在Linux安裝軟體

Reading material

- 軟體安裝：原始碼與 Tarball
  - http://linux.vbird.org/linux_basic/0520source_code_and_tarball.php

- 軟體安裝 RPM, SRPM 與 YUM
  - http://linux.vbird.org/linux_basic/0520rpm_and_srpm.php#rpmortarball

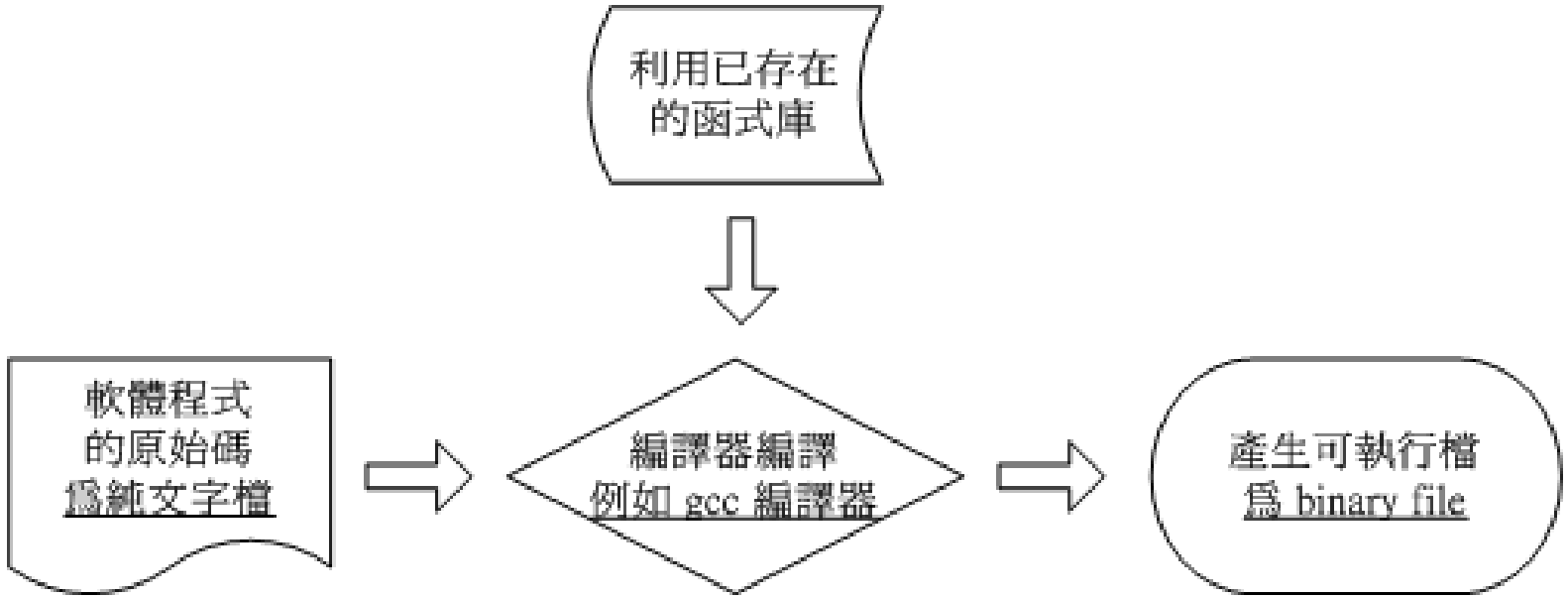
安裝軟體的三種方式

- 從程式碼編譯
  - tarball (各式linux皆適用)

- 軟體管理程式
  - rpm (centos) / dpkg(ubuntu)
  - 安裝已編譯好的執行檔

- 線上安裝
  - yum(centos) / apt-get (ubuntu)

源始碼、編譯器與可執行檔



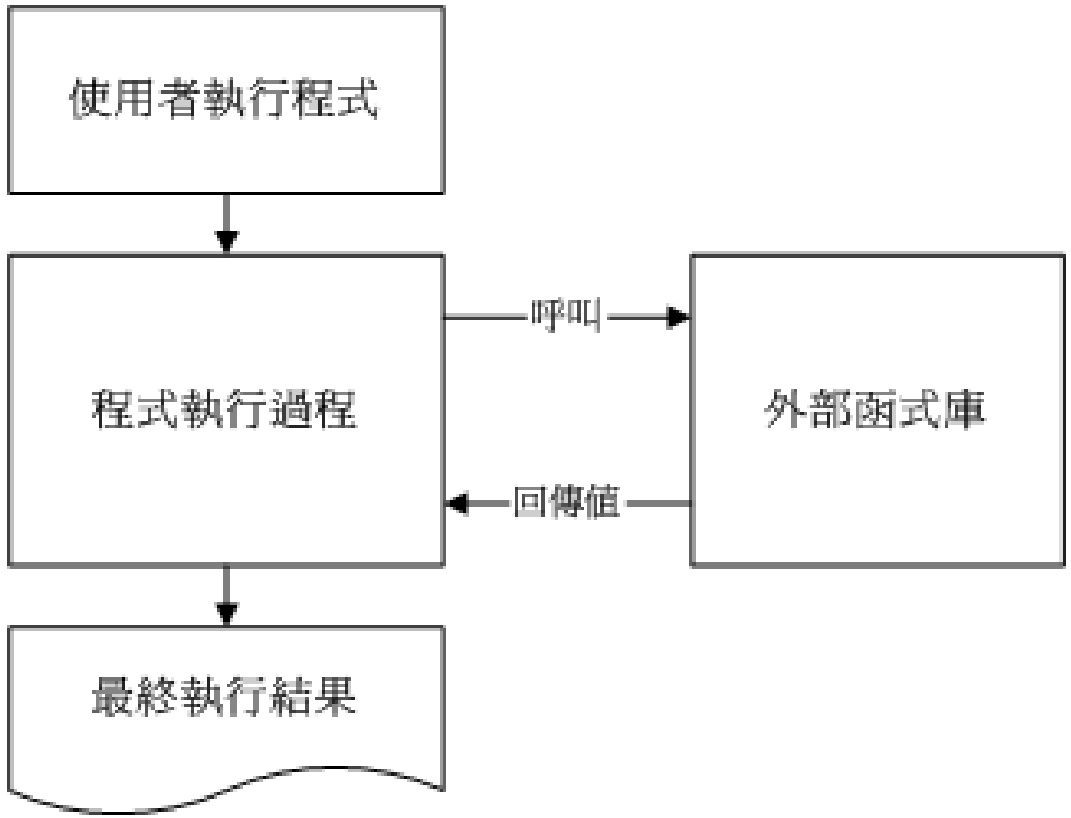
test.c

```
#include <stdio.h>
int main(void){
	printf("Hello World\n");
	thanks_2();
}

void thanks_2(){
	printf("Thank you!\n");
}
```

```
gcc test.c
```

撰寫副程式



thanks.c

```
#include <stdio.h>
int main(void){
	printf("Hello World\n");
	thanks_2();
}
```

thanks_2.c

```
#include <stdio.h>
void thanks_2(void){
	printf("Thank you!\n");
}
```

```
gcc -c thanks.c
gcc -c thanks_2.c
gcc -o main thanks.o thanks_2.o
```

呼叫外部函式庫

```
#include <stdio.h>
#include <math.h>
int main(void)
{
	float value;
	value = sin ( 3.14 / 2 );
	printf("%f\n",value);
}
```

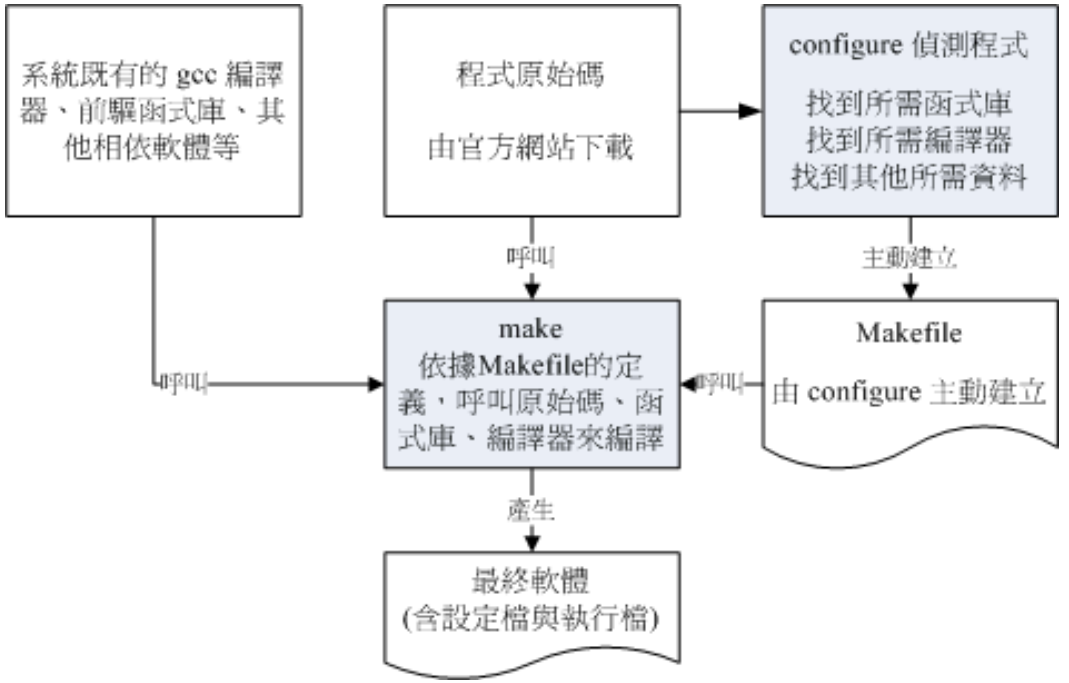
```
gcc -lm -L/lib64 sin.c
```

在/lib64目錄下，尋找libm.so做連結

```
gcc -lxyz -L/mylib test.c
```

在/mylib目錄下，尋找libxyz.so做連結

Configure & make



- 如果libm.so不存在
  - 透過configure檢查所需的library是否存在
  - 不存在的話就要先安裝

- 要手動打一堆gcc xxxx xxxx
  - 簡化編譯時所需要下達的指令；
  - 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；

Tarball安裝的基本步驟

- 取得原始檔
- 取得步驟流程：
  - 查閱 INSTALL 與 README 等相關檔案內容
- 相依屬性軟體安裝
  - 根據 INSTALL/README 的內容察看並安裝好一些相依的軟體 (非必要)；
- 建立 makefile
  - 以configure 偵測作業環境，並建立 Makefile 這個檔案；
- 編譯
  - 以 make 這個程式並使用該目錄下的 Makefile 做為他的參數設定檔，來進行 make (編譯或其他) 的動作；
- 安裝：
  - 以 make 這個程式，並以 Makefile 這個參數設定檔，依據 install 這個標的 (target) 的指定來安裝到正確的路徑！

Make 基本步驟

- ./configure
  - 在建立 Makefile 這個檔案囉！檢查 Linux 系統、相關的軟體屬性等等
- make
  - make 會依據 Makefile 當中的預設工作進行編譯的行為！使用 make 就是要將原始碼編譯成為可以被執行的可執行檔，而這個可執行檔會放置在目前所在的目錄之下， 尚未被安裝到預定安裝的目錄中
- make install
  - make 會依據 Makefile 這個檔案裡面關於 install 的項目，將上一個步驟所編譯完成的資料給他安裝到預定的目錄中，就完成安裝啦！
- make clean
  - make 會讀取 Makefile 中關於 clean 的工作。清除上次編譯過的目標檔案 (*.o) 存在

Make範例: 以安裝ntp為例

- 解壓縮下載的 tarball ，並參閱 README/INSTALL 檔案

```
[root@study ~]# cd /usr/local/src   <==切換目錄
[root@study src]# tar -zxvf /root/ntp-4.2.8p3.tar.gz  <==解壓縮到此目錄
ntp-4.2.8p3/          <==會建立這個目錄喔！
ntp-4.2.8p3/CommitLog
....(底下省略)....
[root@study src]# cd ntp-4.2.8p3
[root@study ntp-4.2.8p3]# vi INSTALL  <==記得 README 也要看一下！
# 特別看一下 28 行到 54 行之間的安裝簡介！可以瞭解如何安裝的流程喔！
```

- 檢查 configure 支援參數，並實際建置 makefile 規則檔

```
[root@study ntp*]# ./configure --help | more  <==查詢可用的參數有哪些
  --prefix=PREFIX         install architecture-independent files in PREFIX
  --enable-all-clocks     + include all suitable non-PARSE clocks:
  --enable-parse-clocks   - include all suitable PARSE clocks:
# 上面列出的是比較重要的，或者是你可能需要的參數功能！

[root@study ntp*]# ./configure --prefix=/usr/local/ntp \
>  --enable-all-clocks --enable-parse-clocks  <==開始建立makefile
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
....(中間省略)....
checking for gcc... gcc          <==也有找到 gcc 編譯器了！
....(中間省略)....
config.status: creating Makefile  <==現在知道這個重要性了吧？
config.status: creating config.h
config.status: creating evconfig-private.h
config.status: executing depfiles commands
config.status: executing libtool commands
```

- 開始編譯與安裝

```
[root@study ntp*]# make clean; make
[root@study ntp*]# make check
[root@study ntp*]# make install
# 將資料給他安裝在 /usr/local/ntp 底下
```

例題:

- 下載screen的tarball (https://ftp.gnu.org/gnu/screen/screen-4.6.2.tar.gz)

- 使用./configure 建立makefile
  - ncurses-devel
- 使用make 進行編譯
- 使用make install 進行安裝

軟體管理程式: RPM

- 從原始碼安裝很麻煩
  - 偵測作業系統與環境、
  - 設定編譯參數、
  - 實際的編譯、
  - 安裝軟體

- 若有人先將軟體編譯好所需要的軟體， 使用者只要安裝這個編譯好的可執行的軟體
  - RPM: 用在centos
  - dpkg: 用在ubuntu

RPM的優點

- RPM 內含已經編譯過的程式與設定檔等資料，可以讓使用者免除重新編譯的困擾；
- RPM 在被安裝之前，會先檢查系統的硬碟容量、作業系統版本等，可避免檔案被錯誤安裝；
- RPM 檔案本身提供軟體版本資訊、相依屬性軟體名稱、軟體用途說明、軟體所含檔案等資訊，便於瞭解軟體；
- RPM 管理的方式使用資料庫記錄 RPM 檔案的相關參數，便於升級、移除、查詢與驗證。

Rpm檔名資訊

- ntp-4.2.6p5-28.el7.centos.x86_64.rpm
  - 軟體名稱
  - 版本資訊
  - 釋出版本次數
  - 操作硬體平台

平台名稱	適合平台說明
i386	幾乎適用於所有的 x86 平台，不論是舊的 Pentium 或者是新的 Intel Core 2 與 K8 系列的 CPU 等等，都可以正常的工作！那個 i 指的是 Intel 相容的 CPU 的意思，至於 386 不用說，就是 CPU 的等級啦！
i586	就是針對 586 等級的電腦進行最佳化編譯。那是哪些 CPU 呢？包括 Pentium 第一代 MMX CPU， AMD 的 K5, K6 系列 CPU (socket 7 插腳) 等等的 CPU 都算是這個等級；
i686	在 Pentium II 以後的 Intel 系列 CPU ，及 K7 以後等級的 CPU 都屬於這個 686 等級！ 由於目前市面上幾乎僅剩 P-II 以後等級的硬體平台，因此很多 distributions 都直接釋出這種等級的 RPM 檔案。
x86_64	針對 64 位元的 CPU 進行最佳化編譯設定，包括 Intel 的 Core 2 以上等級 CPU ，以及 AMD 的 Athlon64 以後等級的 CPU ，都屬於這一類型的硬體平台。
noarch	就是沒有任何硬體等級上的限制。一般來說，這種類型的 RPM 檔案，裡面應該沒有 binary program 存在， 較常出現的就是屬於 shell script 方面的軟體。

Rpm 指令

- 查詢
  - -q

```
[root@study ~]# rpm -qa                                  <==已安裝軟體
[root@study ~]# rpm -q[licdR] 已安裝的軟體名稱           <==已安裝軟體
[root@study ~]# rpm -qf 存在於系統上面的某個檔名         <==已安裝軟體
[root@study ~]# rpm -qp[licdR] 未安裝的某個檔案名稱      <==查閱RPM檔案
選項與參數：
查詢已安裝軟體的資訊：
-q   ：僅查詢，後面接的軟體名稱是否有安裝；
-qa  ：列出所有的，已經安裝在本機 Linux 系統上面的所有軟體名稱；
-qi  ：列出該軟體的詳細資訊 (information)，包含開發商、版本與說明等；
-ql  ：列出該軟體所有的檔案與目錄所在完整檔名 (list)；
-qc  ：列出該軟體的所有設定檔 (找出在 /etc/ 底下的檔名而已)
-qd  ：列出該軟體的所有說明檔 (找出與 man 有關的檔案而已)
-qR  ：列出與該軟體有關的相依軟體所含的檔案 (Required 的意思)
-qf  ：由後面接的檔案名稱，找出該檔案屬於哪一個已安裝的軟體；
-q --scripts：列出是否含有安裝後需要執行的腳本檔，可用以 debug 喔！
查詢某個 RPM 檔案內含有的資訊：
-qp[icdlR]：注意 -qp 後面接的所有參數以上面的說明一致。但用途僅在於找出
            某個 RPM 檔案內的資訊，而非已安裝的軟體資訊！注意！
```

- 安裝
  - -i

```
[root@study ~]# rpm -ivh package_name
選項與參數：
-i ：install 的意思
-v ：察看更細部的安裝資訊畫面
-h ：以安裝資訊列顯示安裝進度

範例一：安裝原版光碟上的 rp-pppoe 軟體
[root@study ~]# rpm -ivh /mnt/Packages/rp-pppoe-3.11-5.el7.x86_64.rpm
Preparing...                          ################################# [100%]
Updating / installing...
   1:rp-pppoe-3.11-5.el7              ################################# [100%]

範例二、一口氣安裝兩個以上的軟體時：
[root@study ~]# rpm -ivh a.i386.rpm b.i386.rpm *.rpm
# 後面直接接上許多的軟體檔案！

範例三、直接由網路上面的某個檔案安裝，以網址來安裝：
[root@study ~]# rpm -ivh http://website.name/path/pkgname.rpm
```

- 移除
  - -e:
  - 因為軟體屬性相依導致無法移除某些軟體的問題

```
# 1. 找出與 pam 有關的軟體名稱，並嘗試移除 pam 這個軟體：
[root@study ~]# rpm -qa | grep pam
fprintd-pam-0.5.0-4.0.el7_0.x86_64
pam-1.1.8-12.el7.x86_64
gnome-keyring-pam-3.8.2-10.el7.x86_64
pam-devel-1.1.8-12.el7.x86_64
pam_krb5-2.4.8-4.el7.x86_64
[root@study ~]# rpm -e pam
error: Failed dependencies:  <==這裡提到的是相依性的問題
        libpam.so.0()(64bit) is needed by (installed) systemd-libs-208-20.el7.x86_64
        libpam.so.0()(64bit) is needed by (installed) libpwquality-1.2.3-4.el7.x86_64
....(以下省略)....

# 2. 若僅移除 pam-devel 這個之前範例安裝上的軟體呢？
[root@study ~]# rpm -e pam-devel  <==不會出現任何訊息！
[root@study ~]# rpm -q pam-devel
package pam-devel is not installed
```

範例:

- wget http://mirror.centos.org/centos/7/os/x86_64/Packages/screen-4.1.0-0.25.20120314git3c2946.el7.x86_64.rpm

- 透過rpm安裝screen

例題:

- $wget http://mirror.centos.org/centos/7/os/x86_64/Packages/ntp-4.2.6p5-28.el7.centos.x86_64.rpm
- 透過rpm 安裝ntp

- 相依rpm
  - http://mirror.centos.org/centos/7/os/x86_64/Packages/autogen-libopts-5.18-5.el7.x86_64.rpm
  - http://mirror.centos.org/centos/7/os/x86_64/Packages/ntpdate-4.2.6p5-28.el7.centos.x86_64.rpm

RPM的缺點

- 軟體間是有相關性:
  - 例如要安裝網路卡驅動程式，就得要有與 gcc 及 make 等軟體

- 軟體的屬性相依
  - RPM 在打包的軟體時，會加入一些訊息，包括軟體的版本、 打包軟體者、相依屬性的其他軟體、等等
  - 安裝某個以 RPM 型態提供的軟體時，RPM 會去檢驗一下資料庫裡面是否已經存在相依的軟體， 如果資料庫顯示不存在，那麼這個 RPM 檔案『預設』就不能安裝

例題:

- 先安裝相依rpm
  - http://mirror.centos.org/centos/7/os/x86_64/Packages/autogen-libopts-5.18-5.el7.x86_64.rpm
  - http://mirror.centos.org/centos/7/os/x86_64/Packages/ntpdate-4.2.6p5-28.el7.centos.x86_64.rpm

- 再透過rpm 安裝ntp
  - http://mirror.centos.org/centos/7/os/x86_64/Packages/ntp-4.2.6p5-28.el7.centos.x86_64.rpm

例題:

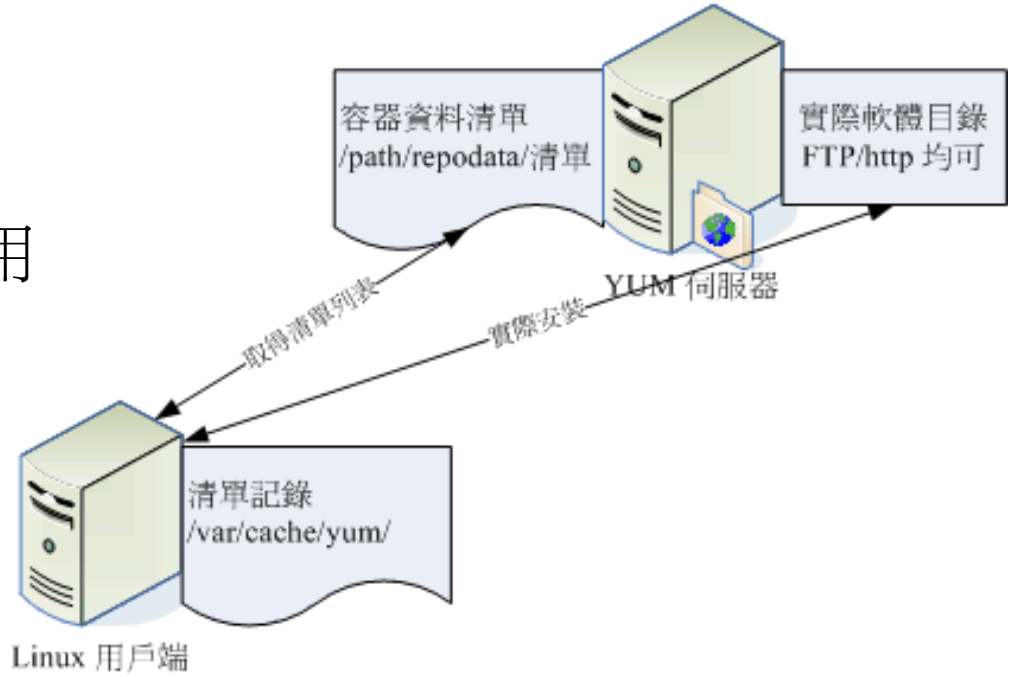
- 透過rpm 移除 ntpdate，請問此時能不能移除

- 先透過rpm移除ntp後，再移除 ntpdate，請問此時能不能移除

YUM 線上升級機制

- Yum 伺服器有所有要安裝的rpm相依資訊
- 當用戶要安裝軟體時，用戶主機會向網路上的 yum 伺服器下載所需的rpm

- 用戶主機要設定合適的yum伺服器位址
  - CentOS 預設已經有多部映射站台可直接使用



Yum指令

- 查詢
  - yum [list|search]查詢項目
- 安裝
  - yum [install|update] 軟體
- 移除
  - yum [remove] 軟體

範例:

- 透過yum 安裝screen
- 透過yum 移除screen

例題:

- 透過yum 安裝ntp
- 透過yum 移除ntp