计算机软件
应用系列

Windows 图文程序设计方法与实例

● 严 文 等 编著

● 科 学 出 版 社

计算机软件应用系列

Windows 图文程序设计方法与实例

严 文 等编著

科学出版社
1995

（京）新登字092号

内 容 简 介

本书由浅入深地介绍了在 Windows 环境下开发（而不是使用）各种应用程序和工具的方法，并用大量程序实例说明了 Windows 编程技巧。全书按设计 Windows 应用程序的思路进行组织，主要内容包括：Windows 环境下的文本输出与处理；应用程序如何获取用户输入（包括键盘、鼠标及定时器输入）；如何在程序中利用 Windows 提供的图形接口绘图。本书的后几章集中讨论了 Windows 环境中可供应用程序调用的数据交换技术。

本书多数程序采用 Borland C++ 语言编写，少量复杂程序中嵌入了编译指令，可在 MS C/C++ 7.0中编译运行。

本书可供 Windows 程序员、微机高级用户及计算机软件人员参考。

计算机软件应用系列

Windows 图文程序设计方法与实例

严 文 等编著

责任编辑 刘晓融 留 薇

科学出版社出版

北京东黄城根北街16号

邮政编码：1000717

化学工业出版社印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1995年1月第 一 版 开本：787×1092 1/16

1995年1月第一次印刷 印张：24¾

印数：1—3 500 字数：582 000

ISBN 7-03-004252-2/TP·384

定价：28.80元

前　言

Microsoft Windows 是一种具有窗口功能和多任务功能的操作环境。自 Windows 3.1推出以来，Windows 已经风靡全球。对使用者而言，Windows 易学易用，而且具有一致的用户界面：画面多彩多姿，图标生动鲜明，颜色明朗活泼，窗口切换速度较快。对程序开发者而言，Windows 突破了640KB 内存限制，提供了多任务环境，设备具有独立性、资源丰富，消息驱动式体系结构新颖别致。因此，很多 DOS 用户开始改用 Windows，许多软件厂商也开始开发并推出 Windows 中的应用程序。目前，Windows 已被公认为微机界面的标准。

Windows 使用起来很简单，但编程却很麻烦。有志成为优秀 Windows 程序员的人们大多要走一些弯路，才能满足大量程序设计的要求。为了让读者少走弯路，本书重点讨论 Windows 环境下的应用程序设计，并给出了大量完整的程序。例如，在讨论内存管理时，我们给出了一个因耗尽 Windows 资源而无法继续运行的程序，以说明设计 Windows 应用程序的一般原则；在有关输入的章节，我们给出了一个计时器程序，它可以模拟具有不同计时单位的时钟；在讲述调色板时，我们给出了一个色彩演示程序，它可以产生类似彩虹的输出。另外，有些应用程序还可以绘制各种图形，管理 Windows 下的 DOS 文件，输出优美的 TrueType 字体等等。书中许多程序短而精，并清晰地演示了各种 Windows 程序设计技巧，还有些程序是极有用的工具程序。如果读者能熟练使用 Windows，并具有一定的 C 语言编程经验，将更有助于理解和灵活运用这些实例程序。

初学 Windows 程序设计的人经常会面对许多全新的概念而不知所措，针对这种情况，本书先用一章的篇幅讲解 Windows 应用程序的结构、开发步骤和设计原则，并逐条语句地剖析了一个简单的 Windows 应用程序，由此总结出设计 Windows 程序的难点所在。第二章涉及内存管理，即应用程序的时空效率问题。初学者可以先跳过这一章，直到阅读完后面的章节后再回头体会这一章的内容。第三章阐述 Windows 下的输入。之后的所有章节则是本书的重点，着重讨论文本和图形输出，内容包括文本、字体、打印机、图形设备接口、绘图及文件管理。

值得指出的是，Windows 是一个复杂的图形环境，本书无法囊括它的全部内容。事实上，设计一个大型 Windows 应用程序时，很可能需要考虑窗口界面、键盘/鼠标输入、图文输出、动态数据交换和链接等问题。限于篇幅，本书重点讨论了输入/输出部分，有关窗口及资源的内容，则穿插在本书的程序实例中，并在必要时略加说明。由于本书针对初、中级 Windows 程序员，因此没有讨论 Windows 下的数据交换和链接，有关这部分内容，请参阅其他书籍。

目　录

第一章　简　介

1.1　概　述

本书没有讲述如何使用 Windows。如果读者尚无使用 Windows 环境的经验，则需要安装并熟悉它。本书也不致力于指导读者怎样编写 C 程序。在使用 Windows 编程之前，必须先对传统环境（如 MS-DOS）下的 C 程序设计有丰富的实践知识。如果对 C 语言过于陌生，则有必要熟悉结构和指针等概念。熟悉 80x86 系列微处理器的分段体系结构将有助于读者理解本书的内容。如果读者了解 80x86 在实模式和保护模式下是如何寻址的以及远程指针、函数等预备知识，学习效果将会更好，并设计程序大有帮助。当然，没有这方面的认识也无关紧要，因为这并不影响阅读本书。

本章介绍一个简单的 Windows 应用程序，让读者对 Windows 程序的结构和流程编制有一个感性认识，同时引伸出 Windows 程序设计的若干重要概念。当然，我们不可能一开始就探究功能齐全的应用程序的复杂特性，所以，接下来的章节着重讨论如何处理 Windows 程序的各种不同组件，程序员可以将这些组件结合在一起，以形成整个程序。

要有效地运行 Windows 和本书中的程序，需要以下硬件支持：

(1) 与 IBM PC-AT 兼容的 PC 机；80286 以上的 CPU，内存容量至少 640KB。

(2) 软盘驱动器：容量为 1.2MB（5.25 英寸[①]）或 1.44MB（3.5 英寸）。

(3) 硬盘驱动器：容量在 40MB 以上。

(4) 显示系统：单色 MGA，彩色低分辨率 CGA 或彩色高分辨率 EGA，VGA。

(5) 鼠标：与 MS 鼠标兼容。

当然，各种硬件设备的档次越高越好。例如，所使用的 CPU 越快越好，这样可缩短编译程序的时间。

软件部分必须具备：

(1) MS-DOS 3.1 版以上。

(2) Microsoft Windows 3.0 版以上。

(3) Borland C++ 3.0 版以上，或

(4) Microsoft C/C++ 7.0。

如果还没有安装编译器，则应知道本书的程序只需要 small 模式下的函数库。也可以使用 Microsoft 和 Borland 之外的 C 编译器，只要这种编译器适合于编译 Windows 程序。注意，大多数普通 C 编译器不能用来编译 Windows 程序。

① 1 英寸＝0.0254 米。

1.2 Windows 与 MS-DOS

尽管 Windows 原本用来运行专为 Windows 环境设计的程序，但它也能运行非 Windows 的 MS-DOS 程序。

MS-DOS 程序可以分为两大类，其中较好的应用程序是那些使用 MS-DOS 和 PC ROM BIOS（基本输入/输出系统）软中断来读键盘和视频显示器的程序，而较差的应用程序是那些直接写视频显示器、使用图形或控制硬件键盘中断的程序。所谓“较差”并非指程序的质量（许多为 PC 编写的最好的应用程序，在 Windows 中却是较差的程序），而是指程序使用 PC 硬件的方式。在基于 286 的机器上运行时，Windows 完全没有办法允许此类程序窗口化或多任务化。不过，Windows 可以用 386 微处理器的“虚 86 模式”使较差的应用程序窗口化或多任务化。

启动 Windows 与在 MS-DOS 下运行其他普通应用程序相同。但是，一旦 Windows 装入内存，它就成为一个功能完备的操作系统。当然，还不能完全说 Windows 是操作系统，因为它是在 MS-DOS 之上运行的。在 Windows 运行时，它与 MS-DOS 共同管理计算机的硬件资源。基本上由 MS-DOS 管理文件系统，而 Windows 做除此之外的一切事情——管理视频显示器、键盘、鼠标、打印机和串行端口，并负责内存管理、程序执行和调度。

Windows 与 MS-DOS 各有所长，互为补充。Windows 几乎不提供对文件 I/O 的支持，但这是最小的操作系统（如 MS-DOS）也具备的最基本的功能之一。这就产生了一些令人啼笑皆非的结果。在 Windows 中，创建一个基于磁盘的、包含了一系列复杂图形绘制命令的元文件比创建一个简单的 ASCII 文件更加容易，因为前者是 Windows 完成的，而后者是 Windows 用 MS-DOS 来完成的（所以，我们在后面专设一章讨论文件 I/O）。

编写 Windows 程序要么是完备的，要么就完全不予编写，无法折衷。例如，不能编写一个 MS-DOS 应用程序（即使是所谓“较好”的程序），而在其中只用 Windows 处理图形。如果想使用 Windows 的部分功能，就必须全身心地投入，编写功能完备的 Windows 程序。

在知道了 Windows 程序的结构之后，原因就变得非常明显了。Windows 中的一切都是相互联系的。如果想在视频显示器上绘制一些图形，就需要一个“设备环境句柄”；要获取设备环境句柄，就需要一个“窗口句柄”；要得到窗口句柄，必须创建一个窗口，并准备接收发送给窗口的“消息”；要接收和处理消息，又需要一个“窗口过程”。这时候，就已经在编写 Windows 程序了。

1.3 Windows 的精髓

早期的视频显示器仅用于回显用户用键盘输入的文本。在图形用户界面（GUI）中，视频显示器成为用户输入的来源。视频显示器以图标和输入设备（如按钮和滚动条）的形式显示各种图形对象。用户可以用键盘（或其他定位设备，如鼠标）直接在屏幕上操

纵这些对象，可以拖曳图形对象、按下鼠标按钮或操作滚动条。因此，用户与程序的交流变得更加密切，这不再是一种从键盘到程序、再到视频显示器的单程循环，用户已能与显示器上的对象直接进行交流。所以，我们说 Windows 是一个以图形为基础的程序开发环境。从用户的角度来看，这个图形环境的特点主要表现在以下几个方面：

(1) 用户界面的一致性

长期以来，许多用户对 C:>或 A:>这种命令行式的界面感到厌倦。如今，Windows 提供了一个友好的图形用户界面，用户再也不用花很长时间来学习如何使用计算机或掌握新程序了。Windows 对初学者帮助很大，因为所有 Windows 程序都具有基本相同的外观。

当前正运行的程序占据一个“窗口”，即屏幕上的一块矩形区域。图 1.1 就是一个典型的 Windows 应用程序窗口，程序名显示在标题条（caption bar）中。大部分操作都是用鼠标完成或以下拉式菜单处理的。大多数 Windows 应用程序兼具鼠标和键盘接口，还用对话框（dialog box）获取外部输入信息。菜单和对话框使用户可以尝试性地使用新程序。

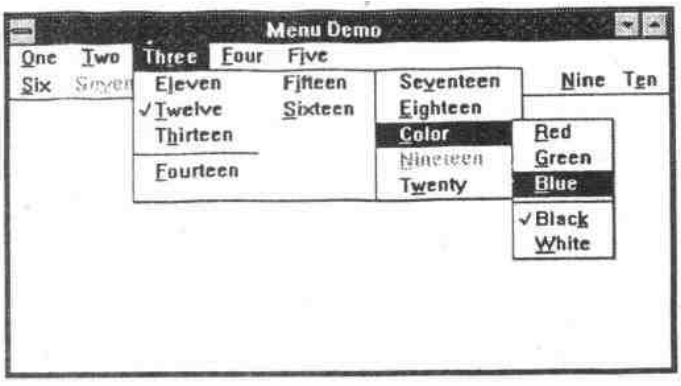

图 1.1　一个典型的 Windows 应用程序窗口

正因为 Windows 应用程序有着大致相同的外观，所以一旦用户熟悉了这种接口方式，就不必再花很长时间去学习如何操作一个新的应用程序。即使碰到一个从未用过的 Windows 应用程序，也能很轻松地掌握它的使用方法。界面的一致性大大降低了学习 Windows 的难度，甚至可以说大大降低了学习计算机操作的难度。

从程序员的观点来看，一致的用户界面是由于使用 Windows 的内部例程构造菜单和对话框而形成的。所有菜单均具有同样的键盘和鼠标界面，是因为这项工作由 Windows 处理，而不是由应用程序处理的。

(2) 突破 640KB 内存限制

如果不在内存管理方面做一些努力，操作系统是无法实现多任务的。在新程序启动、旧程序终止时，内存会变得零乱。系统必须将空闲的内存空间组织在一起。这就要求系统移动内存中的数据和代码块。

在 8088 微处理器上运行的 Windows 1.0 就已能够实现这种类型的内存管理。在实模式下，这只能被看成是软件工程领域一桩令人惊喜的成就罢了。在 Windows 下运行 的

程序可以超额分配内存——在任意时间，程序可以包含内存中装不下的代码量。程序可以放弃内存中的代码，然后从.EXE 文件中重新装入之。用户可以运行一个程序的多个实例（instance，又称“副本”），所有这些实例共享内存中同样的代码。在 Windows 中运行的程序可以共享位于其他.EXE 文件中的例程，这称作“动态链接”。Windows 提供了一种技术，即在运行时将程序与动态链接库中的例程相链接。Windows 本身就是一个动态链接库的集合。

因此，即使在 Windows 1.0 中，PC 体系结构的 640KB 内存限制实际上已被突破了，而且不要求任何附加内存。但 Microsoft 并未就此止步：Windows 2.0 使得 Windows 应用程序能访问扩展内存（EMS）；Windows 3.1 能在保护模式下运行，使应用程序能访问高达 16MB 的扩充内存。

（3）多任务操作

Windows 环境的最大特色就在于提供了多个窗口。一个应用程序占用一个窗口，用户可以随时在屏幕上移动任何一个窗口，改变窗口的大小，在不同的窗口间切换，窗口间还可相互传递消息（message）。一个操作环境若允许多个应用程序同时运行，就表明该环境具备多任务操作的能力。

计算机主机中只有一个 CPU，实际上它一次还是只能做一件事，只是它从一件工作转换到另一件工作的速度非常快。Windows 将 CPU 分配给要同时运行的各个应用程序使用，这样，运行的程序越多，分工就越明显。

尽管有些人不断询问在单用户计算机上多任务是否是必要的，但用户确实乐于使用多任务，而且从中受益匪浅。MS-DOS 驻留程序（如 Sidekick）的流行就证明了这一点。尽管严格地讲，弹出式程序并不是多任务程序，但它们允许快速的环境切换，这涉及到许多与多任务相同的概念。

在 Windows 下，每个程序实际上都是一个驻留的弹出式程序。用户可以同时显示并运行几个 Windows 程序。每个程序在屏幕上占据一个矩形窗口，如图 1.2 所示。用户可以在屏幕上移动程序图标、在不同程序之间切换，以及在程序之间传输数据。

（4）与设备无关的图形界面

Windows 是一个图形界面，Windows 程序能完全利用视频显示器和打印机上的图形和格式化文本。图形界面不仅在外观上吸引人，而且还能给用户传递比较高级的信息，如图 1.3 所示。

为 Windows 编写的程序不直接访问屏幕和打印机等图形显示设备。相反，Windows 提供了一种图形程序设计语言（称为图形设备接口，即 GDI），它使得显示图形和格式化文本很容易。Windows 虚构了显示硬件，为 Windows 编写的程序支持任意视频卡和任意打印机，只要 Windows 有这种视频卡或打印机的设备驱动程序。程序不需要确定系统连接的是哪种设备。

上述特点为 Windows 用户提供了很大的方便，但程序员却得不到这些特点的“庇护”。如果读者以前没有设计图形用户界面的经验，那么以后肯定会碰到一些陌生的概念。几乎每个刚开始编写 Windows 程序的程序员都经历了思维转向，以消化和吸收这些概念。

如果刚开始时发现编写 Windows 程序困难、枯燥，而且还充满陌生的概念，这是很

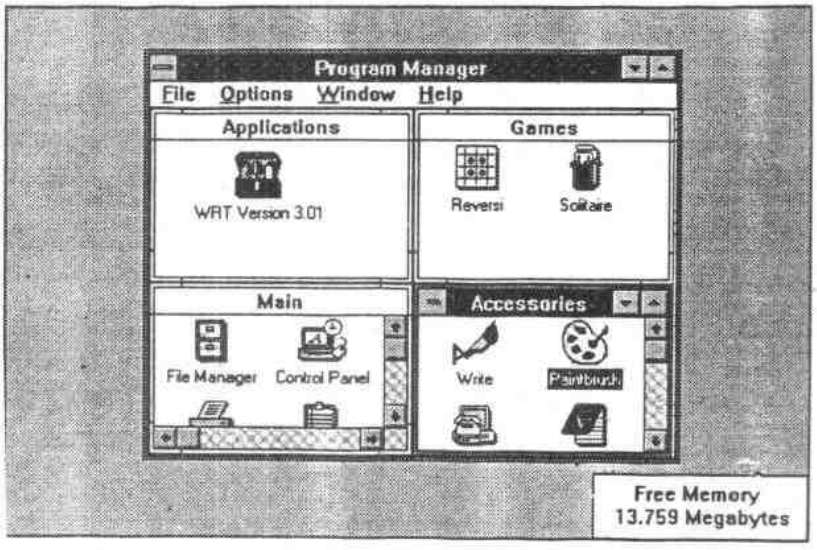

图 1.2　在 Windows 中同时运行多个应用程序

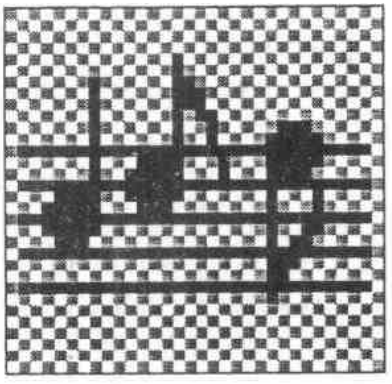

图 1.3　用刷子和画笔画出的简单图形

正常的反应。许多优秀的 Windows 程序员一开始也是如此。只要掌握了这些概念和一般的编程方法，以后编程时就不会感到太棘手了。

下面我们从程序员的角度出发，指出 Windows 应用程序的特点和要素。

(1) 窗口界面

从用户的视觉观点来看，“窗口”(window) 就是屏幕上的一个矩形区域，也就是程序的工作空间。每个 Windows 应用程序通常拥有一个窗口(或再加上一些附属窗口)，应用程序就是在这个窗口中输入、输出和完成各种操作的。

对程序员而言，窗口是一个接收和处理消息的对象，也是一个虚拟屏幕。以往 MS-DOS 下的应用程序可以占据整个实际屏幕，但 Windows 应用程序则不然，所有操作都放在一个窗口中，用户的键盘及鼠标输入都送到该窗口中处理，输出文本或绘图都必须限制在该窗口中。

不论是输入、输出操作，还是其他可能影响窗口的事件，都以“消息”(message)的形式通知该窗口。程序员的责任就是为每个窗口编写一个专用的函数，以处理各种“消息”。程序员通过对消息的处理，达到操作和控制窗口的目的。大部分消息都来自 Windows，但窗口也可以发送消息给自己，或发送消息给别的窗口。例如用户在窗口中按下鼠标按钮，Windows 就必须以一条消息将此事件告诉其专用函数，由该函数决定针对此消息应执行哪些操作。Windows 系统中的消息种类非常繁多（约 220 种），程序员可对每种消息作必要的处理。

(2) 窗口函数

“窗口函数”也称为窗口过程 (window function 或 window procedure)。每个窗口都有一个专用函数，负责处理该窗口的消息，这个专用函数就叫作“窗口函数”。窗口函数的作用在于对 Windows 所传来的消息作适当的响应。程序员的主要工作就是设计窗口函数。

(3) PASCAL 调用方式

Windows 函数的调用类型通常都声明成 FAR PASCAL，FAR 指远程调用 (far call)。因为 Windows 函数和调用该函数的程序代码可以位于不同的代码段 (code segment) 内，因此需用 32 位地址来传送函数所在的“段地址”(16 位的 segment 地址) 及“偏移地址”(16 位的 offset 地址)，所以必须声明成 FAR 类型。PASCAL 类型则是 Windows 惯用的调用方式，它已被公认为标准类型。这种类型的函数在编译器产生机器码时会依据参数在函数中出现的顺序将各参数放入堆栈中，以供函数使用。参数在堆栈中的顺序恰好与普通的 C 函数相反。

(4) Windows 函数调用

目前 Windows 系统中可供应用程序调用的函数已超过 550 个，所有函数名称均根据其代表的含义而定，大小写字母混合书写，不但易懂，且便于查询和使用。例如 CreateWindow 函数显然就是“create a window”的意思。

虽然 Windows 对用户而言是一个易学易用的系统，但对程序员来说却非常困难。试想，一个含有 550 个函数的软件开发工具会容易吗？编写 Windows 应用程序势必要用到这 550 个函数，因此大多数 Windows 程序员在每次编写 Windows 程序时都备有 Windows 程序员参考手册，以便随时查阅。

Windows 的函数调用 (Windows function) 和窗口函数 (window function) 并不相同，这一点用户必须十分清楚。前者是指由 Windows 所提供的库函数，它们有固定的名称，由应用程序来调用；后者是某个窗口的专用函数，由程序员负责编写，并自己定义名称，而且它是供 Windows 调用的函数。

(5) Windows 包含文件

Windows 中 550 个函数的原型 (prototype) 均声明在 WINDOWS.H 文件中，这个文件中还包括标识符以及数据结构的定义。每个 Windows 应用程序都必须在开头将它包含进去：

```
#include  <WINDOWS.H>
```

WINDOWS.H 文件中包括很多东西，它是一个很大的包含文件（又称头文件）。编译 Windows 程序比编译一般 C 程序要花更长的时间，就是因为这个包含文件的缘故。

该文件不但庞大而且相当重要，它包含 1200 个以上的 #include 指令，它们大都用来定义标识符常数，例如：

```
#define  WM_DESTROY  0x0002
```

几乎每个 Windows 使用的常数都在 WINDOWS.H 中有一个对应的标识符。程序员编写程序时，不可能记住所有常数值，因此必须使用这些定义好的标识符（标识符是有意义的字符串，便于记忆）。下面列举一些标识符常数：

```
CS_HREDRAW              IDI_APPLICATION
WS_OVERLAPPED           WM_QUIT
DT_SINGLELING           WS_HSCROLL
IDC_ARROW               CS_VREDRAW
WM_DESTROY              DT_CENTER
CW_USEDEFAULT?
```

前两个字母为某一类东西的缩写，加一下划线，再加上一些有含义的字母拼凑起来，就成为 Windows 标识符。缩写字母的含义如表 1.1 所示。

表 1.1　Windows 中的缩写字母

缩写字母	含　义
CS	窗口类的类型
IDI	图标标识符的 ID 号
IDC	光标标识符的 ID 号
WS	窗口类型（也称窗口风格）
WM	窗口消息
CW	建立窗口（也称创建窗口）
DT	输出文本（在图形环境下也称绘制文本）

WINDOWS.H 还包括 100 个以上的 typedef 指令，它们是关于数据类型和数据结构的声明，例如，

```
typedef  int  BOOL
```

声明 BOOL 这个数据类型是 int（整数）类型。从字面上看，BOOL 这种类型的变量用于“布尔值”(boolean value，真或假)。另外还有一些其他的 typedef 声明，见表 1.2。这里

表 1.2　Windows 中的类型标识符

WINDOWS.H 类型	含　义
BYTE	无符号字符
WORD	无符号整数
LONG	符号长整数
DWORD	无符号长整数
LPSTR	远程字符串指针
FARPROC	远程过程指针

没有一一列举。typedef 也能用来定义结构，WINDOWS.H 定义了很多数据结构，例如，

```
typedef struct tagMsg
```

```
{
    HWND    hWnd;
    WORD    message;
    WORD    wParam;
    LONG    lParam;
    DWORD   time;
    POINT   pt;
} MSG;
```

这个 MSG 结构用来记录一条消息。还有一些由 typedef 定义的其他数据结构，见表 1.3。

表 1.3 Windows 中的数据结构

数据结构	含　义
MSG	定义消息的各个域
WNDCLASS	定义窗口类型的各个域
PAINTSTRUCT	定义在窗口中绘图时所用的绘图结构
RECT	定义一个矩形区域
POINT	定义一个坐标点

WINDOWS.H 定义的结构也非常多，这里不再列举。

总之，编写 Windows 应用程序时，必须记住将 WINDOWS.H 文件包含到程序中，否则函数、标识符、数据结构等都无法使用。

(6) 句柄

“句柄”(handle) 在我们编写 Windows 应用程序时会常常碰到。Windows 系统中的对象都有一个句柄与其对应，例如一个窗口有一个“窗口句柄”，一个图标有一个“图标句柄”，一个画刷有一个“画刷句柄”。

所谓句柄，是一个不带正负号的 16 位数值。在程序中通常通过调用某些 Windows 函数而取得某个对象的句柄。有了句柄才能使用与其对应的对象。所以，句柄就像对象的名字一样。

句柄在 Windows 程序中非常普遍，每种句柄类型都有自己的类型 (typedef) 定义，见表 1.4。

表 1.4 Windows 中的句柄类型

WINDOWS.H 句柄类型	含　义
HANDLE	一般句柄（无符号整数）
HWND	指向一个窗口
HDC	指向一个设备环境
HMENU	指向一个菜单
HICON	指向一个图标
HCURSOR	指向一个光标
HBRUSH	指向一个画刷
HPEN	指向一种画笔
HFONT	指向一种字体