

Windows 98 Developer's Handbook

Windows 98

高级开发指南

[美] Ben Ezzell

Jim Blaney 著

邱仲潘 等译



电子工业出版社
Publishing House of Electronics Industry
URL: <http://www.phei.com.cn>

Windows 98 Developer's Handbook

TP316
X432.1

Windows 98

高级开发指南

〔美〕 Ben Ezzell Jim Blaney 著

邱仲潘 等译

电子工业出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 提 要

本书主要介绍Windows 98事件驱动和消息驱动的编程方法,内容涉及相当广泛,包括32位Windows环境的介绍、键盘和鼠标事件的处理、各种应用程序资源的利用、高级程序的设计思想、各种Windows图形处理、应用程序之间的信息交换以及多媒体的使用。

本书在涉及到每一个编程主题时,都详细地介绍了各种API的使用方法和编程任务的实现,并给出示范程序,同时对各种编程问题进行了说明。

本书结构清晰、内容新颖、由浅入深、例证丰富,可以作为Windows 98应用程序开发人员的使用手册,也可作为Windows用户的参考手册。



Copyright©1999 SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501.
World rights reserved. No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photograph, magnetic or other record, without the prior agreement and written permission of the publisher.

本书英文版由美国SYBEX公司出版,SYBEX公司已将中文版独家版权授予中国电子工业出版社及北京美迪亚电子信息有限公司。未经许可,不得以任何形式和手段复制或抄袭本书内容。

书 名: Windows 98高级开发指南

著 者: [美] Ben Ezzell Jim Blaney

译 者: 邱仲潘 等

特约编辑: 朱志刚

印 刷 者: 北京天竺颖华印刷厂

装 订 者: 三河金马印装有限公司

出版发行: 电子工业出版社 URL:<http://www.phei.com.cn>

北京市海淀区万寿路173信箱 邮编: 100036

北京市海淀区翠微东里甲2号 邮编: 100036

经 销: 各地新华书店

开 本: 787×1092 1/16 印张: 51.125 字数: 1300千字

版 次: 1999年9月第1版 1999年9月第1次印刷

书 号: ISBN 7-5053-4902-3

TP·2396

定 价: 78.00元

版权贸易合同登记号 图字: 01-98-1141

凡购买电子工业出版社的图书,如有缺页、倒页、脱页、所附磁(光)盘有问题者,请向购买书店调换。若书店售缺,请与本社发行部联系调换。电话: 68279077

前 言

与DOS和Windows 3.x相比，Windows 98及其前身Windows 95具有显著的不同点。同时，它们也是对熟悉的3.x编程环境的一种延续。

Windows这些新的版本对我们来说是熟悉的，因为从本质上讲它们提供了和旧版本相同的操作和特征，这些操作和特征曾使Windows的早期版本非常流行。它们还保留了执行现有Windows 3.x应用程序的能力。但是为Windows 98开发应用程序和为Windows 3.x开发应用程序大不相同。新旧系统之间的区别远远不是表面的。很多重大的改变不是显而易见的，但仍然在操作方面提供了极其重要的增强。

与Windows 3.x不同，Windows 98不是在DOS管理内进行操作的一个外壳，而是一种完整的操作系统，不受DOS环境的限制。Windows 98能够利用新的CPU，但不能在旧的8086/80286系统上运行。

Windows 3.x只是在DOS界限内进行操作的一个外壳，而Windows 98则是一种完整的操作系统，能够利用更新的80386/80486/Pentium CPU的优点，并给用户提供这些优点。

本书将向读者说明如何开发在Windows 98下运作的各种类型的应用程序。话题范围从处理键盘和鼠标输入的基本知识到网络和Internet支撑的高级技术。你将会学到如何利用Windows 98的特征来创建程序，使这些程序的外观像你所要求的那样，以及完成你所想完成的功能。

本书的适用对象

不管你先前具有什么样的编程经验，以及无论你是一个编程专家，还是一个编程新手，你都将会从本书得到Windows 98应用程序开发的综合指导。

对于从来没有使用过Windows或OS/2的程序员来说，如果仅仅是因为传统的DOS或Unix编程和Windows环境中提出的要求（和功能）之间的差异，那么Windows 98应用程序开发对他来说可能具有挑战性。然而，本书对其中的议题和说明进行了设计，使得对事件驱动和消息驱动编程，以及Windows API函数与MFC（微软基础类）库的需求和功能，给出了清晰的说明，然而这些要以对C/C++语言编程比较熟悉为前提。

以前使用过Windows 3.x的程序员将会发现，用于32位的Windows 98的例子与用于Windows 3.x的等同程序相比，既有相似性也有区别。本书指出了早先的16位系统API调用和32位系统API调用之间的区别，包括将程序从Windows 3.x转换为Windows 98的建议。

那些已经利用过OS/2编程的人对于事件驱动编程以及使用API函数调用进行编程的基本原理应该是熟悉的。在某种程度上，这种熟悉将既是优点，也是缺点。优点在于OS/2和Windows 98中进程和处理是相似的。然而，你将需要知道新名词，如那些用于消息和API函数的名词以及Windows作为参数所使用的类型和结构。

本书所包含的内容

本书介绍了利用Microsoft的Visual C++进行Windows 98编程的主要元素，包括Windows应用资源。所有被讨论的主题都给出了工作例子，本书选配的光盘上包含了所有这些例子的完整清单。

本书由五部分组成：

- Windows 98编程入门
- Windows 98应用程序资源
- 高级应用程序设计
- Windows 98图形
- 应用程序之间信息的交换

此后还有附录，提供将16位Windows 3.x环境中现有应用程序移植到32位Windows 98环境的过程和建议。

第一部分：Windows 98编程入门

在可以开始开发Windows 98应用程序之前，第一步是了解新操作系统的工作原理和将要使用的工具和选项。因此，本书的第1章从了解Windows 98环境开始，包括对使用该操作系统的要求。

在第2章中，从一个简单的程序开始着手学习编程，设计这个程序是如果要说明为98开发应用程序的基本方法。本章还以Template.I的形式给出了一个模板程序。作者包括了这个文件，在其它演示程序中将要用到这个文件。

在第3章中，将研究消息、消息处理和MFC。并讨论Windows 3.1和Windows 98应用程序源代码之间的区别，包括结构、类型和类。

接下来的三章讲述了键盘和鼠标处理以及窗口控制的基本知识。第4章包含了一个简单的编辑器程序，来演示虚拟键（virtual-key）代码的产生和解释方法。第5章讨论鼠标消息和光标。第6章介绍子窗口控件和控制按钮。

第二部分：Windows 98应用程序资源

不必依靠程序指令来创建对话框、菜单和窗口控制，如按钮和滚动条，更简单的方法是使用资源编辑器，来为你绘制所需的资源元素。其它可见性不是很强的资源元素包括快捷键和串。此外，象位图、图标和定制光标这样的元素也可使用资源编辑器来创建。

资源元素不出现在程序源代码中。相反，在编译应用程序之前，那些元素是由独立的编辑器创建并以ASCII脚本（.RC文件）存储的。将图象存储为.BMP、.ICO或.CUR二进制文件，在脚本中引用这些文件。另外，所有这些资源元素都可以编译后的形式存储在一个.RES文件中。

第二部分介绍应用资源和用于创建这些资源的资源编辑器。第7章对资源类型和两个流行的资源编辑器提供了总述。后续章节讨论位图、对话框、菜单、快捷键和字符串资源。第12章在一个应用程序中演示各种元素的使用。

第三部分：高级应用程序设计

对于那些想超越基本知识的人来说，Windows 98提供了各种各样的工具，供更高级应用程序使用。第13章首先介绍一组扩展资源文本，包括弹出菜单和提示窗口。

第14章讨论多线程程序，这种程序处理异步活动。第15章讨论在独立的线程之间使用进程和管道来进行通信。第16章介绍使用公用对象模型（COM）的多线程程序。

第17章介绍系统注册表和注册表键，这是维护应用程序关键数据和用户选项的机制。第18章描述了在应用程序中采用异常处理机制来解决影响流程正常运行的异常条件问题，第19章阐述Windows 98内存管理，以及进程通过创建文件映射对象如何来共享内存块。两个样本程序中的第一个使用虚拟内存函数构造了一个动态分配数组。第二个程序让用户浏览和编辑共享内存中的内容。

接下来的三章阐述与通过网络连接到其它系统的操作相关的议题。第20章讨论Windows 98安全性函数，第21章阐述如何将Internet支持包含在用户的Windows 98应用程序之中，而第22章说明如何创建网络应用程序。

第四部分：Windows 98图形

第四部分介绍Windows图形操作，从第23章开始，讨论Windows设备描述表和映射模式。第24章继续讨论颜色和调色板并演示了主要映射方式。

第25章描述绘制和填充形状的基本工具，这些形状包括曲线和多边形以及画笔。第26章讲述了更多的绘制工具，包括画刷、填充模式、预定义位图模式和与设备无关的位图（DIB）。

第27章的主题是文本显示。这一章研究字体选择、对齐方式和字型以及如何修改设备描述表来控制文本的显示方式。第28章了解使用图形文件的方法。

第29章介绍其它绘图和图形选择的操作。第30章演示更复杂图象的选择方法，包括交互式图形和复杂图形。第31章讲述有关图形模拟的话题。

这部分的最后一章谈到图形的打印以及如何开发灰度图象，以便在黑白打印机上打印彩色图象。

第五部分：应用程序之间的信息交换

甚至在诸如DOS这样的单任务环境中，其中每个应用程序都是独立存在的，也设计了各种方法以允许一个应用程序在它们之间交换信息。这些信息所涉及的范围从普通文件格式的简明性到更秘密的做法，包括TSR，它将信息粘贴到键盘缓冲区中，供前台应用程序利用。

在Windows这样的多任务环境中，在应用程序之间进行信息交换也是一件现实的事情，往往是一个应用程序不可缺少的组成部分。Windows 98包含有允许和鼓励在应用程序之间共享信息、数据或者甚至操作的特征，这些过程和机制包含在第五部分。

第33章演示如何利用元文件来在一个应用程序中记录操作，在另一个应用程序中回放这些操作。第34章说明用户应用程序可如何使用剪贴板操作传输各种信息。第35章和第36章描述DDE和OLE强大的数据共享工具以及Windows 98对这些功能的支持。

第37章介绍多媒体应用程序的开发，从而结束这一部分和本书，说明如何使用多媒体服务作为对其它应用程序的扩展。

特殊特性

本书以说明和旁白形式提供了一些补充信息。说明分提示、警告和一般说明；旁白涉及各种主题，但总是与手头的问题有关。例如，旁白中介绍了如何处理潜在缺陷、如何提供彩色背景、如何使操作更快捷。

本书经常出现“CD”图标。这些图标表示开始介绍本书选配光盘所带的演示程序。文本中的程序名采用斜体字，表示可以从本书选配光盘中找到。

关于演示程序

本书（和本书选配光盘）中的例子均用Microsoft Visual C++ 5.0写成。包括C/C++的例子和许多使用MFC的例子。通常，这些应用程序同时运行在Windows NT和Windows 95/98中。出现例外时，这些不同以文档的形式组织起来，并用文本进行了说明，它们是在网络和安全特征方面遇到的，Windows NT支持这些特征，但Windows 98却不支持。

在那些API说明文档出现的位置（以文本和源代码的形式），它们通常指的是在编译器源CD上提供的在线说明文档。总是可以得到源CD的，要么在当地的CD上可以得到，要么在共享网络驱动器上得到。拥有了在线说明文档，所节省的开发时间是不可估量的（如果是由于系统管理员的脑筋不活，而没有这样的说明文档，那么可以通过多种渠道来得到它们）。

关于本书选配的光盘

本书讨论的所有演示应用程序，以及它们的资源文件、头文件和相关源代码文件均包含于本书选配的光盘上（见本书最后的购盘说明）。此外，本书未讨论或未提到的几个额外的例子只是简要地包含于光盘文件之中。

源文件放置于目录中，这些目录用章号来标识，其中每个目录下又有子目录，用于存储每个应用程序。在某些例子中，...\RES子目录中含有应用程序资源文件，这些子目录是由MFC和AppWizard所产生的。此外，在大多数情况下，为每个演示程序还包含有EXE文件。

试图编译这些例子中的任何应用程序之前，将应用程序目录、目录中的所有文件以及所有子目录均拷贝到硬盘中去。

因为光盘是一种只读设备，所以光盘文件的文件属性也被设置为只读方式。很多文件管理工具，包括Windows 98 Explorer，均不会修改光盘中拷贝文件的属性。因此，将文件拷贝到硬盘驱动器上以后，应该将文件属性复位。如果没有完成这项工作，那么编译器/编辑器将不允许用户编辑这些文件（除非将它们以不同的文件名存储），在某些情况下，甚至会拒绝编译这些文件。

作者的话

尽管本书介绍了生成Windows 98应用程序的大部分主要方面，但由于时间和篇幅所限，不可能面面俱到。读者可以将本书作为起点，开始往感兴趣和需要的方向继续钻研。

欢迎提出建议和意见。我的e-mail地址为ben@ezzell.org，我的Web页面位于<http://www.ezzell.org>。

祝你成为开发应用程序的好手，更上一层楼，可别让老板换下去。

第一部分 Windows 98编程简介

第1章 32位的Windows 98环境

- Windows 98编程的硬件要求
- 磁盘文件系统——16位和32位
- Windows 98的特性
- 跨平台编程
- 双重引导系统的安装

如果读者正在阅读本书，那么就会对创建Windows 98的应用程序感兴趣。然而，在可以设计和实现应用程序之前，第一步是理解应用程序在其中进行操作的环境，以及支持该环境所需的要求。为此，本章讲述Windows 98操作系统所创建的环境。

作为Windows应用程序开发人员，你可能要让应用程序在几种Windows平台运行。本章从应用程序开发人员的角度概述Windows 98和Windows NT以及Windows 98和Windows 95的差别，如果读者计划在运行Windows 98的同时还要运行其它的操作系统，例如Windows NT，则要做一些特殊工作。本章最后一节介绍如何安装双重引导系统。

硬件环境

尽管MS-DOS和Windows的先前版本都已设法为所有早先的硬件系统提供了支持，但许多这些向后的兼容性都是以限制功能为代价而取得的。对早先计算机模型拥有者的支持，包括对808x和80286 CPU的支持，阻碍了程序员充分利用更新的机器，当然，无论是有意地还是无意地，也将最终用户置于类似的限制之下。

在将来某一时间，Windows 98操作系统所固有的局限性也将同样被视为过时、有约束和过于繁琐。到时，Windows 98也将被一种新的操作系统所替代。

到目前为止，我们正在为32位环境编程，且需要满足其要求。我们将从对计算机系统、RAM和硬盘驱动器空间的要求开始。

机器和RAM要求

首先,也是最重要的,Windows 98将不是运行在8080、8086或80286系统上。两个操作系统都要求具有一个最小平台,该平台由80386、80486或P-5,或者MIPS R4000。

理论上,可以在80386系统上运行Windows 98。理论上,运行Windows 98的最低内存要求为4MB。

但这只是理论上可行。实际上,你不可能在低于486和低于16MB内存的机器上运行Windows 98。事实上,即使16MB内存中运行Windows 98也是慢得很,需要大量磁盘交换,因为系统要用硬盘作为系统内存的扩展。最好有Pentium机和32MB内存,64MB内存则更妙。许多新特性只有在32MB内存的机器上才能玩得顺手,如广播服务和活动桌面。当然,如果你问Microsoft市场人员,他们不会这么说的,如果你相信他们的话,我也没办法。

提示: 尽管Intel人员希望他们能一统PC桌面市场,但实际上还是有許多其它处理器系列,包括AMD K-6、Cyrix 6x86MX和IDT WinChip C6。这些处理器都完全支持Windows,提供了与Pentium Pro系列相当的(甚至更好的)性能。另一方面,如今内存已经越来越便宜,在速度和性能方面都有很好的投资回报。如果真要权衡,可以买便宜的CPU,增加更多内存,效果一定不错。

硬盘驱动器空间的要求

第二个要求是硬盘驱动器空间。众所周知,Windows 98并不是从软盘安装,需要有安装多个驱动程序、帮助文件、系统文件、系统等的空间(这里不一一列出)。

要确定硬盘要求,可以简单看看我自己系统上的文件和子目录,Windows 98目录下面占了超过356MB。当然,这包括DLL动态链接库和其它不属于Windows 98安装本身的文件,是由其它应用程序加进去的,并包括许多来路不明的文件(我不知道其来源)。但这说明仅仅Windows目录中就要占用三分之一GB!

如今几个GB的硬盘已经很普遍,而且也不太昂贵(当然,由于操作系统和应用程序的膨胀,需要易于找到和买得起这么大的硬盘)。因此,应作好计划,仅仅Windows目录中就要分配至少0.5GB的空间。

提示: 不要因为你已买了一块更新、容量更高的硬盘驱动器,而将旧驱动器匆匆地扔掉。虽然大多数IDE硬盘驱动器是由不同的厂家制造,但可以将它们组织在一起,作为主/从盘对。SCSI硬盘更容易组织,一个SCSI控制器可以支持多达六个驱动器。

但要记住,512KB只是最基本要求。如果在系统推荐的缺省盘中安装所有应用程序,即安装在Windows 98同一个盘中,则0.5GB很快就会用完。如果出现这种情况,可以用PowerQuest's Partition Magic之类的实用程序调整分区和磁盘大小。

在紧急情况下,用户也许想使用一个磁盘压缩工具来增加现存硬盘驱动器的容量。然而,在安装压缩工具之前,核对一下,以确保该工具不仅与Windows 98兼容,而且与正在运行的其它操作系统兼容(正在使用多重引导的系统)。

文件系统的演变

几年来，FAT16系统经历了多次演变。

最初，所有文件写入单个文件清单，即写入根目录中。这种单目录格式有许多局限性，包括限制磁盘中可以写入的文件个数。但是，由于当时没有硬盘，而5英寸磁盘只能放100KB（是千，而不是百万）内容，因此比文件个数的局限性更大。在这种条件下，很少人会注意或抱怨，因为它比音乐磁带强多了（出现光盘之前的音乐带）。

后来，随着磁盘容量的增大（先增至360KB，再增至1.2MB，最后出现1.4MB的三寸盘），新格式的需求日益增加。子目录的概念是从Unix世界借用的。有限的根目录项目指向非根目录磁道的地址，用于写入其它目录数据（称为子目录）。这些子目录结果作为特殊类型的文件保存。

最后，Windows 95进行了再次改进，允许长文件名。原先的8.3格式文件名保留，并用目录扩展保存长文件名。当然，Windows 95出现之前，OS/2和Windows NT都允许长文件名，但它们是由不同的HPFS和NTFS文件系统属性支持的，是与DOS/FAT不兼容的。

但是，尽管有了这些改进，目前已经出现了新的FAT32文件系统。

文件系统：FAT16与FAT32

DOS FAT16文件系统一直作为PC世界中的事实标准，垂而不死，到Windows 98才引入新的FAT32文件系统，具有几个重要优势。

也许FAT32（32位）文件系统最大的优点是磁盘分配空间的方法。在旧的FAT16系统采用的16位寻址方案中，一个盘上可以生成的簇数（或物理盘上的逻辑分区数）是有限的。

简而言之，由于FAT16只能处理216个地址，大于512MB的硬盘要用16KB块分配文件空间，即任何小于16MB的文件块仍然占用整个16MB块。大文件占用多个块，但大多数系统中通常都是大量小文件，许多文件都小于100字节，更多的文件在1KB与2KB之间。例如，在我的Windows 98目录中，有几百个小于2KB的文件。在FAT16系统中，这些文件都要占用整个16MB块。

说明：如果你认为我的安装不正常，并不适用你的系统，不妨花几分钟检查你自己的硬盘文件。如果你有一个能方便地按长度排序文件的实用程序，则可以在各个目录中迅速找出一大堆小于100字节的文件以及很多不到1KB的文件。甚至有一些长度为0字节的文件，照样占用整个16MB块。

当然，1GB的盘也局限于65536簇，每簇长度为64KB。

FAT16系统和1GB盘的问题在于任何一个文件都要占用整数个数的簇，即文件至少占一簇。这样，如果文件只使用250MB盘上的16KB，则这个1KB文件照样占用16KB空间，其中93.75%的空间是浪费的。但这只是问题的一方面；由于4.3KB文件在256KB盘上占8KB空间，将浪费减少到46.25%，但这要加进磁盘上其它损失的空间中。在1GB盘上，由于簇长增大，问题变得更严重。

这样，只有很大的文件才能真正有效利用空间，而这种大文件并不多。FAT16文件系统浪费的磁盘空间在20%到30%以上。

切换到FAT32文件系统时，可以处理232个地址，即簇长可以减少到1KB，（但更常见的是4KB）。即使1GB以上的盘不进行分区，也能实现这种效果。结果，浪费的磁盘空间很容易减少到2%以下。

但也有缺点，只有Windows 98能认识FAT32文件系统（就象只有Windows NT能认识NTFS文件系统）。即使用FAT32文件系统的磁盘和分区不能由其它操作系统访问。幸而，从Windows NT 5.0开始，FAT32文件系统将得到更多操作系统的支持。

如果要采用双重引导Windows 98和Windows NT 4，则必须考虑其文件系统间的差别。详见本章稍后“双重引导系统”一节。

提示：市面上有几种实用程序，可以将FAT16文件系统变为FAT32系统，而不会丢失现有文件或要求重新格式化磁盘和分区。我喜欢用PowerQuest's Partition Magic，它不仅可以将FAT16文件系统变为FAT32系统，而且可以改变簇长，缩放，生成和删除分区，而不必删除现有数据。其它FAT16-to-FAT32转换软件包也提供类似的功能。

Windows 98的新特性

尽管Windows 98与Windows 95的差别不象Windows 95与Windows 3.1那么明显，但仍然有一些Windows 98的新特性值得一提。尽管这些新特性备受赞誉，但从应用程序开发人员的角度看，大部分新特性都不是太重要。

即插即用驱动程序

Windows 98带有大量即插即用驱动程序，较少遇到新硬件安装出现问题。但这并不表示Windows 98能自动识别所有外设和加入卡。情况虽然有所改善，但仍然不是十全十美。俏皮话“即插即拷”近期内还无法消失。

说明：大多数编程人员可能会忽略与目前应用程序关系不大的即插即用，但也许在不远的将来，即插即用对应用程序开发人员可能变得重要起来。第18章将介绍Windows 应用程序中的即插即用处理。

新技术的支持

应用程序开发人员可能感兴趣的一个领域是通用串行总线（USB）的支持。这个新标准用于替换现有设备对有限硬件中断个数的依赖性。现有系统的外设不断增加，modem、鼠标、声霸卡，等等，使16个硬件中断显得越来越不能满足要求（这是中央系统保留的中断数）。USB可以对现有设备和新设备提供几乎无限个连接，同时提供比现有通道速度高几个数量级的有效通过速度。

Windows 98号称支持USB（或USB号称支持Windows）。但是，到本书编写时，USB标准刚刚出现，还没有任何跟踪记录，因此还看不出这个标准能否实施。USB对应用程序开发的影响还很难预测和想象。目前，可以密切关注USB领域的进展。

除了USB外设外，Windows 98还支持AGP（加速图形端口）显示卡、MMX处理器和尚有争议的DVD盘。此外，还可以连接多个显示器（使用多个显示卡的多个监视器）。同样，新的多链接通道汇合（MCA）技术可以组合两个或几个通信线，从而增加带宽（尽管目前很少有ISP允许支持MCA连接所需的多重登录）。

新实用程序和向导

Windows 98提供了许多新实用程序和向导, 包括:

Update Wizard (更新向导) 连接Microsoft Web站点, 自动下载最新补包(patch)、驱动程序和改进。

Tune-Up Wizard (调整向导) 计划实用程序, 整理磁盘和删除多余的文件。

System File Checker (系统文件检查器) 检查重要的.DLL、.COM、.VXD文件和其它文件, 测试搞乱/修改情况。必要时可以用这个实用程序恢复原文件。

Help Desk (帮助桌面) 支持故障诊断, 链接本地和Internet帮助资源, 如Online Help、Bug Reporting Tool和Microsoft Knowledge Base。

Disk Defragmenter/Optimization Wizard (磁盘整理/优化向导) 建立清单和组织硬盘, 便于访问常用文件。

Accessibility Settings Wizard (访问性设置向导) 对残疾人桌面提供实用程序和定制, 如弱视用户使用的屏幕放大器。

尽管这些特性对Windows 98应用程序设计没有直接影响, 但可能对开发工作提供一些机会, 提供新的方向。

跨平台应用程序开发

尽管本书是针对Windows 98编程, 但也许你的应用程序要在Windows 98和其它Windows平台上运行, 如Windows 95和Windows NT 4。

在此要说的最简单的事就是, 声明Windows 95/98和Windows NT 4之间没有任何真正的区别, 只是这种说法过于简单。实际上, 二者之间存在很多差别。然而, 从编程的观点来看, 大多数区别是不可见的。大多数所编写的运行在Windows 98下的应用程序也可以在Windows 95/NT下执行, 反过来也一样。有例外, 但也仅仅是例外。

以后各章会指出这些例外, 目前只是一般性地介绍进行跨平台应用程序开发要注意的一些方面。

提示: 任何情况下, 如果正在开发一个跨平台应用程序, 唯一真正的解决办法就是周期性地同时在两个操作系统上试运行所开发的应用程序。然后, 如果应用程序在一个操作系统上能够运行, 而在另一个操作系统上不能运行, 那么就是好戏开始, 祝你走运。

Windows NT应用程序

象Windows 95/98一样, NT本身是一个操作系统, 且不依赖于任何潜在的16位DOS操作系统的存在, 这一点与Windows的早期版本有所不同。与Windows 95/98不同的是, Windows NT 3.1、3.51和现在的4.0都是真32位操作系统。遗憾的是, Windows 95/98是一种混合物, 它既依赖于16位的核心代码, 也依赖于32位的核心代码。

有时, 一个应用程序将在一个系统上运行, 而在另一个系统上却不能运行, 因为开发者已利用了某些技巧, 这些技巧在一种操作系统下是可行的, 但在另一种操作系统中就不行。换句话说, 将应用程序编写为依赖于具体针对操作系统的特征。NT不支持Windows 95中找

不到的某些特征，如网络API。当出现这样的情况时，本书在讨论中将提醒读者注意。

有时应用程序可能不能同时在两个系统上运行，因为它们利用了操作系统特定高级特性。例如 Windows NT支持网络API等特性，这是其它操作系统不支持的（第22章介绍了如何在应用程序中增加网络支持）。

影响应用程序的另一差别是Windows NT中安全性和访问权限的用法。Windows 98中不支持这些权限（第20章介绍Win32安全性和加密API）。

提示：Win32驱动程序模型允许开发人员生成同时适用于Windows 98和Windows NT（5.0）操作系统的硬件驱动程序。过去，开发人员要对每个操作系统采用不同的硬件驱动程序，使开发人员和用户难于升级硬件。

Windows 95应用程序

从编程角度看，Windows 95和Windows 98之间没有严格差别。这并不奇怪，因为Windows 98主要是95操作系统的改进（和补包）。

下面几章要介绍新版本的改进之处。但这些改变都不是特别明显。

说明：关于从Windows 3.x应用程序移植到Windows 98的信息，见附录A。

双重引导系统

如果要在同一系统上同时运行Windows 98和Windows NT，则要保证应用程序在两个系统上都能顺利安装。另外，FAT32与NTFS文件系统的不兼容性也会影响安装。

应用程序安装

Windows 98和NT之间最大的冲突往往不是不兼容性问题，而是简单的误解。在双重引导系统上，在一个操作系统下安装的应用程序在另一个操作系统下是可以看到的（因为其所有文件在硬盘上是可以看到的），但它不能在第二个操作系统上运行。通常这不是因为不兼容性问题，而是因为需要在第二个操作系统上进行重复安装。

使人感到恼火吗？是的，但之所以恼火，是有很好的理由的。当将一个应用程序安装在任何一个操作系统上时，安装程序所完成的工作只是创建目录和拷贝文件。有时，安装程序还拷贝DLL（动态链接库）或其它特殊文件，它们被安装在了操作系统的/SYSTEM目录中。但每个操作系统都有它自己的/SYSTEM目录，且其内容不是相互兼容的；不能将它们合并。安装过程中，几乎总是要对某些选项做出选择，以进行操作系统的注册。这些内容也是独立的，且相互不兼容。

说明：某些简单的应用程序根本不需要进行安装，不用安装它们就可以在任一操作系统下执行。在这样的情况下，应用程序通常是自包含的（self-contained），不依赖于DLL和注册项。

除拷贝到每一个操作系统的/SYSTEM目录中去的文件外，双重安装不要求复制目录和文件，所要求做的是制作重复的注册条目（用户是不想用手工来完成这件工作的，因为太麻烦）。

幸运的是，解决办法相对简单。例如，如果正在安装Microsoft Visual C++，而且想将它同时用于双重引导系统上的Windows 98和NT上，那么按照下列步骤进行：

1. 引导任一操作系统。
2. 正常地完成安装。
3. 重新引导第二个操作系统。
4. 在同一个目录中重复安装过程。

安装完成之后，如果应用程序因某种原因在一个或另一个操作系统上运行失败，那么就核对一下应用程序文档，看看是否有某些特殊要求或系统不兼容问题，然后再次试安装。

文件系统设置

如果正在同时运行Windows 98和NT 4，那么C盘应维护为FAT16盘（或分区），这是正确使用Ntldr（NT Loader）所必须的。

另外，记住FAT32盘和分区的功能在Windows NT 4系统上是看不到的，而NTFS和HPFS盘和分区的功能在Windows 98系统上是看不到的。

介绍应用程序运行环境之后，就可以构造应用程序了。下一章，我们将从构造两个简单的应用程序开始。我们将创建传统的“Hello, World”演示程序的Windows版本以及一个模板应用程序，可将该模板程序用作其它应用程序的基础。

第2章 Windows NT/95应用程序设计

- Windows中的“Hello, World”
- 消息映射函数
- Windows应用程序模板
- Windows编程中所使用的规则

对于DOS程序员来说，为Windows 98（或Windows NT和Windows 95）编程与熟悉而又可以接受的编程方法相比，是一种明显而突然的偏离。事实上，这种偏离非常突然，以致于很多程序员感到他们不知道是否已更改了语言及操作系统。他们非常惊讶于明显地发现现象Pascal一样的元素却出现在C代码中，先前觉得熟悉的捷径和过程，如main（）或exit（0），都不再有效了，或者在某些情况下，这些函数的使用对于应用程序或操作系统来说是致命的错误。

其它，尤其是那些已使用过Windows早期版本或甚至使用过OS/2的人可能会发现，这些新的编程方法使用熟悉的技术，尽管他们有时会遇到新的名词或稍微有点不同的语法。遗憾的是，已为Windows 3.x（3.0或3.1版本）编写过应用程序的人可能会觉得完全被弄糊涂了，因为很多区别都是细微的。可是在很多情况下，这些非常细微的区别却是至关重要的。

所以，即使你是一个有经验的Windows应用程序程序员，也应花时间去学习一下Windows 98的实际编程方法。第1章讨论了Windows 98编程的要求。现在该了解Windows程序的组件了，开始就象White King建议Alice的“开头”一样。

WinHello: 一个Windows入门程序



传统上，入门的C程序例子一直总是一条“Hello, World”消息。它是由几行代码完成的。然而，对于Windows 98来说，同样的入门程序例子将显得稍有点复杂，要求用到大约130多行代码，或者大致是原来程序代码长度的20倍。这不足为奇，因为Windows是比DOS更为复杂的一种操作系统，但更反映了一个事实，即任何Windows应用程序，甚至是一个最基本的例子，与其DOS副本相比，都是运行在一个更复杂的环境下的。相应地，Windows应用程序要求具有某些最小的代码条目以与环境匹配。

说明：因为Windows应用程序通常包括有图象（位映射模式）资源，而其DOS形式的应用程序却没有，所以直接比较.EXE文件的大小是无效的。通过比较源代码长度和开发时间，更能说明两种代码形式的区别。

可是，这个问题的反面并不意味着所有的应用程序将比其DOS副本更长、更复杂。相反，更普通地讲，在Windows下，应用程序越大，与其DOS副本相比，其代码长度越短。这是因为应用程序本身在DOS下所提供的很多功能和服务都是直接作为Windows下的系统功能从外部来调用的。

可是,即使不谈理论,WinHello源代码例子也比其DOS形式要大得多。其实际大小取决于应用程序的创建方法。例如,WinHello1是一个简单的应用程序,它没有使用类定义,由单个源文件组成,而WinHello2是一个MFC (Microsoft Foundation Classes) 应用程序,是利用AppWizard来创建的,含有20个源文件 (WinHello1和WinHello2在本书选配的光盘上)。

说明: WinHello和WinHello2演示程序在本书选配光盘Chapter2文件夹中。

Windows.H头文件

Windows.H头文件是所有Windows源代码都要求使用的一个基本包含文件。其原因很简单: Windows.H中含有所有Windows消息、常量、标志值、数据结构、宏以及其它助记符的定义,这些内容使程序能够不必去记忆成千上万的十六进制值及其功能,就能使用它们。

对于Windows的早期版本来说,Windows.H是一个单一庞大的文本文件(约1000行)。对于Windows新版本,Windows.H文件本身已收缩了,现在它主要是由一系列其它包含文件所组成的,其中最重要的是WinUser.H包含文件。这个包含文件是旧式Windows.H文件的当前版本,象其早期版本一样,也含有大约1000行源代码。

说明: 下列定义的引用是由Windows.H文件提供的,很有可能在WinUser.H或WinDef.H文件中找到。
可是,所有这些引用都可以放置在Windows.H中作为包含文件所列出的其它源代码中。

如果正在使用MFC,那么Windows.H头文件被包含在AppWizard提供的SDTAFX.H头文件中:

```
#define VC_EXTRALEAN           // Exclude rarely used stuff from Windows headers
#include <afxwin.h>             // MFC core and standard components
#include <afxext.h>             // MFC extensions
#ifdef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>             // MFC support for Win98 common controls
#endif // _AFX_NO_AFXCMN_SUPPORT
```

WinMain过程

正象每个DOS C程序都拥有一个名为main的过程作为其核心一样,每个Windows程序也有一类入口点,其名称为WinMain (不错,这个名称对大小写是敏感的)。此外,就象DOS C程序在主过程说明中包含有读取命令行参数的代码条目一样,尽管在Windows下很少使用命令行参数,但WinMain还是以lpszCmParam参数的形式包含有类似的代码条目。

提示: 用MFC基础类和应用向导生成应用程序时,WinMain过程不出现在对各个类生成的框架源文件中。但WinMain过程并没有消失,只是不显示而已,放在预编译的基础类库。尽管WinMain程序看不到,但仍然存在,必要时可以接收指令,改变应用程序的初始样式和特征。

可是,与DOS编程不同,用于WinMain的说明不是可选的,必须以它们出现的顺序和形式确切地予以说明,而不管是否使用其中每个具体的变元。记住,因为用户只是间接地调用应用程序的WinMain过程,实际调用格式由Windows提供,DOS上下文中所存在的调用格式的灵活性在Windows下是不存在的。