未公开的 WINDOWS 核心技术

全 正 闫丰学 梅士兵 译
文 极 赵立伟 郑 方 校

WINDOWS UNDOCUMENTED

ANDREW SCHULMAN, DAVID MAXEY,
AND MATT PIETREK

清华大学出版社

未公开的 Windows 核心技术

[美] *Andrew Schulman　David Maxey　Matt Pietrek* 著

全　正　闫丰学　梅士兵　译
文　极　赵立伟　郑　方　校

清华大学出版社

Undocumented Windows

——A Programmer's Guide to Reserved Microsoft Windows API Functions

Andrew Schulman David Maxey Matt Pietrek

Addison-Wesley Publishing Company

Reading, Massachusetts • Menlo Park, California
New York • Don Mills, Ontario • Wokingham, England
Amsterdam • Bonn • Sydney • Singapore • Tokyo • Madrid
San Juan • Paris • Seoul • Milan • Mexico City • Taipei

(京)新登字 158 号

未公开的 Windows 核心技术

[美] Andrew Schulman David Maxey Matt Pietrek 著
金 正 闫丰学 梅士兵 译
文 极 赵立伟 郑 方 校

清华大学出版社出版
北京 清华园
门头沟胶印厂印刷
新华书店总店科技发行所发行
☆
开本:787×1092 1/16 印张:34.25 字数:852 千字
1993 年 11 月第 1 版 1993 年 11 月第 1 次印刷
印数:00 001—10 000
ISBN 7-302-01411-6/TP • 546
定价:34.00 元

序　言

欢迎诸位前来探索神奇的 Windows 世界的奥秘!

大多数有关 Windows 编程的书籍,就算是其中的佼佼者,你都可以猜测到书中会讲述些什么样的论题以及书中的程序会具有什么样的风格。随便打开一本书,翻到任意一页,你都可能发现 TextOut()、BeginPaint()以及其它类似的 Windows 函数。

如果我们所料不错的话,本书的内容会使大多数 Windows 程序员惊奇不已。随意翻到哪一页,都会让人感到"触目惊心"、"赏心悦目"。本书包含下列的议题:遍历任务表,反汇编 Windows 函数,把 HWND 看作指针,察看 WinExec 产生的中断,将原子句柄右移 2 位……。这些东西对于 Windows 编程都很重要,而在以往的书籍中却被忽略了。另外,本书中程序的外观也与众不同:Main()代替了 WinMain(),printf()代替了 TextOut(),处理单一消息的小函数代替了庞大的 switch 语句。

1. 编写本书的目的

坦率地说,开始时我们中至少有一个人并没有打算写这本书。我们真正想写的是一本有关 Windows 编程的书。我们刚刚写完《Undocumented DOS》(《未公开的 DOS 核心技术》,中文版 1992 年由清华大学出版社出版),希望改变一下风格,转向有关美好、整洁、公开的接口——Windows API 的写作。

未曾料到的是,我们很快碰到了有关未公开 Windows 的难题。由于工作需要,我们中的某位需要查阅一些商品化的 Windows 程序,但是时时碰到一些未曾公开的函数调用。这些函数从未出现在 Windows 软件开发工具包(SDK)、设备驱动程序工具包(DDK)、甚至 Windows "Open Tools"资料中。很多 Windows 程序都调用了 GlobalMasterHandle()、GetHeapSpaces()以及 SetInternalWindowPos()之类的函数,它们从未在任何文档中公开过。真是不可思议!

其实,并非不可思议。编写《Undocumented DOS》的工作经验使我们对于 Windows 的未公开性也有了一定的思想准备;何况在有关 Windows 3.0(1990 年 5 月)的报道中,就已经透露过有关 Windows 内部资料和未公开函数的存在。

我们在 CompuServe 和 Usenet 这些网络的有关资料中,也看到了很多关于未公开 Windows 的零星注释。像未公开的 DOS 一样,这些注释也很令人费解。例如,也许有人发现一些 Windows 程序调用 InitTask(),可是 InitTask()却不在 WINDOWS.H 中! 或许,他们会说,哈,"人赃俱在":Microsoft 使用了未公开函数,美国联邦贸易委员会(FTC)可以据此追究 Microsoft 了。可是,后来的资料表明,InitTask()只是所有 Windows 程序都使用的启动代码的一部分。不管怎样,这使我们认识到:如果将这些资料系统化,使未公开的 Windows 资料成为标准文档,这将是一件有益的工作。

经过两年艰苦工作,本书终于与读者见面了。《Undocumented Windows》(《未公开的 Windows 核心技术》)尽可能详尽地讲述了 Windows 的核心:KERNEL、USER 和 GDI。本书

系统讲述了 Windows 3.0 和 3.1 零售和调试版本在标准和增强模式下未公开的函数、数据结构以及消息。另外，我们还讲述了已公开函数和消息中的未公开属性，例如，InSendMessage()返回值的真实含义、WM_NCPAINT 的 wParam 值、……。

换而言之，整本书的内容只覆盖了 KERNEL、NUER 和 GDI。另外一个全新的低层领域，其中有些内容是未曾公开的，有些内容虽曾公开，却晦涩难懂。该领域包括：DPMI、虚拟设备驱动程序(VxD)、虚拟机器管理程序(VMM)、INT 2FH 以及由 Windows 提供的软中断服务、设备驱动程序、Windows DOS 扩展程序(extender)、文件格式、WinDebug、Windows 与 TSR 和内存管理程序的接口、SmartDrive 接口等等，不一而足，这需要另外一本书来讲解。(不错，我们正在编写！书名可能是《Dirty Windows》)。

本书的大部分工作是在 Windows 3.0 盛行的两年内完成的。一些 3.0 中未公开的函数在 3.1 中已公开。这是为什么呢？实际上，这是由于许多商业开发者需要使用一些未公开的然而是关键的函数，从而迫使 Microsoft 将它们公开。那些已在 Windows 3.1 中公开的函数恰恰证实了本书的实用价值，而不是削弱了本书的价值。本书中其它有用的未公开函数也迟早会被 Microsoft 公开，因此你现在就可以使用，这样在竞争中你又有了"提前两年"的优势。

2. 本书内容

下面列举出我们认为本书中比较精彩的一部分内容：

- 如何在 3.1 和 3.0 中创建"drag 和 drop"服务程序(见第 6 章 DragObject())
- 5 个著名的 DC 在哪里？(见第 6 章 DCE)
- 如何管理原子？(见第 5 章 AtomTable)
- 所有不同的 Windows 句柄(任务，实例，模块，PSP/PDB，任务队列)是如何联系的？另外，给了一种句柄，如何派生出另一种？(见第 5 章的介绍)
- HWND 指向什么？(见第 6 章 WND)
- HDC 指向什么？(见第 8 章 DC)
- Windows 3.1 中的标志是什么？(见第 5 章 GetAppCompatFlags())
- "free system resource"(空闲系统资源)问题究竟是什么，而 USER 的局部堆从 3.0 到 3.1 是如何变化的？(见第 6 章的介绍)
- 消息何处去？(见第 5 章 Task Queue，第 6 章 System Message Queue)
- WinExec()返回什么？(见第 5 章)

实际上，远远不止这些，但是上面列举的问题提供了本书所要回答问题的一些思路。

3. "句柄"指向什么？

当然，如果将未公开的函数、消息以及数据结构应用到 Windows 程序中去，就要像第 1 章所强调的那样，保证安全，防止负作用。除此之外，我们还希望通过本书，读者对 Windows 能有更深的"理解"，能更上一层楼。

我们所指的"理解"，在 Windows 程序设计领域似乎已经变成了一个含糊的字眼。"你只管使用句柄，而不用管它指向什么？"——这句话常常被人说起，以至于成了 Windows 程序设计的"经文"——被念叨来、念叨去，却无人去仔细考虑它的真实含意了。

在本书中，我们将告诉你句柄指向什么。例如，HTASK，GetWindowTask()的返回值，也是 EnumTaskWindow()需要的参数，是指向 Task Database 远指针中的段值(选择符)部分。如果不知道 Task Database 是什么，你可以使用 HTASK 而不必关心它指向什么。但有的情况下，知道了 Windows 内部数据结构就能获得重要的技术。作为例子，你可以参考第 5 章的引言中的 HANDLES. H 和 HANDLES. C。

应该牢记的是："知道，然后忘却"是一种好的教育手段，而"完全不知道"却不是。知道了 Task Database 是什么，而在使用 HTASK 时将其忘记，与根本不知道 HTASK 是什么，有着天壤之别。

言重一些，"你可以使用句柄而不用知道它指向什么"这句话实际上是鼓励不求甚解。虽然短时间内没什么问题，却存在潜在的危险。这样会产生一批只知道从 Petzold 的书中或 SDK 示例程序中抄袭、拼凑程序的程序员。他们，如一本工程书中描述的那样，"……缺少清晰的认识，不知道程序是如何，以及为什么会产生如此结果"。抽象和接口是一条重要的工程原则，然而：

> 抽象本身对于对计算机系统有所了解的专家们已经足够了。当我们谈论计算机科学的时候，理论家们可能无助于程序设计，硬件设计者对于高级系统结构可能一无所知，而程序员对于程序如何以及为什么出现最终结果可能缺少清晰认识。通常，复杂的技术研究导致多种技术专长，而学生们研究的局限性使他们往往不能将系统作为一个整体去理解。这些人能够做很多有用的工作，但由于专业所限，创新的潜能受到限制。他们不愿从熟悉走向陌生，不想重新考虑问题，也不愿变革现存的事物。他们被动地承认神秘的存在，而不愿去揭开它们的面纱，只是用无能为力作为逃避的借口。实际上，他们被自己学习的"抽象"所束缚了。
>
> 与此相反，出类拔萃的学生，发掘出能够阐明技术之间的相互关系，而不仅仅是技术本身的观点，作为他们原则中最重要的结构元素。他们用"尊敬"而不是"敬畏"的态度看待接口。他们考查抽象的二重性，把它看作一次对物体的研究而不仅仅局限于已知与神秘的转换。
>
> (摘自：Stephen A. Ward 和 Rebert H. Halstead，Jr.，Computation Structures，Cambridge MA：MIT Press，1990)

似曾相识？如果 Windows 程序员摈弃了可怕的程序拼凑，抛弃 Windows SDK 例程中那样庞大的 switch 语句，拒绝盲目地追随 SDK 中不适当的范例，那么他在考查 HANDLE 及其真实数据结构时就会大有帮助。

难道这和封装、黑匣子以及"信息隐蔽"等概念不抵触吗？这些虽是绝对重要的 tried-and-true(实践是检验真理的唯一标准)工程原则，但却似乎标明任何事情都是因为"需要知道"(need to know)才去做的。如果按照这种说法，系统内部，如我们本书所讲的内容，就属于你绝对不需要知道，知道了甚至更糟糕的范畴。

封装、黑匣子和信息隐蔽都是很正确的。如果没有事情是隐蔽的，任何事情都是公开的，那我们将一事无成。(Jorge Luis Borges 有一个美丽的故事"Funes the Memorious"，是最好的诠释。)在许多时候，我们希望 Windows API 设计者将 Windows 内部使用的东西尽量加以隐蔽。例如，程序员有什么必要跟 MakeProcInstance()打交道？它对于我们可以完全透明。许

多 Windows 机制可以隐蔽在高级编程接口的后面。正如第 4 章 WINIO 库所示，可以只提供给程序员一个类似于 printf()之类的简单、熟悉的接口，可是在系统内部它仍能“正确工作”，由系统处理其它事务。

但是，同其它事情一样，隐蔽系统内部工作需要付出代价。Jeff Duntemann 在一篇论述“The Tragedy of the Black Box”(黑匣子的悲剧)的文章中谈到了这种代价，“由于对系统隐蔽部分缺乏了解，因此妨碍了对系统可见部分的理解”。正如 Jeff 指出的，当“黑匣子”不完全封闭时，更是如此，你不得不同系统内部打交道。

有时，甚至连 Microsoft 自己也感到：将编程建立在“需要知道”(“need to know”)的基础上这一思想并不一定行得通。例如，Windows SDK 中包含缺省窗口处理过程(DefWindowProc())的源程序。从这些源代码中，你可以知道 Windows 如何缺省处理一些不同的消息。因此 DefWindowProc()是 Windows API 中屈指可数的真正易懂的函数。另外一个例子，DDK 包含许多 Windows 驱动程序的源程序，即使浏览一遍，对于理解 Windows 如何工作也是大有好处的。

那些曾用 CVW 或 Soft-ICE/Windows 跟踪过 Windows API 的人会明白我们将要谈论的是什么。本书的一个目的就是提供一些 Windows 内部如何工作的信息，以及(特别是 1--4 章)为你自己发掘 Windows 秘密提供指南。程序员有必要为系统工作原理创建一个模型。勿庸赘言，正确反映工作机理的模型总会有助于你的工作。

4. 磁盘中有什么?

像《未公开的 DOS 核心技术》(《Undocumented DOS》)一样，与本书配套的磁盘并不是本书中出现程序的简单综合，当然，KERNEL、USER、GDI 每章中的所有源程序都在磁盘中。同时，还包含了一些实用工具。例如：

- MAPWIN(第 2 章)显示程序或 DLL 所调用的 Windows API 函数的名字，以及由 DLL 提供的函数名字。
- EXEUTIL(第 2 章)显示程序或 DLL 中调用的未公开的 Windows API 函数。
- RESDUMP(第 3 章)显示 Windows 程序或 DLL 中所使用资源(对话框、菜单、字符串表、加速键表等)的正文描述。
- CALLFUNC(第 4 章)是一个为 Windows API 调用提供实例的 Windows 解释程序。如果想调试一个函数调用，只需键入函数本身，而无需为它编写调试程序。
- SNOOP(第 4 章)是一个消息监视程序(“spy”)。它只监视未公开的消息以及 Windows 原有的窗口过程。
- WISPY(第 4 章)是一个保护模式下的中断接收程序，它记录一个窗口内的中断。
- ATOMWALK(第 5 章)显示系统中每一个原子表中的每一个原子。
- WINMOD 和 WINTASK(第 5 章)提供有关模块和任务的详细信息。
- USERWALK(第 6 章)和 GDIWALK(第 8 章)提供有关 USER 和 GDI 局部堆，以及 USER 和 GDI 对象，如 WND 和 DC 的详细信息。
- CORONER(第 10 章)是一个事后分析程序，与 Dr. Watson 和 WinSpector 相类似。

由于磁盘空间的原因，一些实用工具没有附带源程序。这些工具在大多数情况下能很好

地工作，你不必关心它的源程序。如果你确实想得到源程序，我们下一本关于如何创建 Windows 工具的书，《The DOS Programmer's Guide to Windows》，将会予以说明。这里我们可不是在玩"隐蔽信息"的把戏，确实是容量有限的缘故。

上面大多数工具是用 WINIO 库开发的，都拷入了随书所附的磁盘中，这些工具都可用 Borland C++和 Microsoft C/C++以. LIB 文件形式调用。

此外，磁盘中还有几个重要的包含文件，像 HANDLES. H(以及 HANDLES. C，列出了 KERNEL 数据结构)、USEROBJ. H 和 GDIOBJ. H。

(读者可与清华大学出版社软件部联系购买此书所附磁盘)。

5. 谁读此书?

如果你熟悉 C 语言，从本书中你将获取更多的知识。本书所有的源程序与 Borland C++ 3.0 和 3.1，以及 Microsoft C 6.0 和 Microsoft C/C++ 7.0 兼容。另外，几乎所有的内容都适用于其它语言，包括 Turbo Pascal for Windows(TPW)和 Visual Basic(VB)。如果你想在 TPW 或 VB 中调用本书中的一些函数，可以参阅书目中提供的几本参考书，以便帮助你进行必要的转换工作。

当然，如果你熟悉公开的 Windows API，那将有助于你理解本书内容。例如，如果你从未用过 GlobalAlloc()，那你就难以理解 GlobalMasterHandle()。另一方面，对 Windows 经验只是一知半解的 DOS 程序员，如果从本书的角度看 Windows 编程，就会觉得 Windows 比他们想象的要更为公开和方便。如果你喜欢涉猎于未公开的 DOS，你将更喜爱 Windows。因为 Windows 是一个比 DOS 大得多的涉猎场所。

对于那些并不认为自己是程序员的 Windows"超级用户"，本书有一部分内容也能引起他们的兴趣。例如，第 5 章 GetAppCompatFlags()一节中有关标识的讨论，第 3 章有关 Task Manager 的讨论。磁盘中的一些实用工具对于那些想在 Windows 中发掘秘密的人会大有帮助。

6. Windows 版本

前面也曾说过，本书的目的是讲述 Windows 3.0 和 3.1 的 KERNEL、USER 和 GDI，支持标准和增强模式下的零售版和调试版。

换而言之，不支持实模式。本书假设 Windows 3.0 的实模式不存在，也永远不会再存在。现在有很多 Windows 编程书籍，是从 Windows 2.x 时代就发行了，现在重新加以修订。本书不同于这些书，而是一本全新的书，因此也没有背上"实模式"这个包袱。我们的观点是建立在"Windows 是一个保护模式下的 DOS 扩展程序"这一基础之上的。不管什么原因，如果你认为 Windows 3.0 实模式非常重要，那对我们的工作可能不会感到满意；如果认为 Windows 3.0 之前的版本也很重要，那自然就更不会满意了。

当然，我们认为 Windows 3.0 还是很重要的。3.1 的出现并不意味 3.0 已被打入冷宫。应该记住的是，Windows 3.0 已流行了将近两年，现在还有大量的 Windows 3.0 版本在运行。与 3.0 的标准和增强模式相兼容是很重要的。书中，我们有时也介绍如何在 3.0 上实现 3.1 特性。

展望未来，Win32 和 NT(New Technology)前景如何？这两者有很大区别：Win32 API

是 Windows 程序设计的未来，不过 NT 也会占有一席之地（有多少用户知道 NT 具有 C2 级安全性，也具有在 RISC 处理器上运行的能力?）。不过，执牛耳者可能是 Win32，Microsoft 计划在 Windows 3.1 上运行 Win32 的"子集"。本书很多地方提到了 Win32 API。或许，在 Windows 3.x 中未公开的函数和消息将会在 Win32 中公开。同时，即使在 Windows 中已公开的特性，像 selector-manipul ation（操作选择）函数，在 Win32 中可能不会再得到支持。值得注意的是，ToolHelp 可能不会成为 Win32 的一部分。

说到 ToolHelp，它是本书重要的一部分。通过 ToolHelp，Microsoft 公开了 Windows 3.0 和 3.1 的小部分核心并提供了接口。ToolHelp 并没有提供存取 Windows 使用的真实数据结构的能力，它仅限于较外面的一层（多数是只读的）。例如，ToolHelp 的 TASKENTRY 结构与 Task Database 并不相同，改变 TASKENTRY 内的一个域并不会导致任何任务行为的改变。但在大多数情况下，Toolhelp 能够达到要求，因此要尽可能地使用 ToolHelp 而不用未公开的数据结构。本书第 10 章专门讲述 ToolHelp，而且远比 Microsoft 在 3.1 SDK 的《Programmer's Reference》一书中讲述的 ToolHelp 详细得多。

使用 ToolHelp 与使用未公开 Windows 技术并不冲突，实际上，它们能够在同一个程序中很好地工作。本书的很多例子使用了 ToolHelp，当 ToolHelp 提供的功能不满足时再转用未公开的 Windows 技术。

7. 我们是如何发现未公开资料的?

几乎每一个人都问我们"你们是如何发现这些未公开资料的?"。本书的第 1 章到第 4 章其实是这个问题的最好答案。首先，Microsoft 并没有刻意隐藏未公开的函数。Windows 使用的新的可执行文件（NE）格式能够暴露未公开函数的名字。我们开始用 Microsoft 提供的工具，像 EXEHDR 和 CodeView for Windows（CVW），实施逆向工程。慢慢地我们又开发出能够更好地发掘 Windows 秘密的工具，许多已装入与本书配套的磁盘中。我们开发的另外一个工具已成为产品，Window Source，可以通过 V Communications 公司买到。当 Nu-Mega 的 Soft-ICE/Windows 出现后，我们又使用它。最后，我们又开发了几个很方便的反汇编工具，但目前还不能成为产品。

与 Windows 调试版一同推出的 Windows 3.1 SDK 测试版，带有完整的 CV 调试符号表；再加上能够使用 CV 符号表的反汇编工具，如 Windows Source，对我们的工作就更为有益。但遗憾的是，3.1 SDK 发行版中不再带有完整的符号表。

除了通过逆向工程，我们还从 Windows 2.x 文档、DDK 包含文件以及 Win32 API 中尽可能地发掘更多的信息。

致　谢

实际上，我们之所以能掌握如此多的资料，这要归功于朋友们的帮助。我们要向下面的朋友表示特别的谢意：

Len Berk，John "Knuckles" Benfatto（Phar Lap），Paul Bonneau（Windows/DOS Developer's Journal），Ralf Brown，Ron Burk（Windows/DOSDeveloper's Journal），Geoff Chappell，Bob Chiverton，Alan cobb，Thuan-Tit Ewe（Metaware），Michael Geary，Frank Grossman（Nu-Mega TE chnologies），Brad Kingsbury（Symantec），David Lection，Bill Lewis（Qualitas），Ron

Mann(Praxsys),Mike Maurice,Darren Miclette(IBM WIN-OS/2),Robert Motte(Phar Lap),Duncan Murdoch,Dan Norton,Andrew Pargeter,JeroenPluimers,Jeff Richter,Art Rothstein,Enrique Salem(Symantec),Brett Salter(Periscope),Michael Shiels,Richard Smith(Phar lap),Victor Stone(Borland),Phil Taylor(Borland),Frank Van Gilluwe(V Commanications)和Jonathan Zuck。

当然,还有许许多多的朋友,由于这样或那样的原因而没有列出他们的名字。我们惟有衷心的感谢!

我们的文学负责人,Claudette Moore,是他首先倡导编写本书的。

Benchmark Production 和 Addison-Wesley 的全体同仁,特别是 Andrew Williams,Amy Pedersen,Jennifer Noble,Chris Williams 和 Abby Cooper,保证了本书的出版。

本书的创作也得助于 CompuServe Information Sevice。实际上与本书有关的所有工作都是在 CompuServe 进行的。每月经济上的资助,更使我们感激不尽。

Andrew Schulman:在我长期从事《Undocumented Windows》的写作过程中,我在 Phar Lap Software 的领导人给了我不同寻常的支持。尽管本书耽误了我很多本职工作,Richard Smith,John "Knuckles" Benfatto 和 Robert Moote 给予我很大的理解和支持。我的同事,特别是 Rob Adams,Andre Sant'Anna,Diego Escobar,Karl Kinsella,Maria Vetrano 和 Alan Convis,也给了我不同程度的帮助。

在《PC Magazine》工作的 Trudy Neuhaus 和 Neil Rubenking 提供了两期有关未公开 Windows 的资料,最后编成了第 1 章。Trudy 由于本书而忍受了由此导致的其它文章的延误。非常感谢!

在《Microsoft Systems Journal》工作的 Gretchen Bilson 和在《Dr. Dobb's Journal》工作的 Jon Erickson,也对因本书而引起的延误深表理解。

最后,我还要感谢 Ray Valdes,Ray Duncan,Claudette Moore,Pete Olympia,Randy-Wallin,Jon Udell 和 Tony Rizzo,感谢他们为我提供的机会以及几年来的鼓励。

特别是本书写作的最后几个月,我的儿子 Matthew 和妻子 Amanda Claiborne 给予我极大的理解,谢谢!尽管本书打扰了 Amanda 很多,她还是赢得了 Brandeis 的 MellonFellowship 奖学金。Matthew 的 15 岁生日即将来临,同时也面临入选 Teenage MutantNinja Turtles 的竞争。小家伙,加油!

Dauid Maxey:我特别感激我的家人。写书期间,我无法与家人共进晚餐,特别是我的妻子,除了自己的工作,还要替我应酬很多。

Matt Pietrek:我要感谢我的妻子,April,还有我的"宝贝",Gunther 和 Theodore,使我在深夜和周末也能工作。我还要感谢我的同事,特别是 E. S.,E. B,和 P. E。(本书编辑觉得有必要指出,上面所说的"宝贝"实际上是两只达克斯狗。)

Andrew Schulman(CIS 76320,302;andrew@pharlap.com)
David Maxey (CIS 70401,3057)
Matt Pietrek (CIS 76117,1720)
June 1992

译 者 序

Microsoft Windows 正风靡世界。众多计算机业者被它漂亮的外表所折服，然而，当他们试图转向 Windows 程序开发时，其中许多人却踌躇不前了——Windows 程序开发似乎高深莫测。然而我敢肯定，当这些人读完《未公开的 Windows 核心技术》之后，他们又会信心倍增，朝着 Windows 世界进发了。

其实，感到困扰的并不仅仅是那些初涉 Windows 软件开发的人，即使是 Windows 编程的老手也时常会碰到许多疑难问题。例如，知道 Windows 支持双字节字符集(如日文、中文)的人可能很多，然而能将其熟练地应用到中文系统开发的人可能就很少了；随便问一个问题：输入一个汉字时，Windows 将会发送何种消息？对此，Microsoft 公司出版的所有 Windows 书籍中从未提及，它属于 Microsoft 公司的不公开资料。这些，在本书中，你都可以找到答案。

先看看下列专题：

- HWND 指向什么？
- Windows 3.1 中的“标志”是什么？
- 消息(message)何处去了？
- WinExec()返回什么？
- 所有不同的 Windows 句柄(任务、实例、模块、任何队列)之间的关系何在？给出一种句柄，如何将其转换为另一种句柄？
- 五个著名的 DC 在哪里？

对于 Windows 编程，它们都很重要，可能正有许多人在为此而苦苦探索。读过本书，我相信他们一定会有一种阿里巴巴“芝麻开门”之后的欣喜若狂！

本书详尽分析了 Windows 的核心：KERNEL，USER 和 GDI。它系统地讲述了 Microsoft Windows 3.0 和 3.1 零售和调试版本在标准和增强模式下的未公开的函数、数据结构以及消息。另外，还讲述了已公开函数和消息中的未公开属性。例如，InSendMessage()返回值的真实含义，WM_NCPAINT 的 wParam 值……

本书中的所有源程序都是与 Borland C++ 3.0 和 3.1，以及 Microsoft C 6.0 和 Microsoft C/C++ 7.0 兼容的。所有这些源程序都可在与本书配套的软盘中找到。配套软盘中除了这些源程序，还包含许多实用工具及其源程序。例如：

- MAPWIN　显示程序 DLL 所调用的 Windows API 函数的名字，以及由 DLL 提供的函数名字。
- RESDUMP　显示 Windows 程序或 DLL 中所使用资源的正文描述。
- CALLFUNC　是一个为 Windows API 调用提供实例的 Windows 解释程序。如果想调试一个函数调用，只需键入函数本身，而无需为它编写调试程序。
- SNOOP　是一个消息监视程序，类似于 SDK 中所提供的 SPY，但它只监视未公开

的消息以及 Windows 原有的窗口过程。

• WISPY　是一个保护模式下的中断接收程序，它记录一个窗口内的中断。

另外，软盘中还包含几个很重要的. H 文件，如 HANDLES. H，USEROBJ. H 和 GDIOBJ. H。

（上述软盘，可与清华大学出版社软件部联系购买，电话：2594891。）

本书原名为《Undocumented Windows：A Programmer's Guide to Reserved Microsoft Windows API Functions》。原作者 Andrew Schulman，David Maxey 和 Matt Pietrek 的名字对于读过《未公开的 DOS 核心技术》（《Undocumented DOS》）一书的读者可能并不感到陌生，Andrew Schulman 和 David Maxey 同样也是《未公开的 DOS 核心技术》一书的作者。《未公开的 Windows 核心技术》与《未公开的 DOS 核心技术》堪称珠联璧合之佳作，它们精彩的章节定会使读者获益匪浅！

本书由全正、闫丰学和梅士兵共同翻译整理而成。全书由全正统稿。我们的老朋友文极、赵立伟、郑方和熊桂喜怀着极大的兴趣参加了本书的校对工作。在本书的翻译及出版过程中，责任编辑姜峰付出了辛勤的劳动，在此深表谢意。本书的翻译浸注了我们极大的热情和精力，我们想向读者奉献一本忠于原著的、既有技术品位又有一定文学品位的佳作；但由于水平所限，书中仍难免存在错误，欢迎批评指正。

译　者

1993 年 8 月 • 于北京 • 燕园

目　录

第 1 章

始 料 未 及

Microsoft Windows 的一个关键目标是实现比 MS-DOS 更好的有序性，DOS 仿佛是一座“卡片库”，包含有内存驻留(TSR)程序、设备驱动程序、磁盘缓冲区、内存管理程序、DOS 扩展程序、网络服务程序，以及多任务环境(如 Windows 本身)，它们都在争夺机器的控制权。在软件开发人员看来，Windows 显得健全多了。它提供的服务覆盖面广，看上去无所不包，如保护模式、多任务、动态连接、窗口管理、图形等，而这些都是一般的 DOS 所不具备的。Windows 可以让开发者专注于程序的功能实现，不必考虑诸如使用未公开的系统功能之类的“把戏”，而这在编制大型 DOS 应用程序时是不可避免的。

举例来说，由于 Windows 3.x 在保护模式下运行应用程序，不需要考虑扩充内存，覆盖以及其他把软件限制在 640K 之内的方法。同任何采用保护模式的 DOS 扩展程序一样，640K 的限制不复存在，与之相关的一系列 DOS 编程问题也随之消失了。

另一个例子是编写 TSR 程序，编写可靠的 TSR 程序，不可避免地要使用未公开的 DOS 功能。而在多任务 Windows 环境下，所有应用程序自动驻留内存，因而免去了编写 TSR 的麻烦。(这里说的只是 Windows 应用程序，如果要编写在 Windows 下正常运行的 DOS TSR，还会有一系列问题。)另外 Windows 中的动态连接免去了大部分(虽然不是全部)在程序中调用中断为其他程序提供服务的必要性。

Windows 的有序性与 DOS 的杂乱性最明显的对比在于图形的编程。对于图形，甚至全屏幕字符模式的程序，如电子表格、字处理、CAD 程序等，DOS 都没有提供相应的服务。它提供的 ROM BIOS 视频服务速度太慢，因此，大多数商业化的 PC 应用程序采取直接写视频硬件的方法进行各种低级的、超越常规的操作。Windows 不再需要使用这些招数，事实上也禁止使用，它提供了一套与设备相对无关的图形函数，Windows 应用程序可以文质彬彬地调用这些功能适中的函数。

这样一来，“未公开 Windows”的想法就有些令人吃惊。使用未公开的功能正是 Windows 要解决的问题。采用 Microsoft 实现的未公开的功能吻合了 DOS 的随意性，却似乎有悖于 Windows 的目标和灵魂。Windows 试图提供比 DOS 的 API 功能更强的 API 来杜绝使用未公开的功能。

使用高级程序语言编程的人不必了解汇编语言和处理器结构，同样，Windows API 试图让程序员避免使用低级技巧。这种想法有些可笑，因为 Windows API 本身就十分低级，它需要 80 行代码才能在显示器上显示出“Hello World!”。但是这 80 行代码可以在任何能运行 Windows 的机器上工作，包括与 PC 不完全兼容的机器，如日本的 NEC 计算机，在这一点上它确实是高级的。

从 Microsoft 的一封 Windows 硬件工程会议的邀请信中，能清楚地看出高级编码是 Windows 的一项要旨：

为何要专注于 Windows PC?

> 因为 Microsoft Windows 为 PC 硬件销售者提供了在设计新的系统和子系统时自由创新的机会。Windows 开启了工程革新之门，它为软件人员免除了所有直接写硬件的操作，而它之前的 DOS 标准都做不到。任何能通畅运行 Windows 的机器都能支持全部的通过 Windows 应用程序接口(API)编制的软件。工程的侧重点转移了。在 Windows 这个新世界中，关注的是硬件的性能和创新，而不是硬件的一致性。("Windows Hardware Engineering Conference ,"1-3 March 1992, Advance Invitation。)

如果真正转移了工程侧重点当然再好不过了。只需使用公开的 Windows API ，我们就能转移到真正的高级平台上去，Intel 可能不愿意看到这一切。如果每人都按规则使用 Microsoft API，Microsoft 甚至会把我们、我们的代码和我们的用户都搬到高级 RISC 结构上，抛开所谓的段，IBM 兼容性，MS-DOS 兼容性和其他所有的在当前显得很重要的繁杂琐事。

换言之，Windows 的一项要旨就是让开发者只对 Windows API 编程而不采用其它手段(目前 API 至少还包括一些 DOS INT 21h 调用)。除编写 Windows 驱动程序外，不用做任何低级编程。当然，也不必使用未公开的 Windows 调用或了解任何 Windows 内部的数据结构。那样做会重蹈 DOS 的覆辙。

这种设想事实上只是一厢情愿，试想，有多少商业化 Windows 应用程序能真正做到"按规则"编程而具有竞争力，具备体面的性能和用户期望的特性？所谓 Windows API 能代替全部低级编程的想法似乎并不比 C++能代替汇编语言的想法更高明，或者说，并不十分高明。

Windows 可以免除程序员低级编程的观点其实并不属实。多数 Windows 程序员都会认为，要成为好的 Windows 应用程序开发者，必须熟知 Intel 机器的体系结构(特别是堆栈)，而在 DOS 中不懂这些也能混得不错。有位程序员讲到他仅仅是在使用 Windows 之后才开始使用汇编语言和检查编译器的启动代码。这又何谈可移植和高级？

我们在本章将要看到，重要的 Windows 商业化应用程序，包括 Microsoft 自己的程序，都使用未公开的 API 调用。Microsoft 因此也公开这类调用，尽管是在开发者未经 Microsoft 同意而使用这些调用之后才公开。正是 Windows API 在实际中的应用促使这些资料的公开。仅用 Windows API 编程听似容易，而在现实中却是失败的。

要杜绝在 Windows 中使用怪招，使用低级的、不可移植的代码和未公开的调用，这一概念有什么问题呢？它的问题主要在于 Windows 的成功。在操作系统大战中获胜后，Windows 正在为成功付出代价；众多程序员在抨击它，他们要求它能完成各种功能，而 Windows 可能从未试图这样做。

换言之，使用未公开的功能是成功所带来的不可避免的代价。MS-DOS 付出过这种代价，现在轮到 Windows 了。有趣的是，Windows 也开始被称作"卡片库"了。

Windows 在解决原有的问题时引入了新问题。如果认真想一想，这并不奇怪：为了提供更丰富的功能，Windows API 必然比字符模式的 DOS 更"富有"；这种"富有"必定引入更多的复杂性以及一系列全新的编程问题。

例如，Windows 应用程序虽然可以运行于保护模式下，享有多于 1MB 的内存，但是它