GCC 的使用

簡介：
GCC 是由 GNU 出的 C 語言編譯器，可將由 ANSI C 或 traditional C 語言寫成的程式碼編譯成可執行檔。由於 GCC 能分別編譯出可執行於許多不同硬體、作業系統下的程式，在 UNIX 系統上是相當多人常用的 C 語言編譯程式。
使用方法簡介

基本的使用方法及常用指令

編譯器在編譯過程中，先將程式碼編譯成 object 檔，然後再和程式庫聯結，成爲可執行檔。故一個編譯器須提供的參數主要有幾類：
指定編譯器編出的 object 檔或是可執行檔檔名。
在編譯過程做最佳化，可提升程式的執行速度。
設定搜尋程式庫的標頭檔 (header file) 及程式庫檔的目錄及指定程式庫檔檔名。
提供進一步的資訊以便使用者找尋程式中的錯誤。
以下便以這四個大類分別介紹。
注意：下面在不同類別中所介紹的參數，幾乎都可以混合著使用。

1. 設定編譯出的 object 檔檔名或是可執行檔檔名：
參數： -o out_put_filename
說明： 指定編譯出的檔名爲 out_put_filename。
範例： 本例將程式碼 'test.c' 編譯成可執行檔，並設定檔名爲 'test'。

```
gcc test.c -o test
```

2. 在編譯過程做最佳化：
參數： -O
說明： 在編譯過程做最佳化，以提升增快程式執行速度。
範例：本例將程式碼 'test.c' 編譯成可執行檔 'test'，並在編譯過程做最佳化'。

```
gcc -O test.c -o test
```

範例： 將 test1.c 和 test2.c 編譯聯結成可執行檔 test 並在編譯過程中做最佳化。

```
gcc -O test1.c test2.c -o test
```

3. 設定搜尋標頭檔目錄、程式庫檔的目錄及指定程式庫檔：

設定搜尋標頭檔目錄

參數： -Idir_name
說明： 將目錄 'dir_name' 設定爲搜尋標頭檔目錄之一。

設定搜尋程式庫目錄

參數： -Ldir_name
說明： 將目錄 'dir_name' 設定爲搜尋程式庫目錄之一。

設定程式庫檔案

參數： -lname
說明： 聯結程式庫 libname.a 。
範例一：本例中假設你的程式檔名爲 test.c，數學函數如 sin 等，所要聯結的程式庫爲 libm.a。

```
gcc -O test.c -o test -lm
```

範例二：本例中假設你的程式檔名爲 test.c，使用到 X window 函數，所需的 include 檔放在 /usr/X11R6/include 中，所須聯結的程式庫放在 /usr/X11R6/lib，所要聯結的程式庫爲 libX11.a。

```
gcc -I/usr/X11R6/include -L/usr/X11R6/lib -lX11 test.c -o test
```

4. 提供進一步的資訊以便使用者找尋程式中的錯誤：

參數： -Wall
說明： 輸出較多的警告訊息，以便找出程式的錯誤。

範例： 編譯 test.c 時輸出較多的警告訊息。

```
gcc -Wall test.c
```

參數： -g
說明： 在編譯出可執行檔時，附加執行時除錯資訊，以供 gdb 讀取 (若要使用 ABSoft 的除錯程式，則須將參數改爲 -gdwarf)。
範例： 將 'test.c' 編譯成可執行檔 'test'，並附加除錯資訊。

```
gcc -g test.c -o test
```

進階參數

1. 僅編譯成 object 檔：

參數： -c

說明： 僅編譯成 object 檔而不進行程式庫聯結。

範例： 將 test.c 編譯成 object 檔 test.o。

```
gcc -c test.c -o test.o
```

2. 聯結數個 object 成可執行檔：

範例： 將 'test1.o'、'test2.o' 和程式庫聯結後成爲可執行檔 test。

```
gcc test1.o test2.o -o test
```

3. 觀察巨集展開情形：

參數： -E

說明：展開程式中的巨集以便了解巨集是否依照預期方式展開。

範例：將 test1.c 中的巨集展開後儲存到 test1.c.ext 。

```
gcc -E test1.c > test1.c.ext
```

4. 產生組合語言程式碼：

參數： -S

範例：編譯 test.c 產生對應的組合語言程式碼檔 test.s。

```
gcc -S test.c -o test.s
```

總整理

編譯參數列表 -o out_put_filename 將編譯後產生的檔名設爲 out_put_filename
-O 編譯時做最佳化，以增加程式執行效率。
-c 將原始語言編譯成 .o 檔(object檔) 不做程式庫連結的工作。
-g 編譯出 .o 檔時，保留除錯的資訊，在連結後產生的可執行檔中包含 gdb 需要的資訊。(若要給 ABSoft 的除錯程式使用，須將此參數改爲 -gdwarg)
-E 將原始語言編譯將 test.c 中的巨集展開，之後輸出到 stdout。
-L/usr/X11R6/lib 連結程式時，搜尋程式庫檔時，將 /usr/X11R6/lib 列入搜尋目錄。
-lm 在連結程式時，將 libm.a 列入搜尋檔案中。
-I/usr/X11R6/include 到 /usr/X11R6/include 找尋被 include 的檔案。
-S 產生 test.c 對應的組合語言程式檔 test.s 。
-Wall 產生比較多的警告訊息 (平常不須使用，但當找不到程式中的錯誤時，可以加以使用。)

更多的說明

man gcc
info gcc

實作-靜態函式庫

參考檔案：fred.c, bill.c, program.c, lib.h

fred.c

```
#include <stdio.h>

void fred(int arg)
{
    printf("fred: you passed %d\n", arg);
}
```

bill.c

```
#include <stdio.h>

void bill(char *arg)
{
    printf("bill: you passed %s\n", arg);
}
```

Program.c

```
#include <stdlib.h>
#include "lib.h"

int main()
{
    bill("Hello World");
    exit(0);
}
```

lib.h

```
void bill(char *);
void fred(int);
```

1. gcc –c bill.c fred.c
   ls *.o
2. gcc –c program.c
   gcc –o prgram program.o bill.o
   ./program