apt-get 與 make 指令之介紹

數學三 405210011 黃伯丞

（註：以下範例於 VirtualBox 上架設的 debian 與系上工作站實作）

一、apt-get

apt-get 可對某些套件進行安裝或移除等行為。 基本格式：**apt-get [選項] [命令] [套件名稱 1, 套件名稱 2, …]**

1. 附加選項：
-h：幫助訊息
-q：讓輸出作為記錄檔，不顯示進度
-qq：除了錯誤以外，什麼都不輸出
-d：僅下載，不馬上安裝或解開套件檔案
-s：不作實際操作，只是模擬執行命令
-y：對所有詢問都作肯定的回答，同時不作任何提示
-f：當沒有通過完整性測試時，仍嘗試繼續執行
-m：當有套件檔案無法找到時，仍嘗試繼續執行
-u：顯示已升級的套件列表
-b：在下載完源碼後，編譯生成相應的套件
-V：顯示詳盡的版本號碼
-c=?：讀取指定的設定檔案
-o=?：設定任意指定的設定選項，例如：-o dir::cache=/tmp

2. 命令：
apt-get update (軟體資料庫同步)
根據 /etc/apt/sources.list 中的設定到 APT Server 去更新軟體資料庫，在任何更新之前最好都先做這一個動作，讓軟體資料保持在最新的狀況之下。另外 /etc/apt/sources.list 可以用 apt-setup 來設定。
apt-get install (軟體安裝)
安裝軟體最怕的就是軟體間的相容性，但是在 Debian 裡頭安裝軟體時所有相依、相斥 Debian 都會幫我們自動解決。依照預設值，透過 sudo apt-get install 安裝軟體時，會將檔案暫存在 /var/cache/apt/archives/ 目錄裡
apt-get remove (軟體移除)
與 install 一樣，Debian 一樣會處理移除軟體時所發生的相依問題。
apt-get -purge remove 則連設定檔也會移除

apt-get autoremove (自動清除下載的暫存檔)

apt-get source

如果想取得某個軟體套件 (packages) 的原始碼可以透過這個指令達成。

apt-get upgrade (軟體升級)

平常我們很難顧慮到系統上所安裝的所有軟體的版本是否有新版出現，只要下這個指令 Debian 便會自動找出所有有新版的軟體套件並逐一升級。

apt-get clean

我們透過 apt-get 安裝的任何軟體都會先下載到 /var/cache/apt/archives/ 及 /var/cache/apt/archive/partial/ 目錄底下，一般預設 apt-get 在安裝完軟體後是不會把上述位置底下的.deb 檔清除，這個指令可讓系統自動清理這些目錄。

apt-get autoclean (自動清除下載的.deb 檔)

apt-get check

當遇到有問題的時候可以利用此指令來診斷問題所在。如果系統沒問題，會顯示下列訊息：

Reading Package Lists… Done

Building Dependency Tree… Done

```
root@debian:/home/banarry# apt-get install sudo
正在讀取套件清單... 完成
正在重建相依關係
正在讀取狀態資料... 完成
下列【新】套件將會被安裝：
  sudo
升級 0 個，新安裝 1 個，移除 0 個，有 0 個未被升級。
需要下載 1,055 kB 的套件檔。
此操作完成之後，會多佔用 3,108 kB 的磁碟空間。
下載:1 http://debian.csie.ntu.edu.tw/debian stretch/main amd64 sudo amd64 1.8.19
p1-2.1 [1,055 kB]
取得 1,055 kB 用了 1s (926 kB/s)
選取了原先未選的套件 sudo。
（讀取資料庫 ... 目前共安裝了 133251 個檔案和目錄。）
Preparing to unpack .../sudo_1.8.19p1-2.1_amd64.deb ...
Unpacking sudo (1.8.19p1-2.1) ...
設定 sudo (1.8.19p1-2.1) ...
Processing triggers for systemd (232-25+deb9u11) ...
Processing triggers for man-db (2.7.6.1-2) ...
```

圖一、使用 apt-get 安裝 sudo 指令

二、make

make 是編譯大量的原始碼的工具，藉由 Makefile 裡面所敘述的目標 (target)、所定義的規則 (rule) 來做編譯，產生出可執行的程式。一個大型的 C 專案會包含不少的原始碼，想要快速了解其脈絡，除了 README 等文件之外，Makefile 也能相當簡易的對專案架構迅速上手。

一般來說，當我們輸入 make 命令，會執行以下內容：

a. make 會在當前目錄下按順序找尋文件名為 GNUmakefile、makefile 或 Makefile 的文件。
b. 在 makefile 文件中的找到第一個目標文件（target），並把這個文件作為最終的目標文件。
c. 如果沒找到或目標文件所依賴的文件，或修改時間要比目標文件新，則 make 將執行後面所定義的命令來生成這個文件，如此遞迴下去找到文件彼此的依賴關係，直到最終編譯出第一個目標文件。

1. Makefile 基本架構

```
Target......:Prerequisite.....

<tab>Command.....
```

Target(目標)

一個目標檔，可以是 Object 檔，也可以是執行檔，還可以是一個標籤(Label)。

Prerequisite(先決條件)

要產生目標檔 (target) 所需檔案

Command(命令)

建立專案時需要執行的 shell 命令，每行一定要從<Tab>開始，之後再開始寫 command。

2. 變數(variable)使用

變數宣告時要使用 = 或 := 給予初始值。
如 obj = hello.o foo.o，取用時寫成 (obj) 或 {obj}。

在一些 make 中常使用**自動化變數**來簡寫規則：

$@：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，但不含副檔名。
$?：代表需要重建（被修改）的相依性項目。

另外在 makefile 規則中所用的萬用配對字元是% (可代替)

3. 常用的目標標籤 (Label)、偽目標

all: 包含編譯程式所需動作
install: 包含安裝執行檔的所需動作
clean: 清除 obj 檔
dist: 包含產生 configure 的動作
distclean: 包含清除 configure 所產生的檔案的步驟。

以下為我在其他課程中所使用的簡易 Makefile：

```
TARGET=shell

$(TARGET): fig1_10.o error.o
	$(CC) -o $@ $^

fig1_10.o: fig1_10.c
error.o: error.c

clean:
	rm -f fig1_10.o error.o $(TARGET)
```

圖二、編譯單一執行檔

```
all:exe_fig5_4  exe_fig5_5

exe_fig5_4: fig5_4.o error.o
	$(CC) -o $@ $^

exe_fig5_5: fig5_5.o error.o
	$(CC) -o $@ $^

fig5_4.o: fig5_4.c
fig5_5.o: fig5_5.c
error.o: error.c

clean:
	rm -f fig5_4.o error.o
	rm -f fig5_4.o error.o
```

圖三、編譯多重執行檔

三、參考資料

1. 簡單學 makefile：makefile 介紹與範例程式| Mr. Opengate (https://mropengate.blogspot.com/2018/01/makefile.html)
2. apt-get 指令一覽| 小倉庫 (https://b9532026.wordpress.com/2010/03/30/apt-get-%E6%8C%87%E4%BB%A4%E4%B8%80%E8%A6%BD-2/)
3. Makefile (https://pws.niu.edu.tw/~ttlee/sp.100.2/make/)
4. 上課筆記