        inches)..
57 .PD d   Set inter-paragraph vertical distance to d (
        if omitted,
        d=0.4v).
59 .SS t   Subheading t (like .SH, but used for a
        subsection).
        Predefined Strings
61
SAFE SUBSET
63      Font changes (ft and the \f escape sequence) should
        only have the
        values 1, 2, 3, 4, R, I, B, P, or CW.
65 NOTES
67 FILES
        /usr/share/groff/[*/]tmac/tmac.an
69      /usr/man/whatis
BUGS
71
AUTHORS
73
SEE ALSO
75      apropos(1), groff(1), man(1), man2html(1), mdoc(7),
        mdoc.samples(7),
        groff_man(7), groff_www(7), whatis(1)
77
Linux                                     2004-07-27
                                     MAN(7)

```

- 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`。
- 利用指令 `groff` 處理使用手冊的原始碼。

1. 指令 `groff`

```

$ groff [-Tm] file
2  -:產生Tascii ASCII 文字。
   -:產生Tps  °PostScript
4  -:告訴man groff 輸入一個使用手冊。

```

2. 將使用手冊 `myapp.1` 以 ASCII 文字產生：

```
[dywang@dywHome2 chapter09]$ groff -Tascii -man myapp.1
```

```
2 MYAPP(1)
   MYAPP(1)
4
6 NAME
   MyApp - A simple demonstration application that does
       very little.
8
10 SYNOPSIS
   myapp [-option ...]
12
14 DESCRIPTION
   myapp is a complete application that does nothing
       useful.
16
18   It was written for demonstration purposes.
20
22 OPTIONS
   It doesn't have any, but lets pretend, to make this
       template complete:
24
26   -option
       If there was an option, it would not be -option
       .
28
30 RESOURCES
   myapp uses almost no resources.
32
34 DIAGNOSTICS
   The program shouldn't output anything, so if you
       find it doing so
       there's probably something wrong. The return value is
       zero.
36
38 SEE ALSO
   The only other program we know with this this little
       functionality is
40   the ubiquitous hello world application.
42
44 COPYRIGHT
```

```

44      myapp is Copyright (c) 2003 Wrox Press.

46      This program is free software; you can redistribute it
        and/or modify it
        under the terms of the GNU General Public License as
        published by the
48      Free Software Foundation; either version 2 of the
        License, or (at your
        option) any later version.

50      This program is distributed in the hope that it will
        be useful, but
52      WITHOUT ANY WARRANTY; without even the implied
        warranty of MER-
        CHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
        the GNU General
54      Public License for more details.

56      You should have received a copy of the GNU General
        Public License along
        with this program; if not, write to the Free Software
        Foundation, Inc.,
58      675 Mass Ave, Cambridge, MA 02139, USA.

60
62      BUGS
        There probably are some, but we don't know what they
        are yet.

64
66      AUTHORS
        Neil Matthew and Rick Stones

68
70      ACKNOWLEDGEMENT
        Thanks to Wrox press for publishing this book.

72
74

```

MYAPP
(1)

- 使用 `man` 命令查詢使用手冊

1. 先以指令 `bzip2` 將使用手冊 `myapp.1` 壓縮；

```

2  [dywang@dywHome2 chapter09]$ bzip2 -z myapp.1
   [dywang@dywHome2 chapter09]$ ls myapp.*

```

```
myapp.1.bz2 myapp.spec
```

2. 將壓縮之使用手冊 myapp.1.bz2 放到 /usr/share/man/man1 目錄下。

```
1 [dywang@dywHome2 chapter09] cp myapp.1.bz2 /usr/share/man/
  man1/
```

練習題

1. UNIX 使用手冊處理工具為何？

Sol. `nroff`

2. Linux 使用手冊處理工具為何？

Sol. `groff`

3. 請說明 `.TH MYAPP 1` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 指令名稱顯示在 `man page` 的開頭第一行；1 代表使用手冊歸屬於第一類；顯示結果 MYAPP(1)

4. 請說明 `.SH NAME` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 段落的抬頭為 NAME；顯示結果 NAME

5. 請說明 `.B myapp [-option ...]` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 字型 `myapp [-option ...]` 為粗體字；`-` 為斜體特殊字元

6. 請說明 `.PP \fImyapp\fP` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 開始新的一行且內縮；顯示 `myapp`；`\f` 為字型改變指令；字型改變為有下線 `I`，後再改變回無下線之正常字體 `P`

7. 請說明 `.TP .BI \-option` 在 Linux 應用程式使用手冊之原始碼中之意義及結果。

Sol. 開始新的一行且外凸為標籤；顯示 `-option` 字型為粗斜體

8. 如何以指令 `groff`，將使用手冊 myapp.1 以 ASCII 文字產生？

Sol. `groff -tascii -man myapp.1`

9. 要讓系統可以使用 `man` 命令查詢使用手冊 myapp.1，必須做那兩個動作？

Sol. 1.壓縮 `bzip2 -z myapp.1` 2.複製 `cp myapp.1.bz2 /usr/share/man/man1/`

9.5 實機練習題

9.5.1 練習一

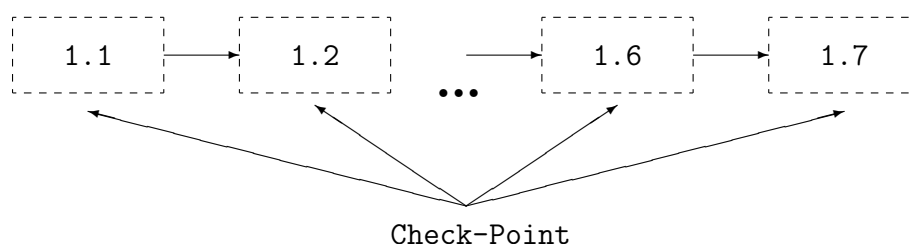
1. 在家目錄下建立 `zzz` 目錄。
2. 切換工作目錄到 `zzz`
3. 下載程式碼
4. 解壓縮 `544977.tgz`。
5. 再切換工作目錄到 `544977-blp3e/chapter09/`
6. 修改 `myapp.1` 手冊，將 `AUTHORS` 內容改成 `1000-xxxxxxx`，`xxxxxxx` 為自己的學號。
7. 修改 `Makefile3` 的 `install`:
 - (a) 建立目錄 `/tmp/local/man`
 - (b) 將手冊安裝至此目錄

Chapter 10

*RCS 版本控制系統

10.1 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，需要重視的修改、暫存、測試以期得到正確的結果，而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。
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. 存取名單記錄着存取權使用者。
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。

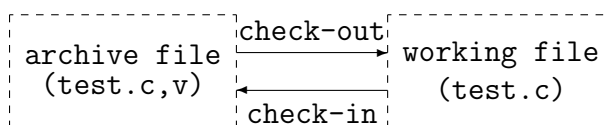
10.2 RCS 基本功能

- 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。



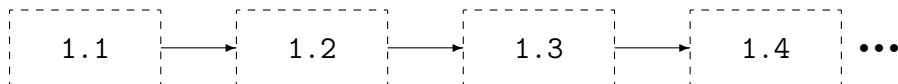
(a) check in (ci)：將檔案版本寫入管理檔案的動作。

(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)

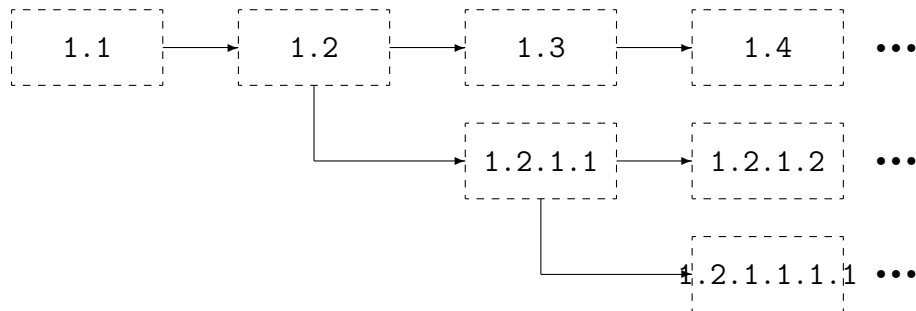
1. 在 RCS 的檔案中每一個版本，都會有一個版本序號以區分各個版本；
2. 對於特定版本的存取都利用版本序號來指定；
3. 版本序號的設定是以 m.n 為單位，m 為主要號碼 (major number)，n 為次要號碼 (minor number)；
4. 預設的版本序號從 1.1 開始。



- 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

```

1 [root@dywHome2 ~]# rcs [-Aabeilmnotu] parameter
2 -A file 將其它檔案的可存取名單寫入該檔的可存取名單
3 -a name 將使用者寫入可存取名單
4 -b branch number 設定主支 ( default branch )
5 -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
7 -l[number] lock 該版本但不取出
8 -m 更改檔案版本的附注 ( log ) 的內容
9 -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
11 -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
13 # 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
2 /*
3     This is an important file for managing the project.
4     It implements the canonical "Hello World" program.
5 */

```

```

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

```

3. 初始 RCS 檔案管理：使用 `rscs -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

```

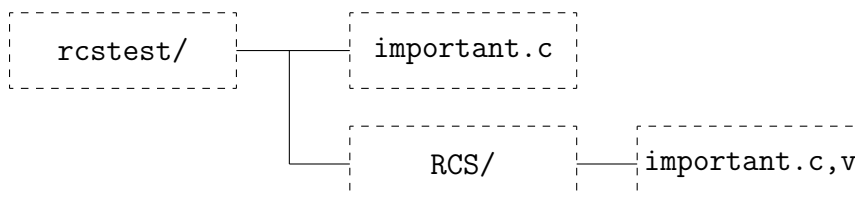
4. `rscs` 命令後會產生一個新的唯讀檔案，其檔尾為(,v)：

```

[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 |

```

```

8 [dywang@dywOffice testrcs]$ rcs -i important.c
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
9
desc
11 @This is an important demonstration file
@

```

● 將程式內容寫入 archive file

1. ci (check in) 命令

```

2 [root@dywHome2 ~]# ci [-dfwmnrtlu] parameter
-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
10 1.1
   date 2008.03.05.06.06.15;   author dywang; state Exp;
   branches;
12 next ;
14
16 desc
   @This is an important demonstration file
   @
18
20 1.1
   log
22 @Initial revision
   @
```



```

24 | text
   | @/*
26 |     This is an important file for managing the project.
   |     It implements the canonical "Hello World" program.
28 | */
30 | #include <stdlib.h>
   | #include <stdio.h>
32 |
   | int main()
34 | {
   |     printf("Hello World\n");
36 |     exit(EXIT_SUCCESS);
   | }
38 |
   | @

```

- 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

```

1 | [root@dywHome2 ~]# co [-dfjlprwu] parameter
   | -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
   | -j 取出並整合指定的版本內容
   | -l          lock 取出版本
   | -p 取出版本內容至標準輸出      ( standard output )
   | -r number 指定取出版本
   | -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
   | -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");
       exit(EXIT_SUCCESS);
16 }

```

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
9 1.2
  date      2008.03.05.06.23.52;    author dywang;  state Exp;
11 branches;
  next      1.1;
13
  1.1
15 date      2008.03.05.06.06.15;    author dywang;  state Exp;
  branches;
17 next      ;
19
  desc
21 @This is an important demonstration file
  @
23
25 1.2
  log
27 @Added an extra line to be printed out
  @
29 text
  @/*
31     This is an important file for managing the project.
     It implements the canonical "Hello World" program.
33 */
35 #include <stdlib.h>
  #include <stdio.h>
37
  int main()
39 {
     printf("Hello World\n");
41     printf("This is an extra line added later\n");
     exit(EXIT_SUCCESS);
43 }
```

```

45 | @
47 |
49 | 1.1
    | log
    | @Initial revision
51 | @
    | text
53 | @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。

```

[dywang@dywOffice testrcs]$ rlog important.c
2
RCS file: RCS/important.c,v
4 Working file: important.c
head: 1.2
6 branch:

```

```

8 locks: strict
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;
Initial revision
22 =====
# 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
RCS file: RCS/important.c,v
4 Working file: important.c
head: 1.2
6 branch:
locks: strict
8 access list:
symbolic names:
10 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。

```

2 [dywang@dywOffice testrcs]$ co important.c
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。

```

2 [dywang@dywOffice testrcs]$ rcs -l1.2 important.c
RCS file: RCS/important.c,v
1.2 locked
4 done
[dywang@dywOffice testrcs]$ ci important.c
6 RCS/important.c,v <-- important.c

```

```
8 new revision: 1.3; previous revision: 1.2
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。

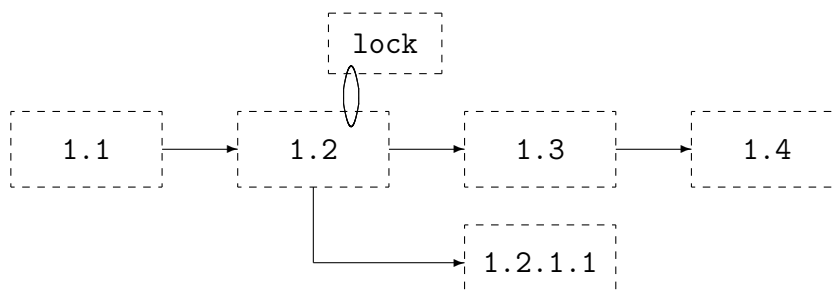
```

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
    int main()
16 {
    printf("Hello World\n");
18     printf("This is an extra line added later\n");
    printf("test for branch\n");
20     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

```

```

5 | enter log message, terminated with single '.' or end of file:
   | >> test for branch
   | >> .
7 | done

```

5. 查看版本訊息

```

1 | [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
  | date: 2008/03/05 06:23:52;  author: dywang;  state: Exp;
  |    lines: +1 -0
21 | branches: 1.2.1;
  | Added an extra line to be printed out
23 | -----
25 | revision 1.1
  | date: 2008/03/05 06:06:15;  author: dywang;  state: Exp;
  | Initial revision
27 | -----
29 | revision 1.2.1.1
  | 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. 輸入一撇(`'`)，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. 未鎖定。
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. 寫入 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. 識別關鍵字串所替方式為 `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. 沒有變化，照舊寫入

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 `rlog -t 1.2 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 -d 1.3 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 -ul 1.3 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. 1.6.1.1

10.3 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)
 +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
 例如：執行 `co -r first`，則 \$Name\$ 展開
 為 'Name: first'.
 \$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.
*/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
13 # 版本(Revision): 1.5
14 # 日期與時間(Date): 2008/03/07 03:30:31
15 # 作者(Author): dywang
16 # 狀態(State): Exp
17 # 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
6 /*
7  * $Id$
   This is an important file for managing the project.
9  It implements the canonical "Hello World" program.
10 */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.
*/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
important:
ident warning: no id keywords in important
```

2. 編輯程式 important.c，加入字串變數 rcsid[] = "\$Id\$"

```
1 [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
#include <stdlib.h>
#include <stdio.h>
static char const rcsid[] = "$Id$";
12
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  
int main()  
25 {  
    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
co RCS/important.c,v important.c
3 RCS/important.c,v --> important.c
revision 1.5
5 done
cc -c -o important.o important.c
7 cc important.o -o important
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
2 total 20
  -rw-r--r-- 1 dywang users  21 Mar  7 10:16 commentfile
4  -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~
6  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. 關於該版本的簡短訊息

8. RCS 識別關鍵字串 `$Name$`，產生什麼資訊？

Sol. 檔名名稱

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. `在式檔案的檔頭`

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. `版本字號 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. `版本狀態為已釋出`

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. `目前版本檔案名為 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 -kkv 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 -kkv 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 -kkvl 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 -kkimportant.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 -rv 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.o` 及 `important.c` 檔案。

10.4 標記符號

- 何謂標記符號(mark symbol)？

- 標記符號是指一個版本除 revision number 外賦予的符號名稱。
- 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。
- 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：

- 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。

```
[dywang@dywOffice testrcs]$ rcs -nmarkSymbol1_6:1.6 important
.c
2 RCS file: RCS/important.c,v
  done
```

- 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

```

```

5  Working file: important.c
   head: 1.6
   branch:
7  locks: strict
   access list:
9  symbolic names:
      r1_0_head: 1.2
11     r1_0: 1.2
      mark: 1.6
13     markSymbol1_6: 1.6
   keyword substitution: kv
15 total revisions: 7
=====

```

- 如何取消標記符號？

1. 以 `rcs` 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]$ rcs -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 `rcs` 指令一次取消多個標記符號

```
1 [dywang@dywOffice testrcs]$ rcs -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:
```



```

11         r1_5: 1.5
        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 爲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，以利了解各版本之關係。

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 -nmark$1_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. \$ \therefore \therefore @ \$ 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 \cdot 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 -rmarkS1_6:1.6 -rmarkS1_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 -rr1_3:1.3 -rr1_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 -rmarkS1_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 -rmarkS1_6_0 important.c`

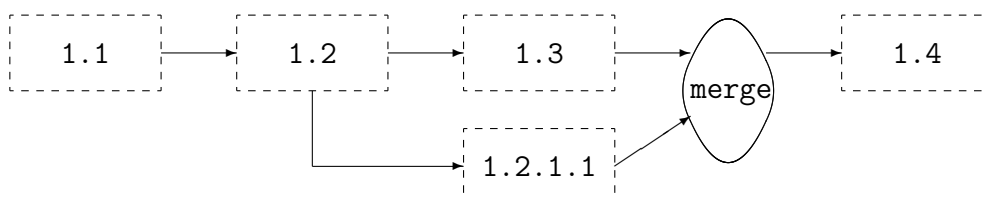
17. 如何使用 rcs 指令，一次取消程式檔案 important.c 中之標記符號 r1_0 及 r1_0_had？

Sol. `rcs -rr1_0 -rr1_0_head important.c`

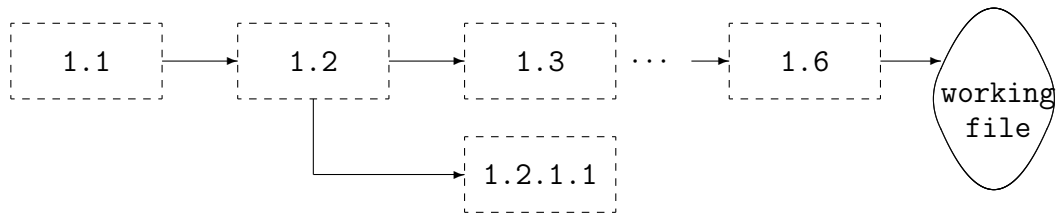
10.5 版本比較與整合

- 為何要比較與整合版本？例如：

- 在版本 1.3 發現錯誤(bug)；
- 回頭找尋錯誤發生的階段，發現錯誤是在版本 1.2；
- 在除錯完成後產生了校正版本 1.2.1.1。
- 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
2 =====
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

```

[dywang@dywOffice testrcs]$ rcsdiff -r1.5 -r1.6 -kkv1
  important.c
2 =====
RCS file: RCS/important.c,v
4 retrieving revision 1.5
  retrieving revision 1.6
6 diff -r1.5 -r1.6
  2c2
8 < * $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 [-kkkkv1r]
  -kk 整合檔案版本不包括 (IDKeyword)
3 -kkv1 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

```
[dywang@dywOffice testrcs]$ co -l -r1.6 important.c
2 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. 除錯完成產生修正版本 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. 向版本比較差異為：第 1 行被刪除，第 2 行新增。

7. 使用 `rcsdiff` 指令，出現訊息 `13a16,17` 代表意義為何？

Sol. 向版本比較差異為：第 13 行被刪除，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`

10.6 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。

1. 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
16 [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
2 RCS file: RCS/important.c,v
3 done

```

2. 查看存取名單

```

1 [dywang@dywOffice testrcs]$ rlog -h important.c
2
3 RCS file: RCS/important.c,v
4 Working file: important.c
5 head: 1.6
6 branch:
7 locks: strict
8 access list:
9     dywang
10     dywtest
11 symbolic names:
12     r1_5: 1.5
13     r1_3: 1.3
14     markSymbol1_6: 1.6
15 keyword substitution: kv
16 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

```
1 [dywang@dywOffice testrcs]$ rcs -e important.c
2 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 -n 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 11

*使用 QT 設計 KDE 視窗程式

11.1 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	http:// gbuilder.sourceforge.net/
Anjuta	GNOME 的 IDE 開發環境	http:// anjuta.sourceforge.net/
Klint	KDE 的 IDE 開發環境	http:// klint.sourceforge.net/
QtEZ	KDE 的 IDE 開發環境	http://projects.uid0.sk/qtez/index.php
RHIDE	文字模式的 IDE 開發環境	http://www.rhide.com/
CRiSP	商用的程式設計編輯器	http://www.crisp.com/
SlickEdit	商用的多程式設計編輯器	http://www.slickedit.com/
Kylix	商用的 C++ 和 Delphi 的 IDE 開發環境	http://www.borland.com/kylix
Eclipse	Java 工具平台和 IDE	http://www.eclipse.org

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. 利用所見即所得的方式，產生 QT 程式的 GUI 程式碼。

11.2 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

```

```

5 | libqt3-devel-3.3.6-18.4mdv2007.0.i586
  | 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/

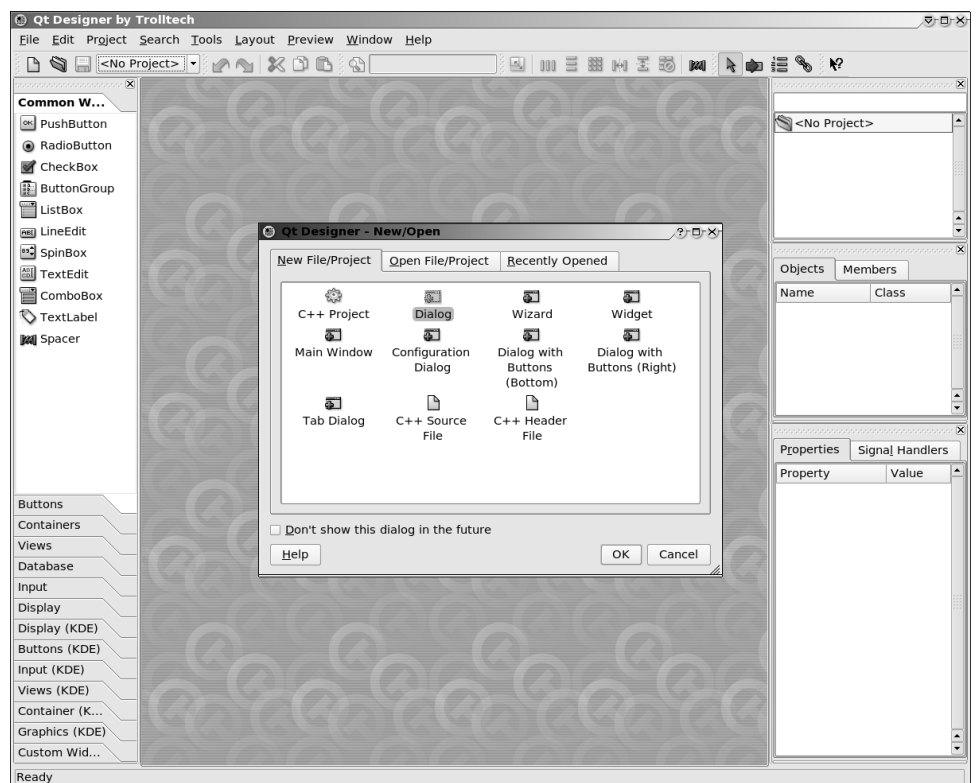
```

3. 執行命令：

```

2 | [dywang@dywOffice ~]$ ll /usr/bin/designer-qt3
  | 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。

11.3 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

```
3 Applications->Programming->Glade Interface Designer文字界面
5 [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 1. table1 Rows: 5 -> 6
2 2. insert ok button in table1 (0,5)
3 3. rename ok button Label and Name to Sqrt
4 4. set Sqrt button clicked Signals to on_Sqrt_clicked
5 5. vim calculatorglade.py
6
7 # coding: utf-8
8 import sys
9 import math # 匯入數學模組
10 try:
11     import pygtk
12     pygtk.require('2.0')
13 except:
14     pass
15 .... omission ....
16
17 class Calculator:
18     .... omission ....
19     "on_Add_clicked" : self.displayAdd,
```

```

20 ||         "on_Sqrt_clicked" : self.displaySqrt, # 定義按鈕接受函
           數sqrt
.... 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): #定義函數displaySqrt
        self.compute("Sqrt")

```

6. 執行

```
1 [dywang@deyu glade]$ python calculatorglade.py
```

11.4 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，提供應用程式的主要視窗功能。 QMainWindow
   # 若需要一個對話窗，要繼承。 QDialog
3  class MyWindow : public QMainWindow
   {
5      Q_OBJECT
   public:
7      MyWindow();
      virtual ~MyWindow();
9      signals:
      void A_Signal();
11 # signals A_Signal() 沒指定參數
   private slots:
13     void doSomething();
   # 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

```
2 void MyWindow::doSomething()
   {
   // 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：

```
1 #include <qmainwindow.h>
2 class ButtonWindow : public QMainWindow
3 {
4     Q_OBJECT
5     public:
6         ButtonWindow(QWidget *parent = 0, const char *name = 0);
7         virtual ~ButtonWindow();
8     private slots:
9         void Clicked();
10 };
```

2. 編輯 ButtonWindow 建構函式

```
1 ButtonWindow::ButtonWindow(QWidget *parent, const char *name)
2     : QMainWindow(parent, name)
3 {
4     # setCaption 是 QMainWindow 的成員函式，可設定視窗標題。
5     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。
7     QPushButton *button = new QPushButton("Click Me!", this, "
8         Button1");
9     # 設定按鈕的幾何大小。
10    button->setGeometry(50,30,70,20);
11    # 將按鈕的 clicked signal 連結到 Clicked() slot 中。
12    connect (button, SIGNAL(clicked()), this, SLOT(Clicked()));
13 }
```

3. 編輯 ~ButtonWindow 解構函式

```
1 ButtonWindow::~ButtonWindow()
2 {
3     # 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()：

```
1 void ButtonWindow::Clicked(void)
2 {
3     std::cout << "clicked!\n";
4 }
```

5. 編輯 main 程式 ButtonWindow.cpp

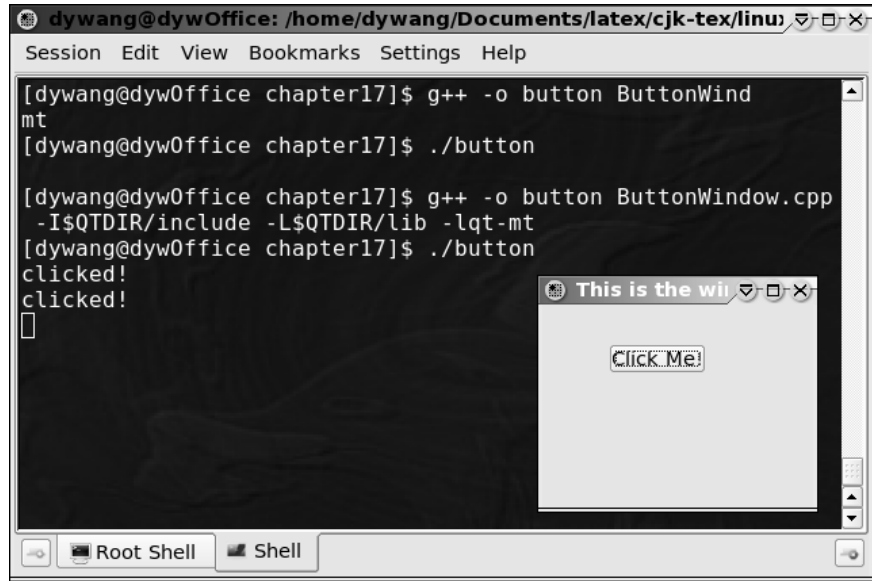
```
1  #include "ButtonWindow.moc"
2  #include <qpushbutton.h>
3  #include <qapplication.h>
4  #include <iostream>
5
6  int main(int argc, char **argv)
7  {
8      QApplication app(argc,argv);
9      # 產生一個 ButtonWindow 的物件，設定應用程式的主視窗，並將視窗
10     顯示在螢幕上。
11     ButtonWindow *window = new ButtonWindow();
12     # 設定應用程式的主視窗。
13     app.setMainWidget(window);
14     # 將視窗顯示在螢幕上。
15     window->show();
16     return app.exec();
17 }
```

6. 編譯前，執行前置處理器 moc：

```
1  $ moc ButtonWindow.h -o ButtonWindow.moc
2
3  # Qt 的 MOC( Meta-Object System )
4  ## 標準的 C++ 無法提供 signal/slot 連結所需之 meta 訊息。
5  ## 標頭檔若包含 \verb|Q_OBJECT| 巨集，MOC 會解析巨集定義，
6  ## 並產生 Qt meta-object 相關的 C++ 程式碼。
7  ## 使用 qmake 產生，就會包括 Makefile 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
```



```

dywang@dywOffice: /home/dywang/Documents/latex/cjk-tex/linu
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!

```

8. 執行程式：

- 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` 要改變大小时，必須人工呼叫 `QWidget::resizeEvent()`。

- `QVBoxLayout` 建構函式說明：

1. `QVBoxLayout` 建構函式 (`QHBoxLayout` 也擁有相同的 API)。

```
2  # QLayout 的 parent 參數，可以是 widget 或其他。 QLayout  
3  QVBoxLayout::QVBoxLayout (QWidget *parent, int margin,  
4                             int spacing, const char *name)  
5  QVBoxLayout::QVBoxLayout (QLayout *parentLayout, int spacing,  
6                             const char * name)  
7  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  QBoxLayout::addWidget (QWidget *widget, int stretch = 0,  
3                          int alignment = 0 )  
4  QBoxLayout::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 `QBoxLayout` 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>  
2  class LayoutWindow : public QMainWindow  
3  {  
4      Q_OBJECT  
5  public:  
6      LayoutWindow(QWidget *parent = 0, const char *name = 0);  
7      virtual ~LayoutWindow();  
8  };
```

2. 程式 LayoutWindow.cpp °

```
1  #include <qapplication.h>
2  #include <qlabel.h>
3  #include <qlayout.h>
4  #include "LayoutWindow.moc"
LayoutWindow::LayoutWindow(QWidget *parent, const char *name)
    :
6  QMainWindow(parent, name)
{
8      this->setCaption("Layouts");

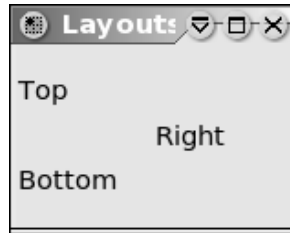
10     # 因為不能直接將 QLayout 加到，所以產生一個假
        的 QMainWindow ° QWidget
        QWidget *widget = new QWidget(this);
12     setCentralWidget(widget);
        QHBoxLayout *horizontal = new QHBoxLayout(widget, 5, 10, \
14         horizontal);
        QVBoxLayout *vertical = new QVBoxLayout();
16     QLabel* 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：

11.5 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,
4                          const char *name = 0 )
  QLineEdit::QLineEdit (const QString &contents,
6                          const QString &inputMask,
                          QWidget *parent, const char *name = 0
                          )
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 lineEdit->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) `\AAAAnn-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。指定行數、列數、邊界設定和間
   |   隔。 widget
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，必須告知行編號、列編號，而起始值為 ( , ) 代表
   |   左上方的格子。 widget00
   |     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();
   | }

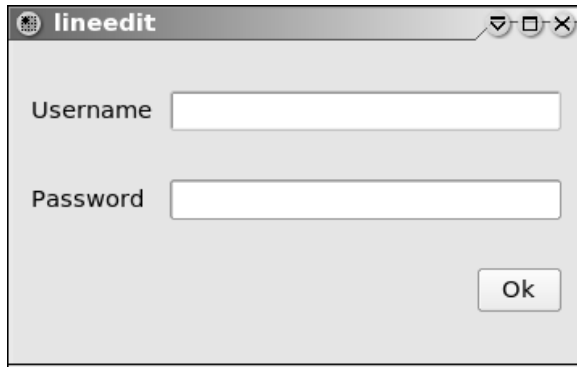
```

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：

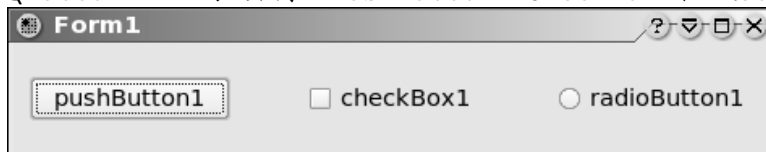
```

#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：PushButtons、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 Cancele。

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 ()
```

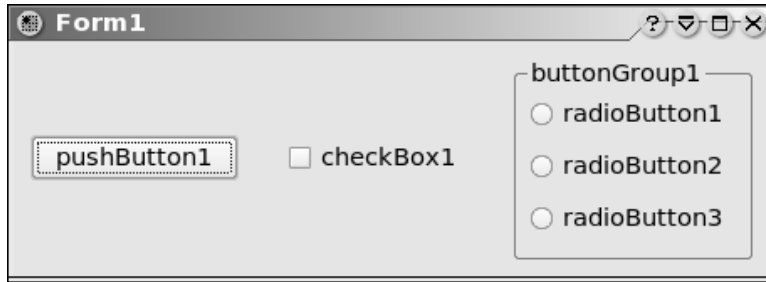
2. 一般只有 on 或 off 狀態資訊；

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, "checkbox");
    |     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：

```

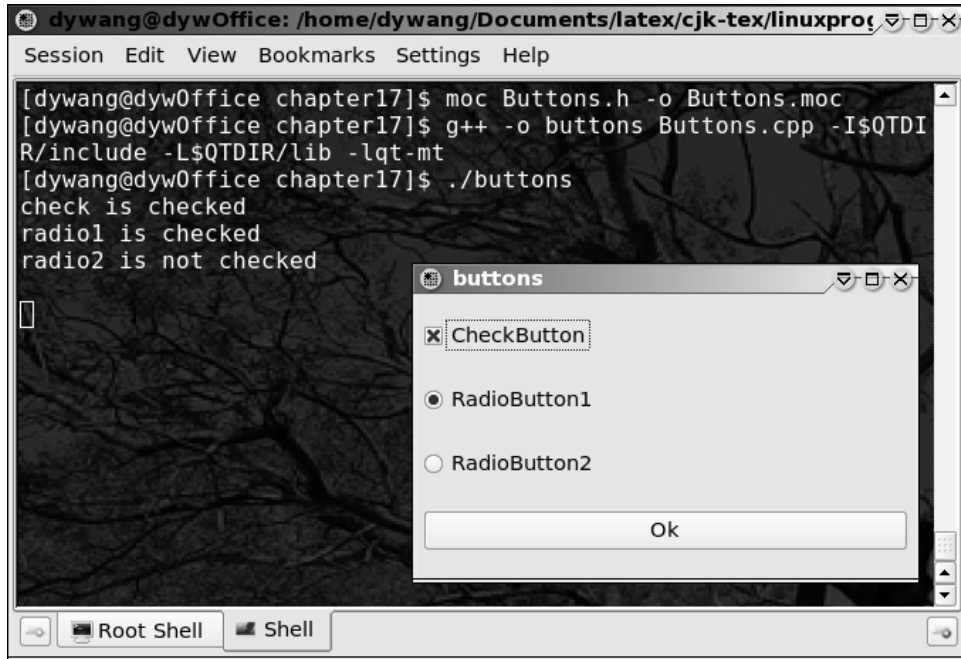
1 | void Buttons::PrintActive(QButton *button)

```

```
3 {
4     if (button->isOn())
5         std::cout << button->name() << " is checked\n";
6     else
7         std::cout << button->name() << " is not checked\n";
8 }
9 void Buttons::Clicked(void)
10 {
11     PrintActive(checkbox);
12     PrintActive(radiobutton1);
13     PrintActive(radiobutton2);
14     std::cout << "\n";
15 }
16 int main(int argc, char **argv)
17 {
18     QApplication app(argc,argv);
19     Buttons *window = new Buttons();
20     app.setMainWidget(window);
21     window->show();
22     return app.exec();
23 }
```

4. 程式執行結果。

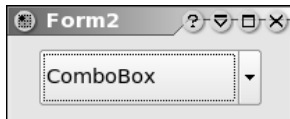
```
2 [dywang@dywOffice chapter17]$ moc Buttons.h -o Buttons.moc
3 [dywang@dywOffice chapter17]$ g++ -o buttons Buttons.cpp\
4 >-I$QTDIR/include -L$QTDIR/lib -lqt-mt
5 [dywang@dywOffice chapter17]$ ./buttons
6 check is checked
radio1 is checked
radio2 is not checked
```



- 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，指定 TRUE 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 QList 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 QList &list, int index = -1)
9  void      insertStrList (const QList *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：

```

2  #include "ComboBox.moc"
   #include <qlayout.h>
   #include <iostream>

```



```

4 | ComboBox::ComboBox(QWidget *parent, const char *name) :
   | QMainWindow(parent, name)
6 | {
   |     QWidget *widget = new QWidget(this);
8 |     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，會被信號傳送。 s
   | 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();
   | }

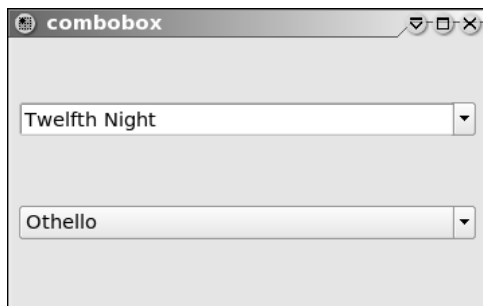
```

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：

```
QListViewItem::QListViewItem ( QListView * parent, QString
    label1,
2  QString label2 = QString::null, QString label3 = QString::
    null,
  QString label4 = QString::null, QString label5 = QString::
    null,
4  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：

```
QListViewItem *toplevel = new QListViewItem(view, "Left Data"
2  ,
  "Right Data");
```

3. 將列元件傳入 toplevel 子節點：

```
2 new QListViewItem(toplevel, "Left Data", "Right Data");
                                     // 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
18     int main(int argc, char **argv)
19     {
20         QApplication app(argc,argv);
21         ListView *window = new ListView();
22         app.setMainWidget(window);
23         window->show();
24         return app.exec();
25     }

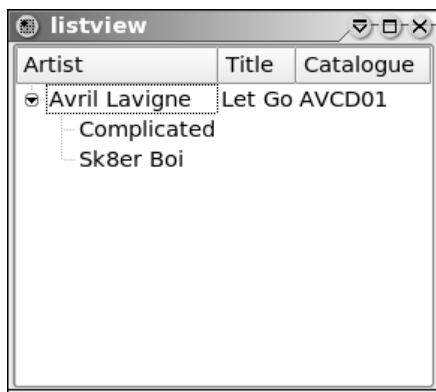
```

2. 編譯和執行 ListView

```

2 [dywang@dywOffice chapter17]$ moc ListView.h -o ListView.moc
3 [dywang@dywOffice chapter17]$ g++ -o listview ListView.cpp \
4 -I$QTDIR/include -L$QTDIR/lib -lqt-mt
5 [dywang@dywOffice chapter17]$ ./listview

```



11.6 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"));
    hbox->addWidget(new QLineEdit());
8    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  {
```

```
16    //Do some other processing
    }
```

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。

```
2    MyDialog *dialog = new MyDialog(this, "mydialog");
    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：

```
2    #include <qmessagebox.h>
    int information (QWidget *parent, const QString &caption,
4        const QString &text,
        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)
    int critical     (QWidget *parent, const QString &caption,
8        const QString &text,
        int button0, int button1, int button2=0)
10
```

2. Each of the buttons, button0, button1 and button2 may be set to one of the following values:

```
2    QMessageBox::Ok
    QMessageBox::Cancel
    QMessageBox::Yes
4    QMessageBox::No
    QMessageBox::Abort
6    QMessageBox::Retry
    QMessageBox::Ignore
```

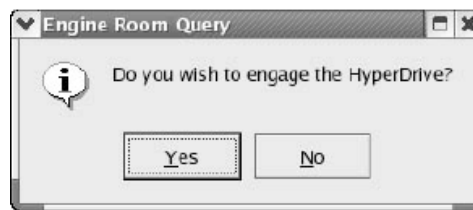
3. 實例：QMessageBox 的使用

```
1    int result = QMessageBox::information(this,
        "Engine Room Query",
3        "Do you wish to engage the HyperDrive?",
```

```

5         QMessageBox::Yes | QMessageBox::Default,
        QMessageBox::No | QMessageBox::Escape);
switch (result) {
7     case QMessageBox::Yes:
        hyperdrive->engage();
9         break;
    case QMessageBox::No:
11         // do something else
        break;
13 }

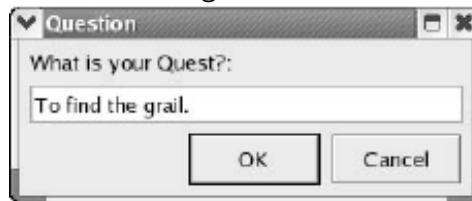
```



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,

```

```
15         bool * ok = 0, QWidget * parent = 0,  
            const char * name = 0)  
double getDouble (const QString &caption, const QString &  
    label,  
17         double num = 0,  
            double from = -2147483647, double to =  
                2147483647,  
19         int decimals = 1, bool * ok = 0,  
            QWidget * parent = 0,  
21         const char * name = 0 )
```

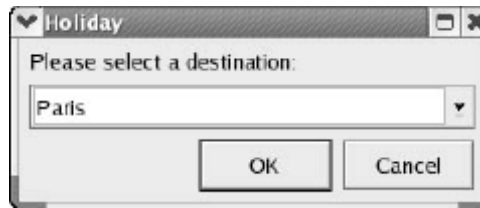
3. 實例：輸入一行的文字：

```
1 bool result;  
  QString text = QDialog::getText("Question",  
3                                     "What is your Quest?:",  
                                     "",  
                                     QLineEdit::Normal,  
5                                     QString::null, &  
                                         result, this,  
                                     "input" );  
7 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 = QDialog::getItem("Holiday",  
5                                     "Please select a  
                                     destination:",  
                                     options, 1, TRUE, &  
7                                     result,  
                                     this, "combo");  
9 if (result)  
    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
  QMAKE_LIBS_X11 += -lkdeui -lkdecore
7 SOURCES = main.cpp window.cpp
  HEADERS = window.h

```

3. 產生 makefile。

```
$qmake file.pro -o Makefile
```

11.7 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， slotslot 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：

```
2 new_file->plug(a_menu);  
  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

```

#include <kmainwindow.h>
2 class KDEMenu : public KMainWindow
{
4     Q_OBJECT
    public:
6         KDEMenu(const char * name = 0);
    private slots:
8         void newFile();
        void aboutApp();
10 };

```

2. 在 KDEMenu.cpp 中引入 widget 相關的檔案：

```

#include "KDEMenu.h"
2 #include <kapp.h>
#include <kaction.h>
4 #include <kstdaccel.h>
#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();
}
13
15 int main(int argc, char **argv)
{
17 KApplication app( argc, argv, "cdapp" );
KDEMenu *window = new KDEMenu("kdemenu");
app.setMainWidget(window);
19 window->show();
return app.exec();
21 }
```

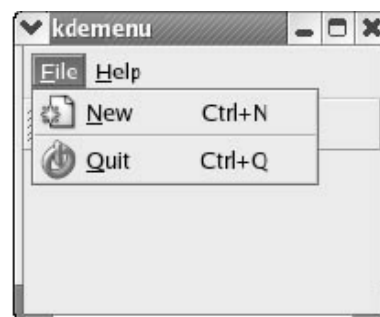
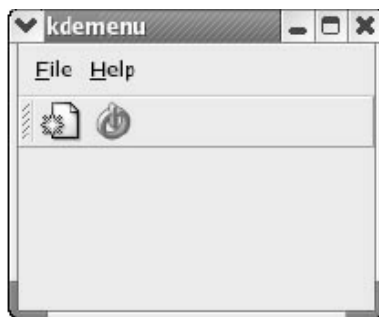
7. 提供一個 menu.pro 檔案給予 qmake：

```
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。

```
$ qmake menu.prop -o Makefile
2 $ make
$ ./kdemenu
```



Chapter 12

Python

12.1 Python 簡介

1. Python 特色

- (a) 與 shell scripts 相似，是直譯式程式。
- (b) 互動式程式，執行過程中可與程式互動。
- (c) 物件導向程式。
- (d) 跨平台，未經修改下可在Linux，Windows，MacOS X 和其他平台上使用。
- (e) 易讀、易學、易維護，適合初學者。
- (f) 可設計各種應用程式，從文字處理、www 網頁瀏覽器至遊戲都可完成。

2. Python 歷史

- (a) Python 構想起於 1980 年代後期，1989 年 12 月 Guido van Rossum 於荷蘭國家數學及計算機科學研究所發展出來。
- (b) Python 由許多程式語言發展出來，包含 ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell 及其他腳本語言。
- (c) Python 由 GNU General Public License (GPL) 授權，原始碼可自由獲得。
- (d) 1994 年 1 月釋出 1.0 版。

3. python 發展前景 參考資料來源

- (a) Python 對於 Linux 管理員是否必要？
LinuxCareer.com 在大約 20% 的職位中將 Python 能力作為關鍵性技術加以考量。

(b) Linux 社群如何看待 Python

Linux Journal 在 2011 年 9 月，網路所進行的調查，24% 將 Python

	Language	Number of votes	%
	Python	2,025	24%
	C	1,661	19%
	C++	1,488	17%
	Java	1,118	13%
列為自己最喜歡的程式語言。	Perl	674	8%
	Other	670	8%
	C#	399	5%
	Ruby	361	4%
	Haskell	126	1%
	Ocaml	47	1%
	Totals	8,569	100%

(c) Python 目前已經由 Google 大範圍使用。

(d) Python 是一種大而全面的程式語言，根據 TIOBE 的索引，截至 2011 年 8 月止，Python 排在十大程式語言榜中的第八位。

(e) Python 能在跨平台的系統中使用，且具備良好的擴展性，相信 Python 的前景是成長。

4. python 的安裝與執行

(a) 在 CentOS 6 安裝 python

```
1 [root@deuu ~]# yum install python -y
   [root@deuu ~]# yum install glade3 -y
```

(b) 執行 python

```
2 [dywang@deyu glade]$ python
   Python 2.6.5 (r265:79063, Nov 12 2010, 00:52:45)
   [GCC 4.4.4 20100525 (Red Hat 4.4.4-5)] on linux2
4   Type "help", "copyright", "credits" or "license" for more
     information.
   >>> a=3
6   >>> b=4
   >>> a+b
8   7
   >>>
10  >>> quit
     Use quit() or Ctrl-D (i.e. EOF) to exit
12 >>> quit()
   [dywang@deyu glade]$
```


(c) 寫成 python 腳本 abc.py 執行：

```
1 [dywang@deyu glade]$ vim abc.py
   a=4
3  b=3
   print a+b
5  [dywang@deyu glade]$ python abc.py
   7
```

(d) 在腳本第一行宣告使用 python 執行 abc.py：

```
[dywang@deyu glade]$ vim abc.py
2  #!/usr/bin/python
   a=4
4  b=3
   print a+b
6  [dywang@deyu glade]$ chmod a+x abc.py
   [dywang@deyu glade]$ ./abc.py
8  7
```

5. python 內縮語法：

- (a) 一般程式語言對於控制流程、函式、類別等區塊程式碼，開始與結束會以指定的碼來對應。例如：shell script 的 if 判斷式，結束程式碼為 fi。但對於程式碼的內縮排列則不規定。
- (b) python 最特別的設計是以內縮來判斷控制流程、函式等區塊程式碼的開始與結束，因此每段區塊程式碼都沒有結束程式碼的設計。
- (c) 我認為這是相當好的設計，這種設計不但可減少程式碼，更可強迫程式撰寫者以階層式撰寫，讓程式更容易解讀。
- (d) 強烈建議以 tab 鍵進行內縮，不要使用空白內縮內縮。要對齊才算是同區塊程式碼，若用 4 個空白內縮，則同區塊程式都要用 4 個空白對齊，即使 tab 設定為 4 個空白位置，也無法執行。
- (e) 錯誤的 python 程式碼

```
[dywang@deyu glade]$ vim error.py
2 [dywang@deyu glade]$ cat error.py
   if True:
4   print "Yes"
   else:
6   print "NO"
   [dywang@deyu glade]$ python error.py
8   File "error.py", line 2
       print "Yes"
10      ^
```

```
IndentationError: expected an indented block
```

(f) 正確的 python 程式：

```
1 [dywang@deyu glade]$ vim correct.py
[dywang@deyu glade]$ cat crooect.py
3 if True:
    print "Yes"
5 else:
    print "NO"
7 [dywang@deyu glade]$ python crooect.py
Yes
```

6. python 括號、引號、換行、註解

(a) 與 shell script 一樣，可使用倒斜線 \ 沿續下一行。

```
2 [dywang@deyu glade]$ vim correct.py
[dywang@deyu glade]$ cat correct.py
if True:
4     print "Yes \
new line"
6 else:
    print "NO"
8 [dywang@deyu glade]$ python correct.py
Yes new line
```

(b) 大中小括號 {} [] () 可以不在同一行。

```
1 days = ['Monday', 'Tuesday', 'Wednesday',
          'Thursday', 'Friday']
```

(c) 註解：

```
2 # 空白行視為註解
# 符號 # 後的文字為註解這是註解
''
4 "這是註解"這是註解
'''
6 """這是註解"""
"""這是註解這是也是註解
8
""" 與 """ 之間文字都為註解
```

```
10 """  
12 # 成對的單引號雙引號之間為字串，且可換行。因不執行故都為註解。  
14 days = ['Monday', 'Tuesday', 'Wednesday',  
          'Thursday', 'Friday']
```

(d) 與 shell script 一樣，可以使用分號；將多個命令放在同一行。

```
a=4; b=3; print a+b
```

12.2 Python 變數

1. python 變數

(a) 變數直接使用，不需宣告：

```
1 #!/usr/bin/python  
3 counter = 100          # An integer assignment  
miles     = 1000.0       # A floating point  
5 name     = "Devid"     # A string
```

(b) 多個變數一起指定：

```
1 a=b=c=1  
a,b,c=1,2,"Devid"
```

(c) 變數指定後，變數型態就固定，重新指定新型態的值，等同重新宣告：

```
2 a=1  
a="Devid"
```

(d) 變數的刪除：

```
2 a=b=c=1  
del a
```

2. python 字串處理

(a) 字串變數以單引號，或雙引號括起來。

```
name = "Devid"      # A string
```

(b) 字串變數為字元陣列：

```
1 [dywang@deyu glade]$ cat str.py
  #/usr/bin/python
3 # coding: utf-8

5 name = "David"
  print name           # 印出整個字串
7 print name[0]        # 印出第 1 個字元
  print name[1:3]      # 印出第 2 至 3 個字元
9 print name[1:]       # 印出第 2 至最後一個字元
  print name * 3       # 印出字串三次
11 print name + " csie" # 印出連續字串

13 [dywang@deyu glade]$ python str.py
David
15 D
  av
17 avid
  DavidDavidDavid
19 David csie
```

3. python 陣列處理

(a) lists 陣列

```
1 [dywang@deyu glade]$ cat lists.py
  #/usr/bin/python
3 # coding: utf-8

5 list = [ "David", 123, 23.4, 'abc', 4.321 ]
  tinylist = [ 98, 'Rita' ]
7 print list           # 印出整個陣列
  print list[0]        # 印出陣列第 1 個元素
9 print list[1:3]      # 印出陣列第 2 至 3 個元素
  print list[1:]       # 印出陣列第 2 至最後一個元素
11 print tinylist * 2   # 印出陣列二次
  print tinylist + list # 印出連續陣列

13 [dywang@deyu glade]$ python lists.py
15 ['David', 123, 23.399999999999999, 'abc', 4.3209999999999997]
David
```

```

17 [123, 23.399999999999999]
    [123, 23.399999999999999, 'abc', 4.3209999999999997]
19 [98, 'Rita', 98, 'Rita']
    [98, 'Rita', 'David', 123, 23.399999999999999, 'abc',
      4.3209999999999997]

```

(b) tuples 陣列：唯讀的 lists

```

[dywang@deyu glade]$ cat tuples.py
2  #/usr/bin/python
   # coding: utf-8
4
   list = ( "David", 123, 23.4, 'abc', 4.321 )
6  tinylist = ( 98, 'Rita' )
   print list                # 印出整個陣列
8  print list[0]              # 印出陣列第 1 個元素
   print list[1:3]            # 印出陣列第 2 至 4 個元素
10 print list[1:]              # 印出陣列第 2 至最後一個元素
   print tinylist * 2          # 印出陣列二次
12 print tinylist + list      # 印出連續陣列
[dywang@deyu glade]$ python tuples.py
14 ('David', 123, 23.399999999999999, 'abc', 4.3209999999999997)
   David
16 (123, 23.399999999999999)
   (123, 23.399999999999999, 'abc', 4.3209999999999997)
18 (98, 'Rita', 98, 'Rita')
   (98, 'Rita', 'David', 123, 23.399999999999999, 'abc',
     4.3209999999999997)

```

(c) lists 與 tuples 陣列比較

```

1  [dywang@deyu glade]$ cat list_tuple.py
   #/usr/bin/python
3  # coding: utf-8
5
   list = [ "David", 123, 23.4, 'abc', 4.321 ]
   tuple = ( "David", 123, 23.4, 'abc', 4.321 )
7  list[3] = 'xyz'            # 有效語法
   tuple[3] = 'xyz'           # 無效語法
9  [dywang@deyu glade]$ python list_tuple.py
   Traceback (most recent call last):
11     File "list_tuple.py", line 7, in <module>
       tuple[3] = 'xyz'        # 無效語法
13   TypeError: 'tuple' object does not support item assignment

```

(d) dictionaries 陣列：

```

1  [dywang@deyu glade]$ cat dict.py
   #!/usr/bin/python
3  # coding: utf-8

5  dict = {}
   dict['peter'] = "1234"
7  dict[2] = "rita"

9  tinydict = {'name': 'devid', 'code': 4538, 'dept': 'csie'}

11 print dict['peter']      # Prints value for 'peter' key
   print dict[2]           # Prints value for 2 key
13 print tinydict          # Prints complete dictionary
   print tinydict.keys()   # Prints all the keys
15 print tinydict.values() # Prints all the values

17 [dywang@deyu glade]$ python dict.py
   1234
   rita
   {'dept': 'csie', 'code': 4538, 'name': 'devid'}
21 ['dept', 'code', 'name']
   ['csie', 4538, 'devid']
23 # dictionaries 沒有順序概念

```

4. if `__name__ == "__main__"`:

```

1  [dywang@deyu python]$ cat mypy.py
   #!/usr/bin/python
3  # coding: utf-8

5  class myPy:
       print "class my first python"
7
   if __name__ == "__main__":
9       print __name__
       myPy()
11  else:
       print __name__
13
   [dywang@deyu python]$ cat test.py
15 #!/usr/bin/python

17 import mypy

19 mypy.__name__

21 [dywang@deyu python]$ python test.py
   mypy

```

```
23 |  
25 | [dywang@deyu python]$ python mpy.py  
    | __main__
```

12.3 python 運算子

1. Python - Basic Operators

12.4 python 流程控制

1. Python - IF...ELIF...ELSE Statement
2. Python - while Loop Statements
3. Python - for Loop Statements
4. Python - break,continue and pass Statements

12.5 python 函式與模組

1. Python - Functions
2. Python - Modules
3. Python - Regular Expressions

12.6 python I/O 與 Exceptions

1. Python - Files I/O
2. Python - Exceptions Handling

Chapter 13

PyGTK

13.1 PyGTK 簡介與開始

1. PyGTK 簡介

- (a) PyGTK 是 GNOME 專案的一部分，作者是著名的 GNOME 開發者 James Henstridge。
- (b) PyGTK 是一套用 Python 封裝的，用於 GTK+ 的 GUI 庫。
- (c) PyGTK 是在 LGPL 授權下的自由軟體，可以自由修改、散佈及研究。
- (d) PyGTK 跨平台，未經修改下可在 Linux，Windows，MacOS X 和其他平台上使用。
- (e) PyGTK 所開發的應用程式更已被選應用於每童一電腦（One Laptop Per Child，OLPC）的系統之上。
- (f) PyGTK 非常容易使用，是最受歡迎的 GTK+ 函式庫之一。
- (g) PyGTK 提供所有圖形界面元件及有用的程式設施來產生桌面應用程式。

2. 第一支程式

```

1 [dywang@deyu python]$ cat mpy.py
2 #!/usr/bin/python
3 # coding: utf-8
4
5 import gtk
6
7 class myPy:
8     def __init__(self):
9         window = gtk.Window(gtk.WINDOW_TOPLEVEL)
10         "lambda 是匿名函式 x 為其參數"
11         window.connect("destroy", lambda x: gtk.main_quit())
12         window.set_title視窗抬頭('')
13         window.set_keep_above(True)
14         window.show()
15 
```

```

17 myPy()
   gtk.main()

```

3. 類別相關聯結

- (a) gtk.Widget class
- (b) gtk.Window class
- (c) gtk Functions
- (d) python: Lambda Functions

13.2 PyGTK 除錯

1. import 的 gtk modules 找不到，錯誤訊息如下：

```

1 [dywang@dywhd2 zzz]$ python mpy1.py
7
3 'import site' failed; use -v for traceback
  Traceback (most recent call last):
5     File "mpy1.py", line 5, in <module>
        import gtk
7 ImportError: No module named gtk

```

2. 查詢安裝套件 pygtk2 已安裝。

```

1 [root@dywhd2 ~]# rpm -qa | grep pygtk
pygtk2-libglade-2.16.0-3.el6.x86_64
3 pygtksourceview-2.8.0-1.el6.x86_64
pygtk2-2.16.0-3.el6.x86_64

```

3. 查詢 pygtk module 在目錄 /usr/lib64/python2.6/site-packages/ 下。

```

2 [root@dywhd2 ~]# find /usr/lib64/python2.6 -name pygtk*
2 /usr/lib64/python2.6/site-packages/pygtk.pyc
  /usr/lib64/python2.6/site-packages/pygtk.py
4 /usr/lib64/python2.6/site-packages/pygtk.pth
  /usr/lib64/python2.6/site-packages/pygtk.pyo

```

4. 執行 python 後進入互動式模式，import pygtk 模組不成功，顯示 sys.path 發覺 /usr/lib64/python2.6/site-packages/ 目錄並不在其中。

```

1 [dywang@dywhd2 zzz]$ python
7
3 'import site' failed; use -v for traceback
Python 2.6.5 (r265:79063, Nov 12 2010, 00:52:45)
5 [GCC 4.4.4 20100525 (Red Hat 4.4.4-5)] on linux2
Type "help", "copyright", "credits" or "license" for more
  information.
7 >>> import pygtk; pygtk.require('2.0')
Traceback (most recent call last):
9   File "<stdin>", line 1, in <module>
ImportError: No module named pygtk
11 >>> import sys
>>> print sys.path
13 ['', '', '/usr/local/google_appengine',
'/usr/local/google_appengine/lib/django_1.4',
15 '/usr/lib64/python26.zip',
'/usr/lib64/python2.6/',
17 '/usr/lib64/python2.6/plat-linux2',
'/usr/lib64/python2.6/lib-tk',
19 '/usr/lib64/python2.6/lib-old',
'/usr/lib64/python2.6/lib-dynload']

```

5. 解決方式：編輯 `/etc/bashrc`，將 `PYTHONPATH` 變數增加 `/usr/lib64/python2.6/site-packages/` 目錄，讓任何用戶登入後就生效。

```

2 [root@dywhd2 ~]# vim /etc/bashrc
[root@dywhd2 ~]# tail -n1 /etc/bashrc
export PYTHONPATH="/usr/lib64/python2.6/site-packages:${PYTHONPATH}"

```

6. 重新取得 `bash` shell 後，查詢 `PYTHONPATH` 變數已包含 `/usr/lib64/python2.6/site-packages/` 目錄。

```

1 [dywang@dywhd2 ~]$ echo $PYTHONPATH
/usr/lib64/python2.6/site-packages::usr/local/google_appengine:/usr/local/google_appengine/lib/django_1.4

```

7. 再執行 `python` 進入互動式模式，`import pygtk` 或 `import gtk` 都成功取得 module，不再出現錯誤訊息。

```

2 [dywang@dywhd2 ~]$ python
Python 2.6.5 (r265:79063, Nov 12 2010, 00:52:45)

```

```
[GCC 4.4.4 20100525 (Red Hat 4.4.4-5)] on linux2
4 Type "help", "copyright", "credits" or "license" for more
  information.
  >>> import pygtk
6 >>> import gtk
  >>>
```

13.3 Buttons 與 Layout

1. Example

```
1 #!/usr/bin/python
  # coding: utf-8
3
4 #import pygtk
5 import gtk
6
7 class myPy:
8     def __init__(self):
9         window = gtk.Window(gtk.WINDOW_TOPLEVEL)
10         "lambda 是匿名函式 x 為其參數"
11         window.connect("destroy", lambda x: gtk.main_quit())
12         window.set_title視窗抬頭('')
13         window.set_keep_above(True)
14
15         vbox1 = gtk.VBox(False, 2)
16         "homogeneous=True 則內含物件平均分佈，spacing=5 間隔5"
17         hbox1 = gtk.HBox(homogeneous=True,spacing=5)
18         hbox2 = gtk.HBox(homogeneous=True,spacing=5)
19         hbox3 = gtk.HBox(homogeneous=True,spacing=5)
20         yes = gtk.Button("yes")
21         no = gtk.Button("no")
22         close = gtk.Button(None,gtk.STOCK_CLOSE)
23         hbox1.add(yes)
24         hbox1.add(no)
25         hbox2.add(close)
26         vbox1.pack_start(hbox1, True, True, 0)
27         vbox1.pack_start(hbox2, True, True, 0)
28         vbox1.pack_start(hbox3, True, True, 0)
29         window.add(vbox1)
30
31         window.show_all()
32
33 myPy()
  gtk.main()
```

2. 類別相關聯結

- (a) gtk.Vbox class
- (b) gtk.Box class
- (c) gtk.Hbox class
- (d) gtk.Button class
- (e) Stock Items

13.4 Signals, Events and Functions

1. Example1

```
#!/usr/bin/python
# coding: utf-8

import pygtk
import gtk

class myPy:
    def callback(self, widget, data):
        if data == "yes":
            print "yes"
        elif data == "no":
            print "no"
        else:
            print "yes|no"

    def __init__(self):
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        "lambda 是匿名函式 x 為其參數"
        window.connect("destroy", lambda x: gtk.main_quit())
        window.set_title視窗抬頭('')
        window.set_keep_above(True)

        vbox1 = gtk.VBox(False, 2)
        hbox1 = gtk.HBox(homogeneous=True, spacing=5)
        hbox2 = gtk.HBox(homogeneous=True, spacing=5)
        hbox3 = gtk.HBox(homogeneous=True, spacing=5)
        yes = gtk.Button("yes")
        no = gtk.Button("no")
        close = gtk.Button(None, gtk.STOCK_CLOSE)
        close.connect("clicked", lambda x: gtk.main_quit())
        yes.connect("clicked", self.callback, 'yes')
        no.connect("clicked", self.callback, 'no')
        hbox1.add(yes)
        hbox1.add(no)
        hbox2.add(close)
        vbox1.pack_start(hbox1, True, True, 0)
```

```

38         vbox1.pack_start(hbox2, True, True, 0)
        vbox1.pack_start(hbox3, True, True, 0)
        window.add(vbox1)
40
        window.show_all()
42
    myPy()
44    gtk.main()

```

2. Example2

```

#!/usr/bin/python
2 # coding: utf-8

4 #import pygtk
import gtk
6 import subprocess

8 icondir="/usr/share/icons/gnome/16x16"
yes_icon = icondir+"/stock/generic/stock_calc-accept.png"
10 no_icon = icondir+"/stock/generic/stock_calc-cancel.png"

12 def img_label_box(parent, img_file, label):
    box = gtk.HBox(False, 0)
14     box.set_border_width(2)
    image = gtk.Image()
16     image.set_from_file(img_file)
    label = gtk.Label(label)
18     box.pack_start(image, False, False, 5)
    box.pack_start(label, False, False, 5)
20     image.show()
    label.show()
22     return box

24 class myPy:
    def callback(self, widget, data):
26         if data == "yes":
            subprocess.Popen("gnome-terminal", shell=True)
28         elif data == "no":
            print "no"
30         else:
            print "yes|no"

32     def __init__(self):
34         window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        "lambda 是匿名函式 x 為其參數"
36         window.connect("destroy", lambda x: gtk.main_quit())
        window.set_title視窗抬頭('')
38         window.set_keep_above(True)

```

```

40     vbox1 = gtk.VBox(False, 2)
    hbox1 = gtk.HBox(homogeneous=True, spacing=5)
42     hbox2 = gtk.HBox(homogeneous=True, spacing=5)
    hbox3 = gtk.HBox(homogeneous=True, spacing=5)
44     yes = gtk.Button()
    no = gtk.Button()
46     close = gtk.Button(None, gtk.STOCK_CLOSE)
    close.set_use_stock(True)
48     close.connect("clicked", lambda x: gtk.main_quit())
    yes.connect("clicked", self.callback, 'yes')
50     no.connect("clicked", self.callback, 'no')
    box_yes = img_label_box(window, yes_icon, "YES")
52     box_no = img_label_box(window, no_icon, "NO")
    yes.add(box_yes)
54     no.add(box_no)
    hbox1.add(yes)
56     hbox1.add(no)
    hbox2.add(close)
58     vbox1.pack_start(hbox1, True, True, 0)
    vbox1.pack_start(hbox2, True, True, 0)
60     vbox1.pack_start(hbox3, True, True, 0)
    window.add(vbox1)
62
    window.show_all()
64
myPy()
66 gtk.main()

```

3. 類別相關聯結

(a) gtk.Button class

13.5 Label and Entry

1. Example1: Label

```

#!/usr/bin/python
2  # coding: utf-8

4  #import pygtk
    import gtk
6  import subprocess

8  icondir="/usr/share/icons/gnome/16x16"
    yes_icon = icondir+"/stock/generic/stock_calc-accept.png"
10  no_icon = icondir+"/stock/generic/stock_calc-cancel.png"

```

```

12 def img_label_box(parent, img_file, label):
    box = gtk.HBox(False, 0)
14     box.set_border_width(2)
    image = gtk.Image()
16     image.set_from_file(img_file)
    label = gtk.Label(label)
18     box.pack_start(image, False, False, 5)
    box.pack_start(label, False, False, 5)
20     image.show()
    label.show()
22     return box

24 class myPy:
    def callback(self, widget, data):
26         if data == "yes":
            subprocess.Popen("gnome-terminal", shell=True)
28         elif data == "no":
            self.label.set_text("no")
30         else:
            print "yes|no"
32
    def __init__(self):
34         window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        "lambda 是匿名函式 x 為其參數"
36         window.connect("destroy", lambda x: gtk.main_quit())
        window.set_title視窗抬頭('')
38         window.set_keep_above(True)

40         vbox1 = gtk.VBox(False, 2)
        hbox1 = gtk.HBox(homogeneous=True, spacing=5)
42         hbox2 = gtk.HBox(homogeneous=True, spacing=5)
        hbox3 = gtk.HBox(homogeneous=True, spacing=5)
44         yes = gtk.Button()
        no = gtk.Button()
46         close = gtk.Button(None, gtk.STOCK_CLOSE)
        close.set_use_stock(True)
48         close.connect("clicked", lambda x: gtk.main_quit())
        yes.connect("clicked", self.callback, 'yes')
50         no.connect("clicked", self.callback, 'no')
        box_yes = img_label_box(window, yes_icon, "YES")
52         box_no = img_label_box(window, no_icon, "NO")
        yes.add(box_yes)
54         no.add(box_no)
        self.label=gtk.Label("Yes|No")
56         hbox1.add(yes)
        hbox1.add(no)
58         hbox2.add(close)
        hbox3.add(self.label)
60         vbox1.pack_start(hbox1, True, True, 0)
        vbox1.pack_start(hbox2, True, True, 0)
62         vbox1.pack_start(hbox3, True, True, 0)

```



```
        window.add(vbox1)
64
        window.show_all()
66
myPy()
68
gtk.main()
```

2. Example2: Entry

```
#!/usr/bin/python
2 # coding: utf-8

4 #import pygtk
import gtk
6 import subprocess

8 icondir="/usr/share/icons/gnome/16x16"
yes_icon = icondir+"/stock/generic/stock_calc-accept.png"
10 no_icon = icondir+"/stock/generic/stock_calc-cancel.png"

12 def img_label_box(parent, img_file, label):
    box = gtk.HBox(False, 0)
14    box.set_border_width(2)
    image = gtk.Image()
16    image.set_from_file(img_file)
    label = gtk.Label(label)
18    box.pack_start(image, False, False, 5)
    box.pack_start(label, False, False, 5)
20    image.show()
    label.show()
22    return box

24 class myPy:
    def callback(self, widget, data):
26        if data == "yes":
            subprocess.Popen("gnome-terminal",shell=True)
28        elif data == "no":
            self.label.set_text(data)
30        else:
            self.label.set_text("Only yes|no")
32
    def entryChange(self,widget):
34        if self.entry.get_text() == "yes":
            subprocess.Popen("gnome-terminal",shell=True)
36            self.label.set_text("yes")
        elif self.entry.get_text() == "no":
38            self.label.set_text("no")
        else:
40            self.label.set_text("Only yes|no")
```

```

42     def __init__(self):
43         window = gtk.Window(gtk.WINDOW_TOPLEVEL)
44         "lambda 是匿名函式 x 為其參數"
45         window.connect("destroy", lambda x: gtk.main_quit())
46         window.set_title視窗抬頭('')
47         window.set_keep_above(True)
48
49         vbox1 = gtk.VBox(False, 2)
50         hbox1 = gtk.HBox(homogeneous=True, spacing=5)
51         hbox2 = gtk.HBox(homogeneous=True, spacing=5)
52         hbox3 = gtk.HBox(homogeneous=True, spacing=5)
53         yes = gtk.Button()
54         no = gtk.Button()
55         close = gtk.Button(None, gtk.STOCK_CLOSE)
56         close.set_use_stock(True)
57         self.label=gtk.Label("Yes|No")
58         self.entry=gtk.Entry()
59         close.connect("clicked", lambda x: gtk.main_quit())
60         yes.connect("clicked", self.callback, 'yes')
61         no.connect("clicked", self.callback, 'no')
62         self.entry.connect("changed", self.entryChange)
63         box_yes = img_label_box(window, yes_icon, "YES")
64         box_no = img_label_box(window, no_icon, "NO")
65         yes.add(box_yes)
66         no.add(box_no)
67         hbox1.add(yes)
68         hbox1.add(no)
69         hbox2.add(close)
70         hbox3.add(self.label)
71         hbox3.add(self.entry)
72         vbox1.pack_start(hbox1, True, True, 0)
73         vbox1.pack_start(hbox2, True, True, 0)
74         vbox1.pack_start(hbox3, True, True, 0)
75         window.add(vbox1)
76
77         window.show_all()
78
79     if __name__ == "__main__":
80         print __name__
81         myPy()
82         gtk.main()
83     else:
84         print __name__

```

3. 類別相關聯結

- (a) gtk.Label class
- (b) gtk.Entry class

Chapter 14

Glade

14.1 簡介

1. 什麼是 glade ?

- (a) Glade 是 Rapid application development (RAD) 工具，提供使用者容易的 GTK+工具箱及 GNOME 桌面環境。
- (b) Glade 設計好的使用者界面存成 XML 檔，供程式使用。
- (c) Glade 的 XML 檔可供無數的程式語言使用，包括 C, C++, C#, Vala, Java, Perl, Python 等。
- (d) Glade 是 GNU GPL License 授權的自由軟體。

2. glade3 安裝

```
1 [root@dywHome2 ~]# [root@deyu ~]# yum install glade3
```

3. 執行 glade designer

1 || 圖形界面

```
3 Applications->Programming->Glade Interface Designer文字界面
5 [dywang@deyu glade]$ glade-3
```

4. Glade Interface Designer Demo

5. 撰寫 python file example1.py

```
2 #!/usr/bin/python
  # coding: utf-8
```

```
import gtk
4 import gtk.glade
import subprocess
6 import sys

8 class Example1:
    def __init__(self):
10         self.gladefile = "example1.glade"
        self.wTree = gtk.glade.XML(self.gladefile)
12         dic = {
            "on_window_example1_destroy" : self.quit,
14             "on_button1_clicked" : self.ok,
            "on_button2_clicked" : self.quit,
16             "on_entry1_changed" : self.entrychanged
        }
18         self.wTree.signal_autoconnect(dic)
        self.window = self.wTree.get_widget("window1")
20         self.window.set_keep_above(True)
        self.window.show()
22
    def entrychanged(self, widget):
24         if self.wTree.get_widget("entry1").get_text() == "show":
            subprocess.Popen("gnome-terminal", shell=True)
26             self.wTree.get_widget("label1").set_text("Show
                terminal")
            else:
28                 self.wTree.get_widget("label1").set_text("Only show")
30
    def ok(self, widget):
32         subprocess.Popen("gnome-terminal", shell=True)
        self.wTree.get_widget("label1").set_text("Open terminal")
34
    def quit(self, widget):
36         sys.exit(0)

38 if __name__ == "__main__":
    Example1()
40     gtk.main()
```

6. 執行 python

```
[dywang@deyu glade]$ python example1.py
```

14.2 Calculator Example

1. 下載計算機範例檔

```
calculator.glade
calculatorglade.py
```

2. 下載計算機範例檔 (增加開根號)

```
calculator1.glade
calculator1.py
```

3. 以glade-3開啓calculator.glade

```

1 1. table1 Rows: 5 -> 6
2 2. insert ok button in table1 (0,5)
3 3. rename ok button Label and Name to Sqrt
4 4. set Sqrt button clicked Signals to on_Sqrt_clicked
5 5. vim calculatorglade.py

7 # coding: utf-8
import sys
9 import math # 匯入數學模組
try:
11     import pygtk
    pygtk.require('2.0')
13 except:
    pass
15 .... omission ....

17 class Calculator:
    .... omission ....
19     "on_Add_clicked" : self.displayAdd,
    "on_Sqrt_clicked" : self.displaySqrt, # 定義按鈕接受函
        數sqrt
21 .... omission ....
    if operator == 'Sqrt':
23         self.firstOperand = self.wTree.get_widget("
            displayText").get_text()
            result = math.sqrt(float(self.firstOperand))
25         self.wTree.get_widget("displayText").set_text(str(
            result))
    .... omission ....

27     def displaySqrt(self,widget): #定義函數displaySqrt
29         self.compute("Sqrt")

```

4. 執行

```
1 || [dywang@deyu glade]$ python calculatorglade.py
```

5. 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](#)

Chapter 15

*Tarball 套件發行

15.1 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. 使用新發現漏洞或原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 或 SRPM 套件。

15.2 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 [-cxtzjvfpPN] 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 : 排除FILE 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  
2 [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：

```
$ mv myapp-1.0.tar.gz myapp_v1.tgz  
2 $ 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.檢測程式檔案 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 與 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 -czvpf /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 -czvf 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，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`，清除目標檔案。

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，產生可執行檔。

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. 程式選舉連結檔

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

15.3 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以下省略
10 -----
```

3. 顯示解開原始碼之目錄：

```
[root@dywHome2 src]# ls -ld kplayer*
2 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
9 Create configure script
11
13     make -f Makefile.dist
15
17 Configure
19
21     ./configure --prefix='kde-config --prefix'
23
25 Compile
27
29     make
31
33 Install
35
37     su -c 'make install'
39
41 Run
43
45     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
7 *** Creating list of subdirectories
8 *** Creating configure.files
9 *** Creating configure.in
10 *** Creating aclocal.m4
11 *** Creating configure
12 *** Creating config.h template
13 *** Creating Makefile templates
14 *** Postprocessing Makefile templates
15 *** Creating date/time stamp
16 *** Finished
    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'
2 [root@dywHome2 kplayer-0.5.3]# ./configure --prefix=/usr/
   local/kplayer
3 checking build system type... i686-pc-linux-gnu
4 checking host system type... i686-pc-linux-gnu中間省略
5 -----
6 configure: creating ./config.status
7 fast creating Makefile
8 fast creating admin/Makefile
9 fast creating doc/Makefile
10 fast creating doc/da/Makefile
11 fast creating doc/en/Makefile
12 fast creating doc/pt/Makefile
13 fast creating doc/sv/Makefile
14 fast creating icons/Makefile
15 fast creating kplayer/Makefile
16 fast creating po/Makefile
17 config.pl: fast created 10 file(s).
18 config.status: creating config.h
19 config.status: config.h is unchanged
20 config.status: executing depfiles commands
21
   Good - your configure finished. Start make now

```

4. Compile

```

1 [root@dywHome2 kplayer-0.5.3]# make以上省略
2 -----
3 make[2]: Entering directory '/usr/src/kplayer-0.5.3'
4 make[2]: Leaving directory '/usr/src/kplayer-0.5.3'
5 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/

```
[root@dywHome2 src]# ll /usr/local/kplayer/
2 total 3
drwxr-xr-x 2 root root 1024 Apr  3 10:09 bin/
4 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
./kplayer-0.5.3/INSTALL中間省略
9 -----
./kplayer-0.5.3/Makefile
11 ./kplayer-0.5.3/libtool
./kplayer-0.5.3/config.status
13 ./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. 解除順序為 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. 以括弧，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 16

*RPM 與 SRPM 套件發行

16.1 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：

name-version-release.architecture.rpm	<== RPM 檔名格式
name-version-release.src.rpm	<== SRPM 檔名格式

- (a) 例如檔案 `kplayer-0.5.3-5mdv2007.0.i586.rpm` 的意義為：

kplayer	-	0.5.3	-	5mdv2007.0	.	i586	.	rpm 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。 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，平台名稱爲 i586。

8. RPM 檔案系統

(a) Mandriva 2007 以 /usr/src/rpm/ 爲工作目錄

(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
4 i386
  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，因打包環境為 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。
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`

16.2 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} 展開成爲，
  而 myapp %{version} 展開爲； 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
  install -m755 myapp.1 $RPM_BUILD_ROOT%{_mandir}/myapp.1
6 # %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：

```
%clean
1 rm -rf $RPM_BUILD_ROOT
```

4. %files: 列出套件包含的檔案。例如：

```
%files
2 %{_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 #
3 # spec file for package myapp (Version 1.0)
4 #
5 Vendor:          Wrox Press
6 Distribution:    Any
7 Name:            myapp
8 Version:         1.0
9 Release:         1
10 Packager:        neil@provider.com
11 License:         Copyright 2003 by Wrox Press
12 Group:          Applications/Media
13 Provides:        goodness
14 Requires:        mysql >= 3.23
15 Buildroot:       %{_tmppath}/%{name}-%{version}-root
16 source:          %{name}-%{version}.tar.gz
17 Summary:         Trivial application
18 %description
19 MyApp Trivial Application
20 A trivial application used to demonstrate development tools.
21 This version pretends it requires MySQL at or above 3.23.
22 Authors: Neil Matthew and Richard Stones
23 %prep
24 %setup -q
25 %build
26 make
27 %install
28 mkdir -p $RPM_BUILD_ROOT%{_bindir}
29 mkdir -p $RPM_BUILD_ROOT%{_mandir}
30 install -m755 myapp $RPM_BUILD_ROOT%{_bindir}/myapp
31 install -m755 myapp.1 $RPM_BUILD_ROOT%{_mandir}/myapp.1
32 %clean
33 rm -rf $RPM_BUILD_ROOT
34 %post
35 mail root -s "myapp installed - please register" </dev/null
36 %files
37 %{_bindir}/myapp
38 %{_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 -Rh 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
+ umask 022
73 + 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. 套件的廠商 Wrox Press

7. RPM 規格檔案中標籤定義 Distribution: Any，意義為何？

Sol. 適用於任何的 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

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. 由於準備動作或不是公式化的動作，故已為成巨集 %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. 建立可應用程式，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 的已裝，`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. 列出套件已含的檔案。
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`

16.3 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 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/
   | BUILD/:
3 |
   | RPMS/:
5 | athlon/ i386/ i486/ i586/ i686/ noarch/
   |
7 | SOURCES/:
   | kplayer-0.5.3.tar.bz2
9 |
   | SPECS/:
11 | 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：

```

   | [root@dywHome2 SOURCES]# mv kplayer-0.5.3 kplayer-0.5.3.
   | systray
2 | [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

```

2 [root@dywHome2 SPECS]# rpmbuild -bb kplayer.spec 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-5
  mdv2007.0.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

```

2 [root@dywHome2 SPECS]# ll ../RPMS/i586/
total 6377
-rw-r--r-- 1 root root 3122228 Apr  5 17:29
4 kplayer-0.5.3.systray-5mdv2007.0.i586.rpm

```



```

6  -rw-r--r-- 1 root root 3378549 Apr  5 17:29
   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

```

2  [root@dywHome2 SPECS]# rpmbuild -ba kplayer.spec 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-5
       mdv2007.0.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
       -rw-r--r-- 1 root root 3122404 Apr  5 17:46
   5  kplayer-0.5.3.systray-5mdv2007.0.i586.rpm
       -rw-r--r-- 1 root root 3378599 Apr  5 17:46
   7  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 17

*套件修補、檢驗與管理

17.1 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：

```

2 [root@linux ~]# diff [-bBiqn] from-file to-file選項：
3
4 from-file : 檔名，作為原始比對檔案的檔名；
5 to-file   : 檔名，作為目的比對檔案的檔名；
6 # from-file 或 to-file 可以 - 取代， - 代表
   『Standard』。input
7
8 -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)
9 -B : 忽略空白行的差異。
10 -i : 忽略大小寫的不同。
11 -q : 只列出檔案是否有差異。
12 -n : 以 RCS 格式輸出檔案之差異。
13 -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`。

```
[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。

```
[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。

```
[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
--- old/passwd 2008-04-16 13:14:50.000000000 -0400
5 +++ new/passwd 2008-04-16 11:15:12.000000000 -0400
@@ -1,9 +1,8 @@
7 root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/sh
9 daemon:x:2:2:daemon:/sbin:/bin/sh
-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 前面省略
2 .....().....
   Only in /etc: paper.config
4 diff /etc/passwd /tmp/test/passwd
   4d3
6 < adm:x:3:4:adm:/var/adm:/sbin/nologin
   6c5
8 < sync:x:5:0:sync:/sbin:/bin/sync
   ---
10 > no six line
   Only in /etc: passwd-後面省略
12 .....().....

```

- `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 選項：
2
   -b  : 列出第一個的不同點之字元。
4   -c  : 同上。
   -i SKIP1:SKIP2 : file1 與 file2 分別忽略前 SKIP1 與 SKIP2 位
   元。
6   -s  : 安靜模式，不顯示任何訊息。

```

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，在檔
  案 a /tmp/test/passwd 的第一個字元為。 l
  # 字元 a 之八進位碼為，字元 141 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

```
[root@dywHome2 tmp]# cd /tmp/old
2 [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

```
[root@dywHome2 old]# patch passwd /tmp/test.patch -o passwd1
2 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

```
[root@dywHome2 old]# patch passwd -i /tmp/test.patch
2 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

```
[root@dywHome2 old]# patch passwd -i /tmp/test.patch
2 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。

```
2 [root@dywHome2 old]# patch -p1 </tmp/test.patch  
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。

```
2 [root@dywHome2 old]# patch -p0 </tmp/test.patch  
can't find file to patch at input line 4  
Perhaps you used the wrong -p or --strip option?  
4 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.
```

```

6 | Hunk #2 succeeded at 4.
   | 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 -n 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 -c 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，該字元輸出為 `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. `printf '\x61\t\x6c\n'` 表示輸出十六進位 6c (即十進位 108) 之 ASCII 字元，故會輸出 `a\tc`。
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儲存於 `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. 系統會將更新後的內容儲存於 `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. `u/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`

17.2 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
```

- 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：

```
1 [root@dywHome2 SOURCES]# md5sum -c kplayer.md5
2 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：

```
1 [root@dywHome2 SOURCES]# md5sum -t readme
2 5a4d99c2a9b759dca679c2c1001cf8d4  readme
```

6. 再檢驗 kplayer.md5 檔案指紋：

```
1 [root@dywHome2 SOURCES]# md5sum -c kplayer.md5
2 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：

```
1 [root@dywHome2 rpm]# wget ftp://ftp.isu.edu.tw/Linux/Mandriva
2 /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.
12 ==> PASV ... done. ==> RETR MD5SUM ... done.
    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：

```

[root@dywHome2 rpm]# cat MD5SUM
2 374fca8ea858a7079c3717acca208e47  hdlist.cz
   38b6616b9514707ff8dc344eadd5468f  synthesis.hdlist.cz

```

3. 下載檔案：

```

1 [root@dywHome2 rpm]# wget ftp://ftp.isu.edu.tw/Linux/Mandriva
  /official/2007.0/i586/media/main/updates/media_info/hdlist.cz
3 --20:58:30--  ftp://ftp.isu.edu.tw/Linux/Mandriva
  /official/2007.0/i586/media/main/updates/media_info/hdlist.cz
5      => 'hdlist.cz'
   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.
   Logging in as anonymous ... Logged in!
9   ==> SYST ... done. ==> PWD ... done.
   ==> TYPE I ... done. ==> CWD /Linux/Mandriva/official
   /2007.0/i586/media/main/updates/media_info ... done.
11  ==> PASV ... done. ==> RETR hdlist.cz ... done.
   Length: 21,737,096 (21M) (unauthoritative)
13 100%[=====]
   21,737,096 102.50K/s ETA 00:00
15 21:02:04 (99.67 KB/s) - 'hdlist.cz' saved [21737096]
17
18 [root@dywHome2 rpm]# wget ftp://ftp.isu.edu.tw/Linux/Mandriva
  /official/2007.0/i586/media/main/updates/media_info/synthesis
19 .hdlist.cz
   --21:02:21--  ftp://ftp.isu.edu.tw/Linux/Mandriva
21 /official/2007.0/i586/media/main/updates/media_info/synthesis
   .hdlist.cz
   => 'synthesis.hdlist.cz'
23 Resolving ftp.isu.edu.tw... 140.127.177.17, 140.127.177.15
   Connecting to ftp.isu.edu.tw|140.127.177.17|:21... connected.
25 Logging in as anonymous ... Logged in!

```



```

27 ==> SYST ... done.      ==> PWD ... done.
    ==> TYPE I ... done.  ==> CWD /Linux/Mandriva/official
    /2007.0/i586/media/main/updates/media_info ... done.
    ==> PASV ... done.    ==> RETR synthesis.hdlist.cz ... done.
29 Length: 156,870 (153K) (unauthoritative)

31 100%[=====]
    156,870      99.67K/s

33 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

- 如何計算檔案 file1 的 md5 指紋？
Sol. `md5sum file1`
- 如何計算檔案 file1 的 md5 指紋，並存成 file1.md5？
Sol. `md5sum file1 >file1.md5`
- 如何計算檔案 file2 的 md5 指紋，並累加至 file1.md5？
Sol. `md5sum file2 >>file1.md5`
- 如何讀取檔案 file1 的 md5 指紋？
Sol. `md5sum file1`
- 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` 檢驗，發現檔案 md5 指紋不符，表示檔案被更動過。

17.3 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

```

2  rpm {-i|--install} [install-options] PACKAGE_FILE ...
    install-: options

```

```

4 | -h : 以安裝資訊列顯示安裝進度
  | -- : 不要去檢查nodeps rpm 套件的相依性。
  | -- : 不要檢查nomd5 rpm 套件的 MD5 資訊。
6 | -- : 不想讓該套件自行啓用或者自行執行某些系統指令。noscripts
  | -- : 直接覆蓋已存在的檔案。replacefiles
8 | -- : 重新安裝某個已經安裝過的套件。replacepkgs
  | -- : force--replacefiles 與 --replacepkgs 。
10 | -- : 測試套件是否可以被安裝到使用者的test Linux 環境當中。
  | --prefix : 安裝到新目錄NEPATH 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-3.7.3-4
  | mdk.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 #
  | ##### [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：

```

rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...
rpm {-F|--freshen} [install-options] PACKAGE_FILE ...
install- : options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：

```

[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.2006mdk.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
.2006mdk.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 --：不要去檢查nodeps rpm 套件的相依性。

```

```
--：只是測試，不是真的移除套件。常與一般選項 test -vv 一起使用來  
除錯
```

2. 測試是否可移除 mplayer：

```
[root@dywHome2 rpm]# rpm -e mplayer --test
error: Failed dependencies:
    mplayer is needed by (installed) kplayer-0.5.3-5
    mdv2007.0.i586
```

3. 測試不要檢查 rpm 套件的相依性下，是否可移除 mplayer：

```
[root@dywHome2 rpm]# rpm -e mplayer --test --nodeps
```

4. 移除套件 baghira：

```
[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：

```
[root@linux ~]# rpm -qa
[root@linux ~]# rpm -q[licdR] 已安裝的套件名稱
[root@linux ~]# rpm -qf 存在於系統上面的某個檔案

-q：僅查詢，後面接的套件名稱是否有安裝；
-qa：列出所有已經安裝在本機 Linux 系統上面的套件名稱；
-qi：列出該套件的詳細資訊 (information)，包含開發商、版本與說明等；
-ql：列出該套件所有的檔案與目錄所在完整檔名 (list)；
-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)
-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：

```
[root@linux ~]# rpm -qp[licdR] 未安裝的某個檔案名稱
-rp[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?

```
[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：

```
[root@linux ~]# rpm -ql logrotate
2 /etc/cron.daily/logrotate
  /etc/logrotate.conf以下省略
4 .....
```

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以下省略
9 -----
```

8. 查詢 logrotate 相依套件

```
1 [root@linux ~]# rpm -qR logrotate
/bin/sh
3 config(logrotate) = 3.7.1-10
libc.so.6以下省略
5 -----
```

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

```
[root@dywHome2 ~]# rpm -qc logrotate
2 /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-3.7.3-4
  mdk.i586.rpm\
3 > --replacepks
```

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：
5
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？

```

1 [root@dywHome2 rpm]# rpm -Vf /etc/crontab
2 .....T c /etc/crontab

```

6. 查詢 kplayer-0.5.3-5mdv2007.0.i586.rpm 內被更動過的檔案？

```

1 [root@dywHome2 rpm]# rpm -Vp kplayer-0.5.3-5mdv2007.0.i586.
rpm
2 S.5....T /usr/bin/kplayer

```

7. 查詢結果第一項資訊說明：

```

1 S : file Size differs檔案的容量大小被改變
2

```

```

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-1.4.5-1
   |   mdv2007.0

```

5. 查詢金鑰檔案及其設定檔：

```

1 [root@dywHome2 ~]# find /etc /usr -name RPM-GPG-KEY*
2 /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/sGlzkZ7WofaPgL2A7Vmi4aMLF9xNkIUzeekk2kxnZ+3
P277i2wckvIox
15 gGl130vqWUf6+20jBPWrOGz6hs+MUve7k+A7sgmg3n2P6fa999nxWTK+7
m7705x+2qXb+dxF中間省
    略

```

```

17 -----
    41trJiEWiEkEGBECAAkFAj/XUE4CGwwACgkQRFk1+
      HjQGfWi6wCeJPiCFBej0nfnBfaOWh98
    aNq3XfUAoKIfvhoiijSRwjSNmygJtZbiKOWC
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. `-vv`
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. `-vvv`
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 --no-md5`
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/local`
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. `-u`
17. rpm 指令升級與安裝之差異為何？
Sol. 升級與安裝的差異在於 rpm 指令升級時，會先檢查是否有舊版安裝，若無舊版安裝，則會直接安裝。
18. rpm 指令更新與安裝之差異為何？
Sol. 更新與安裝的差異在於 rpm 指令更新時，會先檢查是否有舊版安裝，若無舊版安裝，則會直接安裝。
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` & `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` & `rpm -Uvh a.i586.rpm` & `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 -qf 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. 當資料不小心被改；2. 刪除了某個套件的檔案；3. 不知道修改到某個套件的檔案內容。
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. 檔案的容量大小被改變
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. gpg
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