

SAMS

北京科海培训中心

[美] Ben Ezzell 著
廖俊 段爱民 译
成昊 审校

Windows 32位编程指南

清华大学出版社
西蒙与舒斯特国际出版公司

TP312
YZE/1

SAMS

北京科海培训中心

Windows 32 位编程指南

^{B. 伊泽尔}
[美] Ben Ezzell 著

寥 俊 段爱民 译

成 昊 审校

清华大学出版社

西蒙与舒斯特国际出版公司

(京)新登字 158 号

著作权合同登记号:01-96-1211

内 容 提 要

本书讲述跨 Windows 95 和 Windows NT 进行 32 位编程的方法,并以大量的实例说明了 MFC 和公用对话框与自定义对话框的使用方法 with 技巧。

全书共分 4 部分。第 1 部分深入介绍了 32 位程序的运行环境;第 2 部分介绍用户界面的设计;第 3 部分以实例讲述了 Windows 程序设计中至关重要的图形问题;第 4 部分介绍 Microsoft 基础类库,以及如何使用面向对象的程序设计和基础类库创建应用程序的方法。

32-Bit Windows Programming

Copyright © 1996 by Sams Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher.

本书英文版由 Prentice Hall 出版社属下的 SAMS 计算机图书出版公司于 1996 年出版。版权为 SAMS 公司所有。本书的中文版版权由 Prentice Hall 公司授予北京科海培训中心、清华大学出版社和西蒙与舒斯特国际出版公司合作共同出版、发行。未经出版者书面允许不得以任何方式复制或抄袭本书内容。

版权所有,盗版必究。

本书封面贴有 PRENTICE HALL 激光防伪标签,无标签者不得进入各书店。

JSS98/07

书 名:Windows 32 位编程指南

作 者:寥俊 段爱民

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

印刷者:北京门头沟胶印厂

发 行:新华书店总店北京科技发行所

开 本:16 印张:25.25 字数:611 千字

版 次:1996 年 11 月第 1 版 1996 年 11 月第 1 次印刷

印 数:0001~6000

书 号:ISBN 7-302-02389-1/TP·1197

定 价:40.00 元

05887.00

前言

对大多数计算机用户来说,16 位环境好像从来就没有什么变化,至少从个人计算机被发明的那天起就始终如一。这种看法从许多方面来讲都可以说是对的,因为早期 8 位(和 4 位)的操作系统只有少数特殊嗜好者才会感兴趣;早在个人计算机的销售日趋膨胀以致像电视机一样普及之前,这样的操作系统就已经过时了。

从历史的角度看(如果可以把短短的 20 年称为历史的话),PC 机已经经历了几个演变阶段。当前从 16 位到 32 位环境的改变并非是唯一的一次变化,甚至不是一个令人惊奇的变化。因为事实上,这个变化不仅不是新的,甚至可以说是过期的。

16 位操作系统利用了 80286 CPU 的非凡能力,替代了早期 8 位 OS。8 位 OS 速度慢、限制多、使用麻烦,当然这和最早 PC CPU(8080s/8088s/280s)的处理能力有关。老实讲,8 位系统并不具备像它们的总线宽度所暗示的那样多的局限性,但是,它们的能力的确是有限的,人们很快就对它们做了升级。

但是,当 80386 CPU 出现时(该 CPU 具有处理 32 位操作的潜力),并没有发生操作系统的相应改变。尽管如此,这种新型的速度较快的芯片立即产生了新的市场需求,随之而来的是对大于 1M 界限的内存和访问这些内存的 XMS 和 EMS 驱动程序的需求,但是由于没有新的 OS 介入,从而无法使 386 系统尽其所能。

在过渡时期发布了几个新的 DOS 版本,包括 Microsoft, Digital Research 和 IBM 的 DOS,但是所有这些版本,甚至在 80486 CPU 出现以后,仍是 16 位操作系统。由 Microsoft 和 IBM 联合开发的 OS/21.x 版也是 16 位 OS(虽然 OS/2 对扩展内存的支持优于 DOS)。

为什么变化迟迟未到?

因为诸如 Microsoft, Digital Research 和 IBM 等开发商均认为:保持与旧 PC 机(指 8080/8088s 和 80286 系统)兼容的重要性大于为满足极少数使用最新 CPU 的用户而开发全新系统的重要性。

当然,这种讲法过分简单了,事实上这个阶段,做了大量的工作试图开发不向下兼容(不与以前的 CPU 兼容)的新的操作系统。这些新的操作系统比以前的 OS 更为复杂。复杂性不是由于 32 位寻址和 32 位指令引起的,而是由于新的 OS 试图不只是在功能上,而且是在外观上与以前的 OS 完全不同。

当然,今天我们对外观上的不同是如此熟悉,甚至可以说是熟视无睹了。最明显的变化是从面向文本的环境转到了面向图形的环境。然而更重要的也是最不易觉察却无处不在的变化是从单任务环境转变到一个多任务系统。今天,我们认为后台进程的存在是理所当然的——如 e-mail 程序、打印缓冲池、屏幕保护程序以及多个基本进程(如编辑器、编译器和网络浏览器)可以同时打开并成为活动进程。使用 DOS 的 TSR 虽然也可以启动有限的多任务,但是这些任务之间经常发生冲突,成功运行的机会不多。今天,在真正的多任务环境中,不仅多个任务能够毫无差错地执行,而且对一个进程与另一个进程的相互干涉也是不能容忍的。

如果一个 32 位操作系统的创建只意味着产生新版的 MS-DOS 或 PC DOS,那么 16 位

的环境早就消失了。

随着 OS/2 2.0 版本的出现,第一次真正与旧的 16 位环境分离了。这是第一个不是为了向下兼容而设计的操作系统,然而 OS/2 不再是 IBM 和 Microsoft 的联合项目,它是由 IBM 独立开发并推向市场的。

Microsoft 没有把 32 位的市场留给 IBM,事实上,Microsoft 的 Windows NT(3.1 版)正准备出台,Windows NT 不再支持老的 CPU,它只能运行在 80386/80486 CPU 上,它需要至少 12M 内存(16M 或更多内存将更好)。

不幸的是,当 Windows NT 发布时,RAM 的价格仍旧比较昂贵。而且也许部分由于市场的原因,Windows NT 更多地被看作是一个服务器平台,而不是普通用户的操作系统。这是一种遗憾,因为 Windows NT 虽然有些缺点,但是比 DOS/Windows 3.1 并存模式要优越得多。至今,绝大多数用户仍感觉 Windows 3.1 和 Windows NT 的区别只在于 Windows NT 价格昂贵,需要更高级的设备,但在执行性能上没有什么真正的效果。

在同一时间,OS/2 版本 2.0 面临着严峻的市场问题。总的来说,开发人员乐于使用 Windows 3.1 是因为在 Windows 3.1 环境下开发的产品市场看好。开发针对 OS/2 的产品需要更多花费,对大多数软件开发人员来说,他们没有能力支付如此多的花费,而其产品却仅仅面向较小的市场。由于 OS/2 是自成一体的,因此它没有获得足够的市场份额,无法吸引应用程序开发人员,因而也无法依赖这些开发人员来提高市场份额。

观点的改变

一些观察者认为 80586(如 Pentium)芯片对市场的观念改变起了作用;另一些人认为 Microsoft 的市场目的就是推广 32 位的 Windows 95 操作系统;还有一些人坚持:16 位操作系统的灭亡是由于其固有限制因而无法支持现代用户的性能期望;少数人甚至认为 video 和 CD 的特性需求促使了用户向响应速度更快的系统的转变。不管由于什么原因——肯定是多种而非一种原因可以解释的——32 位操作系统不再被视为只是少数专门用户才需要的系统。

今天,众多用户需要响应更快的系统,期待新的 32 位操作系统支持更快、更强壮、更大型以及具有更多特性的应用程序。

无论什么原因(也许只是到了实现 32 位思想的时候了),32 位操作系统终于成为了新的标准。因此,本书的主要内容是在 32 位环境内开发应用程序。

本书的组织结构

在本书中,我们将着眼于以下若干问题:如何编写 32 位应用程序? 32 位应用程序与 16 位应用程序的区别是什么? 如何使用 32 位编译器提供的服务简化应用程序开发? 以及如何使用 32 位版本的 Microsoft 基础类库(MFC)开发应用程序。

在本书第 1 部分,我们看看 32 位环境的特征。我们从对 Windows NT 和 Windows 95 环境的比较入手,对比其硬件和软件需求、文件和目录的格式和这些 OS 格式的其他选项。我们先简单地看一下在 32 位环境中执行 DOS 应用程序(16 位)的情况,然后考虑将 16 位

Windows 3.1 应用程序移植到 32 位 WinNT/Win95 环境时的问题,并特别注意过时的函数、可移植的 API 和这些环境之间的区别。

在本书第 2 部分,我们看看用户界面的设计。您将从消息框的创建和其他应用程序资源的创建开始学起,您还将学习使用 WinNT 和 Win95 的通用对话框资源。接下来学习应用程序泛及到的概念。用户界面可能是任何应用程序最重要的部分,因为它是您的应用程序的脸面,是您的程序首次被评价的基础。

在第 3 部分,话题转向图形操作。图形方面的工作包括一系列主题,从处理图形环境到画图操作;涉及颜色、调色板和图形工具;以及图元文件、字形;同时包括剪贴板(Clipboard 和 ClipBook)。虽然一些很好的应用程序没有依赖精致的图形,然而如果设计者理解图形如何起作用,以及如何在应用程序设计时互相配合,那么即便是一个简单的应用程序也将从中获益。

第 4 部分讨论 Microsoft 基础类库,以及如何使用面向对象的程序设计方法,如何使用 MFC 类创建应用程序和用户自定义类。虽然面向对象程序设计(OOP)不是必须的(当然不是应用程序开发的全部和最终目标),但是如果使用得当,OOP 是一个有用的工具。当使用不当时,OOP 可能给程序员和后来使用此应用程序的人员造成很大的麻烦。

上面是本书将要讨论的主题和内容的简要大纲。您不必逐行通读本书,您可以前后跳跃挑选那些您感兴趣的主题和示例。无论您采用什么读书方法,您将发现本书的内容是有价值的。

本书约定

本书使用一系列编排格式上的约定使内容的表达更为清晰。这些约定有以下几种:

提示指明了作者简单直接的建议。提示的格式如下:

提示: 如果您使用 Generic Thunks,您可以从专门平台的 DLL 中将它们独立出来。另外,您也可以通过提供在运行时检测平台的手段,从而调用与活动平台相应的 API。

注意是与正文相关的注释。注意的格式如下:

注意: 匈牙利命名规则的标准和版本不是唯一的。不同时间曾发表了好几个版本,不同的工作小组根据不同环境采纳了不同的版本。但是,所谓标准的首要也是最重要的用途旨在加强名字的可读性,因此您可以遵循、改变或修改这些标准。

警告让您知道观察或做某些事是非常重要的。警告的格式如下:

警告: 使用 WinNT 的实用程序 RegEdt32(Registry Editor)可以访问注册簿数据库,在 Win95 下用 RegEdit 实用程序也可以访问此库。如果作修改,要特别谨慎,因为错误或偶然的变化可能严重破坏系统操作。

旁注是可读也可跳过的内容,本书将原版书中的旁注内容在正文中以楷体字表示出来,虽然它包含有用信息,但对您理解本章内容无关紧要。标题用黑体字告诉你要讨论什么问题。

程序代码、函数和方法名称、标志、变量和其他特殊正文使用专门的字体,如 GetClientRect()。冗长的代码范例与正文分开显示,如同在编辑器屏幕上显示的那样。

```

////////////////////////////////////////
//CHellowinView drawing

void CHellowinView::OnDraw(CDC * pDC)
{
    CHellowinDoc * pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    //TODO: add draw code for native data here
    CString      csHello = "hello,world";
    CRect        cRect;

    GetClientRect( cRect );
    pDC->SetTextAlign( TA_BASELINE | TA_CENTER );
    pDC->SetTextColor( ::GetSysColor( COLOR_WINDOWTEXT ) );
    pDC->SetBkMode( TRANSPARENT );
    pDC->TextOut(( cRect. right /2 ), (cRect. bottom /2), csHello);
}

```

被 Visual C++ 使用的热键用粗体和带有下划线的字母表示。当您看到菜单命令或对话框的域用此种格式显示时,您就知道可以按下 Alt 键和用粗体及下划线显示的字母启动菜单命令或访问对话框的域,从而无需使用鼠标。例如,当您看到 **File, **New**** 命令时,您就可以按下 Alt+F, 然后释放 F 键,再按下 N 键获取菜单命令。当然,您也可以使用鼠标访问所有菜单命令和对话框选项。

说明: 与原版书配套的光盘装有源代码和 32 位程序。北京科海培训中心可为读者订购。

订购地址: 北京海淀路 82 号科海培训中心

邮政编码: 100080

联系电话: 62589259 62562449 62569289

目 录

第 1 部分 在 32 位环境下操作

第 1 章 理解 32 位环境 (2)

- 1.1 区别和相似 (2)
 - 1.1.1 硬件环境 (2)
 - 1.1.2 软件环境 (3)
 - 1.1.3 Win95/WinNT 的其他特性 (7)
- 1.2 32 位应用程序开发工具 (8)
 - 1.2.1 创建过程 (8)
 - 1.2.2 Microsoft Developer Studio (10)
- 1.3 本章内容综述 (13)

第 2 章 创建一个 MFC 应用程序 (14)

- 2.1 选定文件类型 (15)
- 2.2 New Project 对话框 (15)
 - 2.2.1 Step1: 结构 (17)
 - 2.2.2 Step2: 数据库支持 (18)
 - 2.2.3 Step3: OLE 支持 (19)
 - 2.2.4 Step4: 应用程序特性 (20)
 - 2.2.5 Step5: 源代码选项 (23)
 - 2.2.6 Step6: 类名选项 (24)
 - 2.2.7 新项目的信息 (25)
- 2.3 项目框架 (25)
- 2.4 ReadMe.TXT 文件 (27)
- 2.5 应用程序框架 (29)
- 2.6 本章内容综述 (32)

第 3 章 从 Windows 3.1 到 WinNT/Win95 (33)

- 3.1 Windows 编程约定 (33)
 - 3.1.1 匈牙利命名规则 (33)
 - 3.1.2 其他前缀约定 (34)
 - 3.1.3 Windows 数据类型 (35)
 - 3.1.4 Windows 宏 (35)
 - 3.1.5 Windows 类型和 MFC 类 (36)

3.1.6 AFX 和 AppWizard 前缀	(37)
3.2 MFC 消息处理的改变	(38)
3.3 Windows NT 和 Windows 95 的比较	(40)
3.4 Thunking(转换)	(41)
3.4.1 Generic 转换块和 Universal 转换块的比较	(43)
3.5 从剪贴板到 DDE 和 OLE2	(43)
3.6 OLE 控件中的 MFC 数据库类	(46)
3.7 INI 文件和注册簿的比较	(46)
3.8 OLE2 注册键	(47)
3.9 其他新特性	(48)
3.9.1 Win95 中新的用户界面类	(48)
3.9.2 MAPI 支持	(48)
3.9.3 网络协议:Windows Sockets	(48)
3.9.4 MFC 移植工具包(MFC Migration Kit)	(49)
3.10 本章内容综述	(49)

第 2 部分 用户界面

第 4 章 使用消息框

(51)

4.1 MessageBox 显示	(51)
4.1.1 MessageBox 图标	(52)
4.1.2 MessageBox 按钮	(53)
4.1.3 MessageBox 的反应	(55)
4.1.4 MessageBox 模式	(56)
4.1.5 AfxMessageBox 显示	(56)
4.2 Dialog-1: 消息框的一个演示实例	(57)
4.2.1 将菜单项与类方法链接	(58)
4.2.2 完成一个成员函数	(60)
4.2.3 报告所选的按钮	(63)
4.2.4 CString 类	(65)
4.2.5 创建一个自定义的 Exit	(67)
4.3 源代码	(69)
4.4 本章内容综述	(74)

第 5 章 介绍 AppStudio 和应用程序资源

(75)

5.1 应用程序资源类型	(75)
5.1.1 加速键	(76)
5.1.2 对话框	(76)
5.1.3 字体资源	(76)
5.1.4 图象资源	(77)
5.1.5 菜单资源	(77)

5.1.6 字符串表	(77)
5.1.7 版本信息	(77)
5.2 应用程序文件和文件类型	(78)
5.2.1 资源文件类型	(79)
5.2.2 编译和链接资源	(80)
5.2.3 动态链接库	(80)
5.2.4 头文件	(80)
5.3 AppStudio	(81)
5.4 资源管理	(81)
5.4.1 创建一个新的资源	(82)
5.4.2 将资源作为二进制数据编辑	(82)
5.4.3 导入和导出资源项	(82)
5.4.4 使用 Symbol Browser 对话框	(83)
5.4.5 使用 Set Includes 对话框	(83)
5.5 本章内容综述	(84)
第 6 章 使用菜单、工具栏和状态栏	(85)
6.1 出发点	(85)
6.2 菜单和菜单结构	(85)
6.3 创建一个菜单	(86)
6.4 结构化的菜单	(87)
6.5 菜单约定和格式	(89)
6.6 菜单限制	(89)
6.7 工具栏	(90)
6.7.1 创建工具栏位图	(90)
6.8 将消息和 ID 值链接到工具栏	(91)
6.8.1 工具提示信息	(95)
6.9 切换工具栏	(96)
6.10 切换菜单	(97)
6.11 本章内容综述	(100)
第 7 章 用户自定义对话框	(101)
7.1 对话框的设计	(101)
7.1.1 创建对话框	(101)
7.2 Standard Controls 对话框	(104)
7.2.1 分配成员变量	(104)
7.2.2 初始化成员变量	(105)
7.2.3 从对话框控件取值	(106)
7.2.4 关闭对话框	(107)
7.2.5 初始化对话框元素	(108)
7.2.6 检索对话框结果	(108)
7.2.7 其他响应方法	(109)

7.3 Special Controls 对话框——WinNT 版本	(109)
7.3.1 滚动条事件消息	(110)
7.3.2 关闭 Special Controls 对话框	(112)
7.4 Special Controls 对话框——Win95 版本	(113)
7.4.1 热键控件	(113)
7.4.2 进度控件	(114)
7.4.3 游标控件	(115)
7.4.4 微调按钮控件	(117)
7.4.5 树视图控件	(118)
7.5 本章内容综述	(124)
第 8 章 用附加控件自定义对话框	(125)
8.1 标签控件	(125)
8.1.1 创建标签控件	(125)
8.1.2 使用子对话框	(128)
8.1.3 使用附加的 CTabCtrl 方法	(131)
8.2 动画控件	(132)
8.2.1 动画控件 AVI 的要求	(133)
8.2.2 创建 CAnimateCtrl 对话框	(133)
8.2.3 通知父窗口	(136)
8.2.4 构建 AVI 文件	(136)
8.2.5 有用的 AVI 控件类	(137)
8.3 本章内容综述	(137)
第 9 章 使用自定义的 OCX 控件	(139)
9.1 介绍 OLE 自定义控件(OCX)	(139)
9.1.1 OCX 属性	(139)
9.1.2 OCX 方法	(140)
9.1.3 OCX 事件	(140)
9.1.4 COleControl 类	(140)
9.2 使用 Visual C++ 版本 2.x 构建 OLE2 自定义控件(OCX)	(140)
9.2.1 选定选项	(141)
9.2.2 修改控件	(142)
9.2.3 接受控件信息	(143)
9.3 使用 Visual C++ 版本 4.0 构建 OLE2 自定义控件(OCX)	(144)
9.3.1 ControlWizard 文件	(146)
9.3.2 可选的 ControlWizard 文件	(148)
9.4 初始化 OCX 控件	(150)
9.5 绘制 OCX 控件	(150)
9.5.1 OCX OnDraw 函数	(150)
9.6 OCX 控件中的鼠标消息	(154)
9.7 OCX 控件属性	(155)

9.7.1 给 OCX 控件增加属性	(156)
9.7.2 增加新属性	(157)
9.8 定义 Property Page 对话框	(161)
9.9 测试 OCX 控件	(162)
9.9.1 注册 OCX 控件	(163)
9.9.2 测试一个 OCX 控件	(163)
9.10 源代码	(165)
9.11 本章内容综述	(170)
第 10 章 使用公用对话框、字体和颜色	(171)
10.1 了解公用对话框和数据类型	(171)
10.2 选定字体和颜色	(172)
10.2.1 LOGFONT 结构	(173)
10.2.2 初始化 LOGFONT 结构	(175)
10.3 选择一种字体	(176)
10.4 选定一种颜色	(180)
10.5 使用 File Open, Save 和 Save As 对话框	(183)
10.5.1 OPENFILENAME 结构	(184)
10.5.2 筛选字符串	(187)
10.5.3 多个文件选择	(187)
10.6 使用 Print 对话框	(187)
10.7 使用 Find 和 Find/Replace 对话框	(189)
10.8 本章内容综述	(193)

第 3 部分 图形操作

第 11 章 了解设备特性和映射模式	(195)
11.1 GetDeviceCaps 函数	(195)
11.2 CreateIC 函数	(195)
11.3 驱动程序版本和硬件类型	(196)
11.4 视频和打印机分辨率	(197)
11.5 RASTERCAPS	(200)
11.6 TEXTCAPS	(201)
11.7 LINECAPS	(202)
11.8 CURVECAPS	(203)
11.9 POLYGONALCAPS	(204)
11.10 本章内容综述	(204)
第 12 章 使用映射模式	(205)
12.1 标准映射模式	(206)
12.1.1 SetMapMode 函数	(206)

12.2	原点、窗口和视口	(212)
12.3	SetWindowExt 和 SetViewportExt 函数	(213)
12.4	相关的模式、视口和窗口函数	(214)
12.4.1	GetMapMode 函数	(214)
12.4.2	其他的关于窗口和视口范围的函数	(214)
12.5	窗口和视口原点	(215)
12.6	WinModes 演示程序	(215)
12.7	Life 演示程序	(219)
12.7.1	Life 游戏	(219)
12.7.2	Forest Life(森林演变游戏)	(221)
12.8	本章内容综述	(221)
第 13 章	使用颜色、调色板和绘图模式	(222)
13.1	Windows 调色板	(223)
13.2	颜色定义	(224)
13.3	标准颜色调色板	(225)
13.4	颜色合成	(226)
13.5	Color1 应用程序	(228)
13.6	COLORREF 值的三种格式	(229)
13.6.1	绝对的 RGB COLORREF 值	(229)
13.6.2	调色板索引(Palette-Index) COLORREF 值	(229)
13.6.3	调色板相关的(Palette-Relative) COLORREF 值	(230)
13.7	抖色(Dithered Colors)	(230)
13.8	自定义颜色	(231)
13.9	捕捉键盘事件	(233)
13.10	其他键盘事件	(235)
13.11	捕捉鼠标事件	(235)
13.12	Color2 演示程序中的鼠标消息	(236)
13.13	颜色到灰度的转换	(237)
13.14	光栅绘图操作(ROP2)	(238)
13.15	Color3 演示程序	(239)
13.16	本章内容综述	(241)
第 14 章	使用绘图工具	(242)
14.1	形状(Shape)和线条(Line)工具	(242)
14.1.1	逻辑画笔(Logical Pen)	(242)
14.1.2	逻辑画刷	(245)
14.2	标准绘图形状	(247)
14.2.1	LineTo 函数	(248)
14.2.2	Rectangle 函数	(249)
14.2.3	RoundRect 函数	(250)
14.2.4	Ellipse 函数	(250)

14.2.5	弧、带弦的弧和扇形块	(250)
14.2.6	Draw1 演示程序	(252)
14.2.7	绘制饼图	(252)
14.2.8	PolyGon 和 PolyPolygon 函数	(254)
14.2.9	多边形填充模式	(255)
14.2.10	PolyPolygon 错误	(256)
14.3	其他绘图函数	(256)
14.3.1	PolyPolyline 函数	(256)
14.3.2	PolyBezier 函数	(257)
14.3.3	PolyDraw 函数	(258)
14.4	本章内容综述	(259)
第 15 章	使用位图和屏幕图象	(260)
15.1	BMP 和 DIB 之比较	(260)
15.1.1	DIB 图象文件	(260)
15.1.2	BITMAPINFOHEADER	(261)
15.1.3	BITMAPINFO 和 BITMAPCOREINFO 之比较	(262)
15.1.4	DIB 位图颜色表	(263)
15.1.5	DIB 图象数据	(263)
15.1.6	OS/2 和 Windows 图象之比较	(263)
15.2	屏幕图象的捕捉与显示	(263)
15.2.1	Capture 实用程序	(264)
15.2.2	设置和清除	(264)
15.2.3	进一步初始化	(265)
15.2.4	OnImageDisPlay 过程	(266)
15.2.5	捕捉屏幕信息	(267)
15.2.6	显示剪贴板	(268)
15.2.7	写位图文件	(269)
15.3	位图压缩格式	(272)
15.3.1	16 色图象所用的 BI_RLE4 图象压缩格式	(272)
15.3.2	256 色图象所用的 BI_RLE8 图象压缩格式	(274)
15.4	读取和显示图象	(274)
15.4.1	ReadDIB File API	(275)
15.4.2	SaveDIBFile API	(276)
15.4.3	PaintDIB API	(277)
15.4.4	ImageAPI 中的其他实用函数	(279)
15.5	BmpImage 演示程序	(280)
15.5.1	BmpImage 中的剪贴板操作	(281)
15.6	本章内容综述	(282)

第 4 部分 Visual C++ 中其他成份

第 16 章 用对象类实现数据接口	(284)
16.1 文件访问方法	(284)
16.1.1 MFC 的文件功能	(284)
16.1.2 用 CFile 打开文件	(286)
16.1.3 访问和共享标志	(287)
16.2 序列化	(288)
16.2.1 制定序列化	(289)
16.2.2 实现定制序列化的方法	(289)
16.2.3 插入和析取运算符	(290)
16.3 CArchive 类	(291)
16.4 随机文件访问	(292)
16.5 CStdioFile 类	(293)
16.5.1 用 CStdioFile 进行文本输入	(293)
16.5.2 用 CStdioFile 进行文本输出	(294)
16.6 AARL_HAM 演示程序	(294)
16.6.1 读写 CArchive 记录	(296)
16.6.2 重载的运算符	(297)
16.6.3 GetEntry 函数	(298)
16.6.4 UpdateData 函数	(299)
16.7 本章内容综述	(299)
第 17 章 理解 ODBC 连接	(300)
17.1 创建一个 ODBC 应用程序	(301)
17.1.1 缺省数据库	(306)
17.1.2 回顾	(307)
17.2 演示程序 AARL3	(307)
17.3 间接编辑——不得已的操作	(309)
17.4 本章内容综述	(311)
第 18 章 使用简单 OLE 操作	(312)
18.1 应用程序集成	(312)
18.1.1 剪贴板服务	(312)
18.1.2 动态数据交换	(312)
18.1.3 图元文件	(312)
18.2 对象链接和嵌入	(313)
18.2.1 复合文档	(314)
18.2.2 链接对象与嵌入对象的比较	(314)
18.2.3 包(Packages)	(315)

18.2.4	OLE 动词	(316)
18.2.5	OLE 包容器与 OLE 服务器的比较	(317)
18.3	创建 OLE 包容器	(317)
18.3.1	COle_cnrView 类	(318)
18.3.2	COle_cnrCntrItem(COleClientItem)类	(321)
18.4	OLE 服务器注册与选择	(322)
18.4.1	实用程序 RegEdit	(322)
18.4.2	选定一个服务器:COleInsertDialog	(323)
18.4.3	注册一个 OLE 服务器	(325)
18.5	创建一个 OLE 服务器	(326)
18.5.1	OLE 服务器类型	(326)
18.5.2	使用 AppWizard 创建 OLE 服务器	(327)
18.5.3	COle_srvrSrvrItem 类	(328)
18.5.4	在图元文件环境中绘图	(330)
18.5.5	CInPlaceFrame 类	(331)
18.5.6	COle_srvrApp 类	(332)
18.5.7	InitInstance 方法	(333)
18.5.8	COle_srvrView 类	(335)
18.5.9	一个有错误的例子	(336)
18.5.10	改正错误的一个办法	(337)
18.6	服务器应用程序菜单	(338)
18.7	实用程序 OLE2View	(340)
18.8	本章内容综述	(341)
第 19 章	MFC 类简介	(342)
19.1	MFC 通用类	(342)
19.1.1	CObject	(343)
19.1.2	文件类	(344)
19.1.3	坐标类	(344)
19.1.4	CString	(345)
19.1.5	时间类	(345)
19.1.6	调试实用类	(345)
19.2	MFC 视对象类	(346)
19.2.1	CDC 和 CDC 的派生类	(347)
19.2.2	CGdiObject 和 CGdiObject 派生类	(348)
19.2.3	CWnd 和 CWnd 派生类	(348)
19.2.4	相关对象类	(354)
19.3	MFC 应用程序框架类	(355)
19.3.1	相关对象类	(357)
19.4	MFC 集合类	(357)
19.4.1	数组集合类	(357)
19.4.2	链表集合类	(358)
19.4.3	映射集合类	(359)

19.4.4 封皮(Wrapper)类	(360)
19.4.5 用户相关的基类	(360)
19.4.6 集合类帮助	(360)
19.5 MFC OLE2 类	(361)
19.5.1 用户客户项	(365)
19.5.2 用户服务器项	(365)
19.5.3 相关的 OLE 类	(366)
19.6 MFC 数据库类	(366)
19.6.1 用户记录集	(368)
19.7 MFC Windows 常用控件类	(368)
19.8 MFC Windows Socket 类	(370)
19.9 MFC DAO 类	(371)
19.10 本章内容综述	(374)
第 20 章 理解 DLI 和派生子类	(375)
20.1 动态链接库	(375)
20.1.1 动态链接进程	(375)
20.1.2 载入时刻动态链接	(376)
20.1.3 运行时刻动态链接	(376)
20.1.4 动态链接的优缺点	(377)
20.1.5 创建动态链接库	(377)
20.1.6 入口点函数	(380)
20.1.7 向 DLL 加入函数	(381)
20.1.8 调用 DLL	(382)
20.1.9 在开发过程中测试 DLL	(384)
20.2 派生子类	(385)
20.2.1 开始	(385)
20.2.2 增加新方法	(386)
20.2.3 创建新类的其他途径	(386)
20.2.4 检索祖先类方法	(387)
20.3 本章内容综述	(388)
第 21 章 类的跟踪和调试	(389)
21.1 Developer Studio 工具集	(389)
21.1.1 Quick Watch 窗口	(390)
21.1.2 Call Stack 窗口	(392)
21.2 其他的调试选项	(392)
21.2.1 Memory 窗口	(393)
21.2.2 Registers 窗口	(393)
21.3 本章内容综述	(394)