《深入淺出 MFC》2/e
電子書開放自由下載聲明

致親愛的大陸讀者

我是侯捷（侯俊傑）。自從華中理工大學於 1998/04 出版了我的《深入淺出 MFC》1/e 簡體版（易名《深入淺出 Windows MFC 程序設計》）之後，陸陸續續我收到了許多許多的大陸讀者來函。其中對我的讚美、感謝、關懷、殷殷垂詢，讓我非常感動。

《深入淺出 MFC》2/e 早已於 1998/05 於臺灣出版。之所以遲遲沒有授權給大陸進行簡體翻譯，原因我曾於回覆讀者的時候說過很多遍。我在此再說一次。

1998 年中，本書之發行公司松崗（UNALIS）即希望我授權簡體版，然因當時我已在構思 3/e，預判 3/e 繁體版出版時，2/e 簡體版恐怕還未能完成。老是讓大陸讀者慢一步看到我的書，令我至感難過，所以便請松崗公司不要進行 2/e 簡體版之授權，直接等 3/e 出版後再動作。沒想到一拖經年，我的 3/e 寫作計劃並沒有如期完成，致使大陸讀者反而沒有《深入淺出 MFC》2/e 簡體版可看。

《深入淺出 MFC》3/e 沒有如期完成的原因是，MFC 本體架構並沒有什麼大改變。《深入淺出 MFC》2/e 書中所論之工具及程式碼雖採用 VC5+MFC42，仍適用於目前的 VC6+MFC421（唯，工具之畫面或功能可能有些微變化）。

由於《深入淺出 MFC》2/e 並無簡體版，因此我時時收到大陸讀者來信詢問購買繁體版之管道。一來我不知道是否臺灣出版公司有提供海外郵購或電購，二來即使有，想必帶給大家很大的麻煩，三來兩岸消費水平之差異帶給大陸讀者的負擔，亦令我深感不安。

因此，此書雖已出版兩年，鑑於仍具閱讀與技術上的價值，鑑於繁簡轉譯製作上的費時費工，鑑於我對同胞的感情，我決定開放此書內容，供各位免費閱讀。我已爲《深入淺出 MFC》2/e 製作了 PDF 格式之電子檔，放在 http://www.jjhou.com 供自由下載。北京 http://expert.csdn.net/jjhou 有侯捷網站的一個 GBK mirror，各位也可試著自該處下載。

我所做的這份電子書是繁體版，我沒有精力與時間將它轉爲簡體。這已是我能爲各位盡力的極限。如果（萬一）您看不到檔案內容，可能與字形的安裝有關 — 雖然我已嘗試內嵌字形。anyway，閱讀方面的問題我亦沒有精力與時間爲您解決。請各位自行開闢討論區，彼此交換閱讀此電子書的 solution。請熱心的讀者告訴我您閱讀成功與否，以及網上討論區（如有的話）在哪裡。

曾有讀者告訴我，《深入淺出 MFC》1/e 簡體版在大陸被掃描上網。亦有讀者告訴我，大陸某些書籍明顯對本書侵權（詳細情況我不清楚）。這種不尊重作者的行爲，我雖感遺憾，並沒有太大的震驚或難過。一個社會的進化，終究是一步一步衍化而來。臺灣也曾經走過相同的階段。但盼所有華人，尤其是我們從事智慧財產行爲者，都能夠儘快走過灰暗的一面。

在現代科技的協助下，文件影印、檔案複製如此方便，智財權之尊重有如「君子不欺暗室」。沒有人知道我們私下的行爲，只有我們自己心知肚明。《深入淺出 MFC》2/e 雖免費供大家閱讀，但此種作法實非長久之計。爲計久長，我們應該尊重作家、尊重智財，以良好（至少不差）的環境培養有實力的優秀技術作家，如此才有源源不斷的好書可看。

我的近況，我的作品，我的計劃，各位可從前述兩個網址獲得。歡迎各位寫信給我（jjhou@ccca.nctu.edu.tw）。雖然不一定能夠每封來函都回覆，但是我樂於知道讀者的任何點點滴滴。

關於《深入淺出 MFC》2/e 電子書

《深入淺出 MFC》2/e 電子書共有五個檔案：

檔名	內容	大小 bytes
dissecting MFC 2/e part1.pdf	chap1~chap3	3,384,209
dissecting MFC 2/e part2.pdf	chap4	2,448,990
dissecting MFC 2/e part3.pdf	chap5~chap7	2,158,594
dissecting MFC 2/e part4.pdf	chap8~chap16	5,171,266
dissecting MFC 2/e part5.pdf	appendix A,B,C,D	1,527,111

每個檔案都可個別閱讀。每個檔案都有書籤（亦即目錄連結）。每個檔案都不需密碼即可開啓、選擇文字、列印。

請告訴我您的資料

每一位下載此份電子書的朋友，我希望您寫一封 email 給我（jjhou@ccca.nctu.edu.tw），告訴我您的以下資料，俾讓我對我的讀者有一些基本瞭解，謝謝。

姓名：
現職：
畢業學校科系：
年齡：
性別：
居住省份（如是臺灣讀者，請寫縣市）：
對侯捷的建議：

-- the end

附錄

從搖籃到墳墓 Windows 的完全學習

侯捷 / 1996.08.12 整理

侯俊傑先生邀請我爲他嘔心瀝血的新作 **深入淺出 MFC** 寫點東西。我未寫文章久矣，但是你知道，要拒絕一個和你住在同一個大腦同一個軀殼的人日日夜夜旦旦夕夕的請求，是很困難的 ☺。不，簡直是不可能。於是，我只好重作馮婦！

事實上也不全然是因爲躲不過日日夜夜的轟炸，一部份原因是，當初我還在雜誌上主持無責任書評時，就有讀者來信希望書評偶而變換口味，其中一個建議就是談談如何養成 Windows 程式設計的全面性技術。說到全面性，那又是一個 impossible mission！真的，Windows 程式技術的領域實在是太廣了，我們從來不會說遊戲軟體設計、多媒體程式設計、通訊軟體設計... 是屬於 DOS 程式技術的範疇，但，它們通常都被理所當然地歸類屬於 Windows 程式設計領域。爲什麼？因爲幾乎所有的題目都拜倒在 Windows 作業系統的大傘之下，幾乎每一種技術都被涵蓋在千百計（並且以驚人速度繼續增加中）的 Windows API 之中。

我的才智實不足以涵蓋這麼大面積的學問，更遑論從中精挑細選經典之作介紹給你。那麼，本文題目大剌剌的「完全學習」又怎麼說？呃，我指的是 Windows 作業系統的核心觀念以及程式設計的本質學能這一路，至於遊戲、多媒體、通訊、Web Server、資料庫、統統被我歸類爲「應用」領域。而 Visual Basic、Delphi、Java 雖也都可以開發 Windows 程式，卻又被我屏棄在 C/C++ 的主流之外。

以下謹就我的視野，分門別類地把我心目中認爲必備的相關好書介紹出來。你很容易就可以從我所列出的書名中看出我的淺薄：在作業系統方面，我只涉獵 Windows 3.1 和 Windows 95（Windows NT 4.0 是我的下一波焦點），在 Application Framework 方面，我只涉獵 MFC（OWL 和 Java 是我的下一個獵物）。

Windows 作業系統

◆ **Windows Internals** / Matt Pietrek / Addison Wesley

最能夠反應作業系統奧秘的，就是作業系統內部資料結構以及 API 的內部動作了。本書藉著對這兩部份所做的逆向工程，剖析 Windows 的核心。

一個設計良好的應用程式介面（API）應該是一個不必讓程式員擔心的黑盒子。本書的主要立意並不在爲了對 API 運作原理的討論而獲得更多程式寫作方面的利益（雖然那其實是個必然的額外收穫），而是藉由 API 虛擬碼，揭露出 Windows 作業系統的運作原理。時光漸漸過去，程式員漸漸成長，我們開始對 How 感到不足而想知道 Why 了，這就是本書要給我們的東西。

本書不談 Windows 官方手冊上已有的資訊，它談「新資訊」。如何才能獲得手冊上沒有記載的資訊？呵，原始碼說明一切。看原始碼當然是不錯，問題是 Windows 的原始碼刻正鎖在美國 WA, Redmond（微軟公司總部所在地）的保險庫裡，搞不好就在比爾.蓋茲的桌下。我們唯一能夠取得的 Windows 原始碼大概只是 SDK 磁片上的 defwnd.c 和 defdlg.c（這是 *DefWindowProc* 和 *DefDlgProc* 的原始碼），以及 DDK 磁片中的一大堆驅動程式原始碼。那麼作者如何獲得比你我更多的秘密呢？

Matt Pietrek 是軟體反組譯逆向工程的個中翹楚。本書藉由一個他自己開發的反組譯工具，把獲得的結果再以 C 虛擬碼表現出來。我們在書中看到許許多多的 Windows API 虛擬碼都是這麼來的。Pietrek 還有一個很有名的產品叫做 BoundsChecker，和 SOFT-ICE/W（功能強大的 Windows Debugger，以企鵝爲形象）搭配銷售。

本書主要探討 Windows 3.1 386 加強模式，必要時也會提及標準模式以及 Windows

3.0。書中並沒有涵蓋虛擬驅動程式、虛擬機器、網路 API、多媒體、DDE/OLE、dialog/control 等主題，而是集中在 Windows 啓動程序、記憶體管理系統、視窗管理系統、訊息管理系統、排程管理系統、繪圖系統身上。本書對讀者有三大要求：

- 對 Intel CPU 的保護模式定址方式、segmentation、selector 已有基本認識。
- 擁有 Windows SDK 手冊。
- 對作業系統有基礎觀念，例如什麼是多工，什麼是虛擬記憶體...等等。

作者常借用物件導向的觀念解釋 Windows，如果你懂 C++ 語言，知道類別與物件，知道成員函式和成員變數的意義與其精神，對他的比喻當能心領神會。

對系統感興趣的人，本書一定讓你如魚得水。你唯一可能的抱怨就是：一大堆 API 函式的虛擬碼令人心煩氣燥。文字瀚海圖片沙漠的情形也一再考驗讀者的定力與耐力。然而小瑕不掩大瑜。我向來認爲釀了一瓶好酒的人不必聲嘶力竭地廣告它，這本書就是一瓶好酒。作者 Pietrek 自 1993/10 起已登上 ***Microsoft Systems Journal*** 的 Windows Q&A 主持人寶座，沒兩把刷子的人上這位子可是如坐針氈。現在他又主持同一本刊物的另一個專欄：Under The Hood。***Dr. Dobb's Journal*** 的 Undocumented Corner 專欄也時有 Pietrek 的蹤影。

◆ **Undocumented Windows** / Andrew Schulman, David Maxey, Matt Pietrek / Addison Wesley

朋友們在書店裡選書的方式如何？看不看序？看不看前言？別抓起書像數鈔票般一頁頁流覽，漫無目的的跳躍。從序中可以看出作者的創作心路歷程，作者的抱負理想，還可以看出作者的文筆斤兩。書序，好看得很呢。

大抵你可以從外文書的 Preface 或 Acknowledge 或 Introduction 或 Foreword 看到些類似「序」這樣的輕鬆小品。上一本書 ***Windows Internals*** 的作者在其 Introduction 部份，提到他的感謝，其中對於該書編輯有這麼一段感性談話 ：

首先我要謝謝的，當然是我的編輯。沒有他，這本書幾乎不可能完成。當我們開始爲這本書築夢時，它看起來是那麼可怖，令人畏縮。只因爲我知道他可以助我一臂之力我才有勇氣進行下去。幾乎我所寫的每一筆資料他都有令人驚訝的豐富知識，而且他也注意不讓太多細節扼殺了想像空間。每次當我認爲我已經鉅細靡遺地涵蓋了一整章細部討論，他會以數百個毫不誇張的意見把我推回原點，促使我完成更詳細的討論。我不能夠想像是否還有更好的編輯如他了。

備受 Matt Pietrek 推崇的這位編輯，正是人稱 ¡§Mr. Undocumented¡¨的知名作家 Andrew Schulman，也正是我現在要介紹的 ***Undocumented Windows*** 一書作者。

任何人看到這本書，再看到作者名字，恐怕都有這樣的疑惑：此書和 *Windows Internals* 有何區別又有何關係？Schulman 提出了本書的目標設定：***Windows Internals*** 探討的是 Windows APIs 的內部工作情況，這些 APIs 都是公開的，正式的；***Undocumented Windows*** 探討的則是沒有出現在 Windows 正式文件上的資料。

想學點絕招現買現賣立刻用到自己軟體上的人可能會失望。搞清楚，本書名叫 ***Undocumented Windows*** 而不是 *Undocumented Windows API*。雖然它對 250 個以上的未公開函式都有描述，然而有一半以上的篇幅是在介紹如何以各種工具窺視 Windows 系統，幫助我們了解什麼是 Module、什麼是 Task、什麼是 Instance，這三者是 KERNEL 模組中最重要的成份。當然它也對 GDI 模組和 USER 模組開膛剖腹一番。書裡附了許多軟體工具對 Windows 的偵測報告。這些程式有的是 Phar Lap 公司的正式出品，有的是 Schulman 的私貨。Phar Lap 公司做這類工具軟體真是輕而易舉，別忘記了他們是 DOS Extender 的著名廠商。

本書第一章漫談許多主題，花了相當的篇幅討論 Windows 未公開祕密所引發的美國聯邦交易委員會（U.S. FTC）的關切。第二章至第四章分別介紹一些工具。第五章到第八章是本書第一個重點，介紹 Windows（其實就是 KERNEL、GDI、USER 三大模組）的各個未公開結構、訊息、函式。很多資料不曾在 SDK 中公佈，卻出現在 DDK，想深入了解 Windows 的人不妨有空注意一下 DDK 的文件。這四章佔據 412 頁。

第十章介紹的 ToolHelp 是本書第二個重點。ToolHelp 是 Windows 3.1 新添的一個動態聯結函式庫，可以滿足程式對 Windows 內部資料的查詢。本章對 ToolHelp 的每一個 API 用法、參數、結構、訊息都描述十分詳細。這些 API 允許我們取得 Global Heap、Local Heap、Module Database、Task Database、以及一些系統資訊。

本書附錄B是參考書目。難得的是 Schulman 對每一本書籍都有短評，足見博覽群書，腹笥豐富。我看簡直是在火力展示！

這本書被我看重的地方，在於它提供了許多作業系統的核心資料，至於想撿幾個 Undocumented API 來用的朋友，我要澆你一盆冷水：雖然應用軟體的世界級大廠也都或多或少使用 Undocumented API，但我們的重點不在安全性而在未來性與即時性。你認爲你能夠像上述國際級軟體公司一樣得到微軟公司的第一手資料嗎？這是一件不公平的事，但實力才是後盾。孤臣無力可回天。

著名的 ***Dr. Dobb's Journal***（老字號的天王期刊）在 1992/11 給了本書一個書評，評者是天王巨星 Ray Duncan。Duncan 對於本書作者讚譽有加，事實上他的一本天王巨構 *Extending DOS* 曾收錄有 Schulman 的一章。我把精采的幾句摘譯給各位，春風沐雨一下。

技術文件寫作者（technical writer）是一種被過份苛求而且沒有受到應得尊敬的職業。如果你把焦點再集中到商業雜誌或專業書籍出版社在作業系統、程式介面、發展工具方面的技術文件作者，你就會發現這份職業不但苛求、沒有受到應得的尊敬，而且它還十分地奇特乖辟。再沒有什麼其他領域會像技術文件作者一樣要接受那麼大量的、高水準的讀者的考驗，而且還得和不斷創新的技術拼命，和短的不能再短的產品週期賽跑，和粗劣不堪的產品說明、令人髮指的同意書保証書、模糊的事實、可憐而不可知的市場力量拔河」的技術書籍寫作領域。

其實這是十分公平的！技術文件作者在程式員這一領域的地位如此低落的理由之一是，從業人員的素質良窳不齊。至少 90% 的文章和書籍靠著剪刀和漿糊就做出來了，簡直像是挖泥機一樣，賣力地挖，卻挖出一堆爛泥巴。有的在產品手冊上亂砍幾刀，絲毫沒有加上個人的看法；或是一些半調子學徒爲滿足編輯策劃者的大綱要求，硬拼硬湊，文章中充斥

毫無意義的冗詞贅言。只有 10% 的文章以及書籍，是濁世中的一股清流。這些文章書籍的作者分爲兩個類型：一種是流星型的人物，出了一、兩本有意義的書，如流星畫過天際，閃亮! 然後...沒了，徒留懷念；另一種是一小族群的所謂超級巨星，他們有穩定而質佳的好作品，日復一日，年復一年。

這種超級巨星的特徵是什麼？他們通常都有數年的實際程式經驗，才不至於光說不練；他們對於程式寫作有一股近乎狂熱的感情；他們寫他們所做的，不寫他們所聽的；他們能夠很快認知並接受別人的觀念；他們心胸開闊、博覽群書、通情達理，特別擅長在散亂的斷簡殘篇中理出邏輯結構，並且擅長將此邏輯介紹給別人。他們擁有的最後一個共同特質就是，都有一支生花妙筆。我所指的是 Jeff Prosise、Charles Petzold、Michael Abrash、Jeff Duntemann、Andrew Schulman 等人。

Andrew Schulman 的寫作方式並不是直接給你事實，而是揪著你的衣領讓你自己看事實在哪裡，爲什麼產生這種事實。並且解釋爲什麼這個事實重要，以及如何安全地運用這個事實。第一代 Windows 書籍的代表作品是 Petzold、Yao、Richter、Heller 的書，這一本 Undocumented Windows 將是第二代作品。雖然這本書在表達上還不是盡善盡美，但瑕不掩瑜，它的推出仍是 1992 年此一領域中最重要的一件事情。

痛快之極，痛快之至！

◆ **Windows 95 System Programming Secrets** / Matt Pietrek / IDG Books

注意，前兩本書（***Windows Internals*** 和 ***Undocumented Windows***）都是以 Windows 3.1 爲對探討對象，它們都沒有針對 Windows 95 或 Windows NT 的新版計劃。（微軟曾請 Schulman 寫一本 *Undocumented Windows NT*，他老兄說，等 Windows NT 賣了一千萬套再說。酷！）

本書在作業系統的深度探索方面，可說是對於同一作者的前一本書 ***Windows Internals*** 克紹其裘，但方向不太一樣。本書不再以 Windows API 的內部運作爲出發點，而是以作業系統的大題目爲分野，包括 Modules、Processes、Threads、USER and GDI subsystems、記憶體管理、Win16 模組與其 tasks、Portable Executable 檔案格式與 COFF OBJ 檔案格式。最後兩章頗具實用性質，一是教我們如何自行探勘 Windows 95 的秘密，一是教我們寫出一個 Win32 API Spy 程式（簡直是鬼斧神工）。

Win32 程式設計

◆ **Programming Windows 95** / Charles Petzold / Microsoft Press

文人相輕，中外古今皆然。我們很難看到有一個人，有一本書，受到所有的讀者、同行、媒體、評論一致的推崇。尤其是，一如 Duncan 所言，在這個「必須接受大量高水準的讀者的考驗，和不斷創新的技術拼命，和短的不能再短的產品週期賽跑，和粗劣不堪的產品說明、令人髮指的同意書保証書、模糊的事實、可憐而不可知的市場力量拔河」的技術書籍寫作領域。

但是，有這麼一個例外，那就是 Charles Petzold 及其名著 ***Programming Windows***。***BYTE*** 雜誌稱此書「鉅細靡遺，任何在 Windows 環境下的嚴謹工作者必備」。***Dr. Dobb's Journal*** 的書評則說此書「毫無疑問，是 Windows 程式設計方面舉足輕重的一本書」。我對它

的評價是兼具深度與廣度，不論對初學者或是入門者，此書都值得放在你的書架上，絕不會只是佔據空間而已（不過厚達 1100 頁的它也的確佔了不少空間）。

本書有一個特色，範例很多，而且都很簡潔，旁蕪枝節一概濾除。各位知道，結構化程式設計常會衍生出許多函式，Petzold 的程式儘量不這麼做，這種風格使讀者容易看到程式的重心，不至於跳來跳去。這種方式（短小而直接切入主題，不加太多包裝）的缺點是每一個函式大約沒有什麼重複使用的價值。不過以教育眼光來看，這應該是比較好的作法。一本好書應該教我們釣魚的方法，而不是給我們一條魚。

這本書和所有 Windows 程式設計書籍一樣不能免俗地從 "Hello World !" 開始講起。順流而下的是視窗捲動，基本繪圖，硬體輸入（滑鼠、鍵盤與計時器），子視窗控制元件，各式資源（圖示、游標、圖檔、字串、選單、加速鍵），對話盒，通用型控制元件（Common Controls），屬性表（帶附頁之對話盒），記憶體管理與檔案 I/O，多工與多執行緒，印表機輸出，剪貼簿，動態資料交換（DDE），多文件介面（MDI），動態聯結函式庫（DLL），OLE。

最後一章 OLE，我必須提點看法。依我之見，此章除了讓本書能夠大聲說「我涵蓋了 OLE」之外，一無用處。這其實怪不得執筆人 Paul Yao，在這麼短小的篇幅裡談 OLE，就像在狗籠子裡揮舞丈八蛇矛一樣。

本書文字平易近人，閱讀堪稱順暢。範例程式行雲流水，直接扼要。若要說缺點的話，就是示意圖太少。

此書目前已是第五版，前數版分別針對 Windows 1.0、Windows 2.0、Windows 3.0、Windows 3.1 等版本而作。Petzold 另有爲 OS/2 撰寫的一本 ***OS/2 Presentation Manager Programming***，ZD Press 出版。單從聲勢以及銷售量，你無法想像是同一位作者寫的書。古人母以子貴，今之電腦作家則以寫作對象而揚！嗚乎，有人幸甚，有人哀哉！

◆ **Windows 95 : A Developer's Guide** / Jeffrey Richter, Jonathan Locke / M&T Books

此書誠爲異數。所以這麼說，乃因它是少數不從 Hello、Menu、Dialog、Control... 等初級內容講起的書，可它也不是 DDE 或 OLE 或 DLL 或 Memory 的特殊秀，它講的還是視窗的產生、對話盒技巧、控制元件...，只是深度又多了十幾米。本書的訴求對象是已經具備基本功的人。

本書已是同一系列的第三版，前兩版分別是就 Windows 3.0 和 Windows 3.1 撰寫。新版本在章節的挑選上與前版有相當大的差異，全書主講視窗與視窗類別之深入分析、對話盒的進階技術、訂製型控制元件（custom controls）、Subclassing 與 Superclassing、Hook、檔案拖放技術、鍵盤進階技術和版本控制。原本有的印表機設定、Task and Queues、MDI 程式設計、軟體安裝技術則遭割愛。

有些觀念，看似平凡，其實深入而重要。例如作者在第一章中介紹了許多取得 Windows 內部資訊的 API 函式，並且介紹這些資料的實際意義，從而導出 Windows 作業系統的核心問題。字裡行間曝露許多系統原理，而不只是應用程式的撰寫技巧，這是許多 Windows 程式設計的書難望項背的。

在實作技巧上，Richter 絕對是位高手，每一個程式都考慮極爲週詳。

本書前版曾製作了數幅精巧的示意圖，令人印象深刻，忍不住擊節讚賞。新書未能續此良績，實感遺憾。這是所有書籍的通病：惜圖如金。

◆ **Advanced Windows** / Jeffrey Richter / Microsoft Press

若以出書時機而言，這本書搶在眾多 Windows 95 名書之前出版，拔了個頭籌。封面上烏漆麻黑的法國軍官畫像，令人印象深刻。舊版名曰 ***Advanced Windows NT***，封面上肯定是拿破崙畫像，這新版我就看不出誰是誰來了。

不僅在出書時機拔得頭籌，本書在高階技術（尤其牽扯到作業系統核心）方面也居崇高地位。不少名書也常推薦此書以補不足。

本書基本上以作業系統觀念爲主，輔以範例驗證之。從章名可以發現，全都是作業系統的大題目，包括行程（Process）、執行緒（Thread）、記憶體架構、虛擬記憶體、記憶體映射檔（Memory Mapped File）、堆積（Heap）、執行緒同步控制、Windows 訊息與非同步輸入、動態聯結函式庫、執行緒區域性儲存空間（Thread-Local Storage，TLS）、檔案系統與 I/O、異常現象與處理、Unicode。讀者群設定在具備 32 位元 Windows 程式經驗者。範例程式以 C 寫成。Richter 說他自己發展大計劃時用的是 C++，但他不願意喪失最大的讀者群。

老實說我也很想知道臺灣有多少人真正以 C++ 開發商用軟體。學生不能算，學生是工業體系的種子，卻還不是其中一環。

我曾說 Richter 在實作技巧上是位高手。諸君，試安裝本書所附光碟片你就知道了。我只能用華麗兩字來形容。

◆ **Writting Windows VxD and Device Driver** / Karen Hazzah / R&D Publications

對於想學習 VxD 的人，等待一本「完整的書」而不是「斷簡殘篇」已經很久了。這個主題的書極難求，如果設置金銀銅三面獎牌，大概全世界相關書籍俱有所獲，皆大歡喜。本書穩獲金牌獎給無問題，而金牌和銀牌之間的距離，我看是差得很遠唷。

不少人害怕 VxD，其實它只是新，並不難。VxD 之所以重要，在於 Windows 程式與硬體間的所有溝通都應該透過它；只有透過 VxDs 才能最快最安全地與硬體打交道。VxD 才是你在 Windows 環境中與硬體共舞的最佳選擇。VxD 讓我們做許多 Windows 原不打算讓我們做的事，突破重重嚴密的束縛。你可以乘著 VxD 這部擁有最高特權（Ring0）的黑頭車直闖戒護深嚴的博愛特區（作業系統內部）。有了 VxD，你可以看

到系統上所有的記憶體、攔截或產生任何你希望的中斷、讓硬體消失、或讓根本不存在於你的電腦中的硬體出現。VxD 使你 "go anywhere and do anything"。

本書從最基礎講起，從 VxD 的程式寫法，到 Windows 的虛擬世界，到如何對硬體中斷提供服務，再到效率的提升，DMA 驅動程式，真實模式與標準模式的情況（哦，我忘了說，本書乃針對 Windows 3.1），以及計時器與軟體中斷。所有範例皆以 Assembly 語言完成。

很少書籍在以圖代言這方面令我滿意。本書有許多用心良苦的好圖片，實在太讓我感動了。我真的很喜歡它們。本書已有第二版，可是臺灣未進口（事實上第一版亦無進口），嗚乎，哀哉！

◆ System Programming for Windows 95 / Walter Oney / Microsoft Press

教導 Windows API 程式寫作的書，車載斗量；涉及系統層面者，寥若晨星。

本書介紹「如何」以及「爲什麼」軟體開發者可以整合各式各樣的低階系統呼叫，完成高檔效果。範例程式不使用令人畏懼的 assembly 語言，而是 C/C++ 語言（別懷疑，C/C++ 也可以寫 VxD）。本書打開對微軟作業系統架構的一個全盤視野，並滿足你撰寫系統層面應用程式的需求。它的副標是：C/C++ programmer's guide to VxDs, I/O devices, and operating system extensions.

像前述的 ***Writting Windows VxD and Device Driver*** 那麼好的書，遲遲未見進口臺灣，令人扼腕。這一本 ***System Programming for Windows 95*** 可以稍解我們的遺憾。作者 Oney 曾經在不少期刊雜誌上發表不少深入核心的文章，相當令吾等振奮。他當初一篇發表在 ***Microsoft Systems Journal*** 上的文章：Extend Your Application with Dynamically Loaded VxDs under Windows 95，就已經讓我對這此書充滿信心與期待。想學習 VxD programming 的人，嘿，此書必備。

無責任書評

MFC 討論書籍

關於 MFC 這一主題，在「滄海書訊」版上曾經被討論過的書籍有四本，正是我所列出的這四大天王。看來我心目中的好書頗能吻合市場的反應。

侯捷 / 1997.02 發表於 Run!PC 雜誌

我還記得，無責任書評是在四年前（1993）開春時和大家第一次見面。雖然不是每個月都出貨，但斷斷續續總保持著訊息。在明確宣佈的情況下這個專欄曾經停過兩次，第一次停了三個月，於 1994 年開春復工；第二次停了十五個月，於 1997 年開春的今天，重新與各位說哈囉。

休息整整一個年頭又三個月，寫作上的疲倦固然是因素之一，另外也是因爲這個專欄直接間接引起的讓人意興闌珊的俗人俗務。讀者寫信來說，『總把無責任書評當成休閒散文看。或許您可以考慮寫些休閒小品，定會暢銷』，是呀，我正構思把因這個專欄而獲得的人生經驗寫成一本「現形記」。可是不知道手上「正當」工作什麼時間才能告一段落，也不知道出版社在哪裡。

倦勤過去了，滿腔讀書心得沛然欲發。所以，我又拿起筆「無責任」了。感覺有點陌生，但是回顧讀者們這一年寫來的上百封信，讓我意氣昂揚。這個月我談的是 Visual C++ 與 MFC。此題目我已提過兩次。一來它十分重要，演化的過程也十分快速而明顯，二來這個領域又有一些重量級書籍出現，所以我必須再談一次。

另外，我還是得再強調，侯捷的專長領域有限，離我火力太遠的書我只能遠觀不敢近玩。這個專欄用在拋磚引玉，讓談書成爲一種風氣。***Windows Developer's Journal***（***WDJ***）的 Books in Brief 專欄原先也是主持人 Ron Burk 唱獨角戲，後來（現在）就有了許多讀者的互動。我也希望這樣的事情在這裡發生。

◆ 必也正名乎

常在 BBS 的程式設計相關版面上看到，許多人把 Visual C++ 和 C++ 混淆不清，另則是把 Visual C++ 和 MFC 混爲一談，實在有必要做個釐清。C++ 是語言，Visual C++ 是產品。『我們學校開了一門 Visual C++ 課程』這種說法就有點奇怪，實際意義是『我們學校開了一門 C++ 課程，以 Visual C++ 爲軟體開發環境』。『我會寫 Visual C++ 程式』這種說法也很怪，因爲 Visual C++ 是一種 C/C++ 編譯器，你可以在這套整合開發環境中使用 C 語言或 C++ 語言寫出 DOS 程式或 Windows 程式；如果是 Windows 程式，還可以分爲 Windows API programming 或 MFC programming。所以「我會寫 Visual C++ 程式」表達不出你真正的程度和意思。

Visual C++ 是一套 C/C++ 編譯器產品，內含一套整合開發環境（Integrated Development Environment，IDE），也就是 AppWizard、ClassWizard、編譯器、聯結器、資源編輯器等工具的大集合。你知道，真正的 C++ 程式（而不是披著 C++ 外衣的 C 程式）是以一個個類別（classes）堆砌起來的，爲了節省程式員的負擔，幾乎每一家編譯器廠商都會提供一套現成的類別庫（class libraries），讓程式員站在這個基礎開發應用軟體。MFC 就是這樣一套類別庫。如果以物件導向的嚴格眼光來看，MFC 是比類別庫更高一級的所謂 application framework。PC 上另兩套與 MFC 同等地位的產品是 Borland 的 OWL 和 IBM 的 Open Class Library，前者搭配的開發環境是 Borland C++，後者搭配的是 VisualAge C++。其他的 C++ 編譯器大廠如 Watcom 和 Symantec 和 Metaware，並沒有開發自己的類別庫，他們向微軟取得 MFC 的使用授權，提供 MFC 的原始碼、含入檔、相容的編譯器和聯結器。噢是的，他們要求授權的對象是 MFC，而不是 OWL，這就多少說明了 MFC 和 OWL 的市場佔有率。

產品名稱	廠商	application framework
Visual C++	Microsoft	MFC
Borland C++	Borland	OWL（BC++ 最新版據說也支援 MFC）
VisualAge C++	IBM	Open Class Library
Symantec C++	Symantec	MFC

◆ 滄海書訊

清大 BBS 站台（楓橋驛站，IP 位址爲 140.114.87.5），在「分類討論區」的「電腦與資訊」區之下，有一個「滄海書訊」版，對電腦書籍有興趣的朋友可以去看看。這裡並沒有（還沒有）類似正規書評之類的文章出現，比較多的是讀者們對於坊間書籍的閱後感，以及新鮮讀者的求助函（找某一主題的好書啦、誰要賣書啦、誰要買書啦等等）。

關於 MFC 這一主題，在滄海書訊版上曾經被討論過的書籍有四本，正是我所列出的這四大天王。看來我心目中的好書頗能吻合市場反應。這四本書各有特點，色彩鮮明，統統值得收藏。

◆ 四大天王

一本書能夠有被收藏的價值，可不簡單唷，我不能亂說嘴。諸君，看看我列的理由吧。這四大天王是：

□ ***Inside Visual C++ 4.0***

四大天王之中本書名列老大哥，這排名和天王的「色藝」無關，敬老尊賢的成份多一些。它已是同一本書的第三版，所以才會在書名冠上軟體版本號碼（上一版名爲 ***Inside Visual C++ 1.5***）。書名冠上軟體版本號碼的另一個因素是，本書在教導我們開發程式時，是"tool-oriented"（以工具爲導向），你會看到像「先按下這個鈕，然後填寫這一小段碼，然後在清單中選擇這一項，再回到右邊的視窗上...」這樣的文字說明，所以 Visual C++ 的版本更迭攸關全書內容。

這就引出了本書在程式誘導方面的一個特徵：工具的使用佔了相當吃重的角色。工具的使用難度不高，但非常繁多（從 Visual C++ 新鮮人的眼光看可能是...呃...非常雜亂）。又要學習 MFC，又要配合工具的使用，對初學者而言是雙倍負擔。我曾經在 BBS 上看到一封信，抱怨 ***Inside Visual C++*** 雖是名著，他卻完全看不懂。呵，我完全能夠了解，我不是那種自己懂了之後就忘記痛苦的人。

入選原因：老字號，範例程式內容豐富，220 頁的 OLE 和 110 頁的 Database 是本地唯有的大獨家，別處難找。

☐ ***Programming Windows 95 with MFC***

Ray Duncan（侯捷極爲尊敬的一位老牌作家，近年似乎淡出，沒有什麼新作品）曾經說，這本書是 "the Petzold for MFC programming"，儼然有 Petzold（註）接班人之勢。從其主題的安排，甚至從書籍封面的安排，在在顯示「接班人」的訊息。而它的內容可以證明 Ray Duncan 的推薦並不虛佞。

> 註：Charles Petzold 是 ***Programming Windows 95*** 一書的作者。該書是 SDK 程式設計寶典。這本書近來沒有那麼轟動以及人手一冊了，因爲 MFC 或 OWL 這類 framework 產品不斷精進，Visual Basic、Delphi、C++ Builder 這類快速程式開發工具（Rapid Application Development，RAD）不斷進逼，SDK 程式設計的角色有點像組合語言了。不過我告訴你，學會它，絕對讓你層次不同 -- 不只在程式設計的層次，還在對作業系統的了解層次。

這本書在程式設計的誘導方面，與 ***Inside Visual C++*** 一書有極大的作法差異。本書沒有任何一個程式以 Wizards 完成（我想作者必然曾經借重工具，只是最後再清理一遍），所以你不會看到像 //{{ 和 }}// 這樣的奇怪符號，以及一堆 #ifdef、#undef、#endif。「程式碼是我們自己一行一行寫出來」的這種印象，可能對於消除初學者的焦灼有點幫助。

入選原因：文字簡易，觀念清楚。從章節目錄上你看不到非常特殊的主題，但隱含在各個小節之中有不少珠玉之言。平實穩健。對 MFC 核心觀念如 Document/View、Message Map 的討論雖然淺嚐即止，但表現不俗。

☐ ***MFC Internals***

這是四大天王之中唯一不以教導 MFC「程式設計」爲目的的書。它的目的是挖掘 MFC 的黑箱作業內容，從而讓讀者對 application framework 有透徹的認識。這樣的認識對於 MFC 的應用面其實也是有幫助的，而且不小。

這本書挖掘 MFC 的原始碼至深至多，最後還在附錄A列出 MFC 原始碼的搜尋導引。由於解釋 MFC 的內部運作原理，少不得就有一長串的「誰呼叫誰，誰又呼叫誰」的敘述。這種敘述是安眠藥的最佳藥引，所幸作者大多能夠適時地補上一張流程圖，對於讀者的意識恢復有莫大幫助。

入選原因：獨特而唯一。雖然並非初學者乃至中級程度者所能一窺堂奧，卻是所有資深的 MFC 程式員應該嘗試讀一讀的書籍。

☐ ***Dissecting MFC***

這本書是應用面（各種 MFC classes 之應用）和核心面（隱藏在 MFC 內的各種奇妙機制）的巧妙混合。前一半篇幅爲讀者紮基礎，包括 Win32、C++、MFC 程式的基礎技術環節。後一半篇幅以著名的 Scribble 程式（隨附於 Visual C++ 之中）爲例，從應用面出發，卻以深掘原理的核心技術面收場。看不到豐富絢麗的各種應用技巧，著重在厚植讀者對於 MFC 核心技術的基礎。

入選原因：本書挖掘的 Runtime Class、Dynamic Creation、Message Mapping、Command Routing、Persistence 等主題，解說詳實圖片精采，有世界級水準。並有簡化模擬（使用 console 程式 ），降低入門門檻。SDK 程式員如果想進入 MFC 領域，這本書是最佳選擇。看過 ***Inside Visual C++*** 和 ***Programming Windows 95 with MFC*** 的讀者，這本書會讓你更上層樓，「知其然並知其所以然」。

◆ Inside Visual C++ 4.0

作者：David J. Kruglinski
出版公司：Microsoft Press
出版日期：1996 年初
頁數：29 章，896 頁
售價：US$ 45.00。含光碟一片。

PartI：Windows、Visual C++、and Application Framework Foundamentals
1. Microsoft Windows and Visual C++
2. The MFC Application Framework
PartII：The MFC Library View Class
3. Getting Started wwwith AppWizard - Hello World!
4. Basic Event Handling, Mapping Modes, and a Scrolling View
5. The Graphics Device Interface (GDI), Colors, and Fonts
6. The Modal Dialog and Windows 95 Common Controls
7. The Modeless Dialog and Windows 95 Common Dialogs
8. Using OLE Controls (OCXs)
9. Win32 Memory Management
10. Bitmaps
11. Windows Message Processing and Multithreaded Programming
PartIII：The Document-View Architecture
12. Menus, Keyborad Accelerators, the Rich Edit Control, and Property Sheets
13. Toolbars and Status Bars
14. A Reusable Frame Window Base Class
15. Separating the Document from Its View
16. Reading and Writing Documents - SDI
17. Reading and Writing Documents - MDI
18. Printing and Print Preview
19. Splitter Windows and Multiple Views
20. Context-Senssitive Help
21. Dynamic LInk Libraries (DLLs)
22. MFC Programs Without Document or View Classes
PartIV：OLE
23. The OLE Component Object Model (COM)
24. OLE Automation
25. OLE Uniform Data Transfer - Clipboard Transfer and Drag and Drop
26. OLE Structure Storage
27. OLE Embedded Servers and Containers
PartIV：Database Management
28. Database Management with Microsoft ODBC
29. Database Management with Microsoft Data Access Object (DAO)
Appendix A: A Crash Course in the C++ Language

Appendix B: Message Map Functions in MFC
Appendix C: MFC Library Runtime Class Identification and Dynamic Object Creation

自從 application framework 興起，在 raw API 程式設計之外，Windows 程式員又找到了一條新的途徑。MFC「系出名門，血統純正」，比之其他的 application framework 產品自然是聲勢浩大，MFC 書籍也就因此比其他同等級產品的書籍來得多得多。

群雄並起之勢維持沒有太久，真正的好東西很快就頭角崢嶸了。***Inside Visual C++*** 是最早出線的一本。此書至今已是第三版，前兩版分別針對 MFC 2.0（Visual C++ 1.0）和 MFC 2.5 （Visual C++ 1.5）撰寫。已有評論把此書與 ***Programming Windows*** 並提，稱之爲 MFC/C++ 中的 Petzold 書籍（聽起來猶如錶界中的勞力士，車界中的勞斯萊斯）。Kruglinski 本人爲了卡住這個尊崇的位置，甚至「於數年前的一個冬天，有著風雪的傍晚，冒險進入紐約的 East Village，拜訪 Windows 大師 Charles Petzold，問他關於撰寫 ***Programming Windows*** 的想法...」（語見此書之 Introduction）。

Kruglinski 毫不隱藏他對 MFC 的熱愛，他說這是他等了十年才盼到的軟體開發環境。十年有點跨張，PC 的歷史才多久？但 MFC 與 Visual C++ 整合環境之功能強大卻是不假。這本書劃分爲四大篇。第一篇介紹 application framework 的觀念以及 Visual C++ 整合環境的各個工具元件。第二篇真正進入 MFC 程式設計，不能免俗地從 "Hello World" 開始，焦點放在視窗顯示身上（也就是 *CView* 的運用）。作者嘗試以 C++ 和 MFC 完成一些功能簡單的程式，像是簡易繪圖、圖形捲動、字形輸出、通用對話盒與通用控制元件、OCX 之使用等等。

第三篇才真正進入 MFC 的核心，也就是 Document-View 架構，這也是所謂 application framework 的最大特質。當你學會如何使用 Document 並且把它和 View 連接起來後，你會驚訝資料的檔案動作和印表動作（包括預視功能）是多麼容易加入。這一篇的章節包括漂亮迷人的 UI 元件如工具列、狀態列、分裂視窗、求助系統、屬性對話盒，以及 SDI、MDI、列印、預視、動態聯結函式庫等主題。

第四篇的五章談的全部都是 OLE。不像一般書籍對於 OLE 蜻蜓點水，這一篇是道道地

地的硬扎貨色，範圍包括 COM（Component Object Model）、OLE Automation、Uniform Data Transfer、Structured Storage、Embedded Servers and Containers。

第五篇談的全部是資料庫管理。一章談 ODBC（Open Database Connectivity），另一章談 DAO（Data Access Objects）。

網路上一位讀者抱怨說，本書雖是名著，他卻完全看不懂。呵啊，一切都在意料之內。作者一開始就顧著給我們完全正規的作法，用 AppWizard 產生程式碼，用 ClassWizard 改寫虛擬函式、攔截訊息並撰寫訊息處理常式。剛開始學習 Windows 程式設計的人，甚至已經有 SDK 經驗但沒有物件導向經驗的人，根本昏頭轉向摸不著頭緒。是的，學習 MFC（或其他 Application Framework），先得有許多基礎。包括 C++ 語言基礎、Windows 作業系統基礎、物件導向程式觀念的基礎。

最新消息：本書第五版已有預告，書目上寫的出版日期是 97 年三月。以我對 Microsoft Press 出書進度的瞭解，屆時可能咱們還需再等一等。新書內容針對 Visual C++ 5.0（仍是以 MFC 4.x 爲程式架構核心）但加了不少網路技術，如 Basic TCP/IP、Winsock programming for clients and servers、MFC WinInet、DocObjects and ActiveX controls 等主題。

◆ Programming Windows 95 with MFC

作者：Jeff Prosise
出版公司：Microsoft Press
出版日期：1996 第二季
頁數：14 章，998 頁
售價：US$ 49.95。含光碟一片。

PartI：MFC Basics
1. Hello, MFC
2. Drawing in a Window
3. The Mouse and the Keyboard
4. Menus
5. Controls
6. Dialog Boxes and Property Sheets

7. Timers and Idle Processing
PartII：The Document/View Architecture
8. Documents, Views, and Single Document Interface
9. Multiple Documents and Multiple Views
10. Printing and Print Previewing
11. Toolbars, Status Bars, and Versionable Schemas
PartIII：Advanced Topics
12. Bitmaps, Palettes, and Regions
13. The Common Controls
14. Threads and Thread Synchronization

每一位 MFC 書籍作者，最大的夢想就是其作品被譽爲「C++ 中的 Petzold 書籍」。有人親訪 Petzold，有人則搬出老天王來說幾句話。老天王 Ray Duncan 這麼說： "Jeff Prosise has written the definitive introduction to Windows software development in the era of 32 bits and application frameworks. This book is the Petzold for MFC programming"。這段話被當作本書的廣告主打詞。有趣的是，儘管萬方爭取，Petzold 本人倒是從來沒有說過什麼話。也許他想說的是『我自己來寫本 MFC 經典』，呵呵。

本書有沒有接班人的能耐呢？有！和 ***Inside Visual C++*** 比較，本書在低階部份照顧得多些，程式細節則非常完備。別誤會，我的意思並非說它是那種「把五句話可以說清楚的一段文字，以十句話來表達」的書籍（註），我是說它把各個技術主題挖得很深入，旁徵博引的功夫很好，資料準備得很齊全。

註：另一位大師 Matt Pietrek 的書就有點這種拖拉味道，不過書仍然很棒就是了。他有兩本經典作品：***Windows Internals*** 和 ***Windows 95 System Programming SECRETS***。

本書在導入部份比 ***Inside Visual C++*** 好。作者先安排一個極小的 SDK 程式，解釋 message-based、event-driven 的程式模型，然後再安排一個極小的 MFC 程式，解釋 framework 的運作，告訴你應該改寫什麼函式，標準作法如何，應該攔截什麼訊息，標準作法又如何。雖然程式運行的脈絡仍然不是十分清晰可尋，不過總算是表現不錯了。

從章節目錄可以看出，這本書選題中規中矩，該有的沒遺漏，大獨家倒也沒有。注意，

所有的範例程式都沒有說明其製作過程，只是列出原始碼並以文字解釋原始碼的意義。你知道，視覺性軟體開發過程中，工具的參與絕對少不了，而且角色日形吃重，因此，本書讀者要自己想辦法補足「工具的使用」這一節。***Inside Visual C++*** 就不一樣了，幾乎對於每一個程式，都詳列出工具參與的足跡。

究竟工具的使用要在什麼時候進場，才能夠帶來利益而不是沉重的盲與茫呢？我以爲作者最好先給一個純手工製造的 MFC 程式，用來解釋 MFC 程式的來龍去脈以及程式和 application framework 的互動關係，然後再引進工具的使用說明，然後就安排讓工具強力介入程式設計過程。畢竟，正規的、大型的 MFC 程式開發過程中少不了工具的運用。***Inside Visual C++*** 的作者操之過急，工具一下子全面介入，***Programming Win95 with MFC*** 的作者又避若蛇蠍，完全捨棄工具。

過猶不及！這方面 ***Dissecting MFC*** 的作者就處理得不錯。

這本書沒有談到當紅的 OLE 和 ActiveX。關於這一點，***Windows Developer's Journal***（***WDJ***）的 Books in Brief 專欄（主持人是 Ron Burk）在 1996.10 有這麼一段讀者與評論者的對話：

讀者來函：『我還忘了說，Prosise 的這本書完全沒有討論 OLE。雖然我了解這是這本一般性、介紹性書籍的抉擇，我還是認爲這和書名之間存在著一種矛盾。畢竟，Win95 程式設計一定會牽扯到某些 COM 和 OLE。實際情況的確如此，現在再也不可能和 shell 交談而沒有使用 COM 物件了，Uniform Data Transfer 似乎也已經成爲實作拖放（drag and drop）和剪貼（copy and paste）功能的最佳途徑了。所以，忽略這個主題實在令人有點驚訝。』

Burk 回答：『我同意你的大部份觀點。程式設計書籍的名稱沒有恰如其份地反應出書籍內容是出了名的，所以我無法不同意你的觀點。然而，我絕對不贊成這本書涵蓋 OLE。OLE 複雜到足夠成爲一本書。要在這一本已經過胖的書籍中加入一章 OLE，可想而知必然內容膚淺，就像其他爲了滿足市場因素而強加一章 OLE 的其他書籍一樣。那樣的

書籍在我的架上有一大堆。與其加一章膚淺的 OLE，我寧願作者多花時間讓其他章節更有深度些。...我比任何人忍耐了更多的爛書，所以我寧願看到涵蓋主題不多但是內容十分紮實的書籍。』

「與其加一章膚淺的 OLE，我寧願作者多花時間讓其他章節更有深度些」，唔，就連我當初閱讀 Carles Petzold 的世界名著 ***Programming Windows 95*** 的最後一章（OLE）時，也有相同的感受。如果 Prosise 來到臺灣，發現他的大作被改了名稱，加上了在他抉擇之外的 ActiveX，讓我們猜猜他臉上的表情。這本書的中譯本在原著之外增加了第零章 ActiveX，我願意相信是出版者的用心，而不是如 Ron Burk 所說「爲了滿足市場因素而強加一章膚淺的 OLE」。我不願評論中文版新加的一章內容如何，畢竟用心良苦。但是把書名也給改了，掛上斗大的 ActiveX，這種行徑曾經在 BBS 上引起讀者的強烈抗議，他們說這是「掛羊頭賣狗肉」。

Ron Burk 說「程式設計書籍的名稱沒有恰當反應出其內容，是出了名的」，嗯...嗯...我也曾感受深刻，但沒有這次這麼深刻。

◆ MFC Internals

作者：George Shepherd, Scot Wingo
出版公司：Addison Wesley
出版日期：1996 第二季
頁數：15 章，709 頁
售價：US$ 39.95

1. A Conceptual Overview of MFC
2. Basic Windows Support
3. Message Handling in MFC
4. The MFC Utility Classes
5. All Roads lead to CObject
6. MFC Dialog and Control Classes
7. MFC's Document/View Architecture
8. Advanced Document/Vieww Internals
9. MFC's Enhanced User-Interface Classes
10. MFC DLLs and Threads
11. How MFC Implements COM

12. Uniform Data Transfer and MFC
13. OLE Documents the MFC Way
14. MFC and Automation
15. OLE Controls
Appendix A: A Field Guide to the MFC Source Code
Appendix B: The MFC Internals Floppy

Addison Wesley 出版公司似乎最喜歡出一些未公開秘密、內部運作奧秘之類的書籍。這是繼 ***Windows Internetls*** 和 ***DOS Internals*** 之後又一本黑箱書。

本書挖盡 MFC 原始碼，解釋 MFC 的黑箱作業原理與動作流程。這書的訴求對象已經不是想以 MFC 撰寫一般程式的人，而是 MFC 玩了相當一段時間，欲有所突破的人。

應用技術欲有所突破，核心技術就得加強。很多人對於「了解 MFC 的黑箱作業」心存疑惑，總認爲這違反物件導向的封裝繼承性以及 application framework 的精神。啊，不是這麼說！你買一本汽車百科，了解汽車的構造原理，並不妨礙你「希望在沒有任何機械背景的情況下，學會駕馭這一堆鐵」的心願。然而，當你看過汽車的解剖圖，知道變速箱、離合器、萬向傳動軸、引擎燃料系統、動力傳達裝置、懸吊裝置、煞車裝置...，是否開起車來比較實實在在？了解構造原理之後，要來個甩尾大迴旋，比較知道該怎麼做吧，基本操作也比較不會出錯（很多人煞車時順帶把離合器踏板給踩下去，怕熄火。這習慣養成之後，高速公路上就會要你的命）。

依我之見，了解 MFC 原始碼是有必要的，尤其在導入部份 -- 這是影響一個人能否學成 MFC 的關鍵。一本好的 MFC 書籍應該讓程式員完全了解每一個奇怪巨集（像是 DECLARE_MESSAGE_MAP 、 BEGIN_MESSAGE_MAP 、 END_MESSAGE_MAP 、 DECLARE_SERIAL、IMPLEMENT_SERIAL 等等等）的背後所代表的機制，以及每一個必須改寫的虛擬函式（例如 *CWinApp::InitInstance*、*CDocument::Serialize*、*CView::OnDraw* 等等等）背後所代表的意義與動作。但是當程式的主軸精神完全掌握之後，旁支應用（例如對話盒、控制元件、列印、預視）就不需要再那麼深入探究原始碼了。當然，這是指一般性質的 MFC 書籍而言，***MFC Internals*** 本來就是要把 MFC 整個翻兩翻的，所以它照挖不誤。

◆ Dissecting MFC　2nd Edition

作者：侯俊傑
出版公司：松崗
出版日期：1996/10
頁數：13 章，778 頁
售價：NT$ 860.00。含光碟一片。
第一篇 勿在浮砂築高臺 - 技術前提
1. Win32 程式基本觀念
2. C++ 的重要性質
3. MFC 六大關鍵技術之模擬
第二篇　Visual C++ v4.0 開發工具
4. Visual C++ - 整合性軟體開發環境
第三篇　淺出 MFC 程式設計
5. 總觀 Application Framework
6. MFC 程式設計導論 - MFC 程式的生與死
7. 一個簡單而完整的 MFC 骨幹程式
第四篇　深入 MFC 程式設計
8. Document-View 深入探討
9. 訊息映射與繞行
10. 對話盒與 DDX/DDV
11. View 功能之加強與重繪效率之提昇
12. 印表與預視
13. 多重文件與多重顯示
14. MFC 多緒程式設計
15. 定製一個 AppWizard
16. 站上眾人的肩膀 - 使用 Components 和 ActiveX Controls
Appendix A　從搖籃到墳墓 - Windows 的完全學習
　　無責任書評/侯捷：MFC 四大天王
Appendix B　Scribble Step5 程式原始碼列表
Appendix C　Visual C++ 所附範例程式一覽
Appendix D　以 MFC 重建 Debug Window（DBWIN）

我談這本書，可能會被譏以「分身替本尊說話」，但爲了舉薦好書，以及秉持外舉不避仇、內舉不避親的原則，我不想閃躲。

這本書目前只有中文版。已經有國內出版社積極表達爭取出版英文本的意願。大陸方面，則有多家出版社亟願將此書譯爲簡體版，甚至直接 email 與作者聯絡。這本就是前陣子在 BBS 上引起眾多討論的 **深入淺出 MFC**，***Dissecting MFC***。

依我看，本書橫亙在 ***Inside Visual C++*** 和 ***MFC Internals*** 兩書之間，有 ***InsideVisual C++*** 的實用面，而在核心技術更擅勝場。有 ***MFC Internals*** 的深入面，而無其過於晦澀。

所謂核心技術，本書指的是：

1. 應用程式和 MFC framework 的因果關係。這一部份是你學習 MFC 程式設計的成敗關鍵。因爲太多人上不了第一個台階。本書把隱藏的 *WinMain* 函式、視窗類別註冊、視窗誕生、訊息迴路等動作統統挖掘出來，讓屬於 framework 的那半邊曝光，和你的應用程式碼這半邊拼兜出一張完整的邏輯脈絡圖。才不會堆積木老是少一塊。

2. 訊息映射（Message Mapping）和命令繞行（Command Routing）。「物件導向」從來沒有考慮過 Windows 訊息（那當然）。MFC 程式有四大類別（application、frame、document、view），程式員最容易陷入的苦惱是不知道在哪一個類別中攔截並處理命令訊息。那是因爲沒有能夠看清楚訊息在類別中的流動路線。流動路線的整個地圖隱隱在巍巍山巔：在由 DECLARE_MESSAGE_MAP、BEGIN_MESSAGE_MAP、END_MESSAGE_MAP 巨集以及其他各式各樣的 ON_COMMAND、ON_WM_PAINT 等巨集架構起來的巨大網絡中。當你的程式一開始執行，整個 MFC 的絕大部份類別，都已經貢獻出一些資料，組成這張巨幅網絡（噢，是的，當然也耗用了你的記憶體）。

3. Document/View/Template 之間的關係。一個程式如果支援兩份以上的 Documents，應該如何管理？對應的使用者介面應該如何設定？Document Template 究竟是何用途？這是這個主題要探討的題目。

4. Runtime Type Information（RTTI）和 Dynamic Creation。把一份 document 寫入檔案之中，連同類別名稱和成員變數的值，沒有問題。是的，一點問題也沒有，但是讀出

來就有問題了，因爲你不可能讀一個類別名稱到一個字串中然後對這個字串做 new 動作。「從檔案讀出類別名稱然後產生其物件」，就是 "dynamic creation" 的具體表現。C++ 不支援這項能力，但 MFC 非要不可，因爲有太多時候需要 dynamic creation。其實你只要使用笨方法如下，就可以解決 dynamic creation 的問題：

```
// pseudo code
read ClassName to str
if (str == "Class1")
   new Class1;
else if (str == "Class2")
   new Class2;
else if (str == "Class3")
   new Class3;
else if (str == "Class4")
   new Class4;
¡ K
```

MFC 小組比我們聰明嗎？不會。但是他們比我們懂得包裝。他們在 MFC framework 中架構起一個由所有類別相關資訊（包括類別名稱及建構函式）組成的類別型錄網絡（一個串列），然後把類別名稱的比對動作埋藏在 Serialize 虛擬函式中。類別型錄網絡中的每一個成員就是 *CRuntimeClass* 物件，網絡的組成則是藉由類別宣告時的 DECLARE_DYNCREATE 和 IMPLEMENT_DYNCREATE 巨集完成。RUNTIME_CLASS 巨集就是取出「類別型錄網絡」中的一個元素（代表一個類別）。所以，當你的程式一開始執行，整個 MFC 的絕大部份類別，都已經放在這個「類別型錄網絡」之中（噢，是的，當然也耗用了你的記憶體）。有了這網絡，RTTI（執行時期型別辨識）和 Dynamic Creation 都不是問題。

5. Persistence。文件內容要用什麼型式寫到檔案去，才能夠再從檔案讀出來恢復爲一個個的物件？這就是 persistence（MFC 稱之爲 serialization）要探討的題目。本書把 Serialize 虛擬函式的內部行爲全部挖掘出來，並且實際觀察一個文件檔的 hex 傾印內容。

這五個部份是本書最精華的地方，也是它獨步全球的地方。要有這麼深入的了解，非得觀察 MFC 原始碼不可。本書把相關的 MFC 碼整理出來，加上相當多的示意圖，***MFC***

Internals 雖然挖得更廣，整理的功夫卻沒有這本好。

這本書用詞相當精準。用詞精準在容易岐路的物件導向領域中至爲重要，許多細微的觀念就在字句推敲中成形。

實例方面，希望看到琳琅滿目的範例程式的讀者，將會大失所望。這本書使用 Visual C++ 的標準範例 Scribble。只有 13~16 章才有幾個作者自己設計的程式，而且教育價值雖有，實在有點其貌不揚。然而以 Scribble 爲範例主軸，有一個意想不到的好處：常看 ***Microsoft Systems Journal*** 或 ***Windows Developer's Journal*** 的朋友就知道，許多作家喜歡在示範新技術或新構想時，以 Scribble 爲載具。如果你對 Scribble 十分熟悉，閱讀那些文章可就駕輕就熟了。

本書的許多精心插圖，是令人驚喜的地方。一圖解千言萬語，在這裡獲得最佳註腳。

一鍋湯，要放多少鹽才叫好？有人喜歡重口味，有人雅好清如水。

一本程式設計書籍，究竟要放多少碼，才能夠雅俗共賞，人人點頭？

關於這一點，有兩種完全迥異的看法。第一種看法似乎頗佔上風，他們說書籍應該是解釋程式的邏輯，程式人的意念，程式設計的...呃...境界。因此出現在書中的碼應該只能是小小的、片段的、簡捷的。但凡有一大落一大落的碼出現，那便是不入流、難登大雅之堂。我看過好多書評對於那種有著許多程式碼的書明嘲暗諷，甚而大加撻伐（呵呵，外文期刊上的書評很毒的）。

如果程式碼用來充篇幅，那就罵得好。

如果完整的碼用來給予完整的面觀，我就認爲值得。

其實，程式碼是贅餘還是適得其所，完全視讀者的個人情況而定。

Scribble 範例程式從第 4 章的 Step0 出發之後，陸陸續續只有片片斷斷的一個一個函式碼。我認爲有必要把本書所涵蓋的最後一個版本 Step5 的完整原始碼列出，以爲 Step0 之比對。

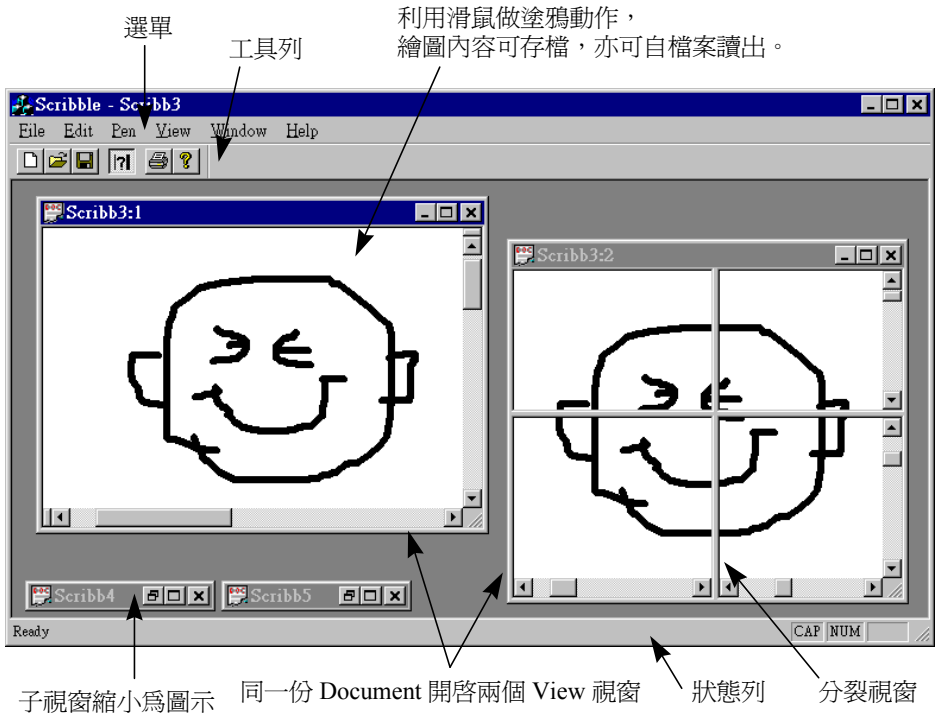
它來了。

抱怨程式碼太多的人，可能是認爲頁數多寡會反應在書籍售價上。有些書是這樣。但這本書，真的，我說，1000 頁或 700 頁，都不會影響它的價格（與價值）。

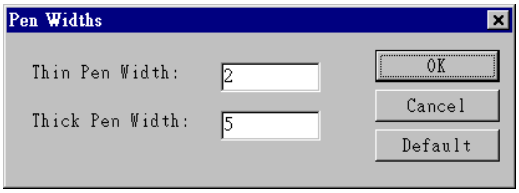
「附錄 B

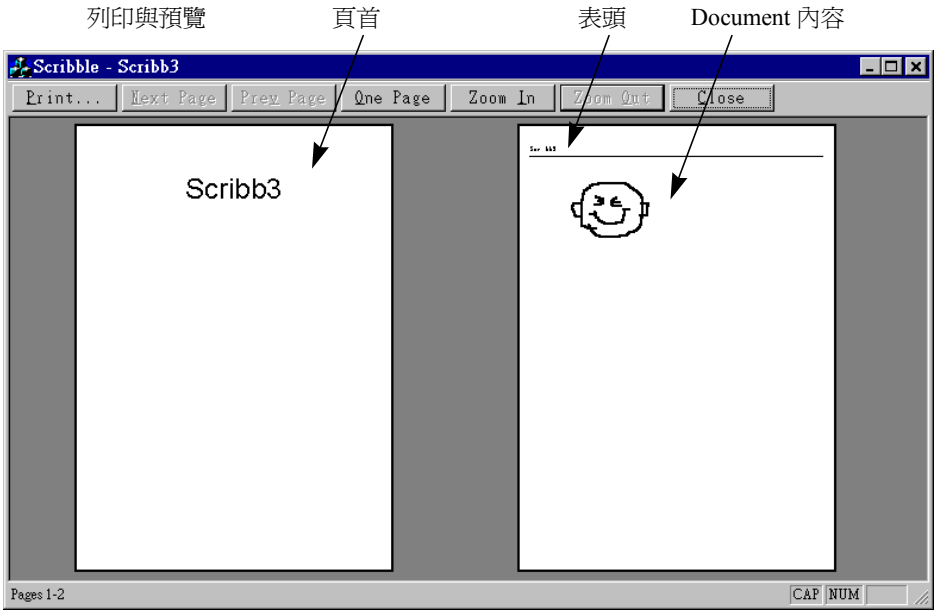
Scribble Step 5
完整原始碼

Scribble Step 5 是一個擁有如下功能的 MDI 塗鴉程式：



對話盒可設定繪筆粗細





SCRIBBLE.H

```
#0001  #ifndef __AFXWIN_H__
#0002         #error include 'stdafx.h' before including this file for PCH
#0003  #endif
#0004
#0005  #include "resource.h"       // main symbols
#0006
#0007  /////////////////////////////////////////////////////////////////////////
#0008  // CScribbleApp:
```

```
#0009  // See Scribble.cpp for the implementation of this class
#0010  //
#0011
#0012  class CScribbleApp : public CWinApp
#0013  {
#0014  public:
#0015         CScribbleApp();
#0016
#0017  // Overrides
#0018         // ClassWizard generated virtual function overrides
#0019         //{{AFX_VIRTUAL(CScribbleApp)
#0020         public:
#0021         virtual BOOL InitInstance();
#0022         //}}AFX_VIRTUAL
#0023
#0024  // Implementation
#0025
#0026         //{{AFX_MSG(CScribbleApp)
#0027         afx_msg void OnAppAbout();
#0028             // NOTE - the ClassWizard will add and remove member functions here.
#0029             //    DO NOT EDIT what you see in these blocks of generated code !
#0030         //}}AFX_MSG
#0031         DECLARE_MESSAGE_MAP()
#0032  };
```

SCRIBBLE.CPP

```
#0001  #include "stdafx.h"
#0002  #include "Scribble.h"
#0003
#0004  #include "MainFrm.h"
#0005  #include "ChildFrm.h"
#0006  #include "ScribDoc.h"
#0007  #include "ScribVw.h"
#0008
#0009  #ifdef _DEBUG
#0010  #define new DEBUG_NEW
#0011  #undef THIS_FILE
#0012  static char THIS_FILE[] = __FILE__;
#0013  #endif
#0014
#0015  /////////////////////////////////////////////////////////////////////////
#0016  // CScribbleApp
#0017
#0018  BEGIN_MESSAGE_MAP(CScribbleApp, CWinApp)
#0019         //{{AFX_MSG_MAP(CScribbleApp)
```

```
#0020         ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
#0021            // NOTE - the ClassWizard will add and remove mapping macros here.
#0022            //    DO NOT EDIT what you see in these blocks of generated code!
#0023         //}}AFX_MSG_MAP
#0024         // Standard file based document commands
#0025         ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
#0026         ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
#0027         // Standard print setup command
#0028         ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
#0029  END_MESSAGE_MAP()
#0030
#0031  /////////////////////////////////////////////////////////////////
#0032  // CScribbleApp construction
#0033
#0034  CScribbleApp::CScribbleApp()
#0035  {
#0036         // TODO: add construction code here,
#0037         // Place all significant initialization in InitInstance
#0038  }
#0039
#0040  /////////////////////////////////////////////////////////////////
#0041  // The one and only CScribbleApp object
#0042
#0043  CScribbleApp theApp;
#0044
#0045  /////////////////////////////////////////////////////////////////
#0046  // CScribbleApp initialization
#0047
#0048  BOOL CScribbleApp::InitInstance()
#0049  {
#0050         // Standard initialization
#0051         // If you are not using these features and wish to reduce the size
#0052         //  of your final executable, you should remove from the following
#0053         //  the specific initialization routines you do not need.
#0054
#0055  #ifdef _AFXDLL
#0056         Enable3dControls();  // Call this when using MFC in a shared DLL
#0057  #else
#0058         Enable3dControlsStatic();  // Call this when linking to MFC statically
#0059  #endif
#0060
#0061        LoadStdProfileSettings();  // Load standard INI file options (including MRU)
#0062
#0063
#0064         // Register the application's document templates.  Document templates
#0065         //  serve as the connection between documents, frame windows and views.
```

```
#0066
#0067         CMultiDocTemplate* pDocTemplate;
#0068         pDocTemplate = new CMultiDocTemplate(
#0069                IDR_SCRIBBTYPE,
#0070                RUNTIME_CLASS(CScribbleDoc),
#0071                RUNTIME_CLASS(CChildFrame), // custom MDI child frame
#0072                RUNTIME_CLASS(CScribbleView));
#0073
#0074         AddDocTemplate(pDocTemplate);
#0075
#0076         // create main MDI Frame window
#0077         CMainFrame* pMainFrame = new CMainFrame;
#0078         if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
#0079                return FALSE;
#0080         m_pMainWnd = pMainFrame;
#0081
#0082         // Enable drag/drop open.  We don't call this in Win32, since a
#0083         //  document file extension wasn't chosen while running AppWizard.
#0084         m_pMainWnd->DragAcceptFiles();
#0085
#0086         // Enable DDE Execute open
#0087         EnableShellOpen();
#0088         RegisterShellFileTypes(TRUE);
#0089
#0090         // Parse command line for standard shell commands, DDE, file open
#0091         CCommandLineInfo cmdInfo;
#0092         ParseCommandLine(cmdInfo);
#0093
#0094         // Dispatch commands specified on the command line
#0095         if (!ProcessShellCommand(cmdInfo))
#0096                return FALSE;
#0097
#0098
#0099         // The main window has been initialized, so show and update it.
#0100         pMainFrame->ShowWindow(m_nCmdShow);
#0101         pMainFrame->UpdateWindow();
#0102
#0103         return TRUE;
#0104  }
#0105
#0106  /////////////////////////////////////////////////////////////////
#0107  // CAboutDlg dialog used for App About
#0108
#0109  class CAboutDlg : public CDialog
#0110  {
#0111  public:
```

```
#0112        CAboutDlg();
#0113
#0114  // Dialog Data
#0115        //{{AFX_DATA(CAboutDlg)
#0116        enum { IDD = IDD_ABOUTBOX };
#0117        //}}AFX_DATA
#0118
#0119        // ClassWizard generated virtual function overrides
#0120        //{{AFX_VIRTUAL(CAboutDlg)
#0121        protected:
#0122        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
#0123        //}}AFX_VIRTUAL
#0124
#0125  // Implementation
#0126  protected:
#0127        //{{AFX_MSG(CAboutDlg)
#0128              // No message handlers
#0129        //}}AFX_MSG
#0130        DECLARE_MESSAGE_MAP()
#0131  };
#0132
#0133  CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
#0134  {
#0135        //{{AFX_DATA_INIT(CAboutDlg)
#0136        //}}AFX_DATA_INIT
#0137  }
#0138
#0139  void CAboutDlg::DoDataExchange(CDataExchange* pDX)
#0140  {
#0141        CDialog::DoDataExchange(pDX);
#0142        //{{AFX_DATA_MAP(CAboutDlg)
#0143        //}}AFX_DATA_MAP
#0144  }
#0145
#0146  BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
#0147        //{{AFX_MSG_MAP(CAboutDlg)
#0148              // No message handlers
#0149        //}}AFX_MSG_MAP
#0150  END_MESSAGE_MAP()
#0151
#0152  // App command to run the dialog
#0153  void CScribbleApp::OnAppAbout()
#0154  {
#0155        CAboutDlg aboutDlg;
#0156        aboutDlg.DoModal();
#0157  }
```

MAINFRAME.H

```
#0001  class CMainFrame : public CMDIFrameWnd
#0002  {
#0003         DECLARE_DYNAMIC(CMainFrame)
#0004  public:
#0005         CMainFrame();
#0006
#0007  // Attributes
#0008  public:
#0009
#0010  // Operations
#0011  public:
#0012
#0013  // Overrides
#0014         // ClassWizard generated virtual function overrides
#0015         //{{AFX_VIRTUAL(CMainFrame)
#0016         virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
#0017         //}}AFX_VIRTUAL
#0018
#0019  // Implementation
#0020  public:
#0021         virtual ~CMainFrame();
#0022  #ifdef _DEBUG
#0023         virtual void AssertValid() const;
#0024         virtual void Dump(CDumpContext& dc) const;
#0025  #endif
#0026
#0027  protected:  // control bar embedded members
#0028         CStatusBar  m_wndStatusBar;
#0029         CToolBar    m_wndToolBar;
#0030
#0031  // Generated message map functions
#0032  protected:
#0033         //{{AFX_MSG(CMainFrame)
#0034         afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
#0035           // NOTE - the ClassWizard will add and remove member functions here.
#0036           //    DO NOT EDIT what you see in these blocks of generated code!
#0037         //}}AFX_MSG
#0038         DECLARE_MESSAGE_MAP()
#0039  };
```

MAINFRAME.CPP

```
#0001  #include "stdafx.h"
#0002  #include "Scribble.h"
#0003
#0004  #include "MainFrm.h"
#0005
#0006  #ifdef _DEBUG
#0007  #define new DEBUG_NEW
#0008  #undef THIS_FILE
#0009  static char THIS_FILE[] = __FILE__;
#0010  #endif
#0011
#0012  /////////////////////////////////////////////////////////////
#0013  // CMainFrame
#0014
#0015  IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)
#0016
#0017  BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
#0018         //{{AFX_MSG_MAP(CMainFrame)
#0019             // NOTE - the ClassWizard will add and remove mapping macros here.
#0020             //    DO NOT EDIT what you see in these blocks of generated code !
#0021         ON_WM_CREATE()
#0022         //}}AFX_MSG_MAP
#0023  END_MESSAGE_MAP()
#0024
#0025  static UINT indicators[] =
#0026  {
#0027         ID_SEPARATOR,          // status line indicator
#0028         ID_INDICATOR_CAPS,
#0029         ID_INDICATOR_NUM,
#0030         ID_INDICATOR_SCRL,
#0031  };
#0032
#0033  /////////////////////////////////////////////////////////////////
#0034  // CMainFrame construction/destruction
#0035
#0036  CMainFrame::CMainFrame()
#0037  {
#0038         // TODO: add member initialization code here
#0039
#0040  }
#0041
#0042  CMainFrame::~CMainFrame()
#0043  {
#0044  }
```

```
#0045
#0046  int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
#0047  {
#0048         if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
#0049                return -1;
#0050
#0051         if (!m_wndToolBar.Create(this) ||
#0052                !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
#0053         {
#0054                TRACE0("Failed to create toolbar\n");
#0055                return -1;     // fail to create
#0056         }
#0057
#0058         if (!m_wndStatusBar.Create(this) ||
#0059                !m_wndStatusBar.SetIndicators(indicators,
#0060                  sizeof(indicators)/sizeof(UINT)))
#0061         {
#0062                TRACE0("Failed to create status bar\n");
#0063                return -1;     // fail to create
#0064         }
#0065
#0066         // TODO: Remove this if you don't want tool tips
#0067         m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
#0068                CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
#0069
#0070         // TODO: Delete these three lines if you don't want the toolbar to
#0071         //  be dockable
#0072         m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
#0073         EnableDocking(CBRS_ALIGN_ANY);
#0074         DockControlBar(&m_wndToolBar);
#0075
#0076
#0077         return 0;
#0078  }
#0079
#0080  BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
#0081  {
#0082         // TODO: Modify the Window class or styles here by modifying
#0083         //  the CREATESTRUCT cs
#0084
#0085         return CMDIFrameWnd::PreCreateWindow(cs);
#0086  }
#0087
#0088  /////////////////////////////////////////////////////////////////////////
#0089  // CMainFrame diagnostics
#0090
```

```
#0091  #ifdef _DEBUG
#0092  void CMainFrame::AssertValid() const
#0093  {
#0094         CMDIFrameWnd::AssertValid();
#0095  }
#0096
#0097  void CMainFrame::Dump(CDumpContext& dc) const
#0098  {
#0099         CMDIFrameWnd::Dump(dc);
#0100  }
#0101
#0102  #endif //_DEBUG
#0103
#0104  /////////////////////////////////////////////////////////////////
#0105  // CMainFrame message handlers
```

CHILDFRM.H

```
#0001  class CChildFrame : public CMDIChildWnd
#0002  {
#0003         DECLARE_DYNCREATE(CChildFrame)
#0004  public:
#0005         CChildFrame();
#0006
#0007  // Attributes
#0008  protected:
#0009         CSplitterWnd    m_wndSplitter;
#0010  public:
#0011
#0012  // Operations
#0013  public:
#0014
#0015  // Overrides
#0016         // ClassWizard generated virtual function overrides
#0017         //{{AFX_VIRTUAL(CChildFrame)
#0018         public:
#0019         virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
#0020         protected:
#0021    virtual BOOL OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext);
#0022    //}}AFX_VIRTUAL
#0023
#0024  // Implementation
#0025  public:
#0026         virtual ~CChildFrame();
#0027  #ifdef _DEBUG
#0028         virtual void AssertValid() const;
```

```
#0029         virtual void Dump(CDumpContext& dc) const;
#0030  #endif
#0031
#0032  // Generated message map functions
#0033  protected:
#0034         //{{AFX_MSG(CChildFrame)
#0035           // NOTE - the ClassWizard will add and remove member functions here.
#0036           //    DO NOT EDIT what you see in these blocks of generated code!
#0037         //}}AFX_MSG
#0038         DECLARE_MESSAGE_MAP()
#0039  };
```

CHILDFRM.CPP

```
#0001  #include "stdafx.h"
#0002  #include "Scribble.h"
#0003
#0004  #include "ChildFrm.h"
#0005
#0006  #ifdef _DEBUG
#0007  #define new DEBUG_NEW
#0008  #undef THIS_FILE
#0009  static char THIS_FILE[] = __FILE__;
#0010  #endif
#0011
#0012  /////////////////////////////////////////////////////////////////////////
#0013  // CChildFrame
#0014
#0015  IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)
#0016
#0017  BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
#0018         //{{AFX_MSG_MAP(CChildFrame)
#0019             // NOTE - the ClassWizard will add and remove mapping macros here.
#0020             //    DO NOT EDIT what you see in these blocks of generated code !
#0021         //}}AFX_MSG_MAP
#0022  END_MESSAGE_MAP()
#0023
#0024  /////////////////////////////////////////////////////////////////////////
#0025  // CChildFrame construction/destruction
#0026
#0027  CChildFrame::CChildFrame()
#0028  {
#0029         // TODO: add member initialization code here
#0030
#0031  }
#0032
```

```
#0033  CChildFrame::~CChildFrame()
#0034  {
#0035  }
#0036
#0037  BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
#0038  {
#0039         // TODO: Modify the Window class or styles here by modifying
#0040         //  the CREATESTRUCT cs
#0041
#0042         return CMDIChildWnd::PreCreateWindow(cs);
#0043  }
#0044
#0045  /////////////////////////////////////////////////////////////////
#0046  // CChildFrame diagnostics
#0047
#0048  #ifdef _DEBUG
#0049  void CChildFrame::AssertValid() const
#0050  {
#0051         CMDIChildWnd::AssertValid();
#0052  }
#0053
#0054  void CChildFrame::Dump(CDumpContext& dc) const
#0055  {
#0056         CMDIChildWnd::Dump(dc);
#0057  }
#0058
#0059  #endif //_DEBUG
#0060
#0061  /////////////////////////////////////////////////////////////////
#0062  // CChildFrame message handlers
#0063
#0064  BOOL CChildFrame::OnCreateClient(LPCREATESTRUCT /* lpcs */,
                                       CCreateContext* pContext)
#0065  {
#0066         return m_wndSplitter.Create(this,
#0067                2, 2,      // TODO: adjust the number of rows, columns
#0068                CSize(10, 10),  // TODO: adjust the minimum pane size
#0069                pContext);
#0070  }
```

SCRIBBLEDOC.H

```
#0001  /////////////////////////////////////////////////////////////////
#0002  // class CStroke
#0003  //
#0004  // A stroke is a series of connected points in the scribble drawing.
```

```
#0005  // A scribble document may have multiple strokes.
#0006
#0007  class CStroke : public CObject
#0008  {
#0009  public:
#0010         CStroke(UINT nPenWidth);
#0011
#0012  protected:
#0013         CStroke();
#0014         DECLARE_SERIAL(CStroke)
#0015
#0016  // Attributes
#0017  protected:
#0018         UINT    m_nPenWidth;   // one pen width applies to entire stroke
#0019  public:
#0020         CArray<CPoint,CPoint>  m_pointArray;   // series of connected points
#0021         CRect   m_rectBounding;  // smallest rect that surrounds all
#0022                                  // of the points in the stroke
#0023                                  // measured in MM_LOENGLISH units
#0024                                  // (0.01 inches, with Y-axis inverted)
#0025  public:
#0026         CRect& GetBoundingRect() { return m_rectBounding; }
#0027
#0028  // Operations
#0029  public:
#0030         BOOL DrawStroke(CDC* pDC);
#0031         void FinishStroke();
#0032
#0033  public:
#0034         virtual void Serialize(CArchive& ar);
#0035  };
#0036
#0037  /////////////////////////////////////////////////////////////////////
#0038
#0039  class CScribbleDoc : public CDocument
#0040  {
#0041  protected: // create from serialization only
#0042         CScribbleDoc();
#0043         DECLARE_DYNCREATE(CScribbleDoc)
#0044
#0045  // Attributes
#0046  protected:
#0047         // The document keeps track of the current pen width on
#0048         // behalf of all views. We'd like the user interface of
#0049         // Scribble to be such that if the user chooses the Draw
#0050         // Thick Line command, it will apply to all views, not just
```

```
#0051        // the view that currently has the focus.
#0052
#0053        UINT    m_nPenWidth;       // current user-selected pen width
#0054        BOOL    m_bThickPen;       // TRUE if current pen is thick
#0055        UINT    m_nThinWidth;
#0056        UINT    m_nThickWidth;
#0057        CPen    m_penCur;          // pen created according to
#0058                                   // user-selected pen style (width)
#0059  public:
#0060        CTypedPtrList<CObList,CStroke*>    m_strokeList;
#0061        CPen*          GetCurrentPen() { return &m_penCur; }
#0062
#0063  protected:
#0064        CSize          m_sizeDoc;
#0065  public:
#0066        CSize GetDocSize() { return m_sizeDoc; }
#0067
#0068  // Operations
#0069  public:
#0070        CStroke* NewStroke();
#0071
#0072  // Overrides
#0073        // ClassWizard generated virtual function overrides
#0074        //{{AFX_VIRTUAL(CScribbleDoc)
#0075        public:
#0076        virtual BOOL OnNewDocument();
#0077        virtual void Serialize(CArchive& ar);
#0078        virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
#0079        virtual void DeleteContents();
#0080        //}}AFX_VIRTUAL
#0081
#0082  // Implementation
#0083  protected:
#0084        void ReplacePen();
#0085
#0086  public:
#0087        virtual ~CScribbleDoc();
#0088  #ifdef _DEBUG
#0089        virtual void AssertValid() const;
#0090        virtual void Dump(CDumpContext& dc) const;
#0091  #endif
#0092
#0093  protected:
#0094        void           InitDocument();
#0095
#0096  // Generated message map functions
```

```
#0097  protected:
#0098         //{{AFX_MSG(CScribbleDoc)
#0099         afx_msg void OnEditClearAll();
#0100         afx_msg void OnPenThickOrThin();
#0101         afx_msg void OnUpdateEditClearAll(CCmdUI* pCmdUI);
#0102         afx_msg void OnUpdatePenThickOrThin(CCmdUI* pCmdUI);
#0103         afx_msg void OnPenWidths();
#0104         //}}AFX_MSG
#0105         DECLARE_MESSAGE_MAP()
#0106  };
```

SCRIBBLEDOC.CPP

```
#0001  #include "stdafx.h"
#0002  #include "Scribble.h"
#0003
#0004  #include "ScribDoc.h"
#0005  #include "PenDlg.h"
#0006
#0007  #ifdef _DEBUG
#0008  #define new DEBUG_NEW
#0009  #undef THIS_FILE
#0010  static char THIS_FILE[] = __FILE__;
#0011  #endif
#0012
#0013  /////////////////////////////////////////////////////////////////
#0014  // CScribbleDoc
#0015
#0016  IMPLEMENT_DYNCREATE(CScribbleDoc, CDocument)
#0017
#0018  BEGIN_MESSAGE_MAP(CScribbleDoc, CDocument)
#0019         //{{AFX_MSG_MAP(CScribbleDoc)
#0020         ON_COMMAND(ID_EDIT_CLEAR_ALL, OnEditClearAll)
#0021         ON_COMMAND(ID_PEN_THICK_OR_THIN, OnPenThickOrThin)
#0022         ON_UPDATE_COMMAND_UI(ID_EDIT_CLEAR_ALL, OnUpdateEditClearAll)
#0023         ON_UPDATE_COMMAND_UI(ID_PEN_THICK_OR_THIN, OnUpdatePenThickOrThin)
#0024         ON_COMMAND(ID_PEN_WIDTHS, OnPenWidths)
#0025         //}}AFX_MSG_MAP
#0026  END_MESSAGE_MAP()
#0027
#0028  /////////////////////////////////////////////////////////////////
#0029  // CScribbleDoc construction/destruction
#0030
#0031  CScribbleDoc::CScribbleDoc()
#0032  {
```

```
#0033        // TODO: add one-time construction code here
#0034
#0035  }
#0036
#0037  CScribbleDoc::~CScribbleDoc()
#0038  {
#0039  }
#0040
#0041  BOOL CScribbleDoc::OnNewDocument()
#0042  {
#0043        if (!CDocument::OnNewDocument())
#0044              return FALSE;
#0045        InitDocument();
#0046        return TRUE;
#0047  }
#0048
#0049  /////////////////////////////////////////////////////////////////
#0050  // CScribbleDoc serialization
#0051
#0052  void CScribbleDoc::Serialize(CArchive& ar)
#0053  {
#0054        if (ar.IsStoring())
#0055        {
#0056              ar << m_sizeDoc;
#0057        }
#0058        else
#0059        {
#0060              ar >> m_sizeDoc;
#0061        }
#0062        m_strokeList.Serialize(ar);
#0063  }
#0064
#0065  /////////////////////////////////////////////////////////////////
#0066  // CScribbleDoc diagnostics
#0067
#0068  #ifdef _DEBUG
#0069  void CScribbleDoc::AssertValid() const
#0070  {
#0071        CDocument::AssertValid();
#0072  }
#0073
#0074  void CScribbleDoc::Dump(CDumpContext& dc) const
#0075  {
#0076        CDocument::Dump(dc);
#0077  }
#0078  #endif //_DEBUG
```

```
#0079
#0080 /////////////////////////////////////////////////////////////////
#0081 // CScribbleDoc commands
#0082
#0083 BOOL CScribbleDoc::OnOpenDocument(LPCTSTR lpszPathName)
#0084 {
#0085        if (!CDocument::OnOpenDocument(lpszPathName))
#0086               return FALSE;
#0087        InitDocument();
#0088        return TRUE;
#0089 }
#0090
#0091 void CScribbleDoc::DeleteContents()
#0092 {
#0093        while (!m_strokeList.IsEmpty())
#0094        {
#0095               delete m_strokeList.RemoveHead();
#0096        }
#0097        CDocument::DeleteContents();
#0098 }
#0099
#0100 void CScribbleDoc::InitDocument()
#0101 {
#0102        m_bThickPen = FALSE;
#0103        m_nThinWidth = 2;   // default thin pen is 2 pixels wide
#0104        m_nThickWidth = 5;  // default thick pen is 5 pixels wide
#0105        ReplacePen();     // initialize pen according to current width
#0106
#0107        // default document size is 800 x 900 screen pixels
#0108        m_sizeDoc = CSize(800,900);
#0109 }
#0110
#0111 CStroke* CScribbleDoc::NewStroke()
#0112 {
#0113        CStroke* pStrokeItem = new CStroke(m_nPenWidth);
#0114        m_strokeList.AddTail(pStrokeItem);
#0115        SetModifiedFlag();  // Mark the document as having been modified, for
#0116                            // purposes of confirming File Close.
#0117        return pStrokeItem;
#0118 }
#0119
#0120
#0121
#0122
#0123 /////////////////////////////////////////////////////////////////
#0124 // CStroke
```

```
#0125
#0126  IMPLEMENT_SERIAL(CStroke, CObject, 2)
#0127  CStroke::CStroke()
#0128  {
#0129         // This empty constructor should be used by serialization only
#0130  }
#0131
#0132  CStroke::CStroke(UINT nPenWidth)
#0133  {
#0134         m_nPenWidth = nPenWidth;
#0135         m_rectBounding.SetRectEmpty();
#0136  }
#0137
#0138  void CStroke::Serialize(CArchive& ar)
#0139  {
#0140         if (ar.IsStoring())
#0141         {
#0142                ar << m_rectBounding;
#0143                ar << (WORD)m_nPenWidth;
#0144                m_pointArray.Serialize(ar);
#0145         }
#0146         else
#0147         {
#0148                ar >> m_rectBounding;
#0149                WORD w;
#0150                ar >> w;
#0151                m_nPenWidth = w;
#0152                m_pointArray.Serialize(ar);
#0153         }
#0154  }
#0155
#0156  BOOL CStroke::DrawStroke(CDC* pDC)
#0157  {
#0158         CPen penStroke;
#0159         if (!penStroke.CreatePen(PS_SOLID, m_nPenWidth, RGB(0,0,0)))
#0160                return FALSE;
#0161         CPen* pOldPen = pDC->SelectObject(&penStroke);
#0162         pDC->MoveTo(m_pointArray[0]);
#0163         for (int i=1; i < m_pointArray.GetSize(); i++)
#0164         {
#0165                pDC->LineTo(m_pointArray[i]);
#0166         }
#0167
#0168         pDC->SelectObject(pOldPen);
#0169         return TRUE;
#0170  }
```

```
#0171  void CScribbleDoc::OnEditClearAll()
#0172  {
#0173         DeleteContents();
#0174         SetModifiedFlag();  // Mark the document as having been modified, for
#0175                             // purposes of confirming File Close.
#0176         UpdateAllViews(NULL);
#0177  }
#0178
#0179  void CScribbleDoc::OnPenThickOrThin()
#0180  {
#0181         // Toggle the state of the pen between thin or thick.
#0182         m_bThickPen = !m_bThickPen;
#0183
#0184         // Change the current pen to reflect the new user-specified width.
#0185         ReplacePen();
#0186  }
#0187
#0188  void CScribbleDoc::ReplacePen()
#0189  {
#0190         m_nPenWidth = m_bThickPen? m_nThickWidth : m_nThinWidth;
#0191
#0192         // Change the current pen to reflect the new user-specified width.
#0193         m_penCur.DeleteObject();
#0194         m_penCur.CreatePen(PS_SOLID, m_nPenWidth, RGB(0,0,0)); // solid black
#0195  }
#0196
#0197  void CScribbleDoc::OnUpdateEditClearAll(CCmdUI* pCmdUI)
#0198  {
#0199         // Enable the command user interface object (menu item or tool bar
#0200         // button) if the document is non-empty, i.e., has at least one stroke.
#0201         pCmdUI->Enable(!m_strokeList.IsEmpty());
#0202  }
#0203
#0204  void CScribbleDoc::OnUpdatePenThickOrThin(CCmdUI* pCmdUI)
#0205  {
#0206         // Add check mark to Draw Thick Line menu item, if the current
#0207         // pen width is "thick".
#0208         pCmdUI->SetCheck(m_bThickPen);
#0209  }
#0210
#0211  void CScribbleDoc::OnPenWidths()
#0212  {
#0213         CPenWidthsDlg dlg;
#0214         // Initialize dialog data
#0215         dlg.m_nThinWidth = m_nThinWidth;
#0216         dlg.m_nThickWidth = m_nThickWidth;
```

```
#0217
#0218         // Invoke the dialog box
#0219         if (dlg.DoModal() == IDOK)
#0220         {
#0221                // retrieve the dialog data
#0222                m_nThinWidth = dlg.m_nThinWidth;
#0223                m_nThickWidth = dlg.m_nThickWidth;
#0224
#0225                // Update the pen that is used by views when drawing new strokes,
#0226                // to reflect the new pen width definitions for "thick" and "thin".
#0227                ReplacePen();
#0228         }
#0229  }
#0230
#0231  void CStroke::FinishStroke()
#0232  {
#0233         // Calculate the bounding rectangle.  It's needed for smart
#0234         // repainting.
#0235
#0236         if (m_pointArray.GetSize()==0)
#0237         {
#0238                m_rectBounding.SetRectEmpty();
#0239                return;
#0240         }
#0241         CPoint pt = m_pointArray[0];
#0242         m_rectBounding = CRect(pt.x, pt.y, pt.x, pt.y);
#0243
#0244         for (int i=1; i < m_pointArray.GetSize(); i++)
#0245         {
#0246                // If the point lies outside of the accumulated bounding
#0247                // rectangle, then inflate the bounding rect to include it.
#0248                pt = m_pointArray[i];
#0249                m_rectBounding.left    = min(m_rectBounding.left, pt.x);
#0250                m_rectBounding.right   = max(m_rectBounding.right, pt.x);
#0251                m_rectBounding.top     = max(m_rectBounding.top, pt.y);
#0252                m_rectBounding.bottom  = min(m_rectBounding.bottom, pt.y);
#0253         }
#0254
#0255         // Add the pen width to the bounding rectangle.  This is necessary
#0256         // to account for the width of the stroke when invalidating
#0257         // the screen.
#0258         m_rectBounding.InflateRect(CSize(m_nPenWidth, -(int)m_nPenWidth));
#0259         return;
#0260  }
```

SCRIBBLEVIEW.H

```
#0001  class CScribbleView : public CScrollView
#0002  {
#0003  protected: // create from serialization only
#0004         CScribbleView();
#0005         DECLARE_DYNCREATE(CScribbleView)
#0006
#0007  // Attributes
#0008  public:
#0009         CScribbleDoc* GetDocument();
#0010
#0011  protected:
#0012         CStroke*    m_pStrokeCur;   // the stroke in progress
#0013         CPoint      m_ptPrev;  // the last mouse pt in the stroke in progress
#0014
#0015  // Operations
#0016  public:
#0017
#0018  // Overrides
#0019         // ClassWizard generated virtual function overrides
#0020         //{{AFX_VIRTUAL(CScribbleView)
#0021         public:
#0022         virtual void OnDraw(CDC* pDC);  // overridden to draw this view
#0023         virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
#0024         virtual void OnInitialUpdate();
#0025         protected:
#0026         virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
#0027         virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
#0028         virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
#0029         virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
#0030         virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
#0031         //}}AFX_VIRTUAL
#0032
#0033  // Implementation
#0034  public:
#0035         void PrintTitlePage(CDC* pDC, CPrintInfo* pInfo);
#0036         void PrintPageHeader(CDC* pDC, CPrintInfo* pInfo, CString& strHeader);
#0037         virtual ~CScribbleView();
#0038  #ifdef _DEBUG
#0039         virtual void AssertValid() const;
#0040         virtual void Dump(CDumpContext& dc) const;
#0041  #endif
#0042
#0043  protected:
#0044
```

```
#0045  // Generated message map functions
#0046  protected:
#0047         //{{AFX_MSG(CScribbleView)
#0048         afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
#0049         afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
#0050         afx_msg void OnMouseMove(UINT nFlags, CPoint point);
#0051         //}}AFX_MSG
#0052         DECLARE_MESSAGE_MAP()
#0053  };
#0054
#0055  #ifndef _DEBUG  // debug version in ScribVw.cpp
#0056  inline CScribbleDoc* CScribbleView::GetDocument()
#0057     { return (CScribbleDoc*)m_pDocument; }
#0058  #endif
```

SCRIBBLEVIEW.CPP

```
#0001  #include "stdafx.h"
#0002  #include "Scribble.h"
#0003
#0004  #include "ScribDoc.h"
#0005  #include "ScribVw.h"
#0006
#0007  #ifdef _DEBUG
#0008  #define new DEBUG_NEW
#0009  #undef THIS_FILE
#0010  static char THIS_FILE[] = __FILE__;
#0011  #endif
#0012
#0013  /////////////////////////////////////////////////////////////////
#0014  // CScribbleView
#0015
#0016  IMPLEMENT_DYNCREATE(CScribbleView, CScrollView)
#0017
#0018  BEGIN_MESSAGE_MAP(CScribbleView, CScrollView)
#0019         //{{AFX_MSG_MAP(CScribbleView)
#0020         ON_WM_LBUTTONDOWN()
#0021         ON_WM_LBUTTONUP()
#0022         ON_WM_MOUSEMOVE()
#0023         //}}AFX_MSG_MAP
#0024         // Standard printing commands
#0025         ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
#0026         ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
#0027         ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
#0028  END_MESSAGE_MAP()
#0029
```

```
#0030 /////////////////////////////////////////////////////////////////
#0031 // CScribbleView construction/destruction
#0032
#0033 CScribbleView::CScribbleView()
#0034 {
#0035        // TODO: add construction code here
#0036
#0037 }
#0038
#0039 CScribbleView::~CScribbleView()
#0040 {
#0041 }
#0042
#0043 BOOL CScribbleView::PreCreateWindow(CREATESTRUCT& cs)
#0044 {
#0045        // TODO: Modify the Window class or styles here by modifying
#0046        //  the CREATESTRUCT cs
#0047
#0048        return CView::PreCreateWindow(cs);
#0049 }
#0050
#0051 /////////////////////////////////////////////////////////////////
#0052 // CScribbleView drawing
#0053
#0054 void CScribbleView::OnDraw(CDC* pDC)
#0055 {
#0056        CScribbleDoc* pDoc = GetDocument();
#0057        ASSERT_VALID(pDoc);
#0058
#0059        // Get the invalidated rectangle of the view, or in the case
#0060        // of printing, the clipping region of the printer dc.
#0061        CRect rectClip;
#0062        CRect rectStroke;
#0063        pDC->GetClipBox(&rectClip);
#0064        pDC->LPtoDP(&rectClip);
#0065        rectClip.InflateRect(1, 1); // avoid rounding to nothing
#0066
#0067        // Note: CScrollView::OnPaint() will have already adjusted the
#0068        // viewport origin before calling OnDraw(), to reflect the
#0069        // currently scrolled position.
#0070
#0071        // The view delegates the drawing of individual strokes to
#0072        // CStroke::DrawStroke().
#0073        CTypedPtrList<CObList,CStroke*>& strokeList = pDoc->m_strokeList;
#0074        POSITION pos = strokeList.GetHeadPosition();
#0075        while (pos != NULL)
```

```
#0076         {
#0077                CStroke* pStroke = strokeList.GetNext(pos);
#0078                rectStroke = pStroke->GetBoundingRect();
#0079                pDC->LPtoDP(&rectStroke);
#0080                rectStroke.InflateRect(1, 1); // avoid rounding to nothing
#0081                if (!rectStroke.IntersectRect(&rectStroke, &rectClip))
#0082                       continue;
#0083                pStroke->DrawStroke(pDC);
#0084         }
#0085  }
#0086
#0087  /////////////////////////////////////////////////////////////////
#0088  // CScribbleView printing
#0089
#0090  BOOL CScribbleView::OnPreparePrinting(CPrintInfo* pInfo)
#0091  {
#0092         pInfo->SetMaxPage(2);   // the document is two pages long:
#0093                                 // the first page is the title page
#0094                                 // the second is the drawing
#0095         BOOL bRet = DoPreparePrinting(pInfo);   // default preparation
#0096         pInfo->m_nNumPreviewPages = 2;  // Preview 2 pages at a time
#0097         // Set this value after calling DoPreparePrinting to override
#0098         // value read from .INI file
#0099         return bRet;
#0100  }
#0101
#0102  void CScribbleView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
#0103  {
#0104         // TODO: add extra initialization before printing
#0105  }
#0106
#0107  void CScribbleView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
#0108  {
#0109         // TODO: add cleanup after printing
#0110  }
#0111
#0112  /////////////////////////////////////////////////////////////////
#0113  // CScribbleView diagnostics
#0114
#0115  #ifdef _DEBUG
#0116  void CScribbleView::AssertValid() const
#0117  {
#0118         CScrollView::AssertValid();
#0119  }
#0120
#0121  void CScribbleView::Dump(CDumpContext& dc) const
```

```
#0122 {
#0123        CScrollView::Dump(dc);
#0124 }
#0125
#0126 CScribbleDoc* CScribbleView::GetDocument() // non-debug version is inline
#0127 {
#0128        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CScribbleDoc)));
#0129        return (CScribbleDoc*)m_pDocument;
#0130 }
#0131 #endif //_DEBUG
#0132
#0133 /////////////////////////////////////////////////////////////////
#0134 // CScribbleView message handlers
#0135
#0136 void CScribbleView::OnLButtonDown(UINT, CPoint point)
#0137 {
#0138    // Pressing the mouse button in the view window starts a new stroke
#0139
#0140        // CScrollView changes the viewport origin and mapping mode.
#0141        // It's necessary to convert the point from device coordinates
#0142        // to logical coordinates, such as are stored in the document.
#0143        CClientDC dc(this);
#0144        OnPrepareDC(&dc);
#0145        dc.DPtoLP(&point);
#0146
#0147        m_pStrokeCur = GetDocument()->NewStroke();
#0148        // Add first point to the new stroke
#0149        m_pStrokeCur->m_pointArray.Add(point);
#0150
#0151        SetCapture();      // Capture the mouse until button up.
#0152        m_ptPrev = point;   // Serves as the MoveTo() anchor point for the
#0153                           // LineTo() the next point, as the user drags the
#0154                           // mouse.
#0155
#0156        return;
#0157 }
#0158
#0159 void CScribbleView::OnLButtonUp(UINT, CPoint point)
#0160 {
#0161        // Mouse button up is interesting in the Scribble application
#0162        // only if the user is currently drawing a new stroke by dragging
#0163        // the captured mouse.
#0164
#0165        if (GetCapture() != this)
#0166               return; // If this window (view) didn't capture the mouse,
#0167                              // then the user isn't drawing in this window.
```

```
#0168
#0169         CScribbleDoc* pDoc = GetDocument();
#0170
#0171         CClientDC dc(this);
#0172
#0173         // CScrollView changes the viewport origin and mapping mode.
#0174         // It's necessary to convert the point from device coordinates
#0175         // to logical coordinates, such as are stored in the document.
#0176         OnPrepareDC(&dc);  // set up mapping mode and viewport origin
#0177         dc.DPtoLP(&point);
#0178
#0179         CPen* pOldPen = dc.SelectObject(pDoc->GetCurrentPen());
#0180         dc.MoveTo(m_ptPrev);
#0181         dc.LineTo(point);
#0182         dc.SelectObject(pOldPen);
#0183         m_pStrokeCur->m_pointArray.Add(point);
#0184
#0185         // Tell the stroke item that we're done adding points to it.
#0186         // This is so it can finish computing its bounding rectangle.
#0187         m_pStrokeCur->FinishStroke();
#0188
#0189         // Tell the other views that this stroke has been added
#0190         // so that they can invalidate this stroke's area in their
#0191         // client area.
#0192         pDoc->UpdateAllViews(this, 0L, m_pStrokeCur);
#0193
#0194         ReleaseCapture();   // Release the mouse capture established at
#0195                             // the beginning of the mouse drag.
#0196         return;
#0197 }
#0198
#0199 void CScribbleView::OnMouseMove(UINT, CPoint point)
#0200 {
#0201         // Mouse movement is interesting in the Scribble application
#0202         // only if the user is currently drawing a new stroke by dragging
#0203         // the captured mouse.
#0204
#0205         if (GetCapture() != this)
#0206                 return; // If this window (view) didn't capture the mouse,
#0207                         // then the user isn't drawing in this window.
#0208
#0209         CClientDC dc(this);
#0210         // CScrollView changes the viewport origin and mapping mode.
#0211         // It's necessary to convert the point from device coordinates
#0212         // to logical coordinates, such as are stored in the document.
#0213         OnPrepareDC(&dc);
```

```
#0214         dc.DPtoLP(&point);
#0215
#0216         m_pStrokeCur->m_pointArray.Add(point);
#0217
#0218         // Draw a line from the previous detected point in the mouse
#0219         // drag to the current point.
#0220         CPen* pOldPen = dc.SelectObject(GetDocument()->GetCurrentPen());
#0221         dc.MoveTo(m_ptPrev);
#0222         dc.LineTo(point);
#0223         dc.SelectObject(pOldPen);
#0224         m_ptPrev = point;
#0225         return;
#0226  }
#0227
#0228  void CScribbleView::OnUpdate(CView* /* pSender */, LPARAM /* lHint */,
#0229         CObject* pHint)
#0230  {
#0231         // The document has informed this view that some data has changed.
#0232
#0233         if (pHint != NULL)
#0234         {
#0235                if (pHint->IsKindOf(RUNTIME_CLASS(CStroke)))
#0236                {
#0237                       // The hint is that a stroke as been added (or changed).
#0238                       // So, invalidate its rectangle.
#0239                       CStroke* pStroke = (CStroke*)pHint;
#0240                       CClientDC dc(this);
#0241                       OnPrepareDC(&dc);
#0242                       CRect rectInvalid = pStroke->GetBoundingRect();
#0243                       dc.LPtoDP(&rectInvalid);
#0244                       InvalidateRect(&rectInvalid);
#0245                       return;
#0246                }
#0247         }
#0248         // We can't interpret the hint, so assume that anything might
#0249         // have been updated.
#0250         Invalidate(TRUE);
#0251         return;
#0252  }
#0253
#0254  void CScribbleView::OnInitialUpdate()
#0255  {
#0256         SetScrollSizes(MM_LOENGLISH, GetDocument()->GetDocSize());
#0257         CScrollView::OnInitialUpdate();
#0258  }
#0259
```

```
#0260  void CScribbleView::OnPrint(CDC* pDC, CPrintInfo* pInfo)
#0261  {
#0262         if (pInfo->m_nCurPage == 1)  // page no. 1 is the title page
#0263         {
#0264                PrintTitlePage(pDC, pInfo);
#0265                return; // nothing else to print on page 1 but the page title
#0266         }
#0267         CString strHeader = GetDocument()->GetTitle();
#0268
#0269         PrintPageHeader(pDC, pInfo, strHeader);
#0270         // PrintPageHeader() subtracts out from the pInfo->m_rectDraw the
#0271         // amount of the page used for the header.
#0272
#0273         pDC->SetWindowOrg(pInfo->m_rectDraw.left,-pInfo->m_rectDraw.top);
#0274
#0275         // Now print the rest of the page
#0276         OnDraw(pDC);
#0277  }
#0278
#0279  void CScribbleView::PrintTitlePage(CDC* pDC, CPrintInfo* pInfo)
#0280  {
#0281         // Prepare a font size for displaying the file name
#0282         LOGFONT logFont;
#0283         memset(&logFont, 0, sizeof(LOGFONT));
#0284         logFont.lfHeight = 75;  //  3/4th inch high in MM_LOENGLISH
#0285                                                 // (1/100th inch)
#0286         CFont font;
#0287         CFont* pOldFont = NULL;
#0288         if (font.CreateFontIndirect(&logFont))
#0289                pOldFont = pDC->SelectObject(&font);
#0290
#0291         // Get the file name, to be displayed on title page
#0292         CString strPageTitle = GetDocument()->GetTitle();
#0293
#0294         // Display the file name 1 inch below top of the page,
#0295         // centered horizontally
#0296         pDC->SetTextAlign(TA_CENTER);
#0297         pDC->TextOut(pInfo->m_rectDraw.right/2, -100, strPageTitle);
#0298
#0299         if (pOldFont != NULL)
#0300                pDC->SelectObject(pOldFont);
#0301  }
#0302
#0303  void CScribbleView::PrintPageHeader(CDC* pDC, CPrintInfo* pInfo,
#0304         CString& strHeader)
#0305  {
```

```
#0306         // Print a page header consisting of the name of
#0307         // the document and a horizontal line
#0308         pDC->SetTextAlign(TA_LEFT);
#0309         pDC->TextOut(0,-25, strHeader);  // 1/4 inch down
#0310
#0311         // Draw a line across the page, below the header
#0312         TEXTMETRIC textMetric;
#0313         pDC->GetTextMetrics(&textMetric);
#0314         int y = -35 - textMetric.tmHeight;       // line 1/10th inch below text
#0315         pDC->MoveTo(0, y);                      // from left margin
#0316         pDC->LineTo(pInfo->m_rectDraw.right, y);    // to right margin
#0317
#0318         // Subtract out from the drawing rectange the space used by the header.
#0319         y -= 25;    // space 1/4 inch below (top of) line
#0320         pInfo->m_rectDraw.top += y;
#0321  }
```

PENDLG.H

```
#0001  class CPenWidthsDlg : public CDialog
#0002  {
#0003  // Construction
#0004  public:
#0005         CPenWidthsDlg(CWnd* pParent = NULL);   // standard constructor
#0006
#0007  // Dialog Data
#0008         //{{AFX_DATA(CPenWidthsDlg)
#0009         enum { IDD = IDD_PEN_WIDTHS };
#0010         int           m_nThinWidth;
#0011         int           m_nThickWidth;
#0012         //}}AFX_DATA
#0013
#0014
#0015  // Overrides
#0016         // ClassWizard generated virtual function overrides
#0017         //{{AFX_VIRTUAL(CPenWidthsDlg)
#0018         protected:
#0019         virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
#0020         //}}AFX_VIRTUAL
#0021
#0022  // Implementation
#0023  protected:
#0024
#0025         // Generated message map functions
#0026         //{{AFX_MSG(CPenWidthsDlg)
#0027         afx_msg void OnDefaultPenWidths();
```

```
#0028         //}}AFX_MSG
#0029         DECLARE_MESSAGE_MAP()
#0030 };
```

PENDLG.CPP

```
#0001 #include "stdafx.h"
#0002 #include "Scribble.h"
#0003 #include "PenDlg.h"
#0004
#0005 #ifdef _DEBUG
#0006 #undef THIS_FILE
#0007 static char THIS_FILE[] = __FILE__;
#0008 #endif
#0009
#0010 /////////////////////////////////////////////////////////////////
#0011 // CPenWidthsDlg dialog
#0012
#0013
#0014 CPenWidthsDlg::CPenWidthsDlg(CWnd* pParent /*=NULL*/)
#0015         : CDialog(CPenWidthsDlg::IDD, pParent)
#0016 {
#0017         //{{AFX_DATA_INIT(CPenWidthsDlg)
#0018         m_nThinWidth = 0;
#0019         m_nThickWidth = 0;
#0020         //}}AFX_DATA_INIT
#0021 }
#0022
#0023
#0024 void CPenWidthsDlg::DoDataExchange(CDataExchange* pDX)
#0025 {
#0026         CDialog::DoDataExchange(pDX);
#0027         //{{AFX_DATA_MAP(CPenWidthsDlg)
#0028         DDX_Text(pDX, IDC_THIN_PEN_WIDTH, m_nThinWidth);
#0029         DDV_MinMaxInt(pDX, m_nThinWidth, 1, 20);
#0030         DDX_Text(pDX, IDC_THICK_PEN_WIDTH, m_nThickWidth);
#0031         DDV_MinMaxInt(pDX, m_nThickWidth, 1, 20);
#0032         //}}AFX_DATA_MAP
#0033 }
#0034
#0035
#0036 BEGIN_MESSAGE_MAP(CPenWidthsDlg, CDialog)
#0037         //{{AFX_MSG_MAP(CPenWidthsDlg)
#0038         ON_BN_CLICKED(IDC_DEFAULT_PEN_WIDTHS, OnDefaultPenWidths)
#0039         //}}AFX_MSG_MAP
#0040 END_MESSAGE_MAP()
```

```
#0041
#0042 /////////////////////////////////////////////////////////////
#0043 // CPenWidthsDlg message handlers
#0044
#0045 void CPenWidthsDlg::OnDefaultPenWidths()
#0046 {
#0047        m_nThinWidth = 2;
#0048        m_nThickWidth = 5;
#0049        UpdateData(FALSE);  // causes DoDataExchange()
#0050               // bSave=FALSE means don't save from screen,
#0051               // rather, write to screen
#0052 }
```

STDAFX.H

```
#0001 #include <afxwin.h>        // MFC core and standard components
#0002 #include <afxext.h>        // MFC extensions
#0003 #include <afxtempl.h>      // MFC templates
#0004
#0005 #ifndef _AFX_NO_AFXCMN_SUPPORT
#0006 #include <afxcmn.h>     // MFC support for Windows 95 Common Controls
#0007 #endif // _AFX_NO_AFXCMN_SUPPORT
```

STDAFX.CPP

```
#0001 #include "stdafx.h"
```

RESOURCE.H

```
#0001 //{{NO_DEPENDENCIES}}
#0002 // Microsoft Visual C++ generated include file.
#0003 // Used by SCRIBBLE.RC
#0004 //
#0005 #define IDD_ABOUTBOX                100
#0006 #define IDR_MAINFRAME               128
#0007 #define IDR_SCRIBBTYPE              129
#0008 #define IDD_PEN_WIDTHS              131
#0009 #define IDC_THIN_PEN_WIDTH           1000
#0010 #define IDC_THICK_PEN_WIDTH          1001
#0011 #define IDC_DEFAULT_PEN_WIDTHS       1002
#0012 #define ID_PEN_THICK_OR_THIN         32772
#0013 #define ID_PEN_WIDTHS               32773
#0014
#0015 // Next default values for new objects
#0016 //
```

```
#0017  #ifdef APSTUDIO_INVOKED
#0018  #ifndef APSTUDIO_READONLY_SYMBOLS
#0019  #define _APS_3D_CONTROLS             1
#0020  #define _APS_NEXT_RESOURCE_VALUE      132
#0021  #define _APS_NEXT_COMMAND_VALUE       32774
#0022  #define _APS_NEXT_CONTROL_VALUE       1003
#0023  #define _APS_NEXT_SYMED_VALUE         101
#0024  #endif
#0025  #endif
```

SCRIBBLE.RC

```
#0001  //Microsoft Developer Studio generated resource script.
#0002  //
#0003  #include "resource.h"
#0004
#0005  #define APSTUDIO_READONLY_SYMBOLS
#0006  /////////////////////////////////////////////////////////////////////
#0007  //
#0008  // Generated from the TEXTINCLUDE 2 resource.
#0009  //
#0010  #include "afxres.h"
#0011
#0012  /////////////////////////////////////////////////////////////////////
#0013  #undef APSTUDIO_READONLY_SYMBOLS
#0014
#0015  /////////////////////////////////////////////////////////////////////
#0016  // English (U.S.) resources
#0017
#0018  #if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#0019  #ifdef _WIN32
#0020  LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#0021  #pragma code_page(1252)
#0022  #endif //_WIN32
#0023
#0024  #ifdef APSTUDIO_INVOKED
#0025  /////////////////////////////////////////////////////////////////////
#0026  //
#0027  // TEXTINCLUDE
#0028  //
#0029
#0030  1 TEXTINCLUDE DISCARDABLE
#0031  BEGIN
#0032      "resource.h\0"
#0033  END
#0034
```

```
#0035  2 TEXTINCLUDE DISCARDABLE
#0036  BEGIN
#0037      "#include ""afxres.h""\r\n"
#0038      "\0"
#0039  END
#0040
#0041  3 TEXTINCLUDE DISCARDABLE
#0042  BEGIN
#0043      "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
#0044      "#define _AFX_NO_OLE_RESOURCES\r\n"
#0045      "#define _AFX_NO_TRACKER_RESOURCES\r\n"
#0046      "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
#0047      "\r\n"
#0048      "#include ""res\\Scribble.rc2""  // non-MS VC++ edited resources\r\n"
#0049      "#include ""afxres.rc""         // Standard components\r\n"
#0050      "#include ""afxprint.rc""       // printing/print preview resources\r\n"
#0051      "\0"
#0052  END
#0053
#0054  #endif    // APSTUDIO_INVOKED
#0055
#0056
#0057  /////////////////////////////////////////////////////////////////////
#0058  //
#0059  // Icon
#0060  //
#0061
#0062  // Icon with lowest ID value placed first to ensure application icon
#0063  // remains consistent on all systems.
#0064  IDR_MAINFRAME          ICON    DISCARDABLE     "res\\Scribble.ico"
#0065  IDR_SCRIBBTYPE         ICON    DISCARDABLE     "res\\ScribDoc.ico"
#0066
#0067  /////////////////////////////////////////////////////////////////////
#0068  //
#0069  // Bitmap
#0070  //
#0071
#0072  IDR_MAINFRAME          BITMAP  MOVEABLE PURE   "res\\Toolbar.bmp"
#0073
#0074  /////////////////////////////////////////////////////////////////////
#0075  //
#0076  // Toolbar
#0077  //
#0078
#0079  IDR_MAINFRAME TOOLBAR DISCARDABLE  16, 15
#0080  BEGIN
```

```
#0081     BUTTON     ID_FILE_NEW
#0082     BUTTON     ID_FILE_OPEN
#0083     BUTTON     ID_FILE_SAVE
#0084     SEPARATOR
#0085     BUTTON     ID_PEN_THICK_OR_THIN
#0086     SEPARATOR
#0087     BUTTON     ID_FILE_PRINT
#0088     BUTTON     ID_APP_ABOUT
#0089  END
#0090
#0091
#0092  //////////////////////////////////////////////////////////////////
#0093  //
#0094  // Menu
#0095  //
#0096
#0097  IDR_MAINFRAME MENU PRELOAD DISCARDABLE
#0098  BEGIN
#0099     POPUP "&File"
#0100     BEGIN
#0101        MENUITEM "&New\tCtrl+N",       ID_FILE_NEW
#0102        MENUITEM "&Open...\tCtrl+O",    ID_FILE_OPEN
#0103        MENUITEM SEPARATOR
#0104        MENUITEM "P&rint Setup...",     ID_FILE_PRINT_SETUP
#0105        MENUITEM SEPARATOR
#0106        MENUITEM "Recent File",        ID_FILE_MRU_FILE1, GRAYED
#0107        MENUITEM SEPARATOR
#0108        MENUITEM "E&xit",             ID_APP_EXIT
#0109     END
#0110     POPUP "&View"
#0111     BEGIN
#0112        MENUITEM "&Toolbar",          ID_VIEW_TOOLBAR
#0113        MENUITEM "&Status Bar",        ID_VIEW_STATUS_BAR
#0114     END
#0115     POPUP "&Help"
#0116     BEGIN
#0117        MENUITEM "&About Scribble...",  ID_APP_ABOUT
#0118     END
#0119  END
#0120
#0121  IDR_SCRIBBTYPE MENU PRELOAD DISCARDABLE
#0122  BEGIN
#0123     POPUP "&File"
#0124     BEGIN
#0125        MENUITEM "&New\tCtrl+N",       ID_FILE_NEW
#0126        MENUITEM "&Open...\tCtrl+O",    ID_FILE_OPEN
```

```
#0127        MENUITEM "&Close",            ID_FILE_CLOSE
#0128        MENUITEM "&Save\tCtrl+S",      ID_FILE_SAVE
#0129        MENUITEM "Save &As...",        ID_FILE_SAVE_AS
#0130        MENUITEM SEPARATOR
#0131        MENUITEM "&Print...\tCtrl+P",   ID_FILE_PRINT
#0132        MENUITEM "Print Pre&view",     ID_FILE_PRINT_PREVIEW
#0133        MENUITEM "P&rint Setup...",    ID_FILE_PRINT_SETUP
#0134        MENUITEM SEPARATOR
#0135        MENUITEM "Sen&d...",          ID_FILE_SEND_MAIL
#0136        MENUITEM SEPARATOR
#0137        MENUITEM "Recent File",        ID_FILE_MRU_FILE1, GRAYED
#0138        MENUITEM SEPARATOR
#0139        MENUITEM "E&xit",             ID_APP_EXIT
#0140    END
#0141    POPUP "&Edit"
#0142    BEGIN
#0143        MENUITEM "&Undo\tCtrl+Z",      ID_EDIT_UNDO
#0144        MENUITEM SEPARATOR
#0145        MENUITEM "Cu&t\tCtrl+X",       ID_EDIT_CUT
#0146        MENUITEM "&Copy\tCtrl+C",      ID_EDIT_COPY
#0147        MENUITEM "&Paste\tCtrl+V",     ID_EDIT_PASTE
#0148        MENUITEM "Clear &All",         ID_EDIT_CLEAR_ALL
#0149    END
#0150    POPUP "&Pen"
#0151    BEGIN
#0152        MENUITEM "Thick &Line",        ID_PEN_THICK_OR_THIN
#0153        MENUITEM "Pen &Widths...",     ID_PEN_WIDTHS
#0154    END
#0155    POPUP "&View"
#0156    BEGIN
#0157        MENUITEM "&Toolbar",          ID_VIEW_TOOLBAR
#0158        MENUITEM "&Status Bar",        ID_VIEW_STATUS_BAR
#0159    END
#0160    POPUP "&Window"
#0161    BEGIN
#0162        MENUITEM "&New Window",        ID_WINDOW_NEW
#0163        MENUITEM "&Cascade",          ID_WINDOW_CASCADE
#0164        MENUITEM "&Tile",             ID_WINDOW_TILE_HORZ
#0165        MENUITEM "&Arrange Icons",     ID_WINDOW_ARRANGE
#0166    END
#0167    POPUP "&Help"
#0168    BEGIN
#0169        MENUITEM "&About Scribble...",  ID_APP_ABOUT
#0170    END
#0171 END
#0172
```

```
#0173
#0174 //////////////////////////////////////////////////////////////////
#0175 //
#0176 // Accelerator
#0177 //
#0178
#0179 IDR_MAINFRAME ACCELERATORS PRELOAD MOVEABLE PURE
#0180 BEGIN
#0181     "N",           ID_FILE_NEW,          VIRTKEY, CONTROL
#0182     "O",           ID_FILE_OPEN,         VIRTKEY, CONTROL
#0183     "S",           ID_FILE_SAVE,         VIRTKEY, CONTROL
#0184     "P",           ID_FILE_PRINT,        VIRTKEY, CONTROL
#0185     "Z",           ID_EDIT_UNDO,         VIRTKEY, CONTROL
#0186     "X",           ID_EDIT_CUT,          VIRTKEY, CONTROL
#0187     "C",           ID_EDIT_COPY,         VIRTKEY, CONTROL
#0188     "V",           ID_EDIT_PASTE,        VIRTKEY, CONTROL
#0189     VK_BACK,       ID_EDIT_UNDO,         VIRTKEY, ALT
#0190     VK_DELETE,     ID_EDIT_CUT,          VIRTKEY, SHIFT
#0191     VK_INSERT,     ID_EDIT_COPY,         VIRTKEY, CONTROL
#0192     VK_INSERT,     ID_EDIT_PASTE,        VIRTKEY, SHIFT
#0193     VK_F6,         ID_NEXT_PANE,         VIRTKEY
#0194     VK_F6,         ID_PREV_PANE,         VIRTKEY, SHIFT
#0195 END
#0196
#0197
#0198 //////////////////////////////////////////////////////////////////
#0199 //
#0200 // Dialog
#0201 //
#0202
#0203 IDD_ABOUTBOX DIALOG DISCARDABLE  0, 0, 217, 55
#0204 STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
#0205 CAPTION "About Scribble"
#0206 FONT 8, "MS Sans Serif"
#0207 BEGIN
#0208     ICON           IDR_MAINFRAME,IDC_STATIC,11,17,20,20
#0209     LTEXT          "Scribble Version 1.0",IDC_STATIC,40,10,119,8
#0210     LTEXT          "Copyright * 1995",IDC_STATIC,40,25,119,8
#0211     DEFPUSHBUTTON   "OK",IDOK,178,7,32,14,WS_GROUP
#0212 END
#0213
#0214 IDD_PEN_WIDTHS DIALOG DISCARDABLE  0, 0, 203, 65
#0215 STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
#0216 CAPTION "Pen Widths"
#0217 FONT 8, "MS Sans Serif"
#0218 BEGIN
```

```
#0219     DEFPUSHBUTTON   "OK",IDOK,148,7,50,14
#0220     PUSHBUTTON      "Cancel",IDCANCEL,148,24,50,14
#0221     PUSHBUTTON      "Default",IDC_DEFAULT_PEN_WIDTHS,148,41,50,14
#0222     LTEXT          "Thin Pen Width:",IDC_STATIC,10,12,70,10
#0223     LTEXT          "Thick Pen Width:",IDC_STATIC,10,33,70,10
#0224     EDITTEXT       IDC_THIN_PEN_WIDTH,86,12,40,13,ES_AUTOHSCROLL
#0225     EDITTEXT       IDC_THICK_PEN_WIDTH,86,33,40,13,ES_AUTOHSCROLL
#0226 END
#0227
#0228
#0229 #ifndef _MAC
#0230 /////////////////////////////////////////////////////////////////////
#0231 //
#0232 // Version
#0233 //
#0234
#0235 VS_VERSION_INFO VERSIONINFO
#0236  FILEVERSION 1,0,0,1
#0237  PRODUCTVERSION 1,0,0,1
#0238  FILEFLAGSMASK 0x3fL
#0239 #ifdef _DEBUG
#0240  FILEFLAGS 0x1L
#0241 #else
#0242  FILEFLAGS 0x0L
#0243 #endif
#0244  FILEOS 0x4L
#0245  FILETYPE 0x1L
#0246  FILESUBTYPE 0x0L
#0247 BEGIN
#0248     BLOCK "StringFileInfo"
#0249     BEGIN
#0250         BLOCK "040904B0"
#0251         BEGIN
#0252             VALUE "CompanyName", "\0"
#0253             VALUE "FileDescription", "SCRIBBLE MFC Application\0"
#0254             VALUE "FileVersion", "1, 0, 0, 1\0"
#0255             VALUE "InternalName", "SCRIBBLE\0"
#0256             VALUE "LegalCopyright", "Copyright * 1995\0"
#0257             VALUE "LegalTrademarks", "\0"
#0258             VALUE "OriginalFilename", "SCRIBBLE.EXE\0"
#0259             VALUE "ProductName", "SCRIBBLE Application\0"
#0260             VALUE "ProductVersion", "1, 0, 0, 1\0"
#0261         END
#0262     END
#0263     BLOCK "VarFileInfo"
#0264     BEGIN
```

```
#0265         VALUE "Translation", 0x409, 1200
#0266     END
#0267 END
#0268
#0269 #endif    // !_MAC
#0270
#0271
#0272 /////////////////////////////////////////////////////////////////
#0273 //
#0274 // DESIGNINFO
#0275 //
#0276
#0277 #ifdef APSTUDIO_INVOKED
#0278 GUIDELINES DESIGNINFO DISCARDABLE
#0279 BEGIN
#0280     IDD_ABOUTBOX, DIALOG
#0281     BEGIN
#0282         LEFTMARGIN, 7
#0283         RIGHTMARGIN, 210
#0284         TOPMARGIN, 7
#0285         BOTTOMMARGIN, 48
#0286     END
#0287
#0288     IDD_PEN_WIDTHS, DIALOG
#0289     BEGIN
#0290         LEFTMARGIN, 10
#0291         RIGHTMARGIN, 198
#0292         VERTGUIDE, 1
#0293         TOPMARGIN, 7
#0294         BOTTOMMARGIN, 55
#0295     END
#0296 END
#0297 #endif    // APSTUDIO_INVOKED
#0298
#0299
#0300 /////////////////////////////////////////////////////////////////
#0301 //
#0302 // String Table
#0303 //
#0304
#0305 STRINGTABLE PRELOAD DISCARDABLE
#0306 BEGIN
#0307     IDR_MAINFRAME   "Scribble Step5"
#0308     IDR_SCRIBBTYPE  "\nScribb\nScribb\nScribble Files
(*.scb)\n.SCB\nScribble.Document\nScribb Document"
#0309 END
```

```
#0310
#0311  STRINGTABLE PRELOAD DISCARDABLE
#0312  BEGIN
#0313      AFX_IDS_APP_TITLE      "Scribble"
#0314      AFX_IDS_IDLEMESSAGE    "Ready"
#0315  END
#0316
#0317  STRINGTABLE DISCARDABLE
#0318  BEGIN
#0319      ID_INDICATOR_EXT       "EXT"
#0320      ID_INDICATOR_CAPS      "CAP"
#0321      ID_INDICATOR_NUM       "NUM"
#0322      ID_INDICATOR_SCRL      "SCRL"
#0323      ID_INDICATOR_OVR       "OVR"
#0324      ID_INDICATOR_REC       "REC"
#0325  END
#0326
#0327  STRINGTABLE DISCARDABLE
#0328  BEGIN
#0329    ID_FILE_NEW         "Create a new document\nNew"
#0330    ID_FILE_OPEN        "Open an existing document\nOpen"
#0331    ID_FILE_CLOSE       "Close the active document\nClose"
#0332    ID_FILE_SAVE        "Save the active document\nSave"
#0333    ID_FILE_SAVE_AS     "Save the active document with a new name\nSave As"
#0334    ID_FILE_PAGE_SETUP   "Change the printing options\nPage Setup"
#0335    ID_FILE_PRINT_SETUP  "Change the printer and printing options\nPrint Setup"
#0336    ID_FILE_PRINT       "Print the active document\nPrint"
#0337    ID_FILE_PRINT_PREVIEW "Display full pages\nPrint Preview"
#0338    ID_FILE_SEND_MAIL    "Send the active document through electronic
mail\nSend Mail"
#0339  END
#0340
#0341  STRINGTABLE DISCARDABLE
#0342  BEGIN
#0343    ID_APP_ABOUT  "Display program information, version No. and copyright\nAbout"
#0344    ID_APP_EXIT   "Quit the application; prompts to save documents\nExit"
#0345  END
#0346
#0347  STRINGTABLE DISCARDABLE
#0348  BEGIN
#0349      ID_FILE_MRU_FILE1    "Open this document"
#0350      ID_FILE_MRU_FILE2    "Open this document"
#0351      ID_FILE_MRU_FILE3    "Open this document"
#0352      ID_FILE_MRU_FILE4    "Open this document"
#0353  END
#0354
```

```
#0355  STRINGTABLE DISCARDABLE
#0356  BEGIN
#0357      ID_NEXT_PANE         "Switch to the next window pane\nNext Pane"
#0358      ID_PREV_PANE         "Switch back to the previous window pane\nPrevious Pane"
#0359  END
#0360
#0361  STRINGTABLE DISCARDABLE
#0362  BEGIN
#0363    ID_WINDOW_NEW     "Open another window for the active document\nNew Window"
#0364    ID_WINDOW_ARRANGE "Arrange icons at the bottom of the window\nArrange Icons"
#0365    ID_WINDOW_CASCADE "Arrange windows so they overlap\nCascade Windows"
#0366    ID_WINDOW_TILE_HORZ  "Arrange windows as non-overlapping tiles\nTile Windows"
#0367    ID_WINDOW_TILE_VERT  "Arrange windows as non-overlapping tiles\nTile Windows"
#0368    ID_WINDOW_SPLIT      "Split the active window into panes\nSplit"
#0369  END
#0370
#0371  STRINGTABLE DISCARDABLE
#0372  BEGIN
#0373      ID_EDIT_CLEAR        "Erase the selection\nErase"
#0374      ID_EDIT_CLEAR_ALL     "Clears the drawing"
#0375      ID_EDIT_COPY         "Copy the selection and put it on the Clipboard\nCopy"
#0376      ID_EDIT_CUT          "Cut the selection and put it on the Clipboard\nCut"
#0377      ID_EDIT_FIND         "Find the specified text\nFind"
#0378      ID_EDIT_PASTE        "Insert Clipboard contents\nPaste"
#0379      ID_EDIT_REPEAT       "Repeat the last action\nRepeat"
#0380      ID_EDIT_REPLACE      "Replace specific text with different text\nReplace"
#0381      ID_EDIT_SELECT_ALL    "Select the entire document\nSelect All"
#0382      ID_EDIT_UNDO         "Undo the last action\nUndo"
#0383      ID_EDIT_REDO         "Redo the previously undone action\nRedo"
#0384  END
#0385
#0386  STRINGTABLE DISCARDABLE
#0387  BEGIN
#0388      ID_VIEW_TOOLBAR      "Show or hide the toolbar\nToggle ToolBar"
#0389      ID_VIEW_STATUS_BAR    "Show or hide the status bar\nToggle StatusBar"
#0390  END
#0391
#0392  STRINGTABLE DISCARDABLE
#0393  BEGIN
#0394      AFX_IDS_SCSIZE       "Change the window size"
#0395      AFX_IDS_SCMOVE       "Change the window position"
#0396      AFX_IDS_SCMINIMIZE    "Reduce the window to an icon"
#0397      AFX_IDS_SCMAXIMIZE    "Enlarge the window to full size"
#0398      AFX_IDS_SCNEXTWINDOW  "Switch to the next document window"
#0399      AFX_IDS_SCPREVWINDOW  "Switch to the previous document window"
#0400      AFX_IDS_SCCLOSE  "Close the active window and prompts to save the documents"
```

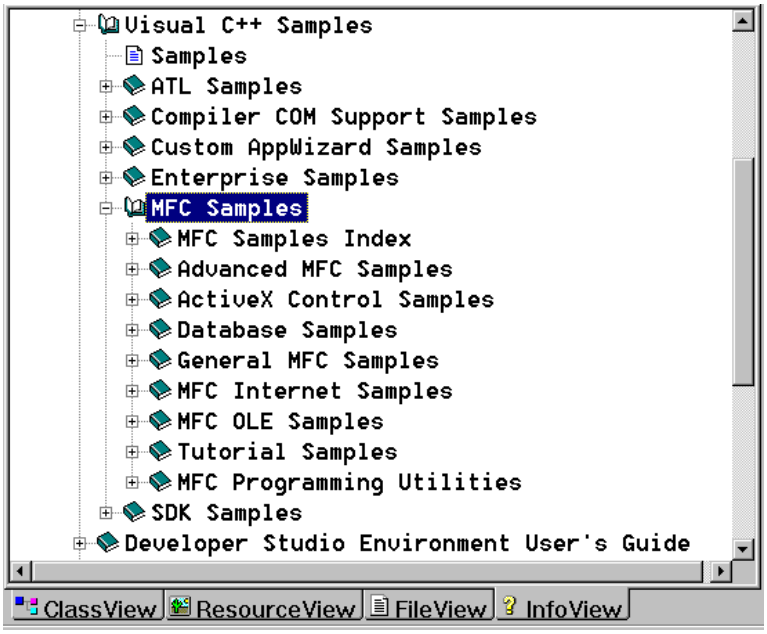
```
#0401  END
#0402
#0403  STRINGTABLE DISCARDABLE
#0404  BEGIN
#0405      AFX_IDS_SCRESTORE     "Restore the window to normal size"
#0406      AFX_IDS_SCTASKLIST    "Activate Task List"
#0407      AFX_IDS_MDICHILD      "Activate this window"
#0408  END
#0409
#0410  STRINGTABLE DISCARDABLE
#0411  BEGIN
#0412      AFX_IDS_PREVIEW_CLOSE "Close print preview mode\nCancel Preview"
#0413  END
#0414
#0415  STRINGTABLE DISCARDABLE
#0416  BEGIN
#0417      AFX_IDS_DESKACCESSORY "Opens the selected item"
#0418  END
#0419
#0420  STRINGTABLE DISCARDABLE
#0421  BEGIN
#0422      ID_PEN_THICK_OR_THIN  "Toggles the line thickness between thin and
thick\nToggle pen"
#0423      ID_PEN_WIDTHS        "Sets the size of the thin and thick pen"
#0424  END
#0425
#0426  #endif    // English (U.S.) resources
#0427  /////////////////////////////////////////////////////////////////////
#0428
#0429
#0430
#0431  #ifndef APSTUDIO_INVOKED
#0432  /////////////////////////////////////////////////////////////////////
#0433  //
#0434  // Generated from the TEXTINCLUDE 3 resource.
#0435  //
#0436  #define _AFX_NO_SPLITTER_RESOURCES
#0437  #define _AFX_NO_OLE_RESOURCES
#0438  #define _AFX_NO_TRACKER_RESOURCES
#0439  #define _AFX_NO_PROPERTY_RESOURCES
#0440
#0441  #include "res\Scribble.rc2"  // non-Microsoft Visual C++ edited resources
#0442  #include "afxres.rc"        // Standard components
#0443  #include "afxprint.rc"      // printing/print preview resources
#0444
#0445  #endif    // not APSTUDIO_INVOKED
```

「附錄 C

Visual C++ 5.0
MFC 範例程式一覽

經過整本書的鍛鍊，我想你已經對於整個 MFC 的架構有了相當紮實的瞭解，對於我所謂的「程式設計主軸」已經能夠掌握。接下來，就是學習十八般武藝的 MFC classes。

Visual C++ 附有極爲豐富的範例程式，包括各種主題如下：



其中使用 MFC 來設計程式的例子極多（其他部份使用 SDK 工具），是一個大寶庫。我把所有以 MFC 開發的範例程式以字母爲序，列於下表供你參考。

程式名稱	說明
ACDUAL	Demonstrates how to add dual interface support to an MFC-based Automation server.
AUTOCLIK	Tutorial example illustrating Automation features in Visual C++ Tutorials.
AUTODRIV	A simple Automation client application that drives the AUTOCLIK tutorial sample application.
BINDENRL	Databound controls in a dialog-based application with property pages.
BINDSCRB	Illustration of the use of new COM interfaces to components currently supported by the Microsoft Office suite of products.
CALCDRIV	Automation client.
CATALOG	Illustration of direct calls to ODBC functions in general, and the ODBC functions SQLTables and SQLColumns in particular.
CATALOG2	Illustration of direct calls to ODBC functions in general using Windows Common Controls.
CHATSRVR	Discussion server application for CHATTER.
CHATTER	Client application that uses Windows Sockets.
CHKBOOK	Record-based (nonserialized) document.
CIRC	Tutorial sample that teaches you how to create a simple ActiveX control called Circle.
CMNCTRLS	Demonstrates how to create and change the styles of 7 of the Windows Common Controls.
COLLECT	MFC C++ template-based collection classes, and standard prebuilt collection classes.
CONTAINER	Tutorial example illustrating ActiveX Visual Editing container features in Visual C++ Tutorials.
COUNTER	Using an ISAPI DLL to send image data (rather than HTML data) back to a Web browser.
CTRLBARS	Custom toolbar and status bar, dialog bar, and floating palette.

程式名稱	說明
CTRLTEST	Owner-draw list box and menu, custom control, bitmap button, spin control.
CUBE	OpenGL application using MFC device contexts along with OpenGL's resource contexts.
DAOCTL	DAO database class functionality and ActiveX controls let you examine a database.
DAOENROL	Based on ENROLL, but migrated to the DAO database classes. Also serves as Step 4 of the DaoEnrol tutorial.
DAOTABLE	Creates a Microsoft Access database (.MDB file) and its tables, fields, queries, and indexes using MFC DAO database classes.
DAOVIEW	DAO database classes and Windows Common Controls let you view the schema of a database.
DBFETCH	Demonstrates the use of bulk row fetching in the ODBC database classes.
DIBLOOK	Device-independent bitmap and color palette.
DLGCBR32	Adding a toolbar and a status bar to a dialog-based application.
DLGTEMPL	Shows how to create dialog templates dynamically.
DLLHUSK	Sharing the DLL version of the Foundation class library with an application and custom DLL.
DLLTRACE	Statically linking the MFC library to a custom DLL.
DOCKTOOL	Dragging and floating toolbars that are "dockable".
DRAWCLI	Full-featured object-oriented drawing application that is also an ActiveX Visual Editing container.
DRAWPIC	Getting a Windows handle to a bitmap or icon from an LPPICTUREDISP.
DYNABIND	Dynamic binding of database columns to a recordset.
DYNAMENU	Dynamically modifying list of items in menus; handling commands not known at compile time; and updating the status bar command prompt for such commands.
ENROLL	Tutorial example illustrating database features in Visual C++ Tutorials.

程式名稱	說明
EXTBIND	Shows how to bind data-aware controls across a dialog box boundary.
FIRE	Dialog-based application that demonstrates five Windows Common Controls.
FTPTREE	Displays the contents of an FTP site in a tree control.
GUIDGEN	A dialog-based MFC application used to generate globally unique identifiers, or GUIDs, which identify OLE classes, objects, and interfaces.
HELLO	Simple application with frame window but no document or view.
HELLOAPP	Minimal "Hello World" application.
HIERSVR	ActiveX Visual Editing server application with drag and drop.
HTTPSVR	Uses MFC Windows Socket classes to implement an Internet HTTP server.
IMAGE	Demonstrates an ActiveX control that is capable of downloading data asynchronously.
INPROC	An in-process Automation server that can be loaded as a DLL in the client's address space.
IPDRIVE	A simple Automation client application that drives the INPROC sample application.
MAKEHM	Command line utility for associating resources with Help contexts.
MDI	MDI application that does not use documents and views.
MDIBIND	Demonstrates data binding in *CWnd*-derived windows, but only at run time.
MDIDOCVW	New version of the MDI sample that uses the document/view architecture.
MFCCALC	An Automation server that implements a simple calculator.
MFCUCASE	Demonstrates MFC support for Internet Server filter DLLs.
MODELESS	Demonstrates the use of an MFC *CDialog* object as a modeless dialog.
MTGDI	Multithread illustration, where GDI resources are converted to MFC objects and back.
MTMDI	Multithread illustration, where user-interface events are processed in a separate user-interface thread.

程式名稱	說明
MTRECALC	Multithread illustration, where recalculations are done in a worker thread.
MULTIPAD	Simple MDI text editor using the *CEditView* class.
MUTEXES	Demonstrates the use of the *Cmutex* class for multithreading.
NPP	Editor demonstrating toolbars, status bars, and other controls.
OCLIENT	ActiveX Visual Editing container application, with drag and drop.
ODBCINFO	Shows how to determine various ODBC driver capabilities at run time.
OLEVIEW	Implementing an OLE object browser through custom OLE interfaces.
PROPDLG	Property sheets (dialogs).
ROWLIST	Illustrates full row selection in a list-view common control.
SAVER	Screen saver written with MFC.
SCRIBBLE	Main tutorial example in Visual C++ Tutorials.
SMILEY	Modifying properties, calling methods, and handling events from the SMILE control in the SMILEY container.
SNAPVW	Shows how to use property pages in a MDI child frame window.
SPEAKN	Multimedia sound using user-defined resources.
STDREG	Tool for populating the Student Registration database (used by the Enroll database tutorial) in any format supported by an ODBC driver. Illustrates SQL table creation.
SUPERPAD	ActiveX Visual Editing server that edits text using *CEditView*.
TEAR	MFC console application that uses the WININET.DLL.
TEMPLDEF	Command line tool that expands source files similar to C++ templates.
TESTHELP	An ActiveX control that has its own help file and tooltips.
TRACER	Tool that sets the Foundation class application trace flags kept in AFX.INI.
TRACKER	Illustration of the various *CRectTracker* styles and options.

程式名稱	說明
VCTERM	Simple terminal emulation program illustrating the use of the MSCOMM32.OCX ActiveX control.
VIEWEX	Multiple views, scroll view, splitter windows.
WORDPAD	Uses MFC's support for rich edit controls to create a basic word processor.
WWWQUOTE	Retrieves information from a database and provides it to the user via an HTTP connection to the server.

附錄 D

以 MFC 重建 DBWIN

沒有 DBWIN，TRACE 唱什麼獨角戲？Visual C++ 的 TRACE 字串輸出必須在整合環境的除錯視窗中看到。這太過份了。我們只不過想在射擊遊樂場玩玩，微軟卻要我們扛一尊 155 加農砲。

侯俊傑 / 1997.01 發表於 Run!PC 雜誌

我自己常喜歡在程式加個 *printf*，觀察程式的流程或變數的內容。簡簡單單，不必驚動除錯器之類的大傢伙。

printf、AfxMessageBox 與 TRACE

printf 是 C 語言的標準函式，但只能在文字模式（text mode）下執行。Windows 圖形介面中差可比擬的是 *AfxMessageBox*，可以輕輕鬆鬆地在對話盒中列印出你的字串。然而 *AfxMessageBox* 有個缺點：它會打斷程式的行進，非得使用者按下【OK】按鈕不可。再者，如果 *AfxMessageBox* 對話盒畫面侵佔了原程式的視窗畫面，一旦對話盒結束，系統會對程式視窗發出原本不必要的 *WM_PAINT*，於是 View 類別的 *OnDraw* 會被呼叫。如果你的 *AfxMessageBox* 正是放在 *OnDraw* 函式中，這惡性循環可有得瞧了。

TRACE 沒有這種缺點，它的輸出集中到一個特定視窗中，它不會打斷你測試程式的雅

興，不會在你全神貫注思考程式邏輯時還要你分神來按【OK】鈕。更重要的是，它不會干擾到程式的視窗畫面。

Visual C++ 1.5 時代，*TRACE* 的字串輸出是送到一個名爲 DBWIN 的視窗上。你可以一邊執行程式，一邊即時觀察其 *TRACE* 輸出。許多人早已發現一件不幸的事實：Visual C++ 4.x 之中沒有 DBWIN，生活的步調因此亂了，寫程式的生命有些狼狽不堪。

沒有 DBWIN，*TRACE* 唱什麼獨角戲？Visual C++ 的 TRACE 字串輸出必須在整合環境的除錯視窗中看到。*TRACE* 巨集只在除錯模式的程式碼中才能生效，而 Visual C++ 竟還要求程式必須在整合環境的除錯器內執行，內含的 *TRACE* 巨集才有效。這太過份了。我只不過想在遊樂場玩點射擊遊戲，他們卻要我扛一尊 155 加農砲來。不少朋友都對這種情況相當不滿。

Paul DiLascia

Microsoft Systems Journal（***MSJ***）上的兩篇文章，彌補了 TRACE 的這一點小小遺憾：第一篇文章出現在 ***MSJ*** 1995.10 的 C/C++ 專欄，第二篇文章出現在 ***MSJ*** 1996.01 的 C/C++ 專欄。兩篇都出自 Paul DiLascia 之手，這位先生在 C++ / MFC 享有盛名，著有 ***Windows++*** 一書。

Paul DiLascia 發明了一種方法，讓你的 *TRACE* 巨集輸出到一個視窗中（他把它命名爲 Tracewin 視窗），這個視窗就像 Visual C++ 1.5 的 DBWIN 一樣，可以收集來自八方的 *TRACE* 字串，可以把內容清除，或存檔，或做其他任何文字編輯上的處理。你的程式只要是除錯模式就行。至於除錯器，把它丟開，至少在這個時刻。

我將在這篇文章中敘述 Paul DiLascia 的構想和作法，並引用部份程式碼。完整程式碼可以從 ***MSJ*** 的 ftp site（ftp.microsoft.com）免費下載，也可以從 MSDN（Microsoft Developer's Network）光碟片中獲得。

此法富有巧思，可以豐富我們的 MFC 技術經驗。所有榮耀都屬於作者 Paul DiLascia。

我呢，我只是向大家推薦並介紹這兩篇文章、這個人、這個好工具，並且儘量豐富稍嫌簡陋的兩篇原文。當文章中使用第一人稱來描述 Tracewin 的程式設計理念時，那個「我」代表的是 Paul DiLascia 而不是侯俊傑。

撐眺撐ヨ

要把 *TRACE* 的輸出字串導向，得先明白 *TRACE* 巨集到底是怎麼回事。*TRACE* 事實上呼叫了 Win32 API 函式 *OutputDebugString*。好，可能你會想到以類似 hooking Win32 API 的作法，把 *OutputDebugString* 函式導到你的某個函式來，再予取予求。這種方法在 Windows NT 中不能湊效，因爲 Windows NT 的 copy-on-write 分頁機制（註）使得我一旦修改了 *OutputDebugString*，我就擁有了一份屬於自己的 *OutputDebugString* 副本。我可以高高興興地盡情使用這副本（渾然不覺地），但其他程式呼叫的卻仍然是那未經雕琢的，身處 KRNL386 模組的 *OutputDebugString*。這樣就沒有什麼大用處啦！

註：所謂 copy-on-write 機制，Matt Pietrck 的 ***Windows 95 System Programming SECRETS*** 第 5 章（#290 頁）解釋得相當好。我大略說明一下它的意義。

當作業系統極儘可能地共享程式碼，對除錯器會帶來什麼影響？如果除錯器寫入中斷點指令的那個 code page 是被兩個行程共享的話，就會有潛在問題。要知道，除錯器只對一個行程除錯，另一個行程即使碰到中斷點，也不應該受影響。

高級作業系統對付此問題的方法是所謂的 "copy on write" 機制：記憶體管理器使用 CPU 的分頁機制，儘可能將記憶體共享出來，而在必要的時候又將某些 RAM page 複製一份。

假設某個程式的兩個個體（instance）都正在執行，共享相同的 code pages（都是唯讀性質）。其中之一處於除錯狀態，使用者告訴除錯器在程式某處放上一個中斷點（breakpoint）。當除錯器企圖寫入中斷點指令，會觸發一個 page fault（因爲 code page 擁有唯讀屬性）。作業系統一看到這個 page fault，首先斷定是除錯器企圖讀記憶體內容，這是合法的。然而，隨後寫入到共享記憶體中的動作就不被允許了。系統於是會先將受影響的各頁拷貝一份，並改變被除錯者的 page table，使映射關係轉變到這份拷貝版。一旦記憶體被拷貝並被映射，系統就可以讓寫入動作過關了。寫入動作只影響拷貝內容，

不影響原先內容。

Copy on write 機制的最大好處就是儘可能讓記憶體獲得共享效益。只有在必要時刻，系統才會對共享記憶體做出新的拷貝。

在 MFC 程式中，所有的診斷指令或巨集（包括 TRACE）事實上都流經一個名爲 *afxDump* 的物件之中，那是一個 *CDumpContext* 物件。所有的診斷動作都進入 *CDumpContext*::*OutputString* 成員函式，然後才進入全域函式 *AfxOutputDebugString*，把字串送往除錯器。

攔截除錯器是很困難的啦，但是你知道，字串也可以被送往檔案。如果我們能夠把送往檔案的字串攔截下來，大功就完成了一半。這個通往檔案的奧秘在哪裡呢？看看 MFC 的原始碼（圖一）。啊哈，我們發現，如果 *m_pFile* 有所指定，字串就流往檔案。*m_pFile* 是一個 *CFile* 物件（圖二）。

```
// in MFC 4.x DUMPCONT.CPP
#0001  void CDumpContext::OutputString(LPCTSTR lpsz)
#0002  {
#0003  #ifdef _DEBUG
#0004      // all CDumpContext output is controlled by afxTraceEnabled
#0005      if (!afxTraceEnabled)
#0006          return;
#0007  #endif
#0008
#0009      // use C-runtime/OutputDebugString when m_pFile is NULL
#0010      if (m_pFile == NULL)
#0011      {
#0012          AfxOutputDebugString(lpsz);
#0013          return;
#0014      }
#0015
#0016      // otherwise, write the string to the file
#0017      m_pFile->Write(lpsz, lstrlen(lpsz)*sizeof(TCHAR));
#0018  }
```

圖一　CDumpContext::OutputString 原始碼

```
// in MFC 4.x AFX.H
#0001  class CDumpContext
#0002  {
#0003  ...
#0004  public:
#0005         CFile* m_pFile;
#0006  };
```

圖二　CDumpContext 原始碼

好，如果我們能夠設計一個類別 *CMfxTrace*（圖三），衍生自 *CFile*，然後爲它設計一個初始化成員函式，令函式之中檢查 *afxDump.m_pFile* 內容，並且如果是 *NULL*，就將它指向我們的新類別，那麼 *CDumpContext*::*OutputString* #17 行的 *m_pFile->Write* 就會因此指向新類別 *CMfxTrace* 的 *Write* 函式，於是我們就可以在其中予取予求啦。

注意，*theTracer* 是一個 static 成員變數，需要做初始化動作（請參考**深入淺出 MFC**（侯俊傑 / 松崗）第２章「靜態成員」一節），因此你必須在類別之外的任何地方加這一行：

```
CMfxTrace CMfxTrace::theTracer;
```

```
#0001  class CMfxTrace : public CFile
#0002  {
#0003      static CMfxTrace theTracer;     // one-and-only tracing object
#0004      CMfxTrace();                  // private constructor
#0005  public:
#0006      virtual void Write(const void* lpBuf, UINT nCount);
#0007      static void Init();
#0008  };

#0001  // Initialize tracing. Replace global afxDump.m_pFile with me.
#0002  void CMfxTrace::Init()
#0003  {
#0004      if (afxDump.m_pFile == NULL) {
#0005          afxDump.m_pFile = &theTracer;
#0006      } else if (afxDump.m_pFile != &theTracer) {
#0007          TRACE("afxDump is already using a file: TRACEWIN not installed.\n");
#0008      }
#0009  }
```

圖三　CMfxTrace

行程通訊（InterProcess Communication，IPC）

把 TRACE 輸出字串攔截到 *CMfxTrace*::*Write* 之後，我們得想辦法在 *Write* 函式中把字串丟給 Tracewin 程式視窗。在 Windows 3.1 之中這不是難事，因爲所有的行程（process）共同在同一個位址空間中生存，直接把字串指標（代表一個邏輯位址）丟給對方就是了。在 Windows 95 和 Windows NT 中困難度就高了許多，因爲每一個行程擁有自己的位址空間，你把字串指標丟給別的行程，對別的行程而言是沒有用的。如果你爲此使用到記憶體映射檔（Memory Map File，Win32 之下唯一的一種行程通訊方法），又似乎有點小題大作。

Paul DiLascia 的方法是，利用所謂的 global atom。atom 並不是 Win32 下的新產物，有過 Win16 DDE（Dynamic Data Exchange，動態資料交換）程式經驗的人就知道，atom 被用來當作行程之間傳遞字串的識別物。說穿了它就是一個 handle。global atom 可以跨越 Win32 行程間的位址空間隔離。下面是 *CMfxTrace*::*Write* 函式的動作步驟：

1. 利用 *FindWindow* 找出 Tracewin 視窗。
2. 利用 *GlobalAddAtom* 把字串轉換爲一個 global atom。
3. 送出一個特定訊息給 Tracewin，並以上述的 gloabl atom 爲參數。
4. 刪除 global atom。
5. 萬一沒有找到 Tracewin 視窗，呼叫 ::*OutputDebugString*，讓 *TRACE* 巨集擁有原本該有的行爲。

下回就是 *CMfxTrace*::*Write* 函式碼。其中的兩個常數分別定義爲：

```
#define TRACEWND_CLASSNAME "MfxTraceWindow"  // 視窗類別名稱
#define TRACEWND_MESSAGE   "*WM_TRACE_MSG"   // 自行註冊的 Windows 訊息
```

```
#0001  // Override Write function to Write to TRACEWIN applet instead of file.
#0002  //
#0003  void CMfxTrace::Write(const void* lpBuf, UINT nCount)
#0004  {
#0005    if (!afxTraceEnabled)
```

```
#0006          return;
#0007
#0008    CWnd *pTraceWnd = CWnd::FindWindow(TRACEWND_CLASSNAME, NULL);
#0009    if (pTraceWnd) {
#0010      static UINT WM_TRACE_MSG =
RegisterWindowMessage(TRACEWND_MESSAGE);
#0011
#0012      // Found Trace window: send message there as a global atom.
#0013      // Delete atom after use.
#0014      //
#0015      ATOM atom = GlobalAddAtom((LPCSTR)lpBuf);
#0016      pTraceWnd->SendMessage(WM_TRACE_MSG, (WPARAM)atom);
#0017      GlobalDeleteAtom(atom);
#0018
#0019    } else {
#0020      // No trace window: do normal debug thing
#0021      //
#0022      ::OutputDebugString((LPCSTR)lpBuf);
#0023    }
#0024  }
```

圖四　CMfxTrace::Write 函式碼

如何使用 TRACEWIN.H

雖然除錯工具的最高原則是，不要動用被除錯對象的原始碼，但爲了讓方法簡單一些，力氣少用一些，看得懂的人多一些，Paul DiLascia 還是決定妥協，他設計了上述的 *CMfxTrace* 類別以及一個 Tracewin 工具程式，你必須把 *CMfxTrace* 類別含入到自己的程式中，像這樣：

```
#include "tracewin.h"
```

並在程式的任何地方（通常是 *InitInstance* 函式內）做此動作：

```
CMfxTrace::Init();
```

然後所有的 *TRACE* 字串輸出就會流到 Tracewin 視窗上。太好了！

注意，TRACEWIN.H 中不但有類別宣告，還有類別定義，和一般的表頭檔不太一樣，所以你只能夠在你的程式中含入一次 TRACEWIN.H。

Tracewin 視窗

這是一個隨時等待接受訊息的小程式。它所接受的訊息，夾帶著 global atom 作爲參數。Tracewin 只要把 atom 解碼，丟到一個由它管轄的 Edit 視窗中即可。Paul DiLascia 設計的這個小程式，功能面面俱到，可以把接受的 *TRACE* 輸出字串顯示在視窗中，或放到某個檔案中；也可以把 EDIT 緩衝區內容拷貝到剪貼簿上，或是清除整個 EDIT 緩衝區內容。功能與 Visual C++ 1.5 的 DBWIN 幾乎不相上下，只差沒能夠把除錯字串輸出到 COM1 和 COM2。

Tracewin 並不動用 Document/View 架構。主視窗內含一個 Edit 控制元件作爲其子視窗，事實上，那是一個衍生自 *CEdit* 的 *CBufWnd* 類別，有點像 *CEditView*。

當應用程式以 *TRACE* 巨集送出字串，經過 *CDumpContext*::*OutputString* 的作用，送往 *CMfxTrace*::*Write*，我們在其中以 *FindWindow* 找到 Tracewin 視窗：

```
CWnd *pTraceWnd = CWnd::FindWindow(TRACEWND_CLASSNAME, NULL);
```

要知道，如果 Tracewin 使用的類別是 MFC 預先建立的四個類別，那麼它的類別名稱可能是像 Afx:b:14ae:6:3e8f 這種不太有意義的字串，而且可能在不同的機器不同的時間有不同的名稱。如此一來如何爲 *FindWindow* 指定第一個參數？我們必須有個什麼方法避免使用 MFC 預先建立的四個類別，但又能夠使用其類別設定。

視窗類別名稱 Afx:x:y:z:w

MFC 2.5 在應用程式一開始執行時，便在 *AfxWinInit* 中先行註冊了５個視窗類別備用。MFC 4.x 的行爲稍有修改，它在應用程式呼叫 *LoadFrame* 準備產生視窗時，才在 *LoadFrame* 所呼叫的 *PreCreateWindow* 虛擬函式中爲你產生視窗類別。５個可能的視窗類別分別是：

```
const TCHAR _afxWnd[] = AFX_WND;
const TCHAR _afxWndControlBar[] = AFX_WNDCONTROLBAR;
const TCHAR _afxWndMDIFrame[] = AFX_WNDMDIFRAME;
const TCHAR _afxWndFrameOrView[] = AFX_WNDFRAMEORVIEW;
const TCHAR _afxWndOleControl[] = AFX_WNDOLECONTROL;
```

這些 AFX_xxx 常數定義於 AFXIMPL.H 中，依不同的聯結狀態（除錯模式與否、動態聯結與否）而有不同的值。如果使用 MFC 動態聯結版和除錯版，5 個視窗類別的名稱將是：

```
"AfxWnd42d"
"AfxControlBar42d"
"AfxMDIFrame42d"
"AfxFrameOrView42d"
"AfxOleControl42d"
```

如果是使用 MFC 靜態聯結版和除錯版，5 個視窗類別的名稱將是：

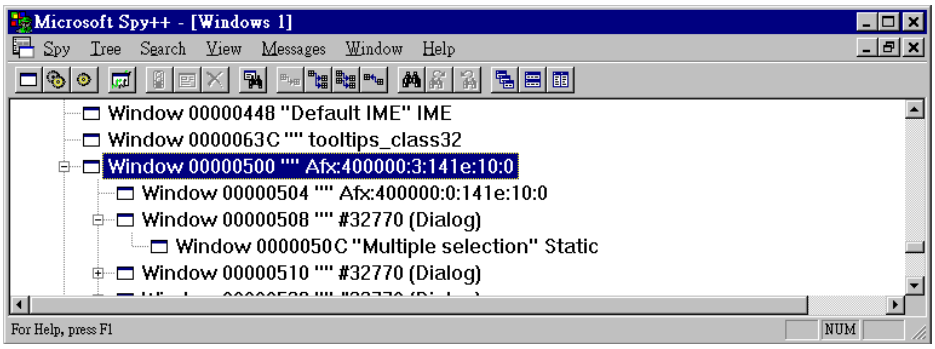
```
"AfxWnd42sd"
"AfxControlBar42sd"
"AfxMDIFrame42sd"
"AfxFrameOrView42sd"
"AfxOleControl42sd"
```

這些名稱的由來，以及它們的註冊時機，請參考**深入淺出 MFC**（侯俊傑 / 松崗）第6章。

然而，這些視窗類別名稱又怎麼會變成像 Afx:b:14ae:6:3e8f 這副奇怪模樣呢？原來是 Framework 玩的把戲，它把這些視窗類別名稱轉換爲 Afx:x:y:z:w 型式，成爲獨一無二之視窗類別名稱：

```
x:  視窗風格（window style）的 hex 值
y:  滑鼠游標的 hex 值
z:  背景顏色的 hex 值
w:  圖示的 hex 值
```

爲了讓 *CMfxTrace* 能夠以 *FindWindow* 函式找到 Tracewin，Tracewin 的視窗類別名稱（也就是那個 Afx:x:y:z:w）必須完全在我們的控制之中才行。那麼，我們勢必得改寫 Tracewin 的 frame 視窗的 *PreCreateWindow* 虛擬函式。



圖五　以 Spy 觀察視窗。請注意每一個視窗類別的名稱都是 Afx:x:y:z:w 型式。fig5.bmp

PreCreateWindow 和 GetClassInfo

圖六是 Tracewin 的 *PreCreateWindow* 內容，利用 *GetClassInfo* 把 MFC 的視窗類別做出一份副本，然後改變其類別名稱以及程式圖示，再重新註冊。是的，重新註冊是必要的。這麼一來，*CMfxTrace*::*Write* 中呼叫的 *FindWindow* 就有明確的搜尋目標了。

```
#0001  BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
#0002  {
#0003    static LPCSTR className = NULL;
#0004
#0005    if (!CFrameWnd::PreCreateWindow(cs))
#0006        return FALSE;
#0007
#0008    if (className==NULL) {
#0009      // One-time class registration
#0010      // The only purpose is to make the class name something
#0011      // meaningful instead of "Afx:0x4d:27:32:hup1hup:hike!"
#0012      //
#0013      WNDCLASS wndcls;
#0014      ::GetClassInfo(AfxGetInstanceHandle(), cs.lpszClass, &wndcls);
#0015      wndcls.lpszClassName = TRACEWND_CLASSNAME;
#0016      wndcls.hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
#0017      VERIFY(AfxRegisterClass(&wndcls));
#0018      className=TRACEWND_CLASSNAME;
```

```
#0019   }
#0020   cs.lpszClass = className;
#0021
#0022   // Load window position from profile
#0023   CWinApp *pApp = AfxGetApp();
#0024   cs.x = pApp->GetProfileInt(PROFILE, "x", CW_USEDEFAULT);
#0025   cs.y = pApp->GetProfileInt(PROFILE, "y", CW_USEDEFAULT);
#0026   cs.cx = pApp->GetProfileInt(PROFILE, "cx", CW_USEDEFAULT);
#0027   cs.cy = pApp->GetProfileInt(PROFILE, "cy", CW_USEDEFAULT);
#0028
#0029   return TRUE;
#0030 }
```

圖六　Tracewin 的 frame 視窗的 PreCreateWindow 函式內容

Tracewin 取出字串並顯示

如果 Tracewin 欲接收由除錯端傳來的自定訊息 *WM_TRACE_MSG*，並交由 *OnTraceMsg* 函式去處理，它就必須在其訊息映射表中有所表示：

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
  //{{AFX_MSG_MAP(CMainFrame)
  ON_REGISTERED_MESSAGE(WM_TRACE_MSG, OnTraceMsg)
  ...
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

並且設計 *OnTraceMsg* 函式如下。

```
#0001  LRESULT CMainFrame::OnTraceMsg(WPARAM wParam, LPARAM)
#0002  {
#0003      if (!wParam || m_nOutputWhere==ID_OUTPUT_OFF)
#0004              return 0;
#0005
#0006      char buf[256];
#0007      UINT len = GlobalGetAtomName((ATOM)wParam, buf, sizeof(buf));
#0008
#0009      if (m_nOutputWhere==ID_OUTPUT_TO_WINDOW) {
#0010
#0011          // Convert \n to \n\r for Windows edit control
```

```
#0012         ...
#0013         // Append string to contents of trace buffer
#0014         ...
#0015     } else if (m_nOutputWhere==ID_OUTPUT_TO_FILE) {
#0016         m_file.Write(buf, len);
#0017     }
#0018     return 0;
#0019 }
```

圖七　CMainFrame::OnTraceMsg 函式內容。

改用 WM_COPYDATA 進行行程通訊（IPC）

Paul DiLascia 的第一篇文章發表後，收到許多讀者的來信，指出以 global atom 完成行程通訊並不是最高明的辦法，可以改用 *WM_COPYDATA*。於是 Paul 從善如流地寫了第二篇文章。

WM_COPYDATA 是 Win32 的新訊息，可以提供一個簡單又方便的方法，把字串送往另一個程式。這正符合 Tracewin 之所需。這個訊息的兩個參數意義如下：

```
wParam = (WPARAM) (HWND) wnd;           // handle of sending window
lParam = (LPARAM) (PCOPYDATASTRUCT) pcds; // pointer to structure with data
```

其中 COPYDATASTRUCT 結構定義如下：

```
typedef struct tagCOPYDATASTRUCT {  // cds
   DWORD dwData; // 隨便你指定任何你需要的額外資訊
   DWORD cbData; // 資料長度
   PVOID lpData; // 資料指標
} COPYDATASTRUCT;
```

dwData 在本例應用中被指定爲 *ID_COPYDATA_TRACEMSG*；Tracewin 將檢查這個識別碼，如果不合格，就忽略該次的 *WM_COPYDATA* 訊息。

Tracewin 新版本的內容我就不再列出了，請直接下載其原始碼看個究竟。

我的使用經驗

現在讓我來談點我使用 Tracewin 的經驗。

我早就需要在 Visual C++ 中使用 DBWIN 了，也早就看到了 Paul DiLascia 的兩篇文章，但是真正研讀它並使用其成果，是在我撰寫 COM/OLE/ActiveX 一書（我最新的一本書，還在孵化之中）的時候。也許當你讀到該書，會感嘆侯俊傑怎麼能夠對 OLE container 和 server 之間的交叉動作瞭若指掌。沒有什麼，我只是在 container 和 server 之中的每一個我感興趣的函式的一開始處，利用 *TRACE* 輸出一些訊息，這樣我就可以從容地從 Tracewin 視窗中觀察那些函式的被呼喚時機了。

所以我在 OLE container 中這麼做：

```
#include "tracewin.h"
...
BOOL CContainerApp::InitInstance()
{
  ...
  pMainFrame->ShowWindow(m_nCmdShow);
  pMainFrame->UpdateWindow();

  CMfxTrace::Init(); // add by J.J.Hou
  return TRUE;
}
```

也在 OLE server 中這麼做：

```
#include "tracewin.h"
...
BOOL CScribbleApp::InitInstance()
{
  ...
  pMainFrame->ShowWindow(m_nCmdShow);
  pMainFrame->UpdateWindow();

  CMfxTrace::Init();
  return TRUE;
}
```

然後我就可以使用 *TRACE* 巨集並搭配 Tracewin 看個過癮了。

很好，當這兩個程式獨立執行的時候，一切盡如人意！但是當我在 container 中即地編輯 Scribble item，我發現沒有任何 Scribble TRACE 字串被顯示在 Tracewin 視窗上。這麼一來我就觀察不到 Scribble 的交叉作用了呀！於是我想，莫不是兩個程式爭用 *afxDump*？或是因爲兩個 Win32 行程使用不同的位址空間？或是因爲...

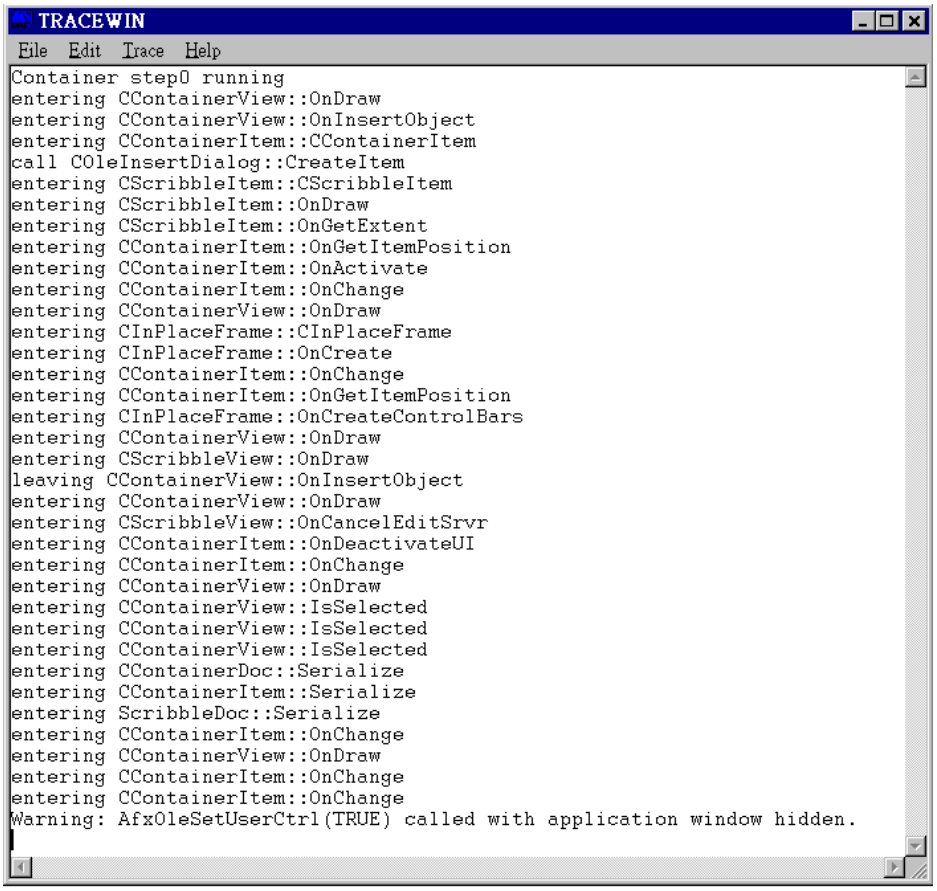
胡說！沒道理呀。如果我交叉使用各自獨立的 Container 和 Scribble（不牽扯即地編輯），它們的 *TRACE* 結果會交叉出現在 Tracewin 視窗上，這就推翻了上述的胡思亂想。

然後我想，會不會是即地編輯時根本沒有進入 server 的 *InitInstance*？如果 *CMfxTrace*::*Init* 沒有先執行過，*TRACE* 當然就不會輸出到 Tracewin 視窗囉。於是我把 *CMfxTrace*::*Init* 改設在...唔...什麼地方才是即地編輯時 server 的第一個必經之地？server item 是也。於是我這麼做：

```
CScribbleItem::CScribbleItem(CScribbleDoc* pContainerDoc)
       : COleServerItem(pContainerDoc, TRUE)
{
       CMfxTrace::Init();

       // TODO: add one-time construction code here
       //  (eg, adding additional clipboard formats
       //   to the item's data source)
}
```

賓果！我看到了預期的東西。圖八就是 Tracewin 視窗畫面。這些輸出結果幫助我分析出 OLE container 和 server 的一舉一動。



圖八 Tracewin 視窗畫面

新的視野

好的除錯工具，不應該要求應用程式碼本身做任何配合性的動作。Paul DiLascia 的 Tracewin 小工具是一個權宜之計。使用上稱不上太方便（你得含入一個 tracewin.h），而且你得有某種程度的技術背景才能把它用得好。不過，說真的，我還是很感謝 Paul DiLascia 的創意，讓我們的視野有了新的角度。

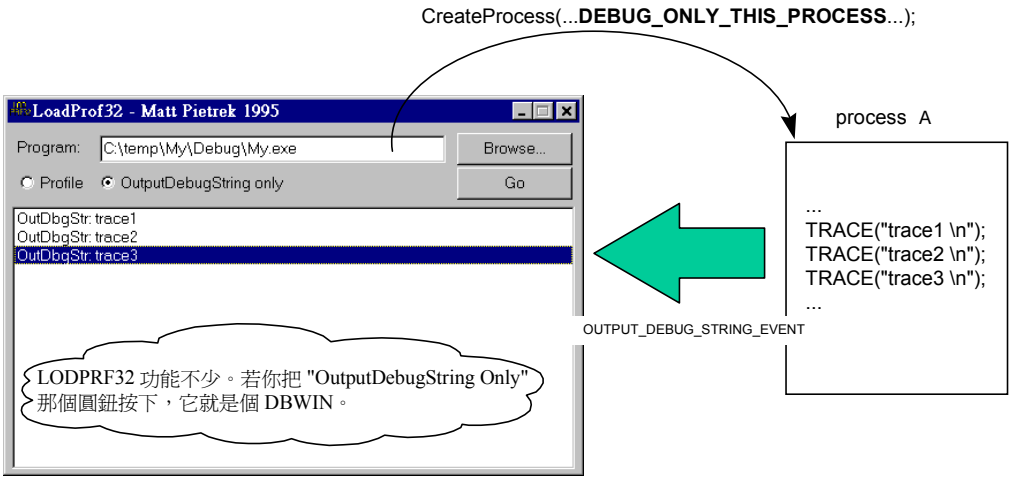
我想你也是。

封裝 DBWIN 之 Debug Event 訊

自從我開始注意到 DBWIN 之後，我就更加注意期刊上有關於 DBWIN 技巧的文章。這才發現，好像大家滿喜歡在這個題目上展現自己傲人的功力。像 Paul Dilascia 這樣，以高階的 MFC 來寫 DBWIN，當然也就不可能太過「威力」-- 雖然從 MFC programming 的技巧來看，我們是學了不少。你知道，Paul 的方法要求你改變你的原始碼，含入一個 .h 檔，並在你的 .cpp 檔中加上一兩行。這在使用的方便性上不怎麼高明。

要高明一點，不落痕跡地抓到 *TRACE* 輸出，就必須懂得 Windows 作業系統內部，以及 Windows system programming。是的，這方面的大師 Matt Pietrek 也寫了一個 DBWIN，不必影響你的原始碼。不過，他用到 debug event，以及許多系統知識，不應該是一本 MFC 書籍適合涵蓋的主題。所以，我只能在這裡告訴你，可以到 ***Microsoft Systems Journal*** 1995.07 的 Windows Q/A 專欄上學習。程式名稱爲 LODPRF32。此程式的主要功能其實是讓你觀察目前所有映射到記憶體中的 DLLs。你可以從中知道你的 EXE 直接或間接使用的所有 DLLs。它還會記錄「最後一次 implicitly loading」至「EXE entry point 執行」之間的經過時間。此期間正是 Win32 載入器呼叫 implicitly loaded DLLs 的初始化程式碼（程式進入點）的時間。最後，它允許使用者濾掉所有的 debug event，只記錄被除錯端的 *OutputDebugString* 輸出的字串 -- 這功能可不正是 DBWIN !?

使用真簡單，不是嗎！這程式若說還有什麼缺點，那就是「只有被此 DBWIN 載入之程式，其 TRACE 輸出字串才能夠被觀察」。有沒有可能像 Win16 的 DBWIN 那樣，做一個系統層面的 "global" DBWIN，像常駐程式那樣隨時偵測等候任何程式發出的任何 *TRACE* 字串呢？可能，但我們還要再下一層地獄，進入 ring0 層次。



圖九　LODPRF32 功能不少。若你把 "OutputDebugString Only" 那個圓鈕按下，它就是個 DBWIN。

重建 DBWIN 之 VxD 篇

若想要寫一個 global DBWIN，困難度陡增。不過，已經有人做出來了。請參考 Ton Plooy 發表於 ***Windows Developer's Journal*** 1996.12 的 "A DBWIN Utility for Win95" 一文。

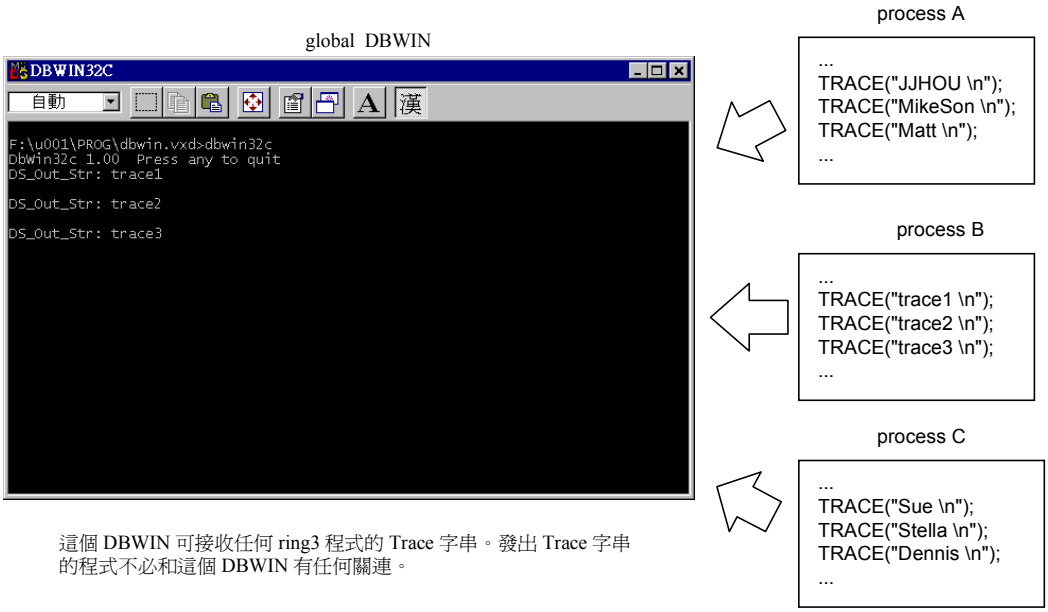
前一個 DBWIN 之所以不能夠做到 "global"，是因爲行程有獨立的位址空間。如欲接收除錯訊息，又必須一一建立「除錯器」與「被除錯端」的關係。那麼，有沒有什麼辦法可以建立一個「系統除錯器」，把所有執行起來的程式統統視爲我的除錯對象？如果這個問題有肯定答案，global DBWIN 就有希望了。

有，VMM 提供了一個 INT 41h 介面。此介面作用起來的條件是，必須有一個「系統除錯器」存在。而「系統除錯器」存在的條件是：當 VMM 以 int41h/AX=DS_DebLoaded 發出訊息時，必須有程式以 AX=DS_DebPresent 回覆之。

可是 *OutputDebugString* 和「系統除錯器」有沒有什麼牽連？如果沒有則一切白搭。幸運的是 *OutputDebugString* 最終會牽動 VMM 的 *Exec_PM_Int41h* service。如果我們能

夠寫一個程式，與 *Exec_PM_Int41h* 掛勾（hooking），使 *Exec_PM_Int41h* 能夠先來呼叫我自己的函式，我就可以悠游自在地在其中處理 TRACE 的除錯字串了。

這個技術最大的難點在於，要與 VMM 打交道，我們得寫 ring0 程式。在 Windows 95 中這意味著要寫 VxD（NT 不支援 VxD）。VxD 的架構其實不太難，**DOS/Windows 虛擬機器作業環境**（侯俊傑 / 旗標，1993）曾經有過詳細的探討。問題在於 VMM 的許多 services 常常要合著用，尤其是面對中斷模擬、事件處理、與 ring3 通訊過程、乃至於 hooking 的處理等等，而這方面的資料與範例相當稀少。此外，ring0 和 ring3 間的同步（synchronous）處理，也很令人頭痛。



圖十　global DBWIN 的執行畫面。它是一個 Console 程式，在接受任何按鍵之前，將一直存在。

不甘示弱

Paul DiLascia 看到百家爭鳴，大概是不甘示弱，在 ***Microsoft Systems Journal*** 1997.04 的

C/C++ Q/A 專欄又發表了一篇文章。他說「理想上 TraceWin 應該無臭無味，如影隨形。沒有 #include，沒有 init 函式...」

於是他又想到一種方法，此法只能在 MFC 動態聯結版身上有效。幸運的是大部份程式都動態聯結到 MFC。要點非常簡單：寫一個 DLL 並在它被載入時設定 afxDump.m_pFile = new CFileTrace。然後讓每個程式載入此 DLL。簡單！

不幸的是，沒有想像中簡單。要讓 DLL 能夠被每一個程式載入，需要用到 Jeffrey Richter 於其名著 ***Advanced Windows*** 第 16 章的 Inject 技術，或是 Matt Pietrek 於其名著 ***Windows 95 System Programming SECRETS*** 第 10 章的 Spy 技術。或是，Paul DiLascia 所採用的 system-wide hook 技術。

好吧，到此爲止。我知道我們每個人都已經頭皮發麻了。有興趣的人，自己去找那些文章和書籍來看。

榮譽

我真希望這些巧奪天工的榮譽都屬於我，可惜都不是，它們分屬於 Matt Pietrek、Paul DiLascia、Ton Plooy。

我喜歡的四本期刊雜誌與四家電腦圖書出版公司的網址

Microsoft Systems Journal（MSJ）	http://www.msj.com/
Windows Developer's Journal（WDJ）	http://www.wdj.com/
Dr. Dobb's Journal（DDJ）	http://www.ddj.com/
PC Magazine	http://www.pcmag.com/
R&D Books	http://www.rdbooks.com/
Microsoft Press	http://www.microsoft.com/mspress/
Addison Wesley	http://www.aw.com/devpress/
O¡Reilly	http://www.ora.com/

哥倫比亞大冰原，位於加拿大亞伯達省落磯山國家公園內。平均厚度 300 公尺，總面積 320 平方公里（比台北市還大），是南極圈外的地球最大冰河遺跡。 大冰原爲八條冰河的源頭，其中以阿塔巴斯卡大冰河（上圖）最壯觀，全長六公里，寬一公里，宛如銀色巨龍，盤桓於崇山峻嶺之間。

人生到處知何似　恰似飛鴻踏雪泥　泥上偶然留指爪　飛鴻那復計東西