**MSVC與CRT的恩怨情仇**

很久沒有寫程式設計入門知識的相關文章了，這篇文章要來談談程式庫 (Library) 連結，以及關於 MSVC 與 CRT 之間的種種恩怨情仇。

如果你使用的作業系統是 Linux、Mac 或其他非 Windows 平台，你可以忽略這篇文章；如果你使用的作業系統是 Windows 平台，但沒有用 Microsoft Visual Studio C++（以下簡稱為 MSVC）軟體撰寫 C++ 程式的話，這篇文章對你的幫助可能很有限；但如果你的作業系統是 Windows，而且你使用的程式整合開發環境是 MSVC 軟體撰寫 C++ 程式的話，這篇文章應該能夠幫助你釐清一些重要的基礎觀念。

身為程式設計者，在學習程式設計的過程中，你是否曾經遇過某些看起來不知所云的錯誤訊息，卻不知該如何解決？例如當你快快樂樂地寫完程式，並且確認所有的程式碼都能成功通過編譯之後，接著執行「建置方案」(Build Solution) 的步驟，結果卻跑出一堆莫名其妙的錯誤：

*LIBCMTD.lib(mlock.obj) : error LNK2005: \_\_lock 已在 MSVCRTD.lib(MSVCR80D.dll) 中定義過了  
LIBCMTD.lib(mlock.obj) : error LNK2005: \_\_unlock 已在 MSVCRTD.lib(MSVCR80D.dll) 中定義過了  
LIBCMTD.lib(crt0.obj) : error LNK2005: \_mainCRTStartup 已在 MSVCRTD.lib(crtexe.obj) 中定義過了*

*…………*

*LINK : warning LNK4098: 預設的程式庫 ‘MSVCRTD’ 與其他使用的程式庫衝突，請使用 /NODEFAULTLIB:library  
LINK : warning LNK4098: 預設的程式庫 ‘LIBCMTD’ 與其他使用的程式庫衝突，請使用 /NODEFAULTLIB:library  
D:\Workspace\CrtLibTest\Debug\CrtLibTest.exe : fatal error LNK1169: 找到有一或多個已定義的符號*

以一般的情況來說，如果在你的程式專案中有使用某些由他人所撰寫的第三方程式庫或是開源專案的程式庫，比較容易會發生上述的錯誤狀況。從上述這些看似離奇而令人摸不著頭緒的錯誤訊息中，我們大概可以猜測問題點應該在於 LIBCMTD.lib 與 MSVCRTD.lib 這兩個程式庫身上。**但到底什麼是 LIBCMTD.lib 和 MSVCRTD.lib？在我們的程式碼中有使用這些程式庫嗎？**

答案是肯定的。

熟悉 C 語言的程式設計者都知道，如果要使用 printf()、scanf() 或者 fopen() 等等 C 語言的基本 I/O 操作函式時，首先必須用 #include 語法將 stdio.h 這個標頭檔納入我們的程式碼中。藉由 stdio.h 中對這些 I/O 操作函式所做出的函式宣告 (function declaration)，編譯器 (Compiler) 才得以確認 printf、scanf 以及 fopen 等等都是合法可用的函式。

而當我們撰寫的程式碼經過編譯器產出 OBJ 形式的檔案之後，需要再經由連結器 (Linker) 的處理程序，將程式碼中全部有使用到的函式定義 (function definition) 連結建置起來，才能夠產生出最後的程式執行檔。問題來了，我們知道 printf、scanf 以及 fopen 的函式宣告存在於 stdio.h 當中，但是這些傢伙的函式定義，也就是真正的實做程式碼，究竟存放在什麼地方呢？

**在 C 語言的標準程式庫中。**

由 C 語言所制訂的標準程式庫，稱之為**「執行階段程式庫」**，也就是 **C Run-Time Library**，通常可簡稱為 **CRT**。在 C 語言的標準程式庫中，包含了一組常用的基礎函式，例如 I/O 處理與字串操作程序等等，所以只要我們使用 C 語言撰寫程式碼，就一定要將編譯完成後的程式碼 OBJ 檔，連結至 C 語言的執行階段程式庫，才能夠產生出合法的 C 語言程式執行檔。

而 CRT 並非只有單一一種版本存在。事實上，除了可以依「除錯」與「釋出」用途分成兩個版本之外，兩者又可分別衍生分出「靜態連結」與「動態連結」兩種形式：

***靜態連結****：*

* *LIBCMTD.lib（除錯版本）*
* *LIBCMT.lib*

***動態連結****：*

* *MSVCRTD.lib（除錯版本）*
* *MSVCRT.lib*

雖然這四個 CRT 版本的用途與使用方式各不相同，但卻有個共通的特點，就是**它們都是滿足執行緒安全需求，可在多執行緒程式碼中安全使用的程式庫版本**。事實上，在過去 MSVC 6 的版本中，本來還有另外兩個 LIBCD.lib（除錯版本）與 LIBC.lib 程式庫，是專門給單執行緒程式使用的 CRT 版本，但是這兩個選項自 MSVC 2005 開始就從設定選項中被刪除掉了，所以現在大多數程式設計者使用的都是多執行緒的 CRT 版本。

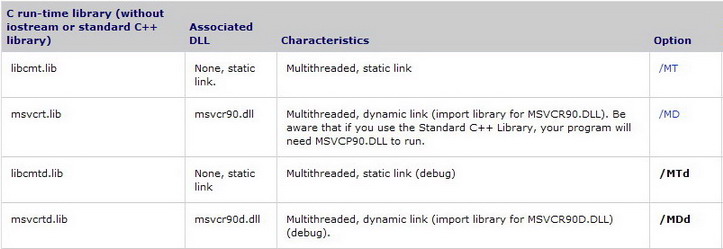
在程式庫連結 (library linking) 的行為中，靜態連結和動態連結的分別，在於使用靜態連結時，會直接將程式庫的函式定義嵌入執行檔之中，而使用動態連結時，程式庫的函式定義則存在於另外的獨立檔案，通常是 DLL 格式的檔案中，然後與程式執行檔一同發佈給使用者。因此在檔案的尺寸上，使用動態連結的執行檔檔案，通常會比使用靜態連結的執行檔檔案來得更小一些。

使用動態連結 CRT 版本的好處，是能夠將經常使用到的標準程式庫們獨立出來，放在 Windows 的系統資料夾中，以減少我們建置出來的執行檔檔案尺寸。但反過來說，**使用動態連結 CRT 版本的缺點也在於這些與執行檔相依為命的 DLL 檔案上**。舉例來說，如果程式以 MSVC 2005 建置出 Debug 組態的執行檔，則此執行檔需要有 msvcr80d.dll 存在才能順利執行；如果是 Release 組態，則相依於 msvcr80.dll。但是如果你把相同的程式碼拿到 MSVC 2008 上建置，產生出來的執行檔則相依於 msvcr90d.dll 與 msvcr90.dll 兩個不同的 DLL 檔案。**不同版本的 MSVC，都會有各自不同的相依 DLL 檔案。**

在 MSVC 的程式專案中，如何指定程式碼要使用靜態連結或者動態連結的 CRT 版本？其實很容易，只要在專案屬性的「C/C++」頁面中，選擇「程式碼產生」(Code Generation) 子頁面，其中有個「執行階段程式庫」(Runtime Library) 的項目，也就是專案中用來設定 CRT 連結版本的地方。其中總共有四個選項，正好對應於上述靜態連結與動態連結的四個不同程式庫版本。

* **多執行緒偵錯 (/MTd)**：對應 LIBCMTD.lib
* **多執行緒 (/MT)**：對應 LIBCMT.lib
* **多執行緒偵錯 DLL (/MDd)**：對應 MSVCRTD.lib
* **多執行緒 DLL (/MD)**：對應 MSVCRT.lib

如果你沒有做任何設定就開始建置程式的話，MSVC 的預設選項則會使用動態連結的版本。



C Runtime Library

請注意，以上只是單純 C 語言的程式庫而沒有包含 C++ 語言在內。如果你的程式系統中，有包含 C++ 語言的程式碼的話，那又是另外一回事了。但是在專案屬性的頁面中，為什麼找不到相關的設定選項呢？**因為 MSVC 悄悄地幫程式設計者代勞處理掉了。**只要在程式碼中使用 #include 語法納入任何一個 C++ 的標頭檔，例如 iostream 或 fstream，MSVC 就會在連結器的運作階段中，自動幫我們連結 C++ 的執行階段程式庫。而 C++ 的執行階段程式庫，同樣可分為四個版本：

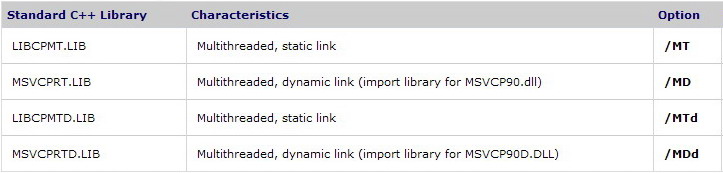
***靜態連結****：*

* *LIBCPMTD.lib（除錯版本）*
* *LIBCPMT.lib*

***動態連結****：*

* *MSVCPRTD.lib（除錯版本）：執行檔相依於 MSVCP90D.dll*
* *MSVCPRT.lib：執行檔相依於 MSVCP90.dll*

至於程式執行檔使用的是靜態連結或者動態連結的版本，就仰賴於 C 語言的版本設定選項了。舉個例子來說，如果你撰寫了一個 Debug 組態的 C++ 程式，並且保留專案原先預設的建置選項（動態連結），那麼最終建置出來的程式執行檔將會相依於 MSVCR90D.dll 以及 MSVCP90D.dll 兩個 DLL 檔案。如果將相同的程式以 Release 組態建置完成，則會相依於 MSVCR90.dll 以及 MSVCP90.dll 二者。



Standard C++ Library

剛學習程式設計的入門者，經常會在滿心歡喜地完成一件程式作品並且傳給其他人使用時，卻發現不能在別人的電腦上啟動程式，其實就是陷入了使用者電腦缺少 DLL 檔案而無法執行程式的窘境。有三種方法可以解決這個令人困擾的問題：

1. 使用者的電腦，必須先安裝「Visual C++ 可轉發套件」（[MSVC 2008](http://www.microsoft.com/downloads/details.aspx?FamilyID=A5C84275-3B97-4AB7-A40D-3802B2AF5FC2&displaylang=zh-tw) 或 [MSVC 2005](http://www.microsoft.com/downloads/details.aspx?FamilyID=200B2FD9-AE1A-4A14-984D-389C36F85647&displaylang=zh-tw) ）。
2. 將所需的 DLL 檔案，例如 MSVCR90D.dll 與 MSVCP90D.dll，直接附在程式的下載包當中。
3. 以靜態連結方式建置程式執行檔。

當你無法確定自己的程式或別人的程式，是否相依於某些特定的 DLL 檔案時，有一個非常好用的免費工具程式 [Dependency Walker](http://www.dependencywalker.com/)，可以開啟 EXE 格式的執行檔或者 DLL 格式的動態程式庫，然後詳細地條列出它們所相依的 DLL 檔案。

瞭解了幾種不同的 CRT 版本選項之後，回到最前面的錯誤訊息問題，相信各位現在應該能夠很清楚地理解，原來會發生這些奇怪的錯誤狀況，**是因為程式同時連結了 LIBCMTD.lib 與 MSVCRTD.lib 所以造成函式定義版本衝突**。也就是說，程式連結器已經在其中一個 CRT 的版本中找到所需的函式定義，但此時卻又跳出另外一位 CRT，也給了一份相同函式的實作版本，所以連結器無法判斷應該忽略誰並且選擇誰。

而這個狀況的發生原因，就是你的程式與程式所連結的外部程式庫，使用了不同的 CRT 版本之故。例如，當你的程式使用了 Lua，自然必須連結至 Lua 的程式庫 lua5.1.lib，但如果 lua5.1.lib 是以靜態連結版本的 CRT 建置而成，而你的程式卻是以預設選項，動態連結 CRT 來建置程式執行檔的話，如此一來就會產生上述這些錯誤訊息了。至此，問題的答案已昭然若揭，解決方法有二種：**其一是將 Lua 重新以動態連結 CRT 的方式建置出一個新的程式庫，其二則是將自己的程式專案改成以靜態連結 CRT 方式建置。**

換個角度想，當你身為一位程式庫的設計開發者，想要將自己寫的東西分享給其他人，但又不想要完全開放自己撰寫的程式源碼時，至少可以同時提供以下四種版本的程式庫，以妥善滿足使用者的各種不同需求：

* Debug：動態連結除錯版本
* Release：動態連結版本
* Debug\_Static：靜態連結除錯版本
* Release\_Static：靜態連結版本

然而，有時候世界並不會運作得如此理想。在某些特殊的狀況下，當我們使用他人所寫的第三方程式庫時，有時可能只拿得到其中某個特定的版本，例如 Release\_Static 版本時，就很有可能會遇到程式庫衝突的錯誤情形。此時就需要視專案的實際需求而定，可以在專案屬性中指定「忽略特定程式庫」(Ignore Specific Library) 這個選項，讓程式碼連結器忽略某些程式庫，以此化解動靜程式庫或新舊程式庫之間的恩怨衝突。

**小測驗**：你所撰寫的程式，必須連結某個以靜態多執行緒 (/MT) CRT 建置而成的程式庫。如果你的程式在 Debug 組態下以多執行緒偵錯 (/MTd) 選項建置，是否會產生衝突？如果你的程式在 Release 組態下以多執行緒 (/MT) 選項建置，是否會產生衝突？是的話，應該如何解決？

**延伸閱讀：**

* [[MSDN] Visual Studio 2008: C Run-Time Libraries](http://msdn.microsoft.com/en-us/library/abx4dbyh.aspx)
* [Dynamically linking with MSVCRT.DLL using Visual C++ 2005](http://kobyk.wordpress.com/2007/07/20/dynamically-linking-with-msvcrtdll-using-visual-c-2005/)

<http://blog.monkeypotion.net/gameprog/beginner/love-and-hate-between-msvc-and-crt>