《侯捷觀點》

但教心似金鈿堅[1]

2002 新春，有些話想和大家聊聊。
讀者來函常能使我的思考聚焦，
所以本文很大比例以來函回覆的方式進行。
本文主要分三個方向：技術、學習、書籍。

Visual Studio.NET

微軟的 Visual Studio.NET（以下簡稱 VS.NET）已於 2002/02/13 正式發表。做爲一個長期以 C++ 爲主要工作領域的人，我和所有 C++ 社群成員一樣關心其中 Visual C++.NET（以下簡稱 VC.NET）的發展。過去以來包括微軟官方雜誌 *MSDN magazine*，雖對 VC.NET 有少量報導（例如 2002/02 兩篇文章），但對 MFC7 著墨極少，近乎空白。我看過唯一一篇夠份量的 MFC 7 討論文章是 Anson Tsao 和 Wlater Sullivan 撰寫的 "*What's New for MFC Developers*?"（http://msdn.microsoft.com/msdnnews/2001/sept/vcnet/vcnet.asp）。

Classes 的運用，其實已屬末節。對於所有 MFC 程式員，或所有《深入淺出 MFC》的讀者，我們比較在意的是 MFC 應用程式的運行主軸是否有大變動。用膝蓋也可以推想出結果：不會有大變動，否則微軟何以面對眾多客戶？

我已經運用 WinDiff 工具觀察了 MFC7 源碼和 MFC4 源碼的差異，以眼睛給予膝蓋有力的支持☺。對 MFC 使用者而言，沒有消息就是好消息。當然，新增 classes

[1] 「但教心似金鈿堅」是白居易《長恨歌》中歌詠唐明皇和楊貴妃愛情的名句。「但教心似金鈿堅，天上人間會相見」。天上人間都能相見了，世上何事不成？太多事情需要我們抱持「吾心信其可行」的信念和堅如金石的意志。新春伊始，以此共勉。

是有的，那是手冊關心的話題，不是我的話題。在 application framework 主軸上，或說在《深入淺出 MFC》一書所關注的主題上，較大的改變一如微軟所宣稱是 Type-safe Message Maps，也就是利用轉型運算子 static_cast<> 增加訊息映射時的型別安全性，作法如下：

```
// 以下是 MFC4 源碼，見 afxmsg_.h（可參考《深入淺出 MFC》2e 第 9 章）
#define ON_COMMAND(id, memberFxn) \
{ WM_COMMAND, CN_COMMAND, (WORD)id, (WORD)id, AfxSig_vv,
    (AFX_PMSG)&memberFxn },

#define ON_WM_CREATE() \
{ WM_CREATE, 0, 0, 0, AfxSig_is, \
  (AFX_PMSG)(AFX_PMSGW)(int (AFX_MSG_CALL CWnd::*)(LPCREATESTRUCT))&OnCreate },


// 以下是 MFC7 源碼，見 afxmsg_.h
#define ON_COMMAND(id, memberFxn) \
{ WM_COMMAND, CN_COMMAND, (WORD)id, (WORD)id, AfxSigCmd_v, \
    static_cast<AFX_PMSG> (memberFxn) },

#define ON_WM_CREATE() \
{ WM_CREATE, 0, 0, 0, AfxSig_is, \
    (AFX_PMSG) (AFX_PMSGW) \
    (static_cast<int (AFX_MSG_CALL CWnd::*)(LPCREATESTRUCT)> (OnCreate)) },
```

用以協助在訊息映射過程中標示型別的一些符號也有了些微變化，但不影響程式運行，例如：

```
// 以下是 MFC4 源碼，見 afxmsg_.h（可參考《深入淺出 MFC》2e 第 9 章）
enum AfxSig
{
 AfxSig_end = 0,  // [marks end of message map]
 AfxSig_bD,       // BOOL (CDC*)
 AfxSig_bb,       // BOOL (BOOL)
 ...
}

// 以下是 MFC7 源碼，見 afxmsg_.h
enum AfxSig
{
 AfxSig_end = 0,  // [marks end of message map]
 AfxSig_b_D_v,    // BOOL (CDC*)
 AfxSig_b_b_v,    // BOOL (BOOL)
 ...
}
```

其他微小改變還有，例如讓 CCmdTarget 和 CView 擁有純虛擬函式、嘗試運用

C++ 關鍵字 `explicit`、在原本以 `UINT` 代表指標之處改用 `UINT_PTR`（可能用到編譯器原生型別 `__int64`，見 basetsd.h）。這些都是爲了讓 MFC 的體質更健康些。

關心 MFC 的人，大概也會關心 STL。噢，persistence（物件永續）依然無解，倒是多了以 hash table 爲基礎而實現的 map, multimap, set。我猜想每 5 年一次的 C++ 標準委員會下次開會應該（一定）會把 hash table 放入 *C++ Standard* 之中了吧，否則就太不上道了。

不在其位不謀其政

下面是讀者給我的一封信（節錄）：

●來函：我當然傾心 C# 了。但它似乎太強大了，真擔心會形成技術壟斷。微軟的技術我都非常熱衷，這有點矛盾，您怎麼看呢?

●侯捷：C# & .NET 是一個絕對值得注意的技術。不說其他，就說它是微軟推出，這就有代表性。代表什麼？首先，微軟是一家怎樣規模的公司，砸下多少銀子在這個被他視爲新世代的技術上？銀子夠就代表人才不愁。領導者爲了維護地位，一定想辦法推出更好的產品。這段分析看來不夠理智，未從技術面觀察。其實很理智，不必討論技術（那要討論到什麼時候，寫多少文字？），只要從人情世故，世事常理來看就夠了。注意我說「絕對值得注意」，我並沒有說它一定最好。

在我對自己的定位（一個對別人的學習帶來幫助的技術作家）裡，我不在乎「壟斷」或「誰壟斷」這種事。有沒有壟斷，誰壟斷，都無所謂。分久必合，合久必分，這個世道有其運行規律。不在其位不謀其政。微軟對整個世界的計算機普及的貢獻很大。誰都知道有不少排版軟件優於 Word，但若沒有微軟的 Windows + Word，現今多少人能夠輕鬆自在地享受個人（桌上）排版便利？TeX 很棒，棒得不得了，可多少人能夠接受「非視覺化」還要「撰寫 macros」的難度？在這大千世界中，存在形形色色的產品，滿足形形色色的人和形形色色的需求。微軟滿足許多許多人的需求（成功不是瞎碰來的）。在你我（以及許多人）的位置上，無需在乎壟斷這樣的事情。當一個公司的技術和形象不足以形成「壟斷」時，自然

會有取而代之者。

鹿與鼎，天下英雄逐之，能者得之。

當然，如果你一定要做逐鹿英雄，一定要把這種事情上綱到國家民族，並決定以國家民族爲己任，我也會很尊敬你。

書作的誕生

讀者對於書籍總有很多期許。非常合理，畢竟讀者希望從書中獲得個人的成長，如果一本書不能帶來幫助，甚至還帶來學習上的歧路，不獨金錢浪費令人惱怒，時間的虛擲更讓人生氣。如果書籍把讀者帶往錯誤的方向，繞了遠路，那麼讀者恨不得把作者殺了，把出版社燒了，也在情理之中。

我發現許多大陸讀者對於寫作、翻譯、出版的過程很不清楚，對於版權、翻譯權的概念也比較模糊。因此，面對某些問題就無法對症下藥罵對人。知情者或因事不關己，或因人情世故，不方便明說。我是客卿，讓我來細說分明。以下所談是我所知道的臺灣出版情況，如果大陸情況不同而我也知道的話，我會特別說明。

書籍的誕生分爲著作和譯作兩種。先談著作。作者把稿件準備好，交給心儀的出版社，合作方式談妥，就可以著手審稿、出版事宜了。也有出版社向作者邀稿的情況，我個人從不敢答應這種邀約，因爲我不喜歡束縛和壓力。

有些作者會將書稿排版妥當（酬勞另計），這對出版社很省事。我曾經提過一個論點：要想成爲**最**優秀的作家，必須學會排版。聽起來很荒謬，好像要學者彎腰處理柴米油鹽。但我認爲，最優秀的作家一定關心最後成品，最後成品的可讀性和版面脫離不了關係。能夠將版面操弄到讓作者稱心如意的，除了作者自己，還有誰？第一流畫家要不要關心畫布材質、顏料成份、兩者作用後產生的渲染或收斂效果？乃至裝裱水平？第一流舞蹈家要不要關心舞台佈置、燈光效果？第一流演奏家要不要關心現場音效、樂器擺設位置？舞台燈光和舞蹈技巧無關，舞台音效和演奏技巧無關，但當舞蹈家和演奏家要演出，他們必須關心這些題目。

寫作生涯的前數年，我不懂排版，也不屑把時間花在上面。1995 年經歷了一次慘

痛教訓，決定學習排版，結果受用無窮。回顧過去與排版人員之間的溝通和拉鋸，彷彿一種生命折磨。至今我的所有書籍都由我自己排版，並且無心插柳地成就個人版面風格（很多人喜歡我的書籍版式）。

我相信至今還有很多人對於「自己的書自己排版」這種論調嗤之以鼻，甚至譏爲傻瓜：『寫作才是正務，內容決定一切，排版算什麼？』唔，我並不打算說服任何人，但侯捷絕不是第一個傻瓜，也不是最大的傻瓜 — Donald Knuth[2]爲了讓他的書達到他所滿意的版面，在寫作過程中擱下一切「正務」，開發出 TeX 排版軟體，並成爲學術界公認的排版標準工具。他（好像）還致力於電子書格式的開發[3]。

譯作的誕生

2001 年大陸出版社製作許多外文書中譯本。我發現大陸讀者對於外文書翻譯的權利義務有許多誤解，例如明明知道某某書籍正在翻譯或即將出版，還來信建議我也翻譯出版。又如建議出版社在中譯本光碟中加附原文書電子檔，又如任意翻譯他人作品在網上公佈。這些都和版權牴觸，簡單說就是和金錢有干係，不能任意而爲。

外文書的翻譯版權通常由出版社爭取，個人行爲很少，但不是不可能（至少在臺灣沒有什麼束縛，只要你有能耐）。個人扮演的角色多半只是提建議，例如讀者或譯者向出版社建議某某書籍很好，請求引進。讀者提的建議幾乎都根據個人需要，出版社自會評估市場價值（可行性）。好出版社不僅評估經濟價值，還評估名聲價值，不過這種令人尊敬的行爲並不多見。一個地區（語種）的翻譯權只有一個，同一本書的兩個版本可視爲兩個實體（兩個 ISBN 書號），可由不同的出版社和不同的譯者完成。

---

[2] Donald Knuth，演算法大師，《*The Art of Computer Programming*》作者，1974 圖靈獎（Turing Award）得主。

[3] Knuth 在 1996 年接受 Dr. Dobb's Journal 專訪時提到：『印刷工業發生了一場革命，它已經成爲電腦科學之一；鉛字轉變成 0/1 bits。過去的技術無法滿足我的書對印刷（排版）的要求。我打算花一年的時間對於書籍的製作給出一個計算機科學家的答案。爲了這個答案，過去以來我花了十年時間。』

出版社獲得譯權後，尋找合適的譯者，就可以開始作業。出版社和讀者之間的緊張關係通常發生在「合適譯者」的認定和選擇。技術翻譯是技術領域之外的另一種專業，但如缺乏專業人才，出版社也只好以學歷或印象來交付任務。沒有人願意把事情做壞，但如果出版社有好人選，卻因爲成本考量做出次等（甚至又次等）的選擇，就是置讀者權益於不顧。久而久之讀者便會反彈。

許多大陸讀者給我的深刻印象是，認定「凡電子型式的都不要錢」。於是要求中譯本附原文書光碟、將電子書任意拷貝傳佈、大聲喧嚷「我有 XXX，要的人進來」。中譯本一旦附上原文書電子檔，影印本出版者肯定要抗議。商業運作有許多配套措施，都可以談，都可能做，但都不是想當然爾。

另一個令我印象深刻的是，譯權觀念普遍薄弱。任何文稿未經同意其實是不能翻譯的。你譯了它，孤芳自賞，可以；公開傳佈，不行。不僅網絡上存在這個問題，期刊雜誌也有大量這類問題。國情或需斟酌，我只論法律。臺灣在 1994/06/12 之前尚未加入國際出版相關組織，翻譯圖書不必付費給外國書商。如今情況已完全改觀。我曾經爲文一篇，紀念 612 圖書大限的重慶南路（台北書店街）盛況，詳見《無責任書評》94/08「錦囊招數齊出 奧密漫天飛舞」。

很多讀者瘋狂收集電子書。我給各位一個良心建議：培養時間成本觀念。IT 產業容不得你浪費時間。你花一些錢，買三兩本經典，好好啃一啃，總體價值遠勝一堆龍蛇雜處、良窳不齊、價值尚待評估、閱讀不便又傷眼力的電子書。換一付眼鏡，要多少錢？時間和身體是你最寶貴的資源，不要貪小失大。電子書的主要用途在於搜尋快捷，巨量內容攜帶方便（需以筆記電腦或閱讀器之普及爲前提），適合當輔助讀物，不適合當學習主菜。

索引與勘誤

外文技術書籍，沒有一本沒有索引。中文書不論譯作或著作，有索引者比例極低。

臺灣在這方面還沒有走出頭緒，製作索引成了一種良心事業。前陣子 CSDN 技術書籍論壇上對於索引有熱烈的討論。侯捷怎麼看這個問題呢？

侯捷觀點

對一本技術書籍而言，索引是必要之物。對一本工具書而言，索引的地位更是關鍵。沒有檢索功能，百科全書尚能飯否？說「沒有索引也沒有帶來什麼不便」這種話者，讀書功夫還在幼幼班程度。

中文計算機書籍長期以來缺乏索引，原因之一是出版界沒有規範，又缺乏自省能力。原因之二是作譯者不願意花時間精力做「正務」以外的「庶務」。原因之三是目前普遍被使用的排版軟體的索引製作能力似有不足。前兩個問題都出於觀念，觀念正確了，事情就做得來。如果把索引視爲必要品，像目錄那樣（沒有目錄的書誰買），無索引不得出版，保證所有的書都有了索引 — 再困難都有人做。

專業排版軟體 TeX 有很強的索引製作功能，但學習和撰寫 macros 都不容易，輸出的 PostScript 或 PDF 又需製版廠商兼容配合（臺灣製版廠許多還只接受 DOC 格式）。普及度最高的個人排版軟體 Word 也有索引製作能力，性能尚待測試。我個人在索引製作上表現也不理想，1997 年出版的《深入淺出 MFC》沒有索引。由於意識到製作索引是對讀者負責，也爲了讓自己的作品更上一個檔次，晚近才硬著頭皮非做不可，海口也先誇下了（用來逼迫自己）。中譯本作法還算簡單，頁頁對譯[4]加上保留不少英文術語，我得以輕鬆留下原書索引。中文著作就難辦，前陣子才以硬功夫做出《STL 源碼剖析》的索引，做到快嘔吐，天可憐見。接下來還有一兩本著作即將完成，我決定好好探探 MS Word 的底線。再吐兩場可受不了。

製作索引是出版社抑或作者的責任？我以爲，索引做不出來，兩造各打五十大板。出版社應該提供資源、訓練、人力協助，作（譯）者應該把製作索引視爲正務 — 沒有你對索引詞條一一標記，哪個軟體聰明到自動探知你的心意？（這又回歸到老問題，作譯者若是沒有能力自己操控排版軟體，這索引難道自己迸出來？）

另一個對讀者負責的態度是勘誤。世上沒有完美無誤的書，各位上經典名書如《*C++ Primer*》、《*The C++ Programming Language*》、《*Effective C++*》、《*The C++ Standard Library*》、《*Design Patterns*》…的支援網站看看，每一本的勘誤都是密密麻麻。出版社如果有「承認錯誤就是砸招牌，自毀長城」的錯誤心態，趁早改過。依我

[4] 頁頁對譯的前提是譯者自己排版，否則來回溝通調整就要讓人瘋狂。人壽幾何，不堪折磨。

之見，沒有勘誤的書還表示沒讀者沒人氣呢。

勘誤比索引好做多了。早些年網際網路沒那麼發達，唯一途徑是在書籍新刷時付上一張勘誤表，或於紙本中直接修正。後者很耗成本，因爲製版和印刷以檯爲單位，一檯等於 16 頁或 8 頁，重新製版成本不小，出版社的意願也就不高。網際網路發達後，到處都是免費的網頁空間和虛擬主機，勘誤不但便宜而且及時。

勘誤是出版社抑或作（譯）者的責任？我以爲全是作譯者的責任。讀者指出疑慮，是否真爲錯誤，還待作譯者定奪。出版社應該站在提供網路資源、配合紙本作業的立場。勘誤是責任而不是服務，連這最基本的責任都不肯承當，出版社和作譯者還有藥救嗎？很忙是嗎？忙著賺錢嗎？我說過，免費網頁空間和虛擬主機到處都有，因此，即使出版社不肯修正紙本，作譯者至少應該提供網上勘誤。勘誤做不出來，兩造各打一百五十大板。

甲級電工啟示錄

客廳打算裝一盞扁扁的照明燈，做爲輔助光源。鄰居是甲級電工，其外甥女又是美靜的鋼琴學生，請他安裝再安心再方便不過了。打電話給他，甲級電工很忙，時間很難安排，來了又客氣不收費。我對不收費的東西向來戒慎恐懼，硬塞給他新台幣 500 元，也不知是多是少。

和室也需要一盞照明，決定如法泡製。甲級電工又來一次，又不收費，又給新台幣 500 元。怪的是每次和室這盞燈一開，時間稍長就有濃烈的焦味。可客廳的那盞沒有異樣。

過年前決定好好進行診斷。打電話給甲級電工，他搬家了。管理員及時介紹一位專門維護本大樓水電的司傅。看樣子是傳統學徒出身。他幫我打開帶有卡楯機關的扁燈，驚呼：電線都燒焦了。

真的，絕緣外衣都快燒光了，連銅線都酥了。估計再沒多久一定電線走火。扁燈上頭兩公分就是木製天花板，家裡都是木製傢俱…。不思不懼，思之悚然。馬上請司傅看看客廳那盞。啊，也焦了。爲什麼沒有焦味？一個待解之謎。

侯捷觀點

扁燈內的空間很小，一大堆應該拉到天花板上隱藏起來的電線，被甲級電工塞在狹小的空間裡。我裝了兩顆一百瓦燈泡，熱氣使得銅線外衣被烤酥烤化了。

我不懷疑甲級電工的實力，國家專業認證考試不是那麼容易通過的，而且他有大型建築實務配電經驗(新竹荷蘭村就有他的作品)。他不可能不知道二百瓦在 20 公分見方不到的密閉燈具內的熱度威力。他有實力，只是沒有用心。這一疏忽險些釀成一場火災。我很高興我還有機會以平靜的語氣告訴你這個故事。

甲級電工失去了一個客戶。如果他繼續不用心，他會失去更多客戶，並帶給別人和自己一些麻煩。至於那位黑手司傅，我留下了他的名片。

技術寫譯出版這條路，有實力的人可能不少，用心的人目前不多。

讀者來函與侯捷回信

●來函：我是才上大學的一名科班學生，由於在中學對計算機有一些了解，到大學來後感覺課程都很輕鬆。最近聽到不少這樣的傳聞：

1. 程序員沒意思，年紀大了就不行了，如果你不懂其他（如管理）之類，就該退休了。
2. 國內計算機行業沒勁，氣氛不好，有本事的人不一定發揮得到它的特長，還是出國去吧，在大學只要學英語就 ok 了，計算機以後再學。

這些傳聞在系中引起不少反應，我想問老師，計算機行業的人才意味什麼，這是一門行業還是屬於科研基礎。請老師對我們這樣才進入這門行業的人一點教導，幫我們指點迷津。

●侯捷：新竹交大（我的母校）計算機系，以前歸屬理學院（1972，計算機科學系），後歸屬工學院（1979，計算機工程學系），後又分展爲資訊工程學系和資訊科學系（1988）。顯然，大家都在尋找 computer science 和 computer engineering 的定位和區隔。不過我告訴各位，這些歸屬或命名或定位，更多是人爲的較勁、資源的重分配、大學聯考排名的考量。經過如此「風風雨雨」，經過科技的進步，一般大眾對電腦的印象，已經從冷氣房內的驕客，轉變爲日常家電用品。搞計算機的人，也從「科學家」變成了工程師。總的來說，一般大眾應該會認爲計算機

的工程性比科學性強些。

任何工程領域之前身（或背後），一定有其科學研究或尖端研究。即使 5000 年歷史之土木工程，也有尖端研究（研究如何在月球上乃至太空中或火星表面蓋房子…）。因此，世界上有大量的計算機工程師，也有爲數不少的計算機科學家。「計算機行業是一門行業還是屬於科研基礎」這樣的討論有什麼意義呢？沒有任何意義。

我雖不在內地生活，不知道計算機學習大環境，但你所說「在大學只要學英語就 ok 了，計算機以後再學」無論如何都不可能是對的。這論調有點像「把 C++ 學好了，編程就沒問題」。英語固然重要，專業領域的基礎訓練也極重要。如果你所處的環境沒有好老師，你自己想辦法找好書學習。

●來函：1 月 4 日最新到電子書籍：...本站所有電子書籍均是免費下載。完全是爲廣大 "無產階級" 的網友而共享。...我們相信一切東西都應是免費和共享的！

●侯捷：共產別人，很簡單；把自己共產給別人，不容易。你把你的一切都拿出來和大家共享，你才有起碼的一點點立場說「我們相信一切東西都應是免費和共享的！」。但即使你把你的一切都拿出來和大家共享，你也沒有資格要大家都和你一樣。盜拷別人的東西，大剌剌地說一切東西都應是免費和共享的，就好像偷了別人財物，然後逢人便說「大家都來拿唷，我好慷慨唷」一樣，可恥。

盜版問題有社會因素，我了解。有些事做了也就算了，爲惡不欲人知畢竟還是廉恥的表現。得了便宜賣乖，就令人難以忍受。

●來函：您提過整理資料的問題，但未提怎樣整理資料。我也 download 了一些資料，並沒有整理的概念，非常非常想知道您是如何和使用整理資料的。

●侯捷：資料如何整理？主要就是在各種地方留下各種型式的 index。書籍有 index，你可以有你所擁有的書籍的 index（如果你的書很多）。硬碟很大，裡面資料很多，你應該對自己的硬碟有一套管理辦法。讀書時多做 cross reference（交叉參考），別怕麻煩。心得眉批儘量多寫，順便記下日期，如此可比較不同學習

階段下的想法（if any）。大量額外文字寫在哪兒好呢？我的心得是寫在書頁空白處最好，永遠與書合爲一體，不怕散落不怕佚失。如果需要夾頁，一定順手寫上頁次。所有一切，都爲了節省將來再閱讀的時間。好書肯定需要一讀再讀的。教外別傳，難以言傳，只要用心，人人都可以發展出一套最適合自己的辦法。

喜歡以潔白無瑕的方式來表達對書籍大愛的朋友，對於直接在書上塗寫，恐怕像拿刀在愛人臉上割劃一樣難忍。每個人愛書方式不同，爾愛其形體，我愛其神髓。一本爬滿心得眉批甚至偶爾心情塗鴉的書，不正是個人學習生涯的最佳映照嗎。

●來函：我是大陸您的一位忠實讀者，最近迷上了 C＋＋和 Windows 編程，看了您關於這方面許多經典書的評價，在華儲網上書店買了您的....對於一個剛畢業的學生來說，這些書的價錢不算便宜，但是買到手裡看了以後，就覺得這些書不應該用金錢來衡量...。要學習一門學問，當你想買書的時候，就一定要買值得一看的、經典的書籍。

●侯捷：當你開始知道閱讀書評、接受不以頁數定價的書籍、慎選書籍，你便走上了一條好的學習道路。

●來函：我有一個建議,同時也是問題:您對於類似 Borland C++ Builder 的 RAD tool 的看法如何?有沒有可能爲這類型工具作深入的介紹?

●侯捷：RAD 很好，但 RAD 使用者需有正確的認識：仍然需要對基礎原理做相當程度的理解。RAD 是軟體開發的好幫手，但若使用者無良好基礎，無法良好駕馭這種高層次工具。我沒有撰寫 Borland C++ Builder 書籍的計劃，很大因素是，我研究的是核心（相信你希望我寫的也是核心），而 Borland C++ Builder的核心 VCL 以 Object Pascal 寫就，這不影響它的功能和用途，卻影響我的興趣。目前我正撥一些時間研究 Qt，它跨平台，我對它的包裝和它的訊息機制很感興趣。

●來函：對於您的書籍的簡體中文版的翻譯，我覺得您可以再斟酌斟酌。例如：...。竊以爲我們會有不同意見，主要源自兩岸語文習慣的不同。正因爲如此，才需要更多的交流。既然您出版簡體中文版本，總得照顧大陸的習慣吧。

侯捷觀點

●侯捷：我的書籍（不論著作或譯作）的簡體版，都是由大陸譯者將繁體版轉譯過去。這麼做有絕對必要，因爲侯捷無法精準掌握大陸日常用語和技術用語。我定了少量幾個絕對必須遵守的原則，其它都由轉譯者自己掌控。如你所說，許多問題源自兩岸語文習慣的不同，大家彼此調適共同努力。我也在改變之中☺

●來函：最近看資料，MICROSOFT 要發佈 VISUAL STUDIO.net，它把所有的語言都統一了起來，那就意味 MFC 快被淘汰，我想就這問題請教一下。

●侯捷：相信你已發現，你的話說得過早。MFC 並未被淘汰，它在 Visual Studio.NET 的 VC7 中活得很好，也有了一些加強。

也許你不知道業界工程師的心情。最近我徵詢好幾位業界朋友（都使用 VC++ 開發系統級軟體），他們對 Visual Studio.NET 的 C#, .NET framework 的推出，態度可用一句話形容：漠不關心。他們唯一關心的是 VC7 的相容性。這種心情可以理解，業界要的是持續穩定，如果你在某個開發環境上發展了許多產品，你最關心的便是保證你的投資和保持你的優勢。這也是一個負責任的工具廠商必須永銘心中的。世上已有許多重量級軟體以 MFC 開發而成，如果哪天微軟不再支援 MFC，或新版 MFC 沒有良好的回溯相容，這些很有份量的應用軟體大廠會不會剝了微軟的皮？回溯相容性不佳的臭名對工具廠商是致命傷 — 誰敢用隨時可能改朝換代的工具？

世界當然要前進。演化（evolution）的代價最小。演化無法讓大家滿足的時候，就會發生革命（revolution）。MFC 顯然還有相當壽命。但當它引退的那天到來，我也毫不意外。MFC 的引退如何才能不引起業界反彈，其後繼產品如何保持最佳相容性，在在考驗微軟的智慧。一旦哪天 C# 有了極強大的勢力（這是微軟的盼望），或多數大廠一如微軟所冀望地逐漸改採 C# and .NET framework 做爲開發平台，微軟便可進行強勢引導。

前面我所說，是「以 VC++ 開發系統層級軟體」的業界朋友的態度。領域不同或不同狀況的朋友，可能對 C#, VB.NET...抱持很大的興趣或期待。這是一場勢力拉鋸。誰手上的籌碼充足，誰就說話大聲；誰有壓倒性籌碼，誰就可以有壓倒性作爲；誰因爲壓倒性勝利而腐化，誰就成爲下一個被推翻的對象。合縱連橫，策略

權謀，分久必合，合久必分，世道其中矣。單純以技術優劣來評論天下，都太過單純。豈不聞「富不過三代」？正因富者可能驕縱、奢華、承平貪歡，貧者可能勤奮、儉約、秣馬厲兵，雙方才有拉近地位甚至互換角色的機會。

拿世事道理看業界發展，我得以自在高吟「隆中對」☺。

●來函：您的演講中，提到翻譯的例子：...可能都有兩岸習慣的差異，因此學生我從理論和實際二個角度都感到沒把握。望指正。

●侯捷：我所說的只是我的看法。我的看法不是教科書。人各有體，無關正誤。文字成就個人風格，個人風格成就個人魅力。我聽過一種說法：技術書籍中不應該（不需要）個人風格。這很可笑。文如其人，言爲心聲，說話寫字（穿衣吃飯走路睡覺）自然而然就是一個人的風格。就算機器寫字說話，不同的機器也會反映出設計者的不同風格。

●來函：我一直是學習 C++以及在 VC++上編程的“狂人分子”但最近聽別人說什麼...本人想請教侯老師，您對此問題是怎麼看待的？也就是非得在二者（Delphi & VC++）比個高低，在綜合評測中到底哪個能略勝一籌呢？

●侯捷：我沒有資格對 VC++ & Delphi 做評比，因爲我對 VC++ 的熟悉遠遠超過我對 Delphi 的熟悉。此外我也沒有興趣評比優劣。我唯一可能寫的評比文章（目前有一篇“Java vs. C++”，尚未發表），只會列出事實，不會給出高下，高下由讀者自己評斷。這類工具發展到今天如此精良、各有所長的地步，我認爲，只有適用與否，沒有高下可言。

●來函：我曾經在 LINUX 下對 SOCKETS 編程做過一定的研究工作，在 LINUX 下各種宏、函數使用都非常的簡便而清晰，而 Microsoft 卻提供了一大堆令人生畏的宏和運作機製，您卻能將其運用得如此爐火純青，實數不易！Visual C++功能極爲龐大，您對其中哪些技術的前景最爲看好呢？在這軟件海洋中，我確實感到有一點迷茫，您能否給我一點明示呢？

●侯捷：MFC macros 的運用，總共不超過 5~10 條規則，何來爐火純青之說。你

若說侯捷把它們的設計原理剖析得淋漓盡致，我就欣然接受你的讚美☺。很多人都對 MFC 那一堆 macros 深懷惡感，我沒有這種感覺。我覺得設計它們的人很厲害。當我清楚了那些 macros 的作用後，使用它們再簡單不過。把 macros 當成語言的延伸，當成一個 operator 或一個 keyword，不就好了。把 `DECLARE_DYNAMIC()` 和 `sizeof()` 視爲同一類東西，何難之有？是的，沒有，而前者是 MFC macros，後者是 C++ operator。

「Visual C++功能極爲龐大，您對其中哪些技術的前景最爲看好呢？」我對它發展多年、市場佔有率高、穩定性高，最看好☺。請注意，我談的都不是哪些技術如何前瞻、哪些特性如何尖端。

許多人批評 MFC 運用了微軟（VC++）的許多**獨特語言特性**，令我不解。我認爲說這話的第一個人對 MFC 和 C++ 語言沒有深刻了解，其他人則是人云亦云。MFC 之所以只能用於 Windows 平台，因爲它所呼叫的底層是 Win32 APIs。你能期望一個呼叫 Win32 API 的 class library 用於 Linux 或 Mac 嗎？MFC 提供了一些 macros，你不能不用，不能用錯，但它們的使用法則十分規律，「不會用」或「誤用」實乃因爲程式員自己未能好好認識這些 macros，換句話說，太多人走都還不會就妄言跑，跑不快又怪天氣不好，風向不對，最後怪老爸老媽給生一付小短腿。MFC 設計出一些 macros、採用自製的 RTTI 機制，但完全符合 C++ 標準。macros 是語言層級的東西，適用於任何 C++ 編譯器。我模擬 MFC 而作的 MFCLite3（見《多型與虛擬》2e 第 6 章），除以文字模式取代圖像模式、以 ANSI file functions 取代 Win32 file APIs 外，主要架構都和 MFC 相同，適用於各種 C++ 編譯器和作業平台，便是明證。Qt 之所以可跨各種作業平台，因爲它不直接呼叫任何平台的 APIs；Qt 身上一樣有類似 MFC 的那些 macros（請參考《程序員》雜誌 2002/02 的 Qt 專欄）：

```
class Demo : public QObject
{
  Q_OBJECT // 這是一個 macro，詳見 src\kernel\qobjectdefs.h
public:
  ...
private:
  ...
};
```

侯捷觀點

```
// 摘自 src\kernel\qobjectdefs.h
#define SLOT(a)       "1"#a     // 這是一個 macro。注意它用的 #
#define SIGNAL(a)     "2"#a     // 這是一個 macro。注意它用的 #
```

●來函：我以前做過中學物理教師 6 年，對計算機感興趣，後來在公司打工，編程、測試、文檔、售前、需求分析、項目規劃都做過，參與過大型項目的開發，也主持開發過中等項目的開發。總體感覺 IT 行業很浮躁，一種語言用過一個月就認爲自己是專家，我很反感。另外，由於在公司幹活，需要用什麼語言就用什麼語言，所以哪一樣都不精，很想深入研究一下。所以拒絕了別人的邀請，干起了教師的行業（在朋友的公司也兼職）。

在學校，由於我做過項目，所以很受學生歡迎，但是基礎感覺有欠缺，在 CSDN 上看到了您的情況，才注意到您，（以前在公司上網規定很嚴，我雖然當了項目經理，還是不能隨意上網，並且經常封閉式開發，所以不知您的大名），很希望在三到四年的時間裡，成爲一個基礎理論扎實並比較熟悉業界最新技術的稱職教師，您的道路就是我的夢想。

中國的程序員沒有規範的培訓，並且有很多的錯誤認識，把寫代碼看成是水平最高的（這種人很多，在我經過的項目中，項目經理必須掌握樣樣比程序員高才行，否則項目很容易流產）。以後幾年，IT 行業會越來越規範，越來越冷靜，中國 IT 行業需要您這樣的專家。

●侯捷：中國 IT 行業需要您這樣胸懷理想又務實的教師。（這封信我的回覆不多，原信照錄，我以爲其中很有省思價值）

●來函：近讀侯 sir 文章，發覺對“因特網”一詞的翻譯頗有嘲笑之意。可能是先入爲主的緣故，我對這個詞的翻譯倒覺得不錯。如同侯 sir 最愛喝的咖啡以及人人都喜歡坐的沙發一樣，將 Internet 翻譯成因特網所採用的也是音譯的方法，讀起來琅琅上口，而且網絡互聯的最終目的不就是要讓人感覺只存在一個網嗎？如此聲情並茂的絕好翻譯，怎麼能說翻的不好？二十年來，恐怕只有基因（gene）一詞的翻譯能夠與之媲美。

●侯捷：「因特」一語，只有音譯，何來聲情並茂？關於咖啡和沙發，我是這麼

想的，科技術語多採意（義）譯。咖啡和沙發，只是一個名稱，談不上什麼意義，採用音譯也理所當然。基因（gene）和萬維網（world wide web），則是音義俱佳的翹楚。

好的譯名需要一個演進過程。一開始接觸 internet 的人，沒有想到「互聯網」這個好詞，隨口說出「因特網」。然後，好詞出現了，被大眾接受了。這是一個演進過程。我並不嘲笑最初創造「因特網」一詞的那個人，只是聽說大陸官方要求媒體必須採用「因特網」，我個人覺得很…呃…該怎麼說才能不傷感情呢…很…呃…有趣吧。

最後，如同我在該文所說，請把「翻的不好」改爲「譯得不好」，好不好 ☺

●來函：我已從事多年的計算機圖書翻譯工作。在這期間，不斷進步，也不斷體驗成功和失敗。從最開始極差的翻譯習作開始，到現在自己覺得“尚可”的水平，感覺再難有一個很大的提高，因此頗爲迷茫。至於原因，主要在於追求新知和花在深思熟慮的時間較少。大陸的翻譯報酬極低（我只能拿到 30～40 元／千字），因此必須多譯，但這樣就造成了一個死循環——爲了生活，我很難花上一年半載的時間去專門學習新知識，專門提高文學素質，專門去深入一門計算機學科。由此造成的翻譯疏錯，恐怕會令許多買書的人感到氣憤吧？但人總是要進步的，只是進步的速度...。

●侯捷：在勞動與酬勞不成比例的情況下，的確很難產生專職優秀寫譯人才。大環境不變，我也想不出什麼好辦法。我個人認爲，技術書籍作者/譯者，並不需要文學素質就可以讓自己的作品到達「達」的境界。只要通順，就「達」。要通順，很簡單，以讀者的心情多看幾遍，多改幾遍，肯定「達」。這是我很強調的沉澱過程。有人說趕快都來不及了，還沉澱呢。這是層次的不同，選擇的不同 — 當然也造就結果的不同。

●來函：我對編程很感興趣，但是關於做一名程序員，我卻有些困惑：計算机技術發展可以說是一日千里，每天都有新的技術不斷湧現，書架上的書每天也在翻新，似乎不管我如何努力，都難以追赶軟件概念制造者們的步伐，更談不上可以有所創新，究竟一個程序員應該如何在追赶最新工具，概念和做在熟悉平台上進

行自己的開發之間找到一個最優點？

●侯捷：基礎學問如萬古長空，開發工具如一朝風月。不可萬古長空不明一朝風月，不可一朝風月昧卻萬古長空。

●來函：經過兩年多的編程實踐，我發現了自己有很多不足，也深深地愛上了程序員這個工作，眼前，就有一個很好的公司在等待我，可是，我不知道是應該進學校深造基礎課，彌補本科學習的不足，還是繼續工作呢？以您的觀點來看，對於一個程序員來說，是不是掌握了某種編程語言的語法，能夠熟練地應用，就算是合格了呢？如果想成爲一個高手，有沒有必要進課堂學習計算機技術的基礎呢？（比如，數據結構，操作系統，編譯原理等，在您的工作中，您覺得這些課程確實對自己很有幫助麼？）

我覺得從發展的角度來看，即使我現在也許可以獨當一面，可是由於基礎的不足，我可能沒有辦法編出來更高水平的程序，競爭不過那些科班出身的程序員們。可是，我又不知道是不是現實中，基礎知識真的那麼重要？是不是會成爲更高發展的羈絆？因爲，好像好多高手，都不是計算機專業出身的，包括您在內。我是真的很矛盾。可能一個是短期利益，一個長遠利益的問題吧？

另外，程序員將來的出路在哪裡呢？也許歲數大了，只能在管理，策劃上進行突破了吧？

●侯捷：程式語言是程式員最基礎的功夫，必須熟練它掌握它。但若只是如此，編寫不出好程式、大程式、專業程式。大家都會說中文，獨獨有人旁徵博引，左右逢源，字字珠璣，文思雋永，這是爲什麼？功夫在語言之外！

語言是首要條件，但不是決勝條件。功夫在語言之外，決勝在語言之外。

PASCAL 語言發明人 Niklaus Worth 博士說：Programming = Data Structures + Algorithms。這兩門功課對於編程非常非常重要。操作系統（你的工作平台）也有必要深入。至於編譯原理，運用在工作上的機會比較少。

基本功有學習的必要，但不一定得在課堂上學習。書籍是你最好的、長長久久的

老師。有些課程需要基礎，基礎又需要基礎，迷霧需要點撥，方向需要指引，有位好老師是最幸福的。但人生之中得遇明師的機率很小，真的。

你需要的基礎知識，視你選擇的路而定。任何技術，都有其基礎知識，而它們都很重要。你說好多高手都不是計算機專業出身，但我看有更多高手是計算機專業出身。你認爲侯捷是高手，可在許多計算機專業領域（只有科班人才會接觸的科目）裡，我的基礎薄弱（這說明「弱水三千取一瓢飲」的重要）。我並不強調出身，我對門第貴族那一套嗤之以鼻，但該有的訓練必須要有。

讓我再強調一次，可以上課學習，可以旁聽學習，可以自修學習。學習和拿學分拿文憑是兩回事。

「程序員將來的出路在哪裡？」你把你的事做好，自然就有出路。景氣差也有人賺大錢，景氣好也有人賠光光。大家別老擔心這種事了。

●來函：我是一名程序員，已從事這個職業兩年半了，曾經買了許許多多的計算機方面的書籍，現在回頭來看，也發現好多都可以看做垃圾了，但是同時要扔呢，有有點捨不得，頗有點“雞肋”的感覺，畢竟那些書都是我用自己的工資買來的。不光是計算機書籍，包括中小學生的參考書籍，真是氾濫成災，沒有什麼內容。還有就是一些不知名的人物寫的一些書，每頁印幾行，每行幾個字，湊成厚厚的一本，如果說書有內容，我很欣賞，但是去一看，我敢保證 98%是垃圾!

●侯捷：濫竽充數的問題，從來不曾斷過。就像光和影一樣，有陽光的地方就有陰影。解決這個問題，需要讀者的覺醒：慎選好書。濫竽沒了市場，就失去生存的養份；好書有了成績，自然有蓬勃的生命，好作者也就能開展更多的努力。

●來函：我很想知道作爲中國人是否應該爲研發一個有中國自主知識產權的產品而努力，更不知道這個目標到底離我們有多遠。

●侯捷：一個世紀以來，中國人做事滲雜了太多民族主義。是的，當然有原因：中國人歷經太多苦難，恨不得強大再強大。所以研發一個屬於自己的 XXX，就成爲常聽的口號，扛起民族的大纛和中國人的希望。

侯捷觀點

很沉重呀。

OS 的研發，是基礎研究水到渠成的結果。如果單單只爲民族主義，研發成果我想不會好到哪裡去。強扭的瓜不甜。Linux 是跨國努力，源碼開放，何不從它著手改造？順帶一提，大家都有權享受 open source 的成果，但中國人（華人）在 open source 領域貢獻了多少？值得深思。

●來函：先生，最近我做 ... 閱讀 boost 庫的代碼...加上最近一段時間我所做的那個 software 完成，其中大量使用 STL 和 OO 特性，前後計 10,000 行代碼左右，算是略有一些實戰心得了。綜合讀書、讀碼、寫碼的感受，我越來越有一種疑慮，想請教先生的看法。

我曾經非常推崇 template 的解決方案，並推薦同好者學習 Boost 中的技巧。但是我現在越來越感到，在實際開發中，我覺得類似 boost 中某些模板技巧，似乎正在重蹈 macros 的覆轍，變得越來越複雜，越來越 tricky，代碼難以編譯不說，更難以閱讀和維護。某些技術，實在有 "爲用模板而用模板" 之嫌。例如 boost::regex++，固然將模板技術運用得高超，但是其效率一般只及 Perl 的 1/4，而美國微軟研究院一研究員，用比較簡單的 template 技術，實現的 GRETA 庫，同樣處理 regular expression，其效率足以與 Perl 抗衡。此事對我震撼不小，難道我們學習模板技術，泛型技術，只是爲了 "炫耀" 一種編程技巧碼?

我現在非常質疑這個方向是否正確。不知道先生怎麼看。

我現在以爲，C++中最重要，最有效，最常用的那部份，還是基本的 C++和標準庫中的那些東西，還是 OO 和 STL 所展示的 generic programming 技術。一味追求 template 編程技巧，恐怕不是正確的方向。即使是 library 的編寫，對於 template 能力也不應該濫用。普通 C++愛好者學習，還是應該以標準庫和 STL 爲准，不宜過早閱讀 Boost source，以免 "走火入魔"。

另外，我感到在一種語言中泡的太久，漸漸有一種局限，無論 C++是多麼偉大的語言，也不可避免地限制了我們的思想。似乎應該擴大自己的眼光，了解一兩種其他語言的思想。我覺得 Python 似乎是個很適合的點。先生的意見如何?

侯捷觀點

●侯捷：你終於有了一些實戰體會，開始 real world project。這和學校中的習作截然不同 — 規模還在其次，最主要是工作態度。你的東西要賣錢，你的產品要維護，你的肩上有了責任。

OO 有其價值，非 GP 可取代。GP 自有擅場，亦非 OO 可到達。application framework 如何以 GP 實現？不可能！STL 揭示的彈性與靈活，如何跳過 template 完成？也不可能！OO 要求把一切相關 data 和 functions 封裝起來，GP 卻要求把 data（containers）和 functions（algorithms）拆開（再透過 iterators 橋接起來），這是相悖的兩種思考。

不同的場合運用不同的技術，才能紅花綠葉相輔相成，威力加乘。GP 在 data structures + algorithms 身上找到了一個絕佳發揮場所，其爲每個程式所必須，又與框架系統不生任何杆格，而且 STL 的複雜度遠小於 application framework，很容易就讓人們良好地接受它、運用它、深入理解它。

任何技術都需要一群急先鋒和偏執狂，扮演探究極限的角色。《*Modern C++ Design*》和 Andrei Alexandrescu 那群人，就是 GP 急先鋒，Alexander Stepanov 則是 GP 教主。沒有這些人的狂熱與偏執，我們不知道 GP 的極限。他們的研究是否已達實用價值，由時間判斷，他們的立論出於偏執或智慧，由我們判斷。

你已經進入業界，你應該已有體會，卡了位的公司要求穩定成長，還在奮鬥求生的公司追求大破大立。不同的位置，不同的角度，每個人期待的都不相同。一旦你經驗老到，你就再也不會對新技術新名詞而瘋狂而一頭栽入了，你會冷靜判斷，決定自己的方向。成熟老練之後，有些人會後悔年少的輕狂孟浪。其實也不必，那就是成長過程。而且，爲某些事不夠理智地瘋狂，是許多人共有的經驗☺。

年輕朋友常落入「以技服人」的炫技窠臼，並且老愛把新技術、新名詞掛在嘴上，爭取在任何時候以經過刻意裝飾的「不經意姿態」流露幾許科技貴族氣質。這種情況我見很多。我不是說你，你的表現很自制沉穩。在近乎狂熱地推崇 template 的過程中，你寫了一些帖文，我看過。正因爲有那樣的過程，你才有能力做今天的深刻反思。

侯捷觀點

網絡上有無數 libraries，其中不乏好作品。但是剖析源碼茲事體大，它需要許多時間和精力。所以一定要慎選對象，不可輕率爲之；你的時間和精力都容不得虛擲。我認爲，只有價值被百分百認定的大型卓越作品，才值得剖析源碼，從中吸取深層技術的養份。一樣米還養百樣人，哪來被百分百認定的大型卓越作品？唔，我說的是被你百分百認定，不是被百分百的人認定。

「我感到在一種語言中泡的太久，漸漸有一種局限」。確是如此，所以我曾要你注意 Java。我自己也研究 Java 和 C#，從中尋找新的養分、新的刺激。但是注意，唯有曾經對某種語言（而且是制高點上的語言）有過深刻的體會，才適合以它爲基礎做這類橫向攻錯。人力有時，以有涯追無涯，殆矣，橫向攻錯並不是要對每一項牽涉領域都專精，而是要從瀏覽之中觸類旁通，觸發靈感。必須先對某項技術或語言非常精專，如此才立於不敗之地，才能做橫向攻錯。

●來函：我是一個程序員，有 4 年工作經驗，最近兩年一直集中在做 j2ee 這方面的開發，這兩年的學習過程中，我自認對 Java 語言本身和相關的核心 api，擴展 api，oop 有了較好的認識和理解，也具備相當的工作經驗。我目前的問題是，現在我想再學習 1，2 門語言，對於 script 語言，我選定 python。對於系統語言，我在考慮是否應該重新學習 C++，我對語言自身和相關 OO 部分的東西感覺不會太困難，但是 C++ 自身的特點要求學習者本身去了解精通特定的平台和框架体系，這又是我所不情願的，一來我沒有實際經驗，很難說我會用 C++ 做哪一方面的東西，我主要還是做 Java。二來學習這些東西要耗費大量的時間和經歷，學習 Java 的各類 extended api 已經讓我處在一種不停奔跑的進程中了。過去兩年裡我除了 Java 沒有接觸任何其他語言，我認爲精通一門，比粗通多門要好得多，更多的精力可以放在更高層次的問題上。問題是很多人的標準裡，不精通 C++ 都算不上真正程序員。尤其現在 Java 火熱，太多的 C++ 程序員轉過來做 Java。在這種情況下，沒有 C++ 背景的人，很多人懷疑你自身的能力。很荒謬，可是是現實，包括大陸某知名的 Linux 開發人員都對我說學不好 C++ 的人，才去學 Java 這樣的話。很多公司寧可招 C++ 的高級程序員，讓他們改學 Java，也不要純懂 Java 的人，很多人認爲我這個時候學習 C++ 年紀已經太大了，我已經過了 25 歲。自己也頗爲困惑。我無意在 30 歲以前去做 SE 和 PM，還是覺得做程序員比較有樂趣，目前對出路比較迷惑，懇請指點一下。

侯捷觀點

另：發現您的大部分作品都局限於比較具體技術方面的東西，對於一些較高層次的，涉及較少，如沒見到您的作品或譯作有 design pattern, anti-pattern，refactor 這些更大局一點的東西，諸多經典的 OO 書籍中，您還是以 OOP 爲主，似乎也只提到 DP，也無和 SE 相關的作品。感覺還是一種在武俠中追求個人練功極致的時代，爲什麼您不投入更多的精力去編寫翻譯一些和具體語言無關較高層次的設計和 SE 方面的東西？

又：在您的網站上發現了您翻譯的 *Refactoring* 一書，此等好書不知大陸是否已有簡體出版計劃？侯先生爲何不向大陸讀者多推荐一些這方面的書？目前唯技術論實在是太濫了，看過太多程序員可以連續熬夜寫幾萬行代碼，卻連基本的代碼規範都不肯遵守，實在是無言以對呀。

●侯捷：我曾有一篇《C++ 的沉迷與愛戀》，但我不知道大陸 IT 業界迷戀 C++ 到了這般地步 — 如果你的話有代表性。看來 C++思想即將取代儒家思想☺。

精通一門語言確實比粗通多種語言要好得多，鼷鼠五技而窮。「學不好 C++ 才去學 Java」這種話太荒謬，太不可思議。就算換成「學不好 C++ 才去學 VB」我都覺得很荒謬。語言的選擇主要在適用性。學術研究是一回事，工業實戰是另一回事。大家都忘了小平同志的教誨，一個勁兒只追求自己心中的白貓黑貓。

你說「C++ 自身的特點要求學習者本身去了解精通特定的平台和框架體系」，這並不正確。你說的是 application framework，是 class library，不是 C++ 語言本身。Java 之所以跨平台，不在語言而在虛擬機器（當然啦，Java 語言和 Java 虛擬機器是焦不離孟）。C++ 也有跨平台的 application framework[5]，Qt 便是；C++ 也有跨平台的小型 framework，STL 便是。只要遵循 C++ 標準，便能跨不同的平台；運用它們，雖不至於「一次編譯到處運行」，卻可以「一次編程到處編譯」。

對於那些不願遵循基本代碼規範的孤芳自賞型「高手」，我個人永遠保持工作上的距離。下面這一段文字，希望對他們有點啓發。Donald Knuth 於 1996 年接受 Dr. Dobb's Journal 訪問時，被要求對美國程式員說幾句話。他說：『我第一想說的是，

[5] Application framework 是一個專有名詞，不是一般的 framework。例如 STL 是一個 framework，但不是一個 application framework。

當你撰寫程式的時候，請想像你正在進行一件文學工作。你應該寫出一些給人類閱讀的詩篇而不只是讓機器遵循的指令。你的程式愈具可讀性，你的工作愈有效率，因爲你可以在今天看懂它，下個星期看懂它，三個月後仍然看懂他，負責維護和修改你的程式的繼任者也可以輕易看懂它。』

「爲什麼您不投入更多的精力去編寫翻譯一些和具体語言無關較高層次的設計和 SE 方面的東西？」呵呵，一步一步來！沒有過去十年堆壘的厚實基礎，我如何構築一個自己完全掌握的堅實體系？我以書籍爲媒介進行教育，我需要一個完完全全放心、值得推薦的好書體系，初階、中階、高階…，語言、應用、系統…，一步一步往上攻堅，也成就我個人的進步。從某個角度說，十年，呃，請讚許我是個堅毅的人。「和具体語言無關較高層次的設計」，你提到的《*Refactoring*》不正是我的開始嗎☺。至於《*Refactoring* 侯捷譯本》發不發行簡體版，主動權不在我，在拿到簡體譯權的出版社身上。

25 歲學 C++，嗯…我 27 歲才學 C 語言，30 年才初次接觸 C++，36 歲才認真學習 C++。希望你信心大振☺ ✍