資料來源: http://www.cc.ntu.edu.tw/chinese/epaper/0020/20120320_2005.html

GNU程式開發工具簡介

作者：周秉誼 / 臺灣大學計算機及資訊網路中心作業管理組碩士後研究人員

GNU開發工具是Linux作業系統、嵌入式系統、與自由及開放原始碼軟體社群中，最常見、使用得最廣泛的程式開發工具，可以協助程式設計人員完成從原始程式碼到二進制機械碼時各項煩瑣的工作，能夠大幅提昇程式的執行效率。不但支援多種程式語言、多種作業系統，也可以產生多種不同處理器的執行檔，是跨平台程式開發和移植的最佳工具。

前言

程式設計和開發是一個撰寫程式碼，實作特定演算法來解決、處理特定問題的過程，最終的目的是要產生一個計算機處理器可以辨視的指令序列、讓計算機處理器遵循這個指令序列，正確地、有效率地處理特定問題。不但要將文字的程式碼，產生為計算機處理器可以辨視的指令序列或二進制機械碼，還需要依作業系統不同，在機械碼中加入不同的資訊和指令才能呼叫和使用系統的資源。因此程式設計人員需要程式開發工具的協助，完成從原始程式碼到二進制機械碼的步驟—編譯 (compile)、組譯 (assemble) 和連結 (link)。

GNU開發工具

GNU開發工具 (GNU toolchain) 是Linux作業系統、嵌入式系統 (Embedded System)、與自由及開放原始碼軟體 (Free and Open Source Software, FOSS) 社群中，最常見、使用得最廣泛的程式開發工具。GNU開發工具是由GNU計劃 (GNU Project) 中的數個程式開發工具所集合而成，可以完成編譯、組譯和連結等步驟，還有自動化設定、自動化編譯、除錯工具等功能。GNU計劃是1983年時理查史托曼 (Richard Stallman) 提出的，目的是要建立一個由自由軟體組成的作業系統。1992年時，Linux核心以GNU通用公共授權條款 (GPL) 釋出，與GNU計劃組成GNU/Linux作業系統，而成為最多人使用的自由軟體作業系統家族，GNU開發工具也成為這些系統最主要的開發工具。

GNU make	自動化編譯工具
GNU Compiler Collection (GCC)	編譯器套件
GNU Binutils	二進制工具
GNU Debugger (GDB)	除錯器
GNU Build System	自動化程式建構系統

表一 GNU開發工具常用工具清單

GNU編譯器

GNU開發工具最重要、也是最常直接使用到的部份，就是GNU編譯器套件 (GNU Compiler Collection, GCC)。GCC最初是在1987年被釋出，一開始只能針對C語言進行編譯，後來陸續加入各種不同程式語言的支援，而成為一個跨語言的編譯器套件。目前最新的穩定版本 (stable version) 是4.6版，可以對C、C++、Fortran、Objective-C、Java、Ada、還有Go等程式語言進行編譯。編譯的目標處理器類型包括了IA-32、IA-64、x86-64、PowerPC、ARM等等，超過20種處理器架構，因此也可以使用GCC進行跨平台編譯 (cross compile)。

使用GCC編譯程式碼只要呼叫各程式語言對應的GCC指令，C語言是gcc、C++是g++、Fortran是gfortran等，就可以產生可以執行的程式檔案。GCC指令也接受命令列的參數來設定編譯的環境選項，如-I參數用來指定標頭檔 (header) 的目錄位置、-o參數可以指定產生的程式檔名、-E參數命令GCC只進行前處理 (preprocessing) 不進行編譯、-O參數用來調整最佳化等級 (optimization level)、-fopenmp參數可以開啟OpenMP平行計算的功能等、-Wall參數會讓GCC顯示編譯時所有的警告 (warning) 資訊。

-I	指定標頭檔位置 (-I/path/to/header)
-o	指定輸出檔名 (-o abc.exe)
-E	只進行前處理，不編譯
-O	調整最佳化等級 (-O0~-O3)
-fopenmp	開啟OpenMP
-Wall	顯示所有警告
-S	只進行編譯，不組譯
-c	只進行組譯，不連結
-L	指定函式庫位置 (-L/path/to/lib)
-l	指定函式庫名稱 (-lm)
-v	顯示所有編譯資訊
-pipe	使用管道傳遞程式碼，不用暫存檔

表二 GCC常用參數清單

GCC最大的難題是如何利用同樣的技術，對不同程式語言及不同處理器架構進行最佳化，為了整合不同程式語言的編譯並提高最佳化的效率，GCC使用中間表示式 (Intermediate Representation) 來最佳化處理及程式碼轉換。GCC針對不同程式語言有不同的前端 (front-end)，用來解析程式碼進行前處理、語法分析 (syntax analysis) 和語意分析 (semantic analysis)，將程式碼轉換為通用的中間表示式。GCC使用的中間表示式又分為三個層級，方便GCC進行不同的最佳化處理，最後產生目標處理器的組合語言。

GNU二進制工具

然而GCC是編譯器，編譯產生的組合語言程式碼又是怎麼自動變成機械碼和可執行的檔案的呢？原來是GCC會呼叫GNU開發工具中的其他部份，如組譯器、連結器等，直接產生可以運行的程式，簡化程式編譯的流程，也節省了程式設計人員花費在程式編譯的

時間。而被GCC呼叫來進行二進制機械碼處理的這些工具，就合稱為GNU二進制工具 (GNU Binutils)，這些工具包括了組譯器as、連結器ld、靜態函式庫操作工具ar等等。如果希望GCC只進行編譯，不做組譯及連結等後續動作，可以用-S參數就只會產生副檔名為.s的組合語言程式檔。

as	組譯器
ld	連結器
gprof	效能分析工具
ar	靜態函式庫工具
objdump	目標檔傾印工具

表三 GNU二進制工具常用工具清單

GNU組譯器

GNU組譯器 (GNU Assembler, GAS) 是GCC預設的後端工具，用來將GCC輸出的組合語言程式檔，轉換為處理器可以辨視的二進制機械碼，產生副檔名為.o的目標檔 (object file)。目標檔通常會包括符號表 (symbol table)、重新定址表 (relocation table)等資訊，方便連結器及共享函式庫 (shared library) 等執行期間才能決定的位址進行連結。在開發大型程式時，也可以將每一個C語言程式檔，先組譯成一個個目標檔，之後再用連結器將每個目標檔連結成為可執行檔；當單一程式碼有修改時，就不用把整個程式重新編譯，只要針對有修改的程式碼產生目標檔，與現有的目標檔進行連結就可以了，能夠大大減少編譯程式花費的時間。在GCC中，如果只要產生目標檔而不進行連結，可以使用-c參數。GNU組譯器的指令名稱是as，除了透過GCC呼叫外，也可以直接組譯現有的組合語言程式。

GNU連結器

GNU連結器 (GNU linker, GNU ld) 最主要的工作就是把不同的目標檔和函式庫進行連結，產生出單一個可執行檔、或把多個目標檔整合成函式庫供其他程式使用。連結器不只要把目標檔的指令區塊及資料區塊重新組織排列到執行檔中，依據排列的結果更新函數呼叫的位址，還要加入作業系統和動態連結 (dynamic linking) 的指令碼和位址資訊，所進行的計算相當煩瑣複雜。因此對程式設計人員來說，連結器能讓程式的各個元件以模組方式開發，又完成最煩瑣的工作，可說是GNU開發工具中最辛苦的幕後黑手。要連結的函式庫名稱和路徑可以透過-L及-l參數由GCC傳遞給連結器，也可以用-static參數來強制程式要使用靜態連結。

GNU C函式庫

在GNU開發環境中，最常被連結到的函式庫就是GNU C函式庫 (GNU C Library, glibc)，它是一個GNU計劃開發的C語言標準函式庫 (C Standard Library, libc)，包括了C語言中各種資料型態 (data type)、字串、數學計算、輸入輸出、記憶體配置等函數。GCC編譯出來的程式都會以動態連結的方式和GNU C函式庫做連結，可以有效地縮小程式執行檔的檔案大小。因此在Linux作業系統中，大部份的程式都和GNU C函式庫有連結，所以GNU C函式庫版本的更新會對整個系統的層面造成影響。GNU C函式庫在Linux作業系統中，通常會把數學函式庫 (libm) 獨立出來，如果程式中有使用到數學函數，如exp()、sin()等，在編譯時就要加上-lm參數，讓程式與數學函式庫連結。

GCC的編譯工作是由數個不同的GNU開發工具一同完成，GCC會自動地收集和整理相關設定和參數，傳遞給各個工具，讓編譯工作進行得更順利。想要了解各個工具的參數，可以使用-v參數讓GCC把每個步驟的指令和參數顯示出來。因為GCC會將編譯完成的組合語言程式碼存在暫存檔中，交給組譯器進行組譯，如果大量編譯程式碼時，就會產生大量的磁碟存取，會影響編譯速度和系統效能，所以也可以加上-pipe參數讓GCC使用系統的管道 (pipe) 來傳送程式碼。

```
$ gcc -fopenmp -Wall -O2 -o pgmp-chudnovsky pgmp-chudnovsky.c -lgmp -lm -v
Using built-in specs.
Target: x86_64-linux-gnu
(…skip…)
gcc version 4.3.2 (Debian 4.3.2-1.1)
COLLECT_GCC_OPTIONS='-fopenmp' '-Wall' '-O2' '-o' 'pgmp-chudnovsky' '-v' '-mtune=generic' '-pthread'
/usr/lib/gcc/x86_64-linux-gnu/4.3.2/cc1 -quiet -v -D_REENTRANT pgmp-chudnovsky.c -quiet -dumpbase
pgmp-chudnovsky.c -mtune=generic -auxbase pgmp-chudnovsky -O2 -Wall -version -fopenmp -o
/tmp/ccuMZcHP.s
(…skip…)
as -V -Qy -o /tmp/ccdXACoZ.o /tmp/ccuMZcHP.s
GNU assembler version 2.18.0 (x86_64-linux-gnu) using BFD version (GNU Binutils for Debian)
2.18.0.20080103
(...skip...)
/usr/lib/gcc/x86_64-linux-gnu/4.3.2/collect2 --eh-frame-hdr -m elf_x86_64 --hash-style=both -dynamic-
linker /lib64/ld-linux-x86-64.so.2 -o pgmp-chudnovsky
(…skip…)
$ ldd pgmp-chudnovsky
linux-vdso.so.1 =>  (0x00007fff437ff000)
libgmp.so.3 => /usr/lib/libgmp.so.3 (0x00007f5d4c0ab000)
libm.so.6 => /lib/libm.so.6 (0x00007f5d4be28000)
libgomp.so.1 => /usr/lib/libgomp.so.1 (0x00007f5d4bc20000)
libpthread.so.0 => /lib/libpthread.so.0 (0x00007f5d4ba04000)
libc.so.6 => /lib/libc.so.6 (0x00007f5d4b6b1000)
librt.so.1 => /lib/librt.so.1 (0x00007f5d4b4a8000)
/lib64/ld-linux-x86-64.so.2 (0x00007f5d4c2ea000)
```

表四 GCC編譯與輸出範例

其他GNU開發工具還有用來產生靜態函式庫的ar、用來傾印目標檔資訊的objdump、效能分析程式gprof、除錯器gdb、自動化編譯工具gmake、及用來自動化編譯程式的GNU程式建構系統 (GNU Build System) 等等不同功能的工具。其中的GNU make是專案管理的好工具，它會依照Makefile檔案裡的規則，協助程式設計人員只針對有修改的程式碼進行編譯，減少等待重複編譯的時間，也能用來進行latex文件編譯、版本控制系統的同步、或進行其他自動化工作，功能強大而且用途廣泛。

結語

GNU開發工具能協助程式設計人員完成從原始程式碼到可執行檔的每一個步驟，不只能產生正確的程式，還注重程式執行檔的效能。也提供了自動化的程式建構工具，讓大型軟體在跨平台的開發和移植工作更為簡單。和其他整合開發環境 (Integrated Development Environment, IDE) 最大的不同在於GNU開發工具不綁定程式碼編輯器，程式設計人員可以使用任何的文字編輯器，如GNU Emacs、vi/vim、UltraEditor等等來撰寫程式碼。沒有圖形化介面也讓GNU開發工具執行上更有效率、更適合用在大型軟體編譯、大量軟體套件及跨平台的自動化建構系統。因此也讓GNU開發工具成為Linux作業系統、嵌入式系統、及開放原始碼、自由軟體社群中，最常見、使用得最廣泛的程式開發工具。