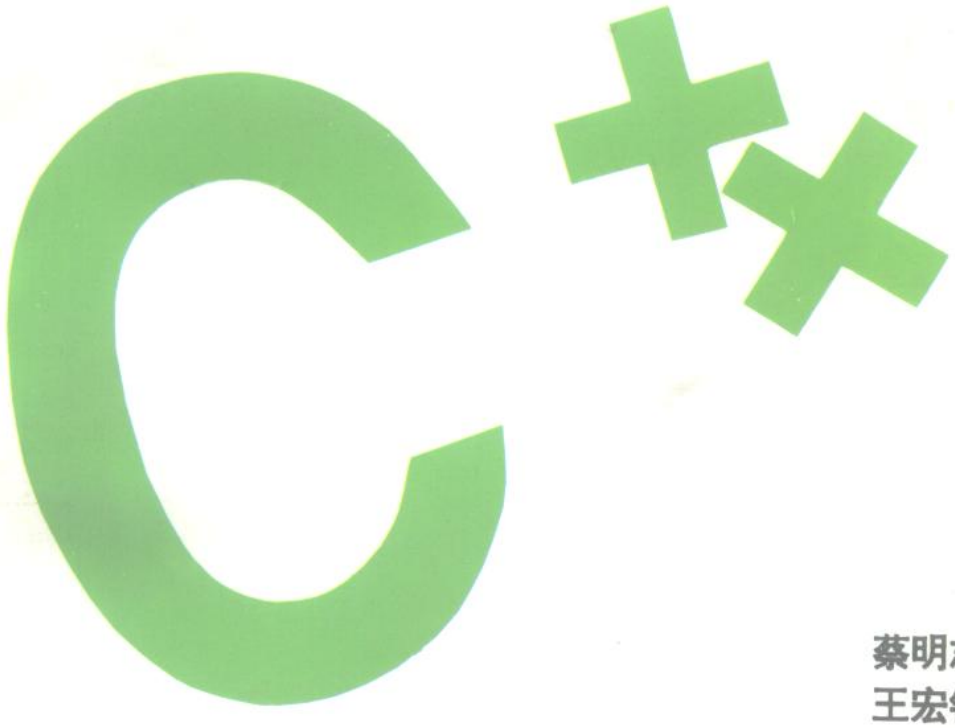
如何用 BORLAND® C++

设计 Windows 应用程序

——附大量实例及配套软盘



蔡明志
王宏铭 著

海洋出版社

如何用 Borland C++设计 Windows 应用程序

—附大量实例及配套软盘

蔡明志　王宏铭　　著
文　都　何利古　　改编
欧阳琼　希　望　　审校

海洋出版社
1993 年·北京

内 容 提 要

本书分成两部分，第一部分以专题形式讨论 Windows 应用程序设计技巧，包括窗口、字体、鼠标和键盘接口、Windows 资源等方面．当然，其中还涉及到所需的各种背景知识．第二部分以实例形式剖析编程的具体方法，这些实例很有代表性，对实例的讲解也很透彻．本书适合大、中专院校师生以及 Windows 程序设计人员阅读．

本书中的所有程序均可在 Borland C++环境下运行．事实上，介绍如何用 Borland C++设计 Windows 应用程序，这正是本书的宗旨．

本书含配套软盘．

在本书的改编过程中，得到北京希望电脑公司秦人华老师和海洋出版社阎世尊老师的大力支持，在此谨表感谢．

(京) 新登字 087 号

如何用 Borland C++设计 Windows 应用程序
——附大量实例及配套软盘

蔡明志　王宏铭　著
文　都　何利古　改编
欧阳琼　希　望　审校

*

海洋出版社出版(北京市复兴门外大街 1 号)
海洋出版社发行　双青印刷厂印刷
开本：787×1092　1／16　印张：35.125　字数：790 千字
1993 年 11 月第一版　1993 年 11 月第一次印刷
印数：1—5000

ISBN：7—5027—3921—7/TP・247　定价：49.00 元(含盘)

目　录

第一章　通用 Windows 程序设计技巧

1.1　Windows 的历史

Microsoft Windows 是由 Microsoft 公司设计出来的，它是一种在 DOS 操作系统上运行的操作环境(operating environment)．这个操作环境具有窗口功能和多任务(multitasking)功能，并且提供了一个图形用户接口(Graphical User Interface，简称 GUI)．Windows 的最大贡献在于"易学易用"，消除了以前学习计算机的种种限制，让学习计算机以及应用软件变得更加容易，更有效率．

自从 Windows 3.0 版推出以来，Windows 已经风靡全球，连续数月勇夺软件畅销排行榜冠军，可见其受欢迎的程度．大多数软件设计人员也正积极地投入 Windows 应用程序的开发行列，因为他们都了解到，Windows 已经带来了革命性的冲击，它为计算机的操作界面指出了新方向，使得信息科技的应用更为简单，更为方便．

进入 Windows 后，相信用户一定会有这样的想法，即自己是否可以写出一个优秀的 Windows 应用软件呢？当然可以．

今天，Windows 发展到 3.1 版，其成功并非偶然．Microsoft 公司为这一系列产品花了八年多的时间，经过不断改进，Windows 才有今天的市场占有率．

当初，"窗口"(window)这种概念是如何形成的呢？大约在 1970 年，由 XEROX Palo Alto 研究中心率先提出这种概念．这个研究中心的计算机专家想让计算机变得更友好、更方便．他们发现，如果要使计算机有一个新的突破，一定要改造传统的用户界面．当用户使用 DOS 启动计算机后，出现的 C:> 就属于传统界面，这种界面常常使一个初学计算机的人感到茫然无措，于是"窗口"这种概念便应运而生．利用窗口能够让用户纵观整个计算机的操作，每个窗口都存储各种不同的信息，屏幕上的窗口之间可以重叠，而又像纸张一样可以更换，想看某个窗口时可以切换到那个窗口中．当时只衍生出窗口这种概念，但实用性的窗口界面却没有被有效地推广开来．

直到 198_ 年初，Apple 公司推出 Lisa，才将窗口概念实际呈现在用户眼前，但因为价格昂贵，所以 Lisa 并不成功．到了 1984 年初，Apple 公司终于推出轰动全世界的 Macintosh，这是第一台突破种种限制的计算机．对一台容易接近的计算机，用户不必学习什么计算机科技知识就能轻松地使用．Macintosh 成功的因素是：造价合理而且改用友好的"窗口界面"，而这正是其最大优点．

Macintosh 的确深深地打动了用户而成为新一代计算机的标准，学习使用 Macintosh 要比学习使用 IBM PC 系统来得轻松愉快，这当然影响到 Microsoft 公司的 MS-DOS 市场．Microsoft 公司认识到 MS-DOS 的传统界面比不上 Macintosh，于是在 1983 年 11 月，Microsoft 公司也投入窗口界面的研究．他们希望在 IBM PC 上也能有一个使用窗口界面的操作环境，这个操作环境将比 DOS 操作系统更能够让用户接受．

经过两年的努力，Windows 1.01 版在 1985 年 11 月上市．这个软件是一种多任务环境，同时提供了窗口功能，还有一些标准应用程序．它的运行速度太慢，而且不是每一种

硬件和外围设备都与 Windows 兼容，再加上 Windows 应用软件不够多，因此 Windows 1.01 版并没有被用户接受。

直到 1987 年 11 月，Windows 2.0 版问市，这个版本修改了 1.01 版的整齐的并列式窗口排列方式，把它变成重叠式窗口(overlapped window)排列方式，也就是今天我们所常见的窗口排列方式。更多的机型和打印机、键盘、屏幕等外围设备都可与 Windows 兼容，同时增强了键盘与鼠标功能，改善了内存管理。当初妨碍 1.01 版流行的最大因素(即可在 Windows 环境下运行的应用程序太少) 也有所改善，许多软件厂商渐渐地把其应用软件移植到 Windows 上，Windows 的市场开始展露曙光。

随后，Windows 2.0 版分成两种版本：一种是 Windows／286，它是在 80286 CPU 上运行的版本，另一种是 Windows／386，它是在 80386 CPU 上运行的版本。Windows 386 主要用来支持 80386 CPU 的保护模式(protected mode)，让 Windows 运行得更平稳、更有效。

1990 年 5 月 22 日，Microsoft 公司推出 Windows 3.0 版，立刻震惊了全球计算机界。Windows 3.0 版将 Windows／286 与 Windows／386 合为一体，并大幅度地调整了图形用户接口，使得画面显得多采多姿，生动鲜明的图标(icon)也极具"魅力"，颜色调配得明朗活泼，并采用了令人耳目一新的系统字体，窗口的切换速度加快，内存突破 640K 的限制，Windows 应用程序可以直接访问的存储容量高达 16M 之多，多任务环境下可以运行更多的应用程序，各种附件也变得更精彩，提供了 Paintbrush，Write，Notepad，Calculator，Solitaire 等工具或游戏，另外还支持网络功能……光看外表就已令人爱不释手，难怪 Windows 3.0 一出场便不同凡响。

1992 年 4 月，Microsoft 进一步推出 Windows 3.1。Windows 3.1 版所带给我们的，并非只是 3.0 版的一个小小改进，事实上它具备更高的运行效率，更适合初学计算机的用户，程序错误用更精确的方式予以检测并修正，加入许多新的重要功能，其中提供的 True Type 向量字体非常漂亮，可放大任意倍数而不失真，这更是引人注目的特色。

绝大部分软件厂商在 Windows 这股潮流涌来之际，纷纷将其应用软件移植到 Windows 系统内运行，包括一般商业应用程序、电子报表、数据库系统、文本编辑软件、排版软件、绘图软件等等。据估计，已有近 1500 个产品可供使用。毫无疑问，Windows 已获得绝大多数人的认同，以往在 DOS 操作系统下操作计算机的用户，都已转入 Windows 环境中。

与此同时，Microsoft 公司还推出 Microsoft Windows Software Development Kit 3.0 版(简称 SDK)。这是一套 Windows 的软件开发工具，让计算机用户甚至软件厂商能够开发 Windows 应用程序，但它还必须配合 Microsoft C Compiler 5.1 以上的版本，在软件开发的步骤上比较麻烦，对初学 Windows 的程序设计者而言，往往望而止步。

幸好 Borland 公司及时推出 Borland C++ 2.0，3.0 及最近的 3.1 版，它们也可以用来开发 Windows 应用程序，而且比较简便。这对于习惯使用 Turbo C 系列编译器的用户而言，真是一大福音。本书中所有实例程序都可用 Borland C++进行编译。

1.2 Windows 的精髓

Windows 是一个面向图形的环境，Macintosh 便是这种 Graphic User Interface 的成功实例，它提供了易学易用的用户界面，操作起来比 PC 上运行的 MS-DOS 更为方便。Windows 的精髓到底是什么？

一致的用户界面

长期以来，有众多用户面对 C:>或 A:>这种命令行式的接口感到厌倦。如今，Windows 提供了一个友好的图形用户接口，它用图标代表每个程序，程序名称显示在标题条(caption bar)中，大部分操作都是以鼠标来完成或以下拉式菜单来处理的。大部分 Windows 应用程序兼具鼠标和键盘接口，Windows 应用程序还用对话框(dialog box)来取得外部输入信息。

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，就不必再花很长的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。接口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的难度。

多窗口、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，改变一个窗口的大小，在不同的窗口间切换，窗口间还可传递消息。一个操作环境中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，所以感觉上像是在同一时间运行。Windows 会将 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。

用户可能会怀疑多任务作业的必要性。假设用户正用 Paintbrush 画图，下笔前，一定需要花时间思考如何布局，要使用什么颜色等问题，在这段思考的时间内，与其让 CPU 停在那里无事可干，不如利用 CPU 在这段空余时间内完成其他工作，例如，用 Print Manager 打印一些报表。以上是多任务作业的意义所在。

完善的内存管理

在 Windows 环境中，内存是一种非常重要的共享资源。因为在同一时间内会有多个应用程序运行，所以任何一个 Windows 应用程序都不能将内存用尽。一个程序开始运行，必须占去一块内存；一个程序结束运行，又会释放回一块内存，因此使内存变得支离破碎。这时，Windows 可以移动代码，使小块的空白内存合并成一块大的空白内存。

Windows 还允许应用程序的代码长度大于空白内存的大小。Windows 的作法是：当装入一段程序代码却发现内存不够时，会将可删除的程序码删除，以空出内存。如果下一次要运行这段被删除的程序代码，Windows 会从运行文件(.EXE)中重新装入这段程序代码。对于这些内存空间的分区、运用与记录，Windows 提供了一套完善的内存管理方

式.

设备独立性

计算机硬件的面貌日新月异，这给软件设计人员带来了不少麻烦：要随时根据硬件的升级而修改软件．对程序员而言，这是一件很麻烦的事．用 Windows 的设备驱动程序(device driver)来控制硬件，不论硬件如何更新，只要硬件厂商提供适当的设备驱动程序，把它安装在 Windows 环境中，应用程序便可通过它来控制新的硬件，而不必修改程序代码．

用户使用 Windows 时会时发现，可在 Windows 环境中使用的打印机，范围极为广泛，从最简单的九针打印机、24 针打印机到激光打印机，而且支持各家厂商．很显然，就算以后再有更新型的打印机问世也不会出现麻烦．

集成环境

Windows 的成功还有一项很重要的因素，即它具有数据传递的能力．在 Windows 环境中，每个应用程序都能分享其他应用程序的结果，例如文字、图片或其他数据．

剪贴板(Clipboard)是传递数据的"转运站"．传递数据有三个步骤：裁剪、拷贝、粘贴．剪贴板可以接收某个应用程序的数据，并把它转换成另一应用程序所需的类型，然后再传递过去．当用户在不同应用程序之间切换时，剪贴板上的数据不会消失，它被所有应用程序所共享．借助剪贴板，每个应用程序都能很轻松地和其他应用程序交换数据，以达到集成的目的．

1.3 检查软硬件设备

工欲善其事，必先利其器．要想学好 Windows 程序设计，只读本书是不够的，一定要实际练习，才能真正掌握．为了顺利编译并运行 Windows 应用程序，应先准备好以下的软硬件设备．

硬件设备部分必须具备：

- 与IBM PC-AT兼容的PC机，80286以上的CPU，至少640KB内存．
- 1.2MB(5.25英寸)或1.44MB(3.5英寸)的软盘驱动器．
- 容量在40MB以上的硬盘驱动器．
- 一套显示系统：单色MGA、彩色低分辨率CGA或彩色高分辨率EGA、VGA．
- 鼠标．

各种硬件设备的档次当然是越高越好，例如，所使用的 CPU 越快越好，这样可缩短编译程序的时间，又如采用比较高级的 VGA 彩色显示器，可使程序运行时的画面比较漂亮．

软件设备部分必须具备：

- MS-DOS 3.1版以上．
- Microsoft Windows 3.0版以上．

■ Borland C++ 2.0版以上。

当然，并不一定非要 Borland C++不可，只要能编译 Windows 的编译器即可，例如 Zortech C++ Compiler 或 Microsoft C／C++ 7.0 Compiler 也可以。但是，请注意，本书中所有实例程序都以 Borland C++ Compiler 为准。

除上述设备外，有下列工具将会更好:

■ 《Windows 3.1程序设计》。
■ 《Windows问题精粹集》。
■ SDK Paint辅助开发工具，附于SDK中。
■ Dialog Box Editor辅助开发工具，附于SDK中。
■ 所有Borland C++的参考文件。

1.4 增强背景知识

假设用户已具备下列两种经验:

■ 熟悉Windows的操作环境，对于Windows的一些专有名词，例如dialog box、menu、scroll bar、font等，都应该有相当程度的理解。
■ 有相当程度的C语言程序设计能力。虽然用Pascal语言或汇编语言也能设计Windows程序，但不如使用 C 语言方便。C 语言是最适合于编写 Windows 程序的程序语言。

如果对 Intel 8086 CPU 系列的结构已有所了解，学习效果将会更好。这对用户程序设计大有帮助。没有这方面的认识也不要紧，因为这并不影响阅读本书。

编写 Windows 程序的注意事项

Windows 是一个多任务操作环境，应用程序必须依照一定的规则编写。当用户开发 Windows 程序时，请注意下列事项:

■ 不要完全占用CPU时间，因为CPU是一种共享资源。虽然Windows具有多任务功能，但一个应用程序不能强制要求另一应用程序让出 CPU 控制权，而必须等到应用程序自己交出控制权后，才能轮到其他应用程序运行。所以 Windows 下的应用程序应避免长时间地占用 CPU 时间，这是编写 Winows 程序的"规范"。
■ 不可对内存或硬件设备(例如键盘、鼠标、计时器、屏幕、显示器等)进行直接的存取控制，Windows 系统必须完全地控制和管理所有资源。应用程序应避免使用中断(interrupt)的方式来提出存取要求。
■ 请勿使用C语言函数库中的CONSOLE I／O函数，例如printf()、scanf()、putchar()、getchar()等。
■ 请勿将C语言函数库中的文件I／O函数用于串行端口(serial ports)及并行端口(parallel ports)的存取控制上，而应使用 Windows 的通信函数(communication functions)。

其他注意事项可参见本书的有关章节。

新的可执行文件格式

DOS 内的传统可执行文件的扩展文件名为.EXE，而 Windows 应用程序经编译后产生的可执行文件，其扩展名也是.EXE，但这是一种用新的文件格式来存储的.EXE 文件。这种新的文件格式含有 Windows 用来管理其程序代码与数据段的动态链接信息，因此无法在一般 DOS 下运行，一定要在 Windows 环境下运行。若强制这种新格式的可执行文件在 DOS 下运行，就会出现类似于"This program must be run under Microsoft Windows."的消息。

用户应有的一些心理准备

Windows 的操作界面对用户来说十分友好方便，但是 Windows 程序设计对程序设计员来说，就不是那么回事了。尤其对初学 Windows 程序设计的人而言，他也许将面临一条艰难的道路。一开始会觉得信息过多，难以消化，但经一段时间适应后，就会慢慢得心应手，步入坦途。学习的过程中难免有挫折，不必害怕。

以上都是在纸上谈兵的准备阶段，现在我们开始接触实践部分。请思考并操作本书下一章的前几个练习，然后继续阅读以下内容。

1.5 术语说明

在真正进入主题前，用户最好对某些名词术语有个大概的认识。

什么是窗口

从用户的视觉观点而言，"窗口"(window)就是一个在屏幕上的矩形区域，也就是程序的工作空间。Windows 的应用程序通常拥有一个窗口(或再加上一些附属窗口)，应用程序就在这个窗口中输入、输出和完成各种操作的。

对程序设计员而言，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 应用程序则不行，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。程序设计员的责任便是为每个窗口编写一个专用的函数，处理各种"消息"。程序设计员通过对消息的处理，达到操作和控制窗口的目的。

究竟这些消息是从什么地方来的？是谁传送它们过来的？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 就必须将此事件以一个消息告诉其专用函数，由该函数决定针对此消息应做哪些操作。Windows 系统中的消息种类非常繁多(约 220 种)，程序设计员可对每种消息作必要的处理。

窗口函数

"窗口函数"(window function 或 window procedure)其实在前面的内容中已经提到

过．每个窗口都有一个专用函数，负责处理该窗口的消息，这个专用函数就叫做"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．程序设计员的主要工作就是设计窗口函数．

Windows 的函数调用

当前 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意而定，不但易懂．且便于查询和使用．例如 CreateWindow( )函数很显然就是"create a window"的意思．

虽然 Windows 对用户而言是一个易学易用的系统，但对程序设计员来说却非常困难．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，因此编写程序时通常手边必须准备一本库函数手册，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，由程序设计员负责编写，并自定义名称，而且它是供 Windows 调用的函数．

PASCAL 调用方式

Windows 函数的调用类型通常都声明成 FAR PASCAL．FAR 是指远程调用(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程序段内(code segment)，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．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．

Windows 的包含文件

Windows 的 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．编写每个 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，这是一个很大的包含文件．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．

此文件不但大而且相当重要，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，程序设计员编写程序时，不可能记得住所有常数值，因此必须使用这些定义好的标识符（因为标识符是有意义的字符串，便于记忆）．以下列举一些标识符常数：

CS_HREDRAW	WS_HSCROLL
IDI_APPLICATION	IDC_ARROW
WS_OVERLAPPED	CS_VREDRAW
WM_QUIT	WM_DESTROY
DT_SINGLELING	DT_CENTER

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．缩写字母的含意如下：

缩写字母	原 意	中文译为
CS	class style	类类型
IDI	ID for an icon	图标标识符
IDC	ID for a cursor	光标标识符
WS	window style	窗口类型
WM	window message	窗口消息
CW	create window	建立窗口
DT	draw text	输出文本

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．由 BOOL 字面上看，这种类型的变量是用在"布尔值"(boolean value，真或假)上．另外还有一些其他的 typedef 声明：

WINDOWS.H 类型	代表含义
BYTE	unsigned char
WORD	unsigned integer(16 bits)
LONG	signed long integer(32 bits)
DWORD	unsigned long integer(32 bits)
LPSTR	far pointer to a character string
FARPROC	far pointer to a procedure

这里不一一列举．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;
```

个 MSG 结构用来记录一个消息．还有一些由 typedef 定义的其它数据结构：

数据结构	代 表
MSG	定义消息的各个字段
WNDCLASS	定义窗口类型的各个字段
PAINTSTRUCT	定义在窗口绘图时所用的绘图结构
RECT	定义一矩形区域
POINT	定义一坐标点

WINDOWS.H 定义的结构也非常多，这里不再列举.

总而言之，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.

什么是句柄

"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，句柄就象对象的名字一样.

句柄在 Windows 程序中非常普遍，每种句柄类型都各有自己的 typedef 定义:

WINDOWS.H 句柄类型	代表含义
HANDLE	一般句柄(unsigned integer)
HWND	指向一个窗口
HDC	指向一个设备环景
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	指向一种字型

以后参看程序实例时，用户将学习到句柄的用途和用法.

面向对象的程序设计

面向对象的程序设计(Object-Orientecd Programming)现在非常流行，它以一种新的观点来从事软件开发工作，有人预言它将导致一次"软件工业革命". 事实上，在 Windows 下编写程序，无形中就已遵循了面向对象的程序概念. 在这种概念下，"对象"(object)是一个用户定义(user defined)的数据类型，它包含数据结构和作用在这些数据结构上的一些函数.

用户可以把 Windows 想象成一堆对象的组合，例如窗口(window)、滚动条(scroll bar)、下压式按钮(push buttom)、单选按钮(radio buttons)等. 有趣的是，从面向

对象的观点来看，程序设计员在程序中也把这些东西看成是一个个的对象(一个个抽象(abstract)的数据类型)．窗口中发生的事件会产生"消息"，而消息也可以被当成一种对象．

本书的重点不是面向对象的程序设计，有兴趣的读者请参考其它书籍．

消息驱动结构

在介绍窗口时已提到过"消息"的概念．Windows 系统以传递消息的方式来实现相互通信的目的．从应用程序的角度来看，发送一条消息，是对应某个事件的产生而发出的通知．事件发生的原因可能是用户移动鼠标、按下鼠标按钮、按下某键盘键、改变了窗口大小等，也可能是程序发送消息给自己(例如要求重画或更新窗口)，或者是别的程序送来消息(例如两个相关的程序，其中一个即将结束，便用消息通知另一个程序)．用户不妨将消息传递看成是电子邮件传递．这样，窗口可与窗口通信，用户可与窗口通信，Windows 系统也可与窗口通信．

消息在 Windows 系统中是不可缺少的重要角色，它的用途可归纳如下：

1. 外在环境所发生的事件，通过消息通知应用程序．
2. 应用程序之间可借助消息相互通信．
3. 事件发生后，Windows传送消息给应用程序，应用程序可针对消息作适当响应．
4. 应用程序可以发送消息给自己，避免程序代码的重覆．
5. 应用程序只有在收到消息后，才有权使用CPU．因此，消息系统使Windows达到"多任务"(multi-tasking)的目的．

简单地讲，编写 Windows 应用程序，主要就是编写窗口函数，而编写窗口函数，主要就是编写处理消息的步骤．

动态链接库

编写传统 C 程序时，调用的函数属于"静态链接库"(static link library)里的函数．编译与链接之后，该函数的机器代码便复制一份到.EXE 可执行文件中．

但是在 Windows 系统下，它提供三个"动态链接库"(dynamic link library)，包含 Windows 所提供的 550 个函数．用户编写 Windows 应用程序时所调用的 Windows 函数在编译与链接后并不会把机器代码复制到.EXE 可执行文件中，调用处只有一个"序码"(ordinal number)，指出是动态链接库中的哪一个函数．当用户将应用程序装入内存时，Windows 利用此"序码"生成一远程指针，直接跳到动态链接库中的那个函数去运行(当然，动态链接库在开始运行 Windows 时就已被装入内存)．

1.6 消息驱动结构

消息的传递在 Windows 系统中非常重要，所以我们最好对"消息驱动结构"(message-driven architecture)作一番探讨．

前面提到的 Windows 所提供的 550 个函数，分别属于三个"动态链接库"，这三个模块分别叫做 KERNEL，USER 和 GDI．用户可以在 Windows 下的 SYSTEM 子目录中

看到这三个文件。KERNEL 模块专门负责内存管理、程序的装入与运行以及程序调度(scheduling)和规划等，USER 模块专门负责用户界面与窗口消息的传递，GDI 模块则负责绘图(包括视频显示器及打印机)。

USER 和 GDI 模块能调用 Windows 系统中的设备驱动器程序。设备驱动器通常都以.DRV 为扩展文件名。设备驱动程序直接控制计算机的硬件或 ROM BIOS。在正常情况下，Windows 应用程序绝不会直接控制计算机硬件或 ROM BIOS。应用程序使用高层次的支持，低层次的控制由设备驱动程序去完成。Windows 应用程序、Windows 的三个动态模块、各设备驱动程序以及 MS-DOS 间的关系如图 1.1 所示。

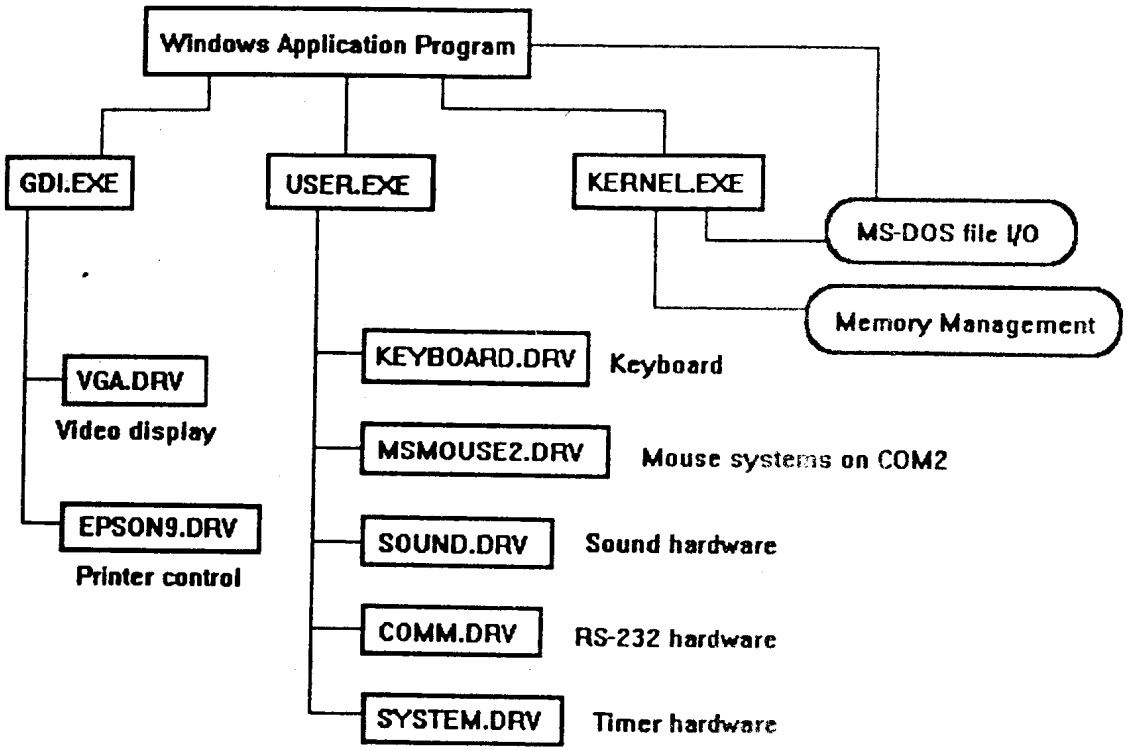


图 1.1 Windows 模块与设备驱动程序之间的关系

图 1.1 使用户一目了然。应用程序调用 Windows 系统，Windows 系统调用它的设备驱动程序，而设备驱动程序直接控制硬件。用户通过设备驱动程序的文件名便可以了解所使用的外围设备。以图 1.1 为例，我们使用的是：VGA 视频显示器(VGA.DRV)、EPSON 九针打印机(EPSON9.DRV)、鼠标插在 COM2 串行口上(MSMOUSE2.DRV)、键盘、RS-232 卡、计时器、扬声器等。

我们的重点是 USER 模块，它会把发生的事件转换成消息。请参考图 1.2，键盘和鼠标驱动程序处理从键盘和鼠标而来的中断。当键盘或鼠标事件发生时，驱动程序便调用 USER 模块中的函数，把这个事件转换成格式化的消息，然后放到"系统消息队列"(system message queue)中。例如，当按下键盘上的 A 键或按下鼠标左按钮时，上述操作就会执行。