HZ BOOKS

Microsoft Press

经典重读

# Windows 核心编程

# Programming Applications for Microsoft Windows

## Fourth Edition

## 原书第4版

(美) Jeffrey Richter 著

黄陇 李虎 译

附赠光盘

机械工业出版社
China Machine Press

# Windows 核心编程

## Programming Applications for Microsoft Windows

Fourth Edition

原书第4版

(美) Jeffrey Richter 著

黄陇 李虎 译

机械工业出版社
China Machine Press

本书是讲解Windows操作系统内部机制的一本专著。作者从基本概念入手，全面系统地介绍了Windows底层实现机制、Windows应用程序的基本构件（包括进程、线程、内存管理、动态链接库、线程本地存储和Unicode）以及各类Windows API等，并列举了大量应用程序示例，精辟地分析了Windows编程的各个难点和要点，为掌握Windows编程技巧提供了一条有效的捷径。

本书适合Windows编程人员参考。

Jeffrey Richter: *Programming Applications for Microsoft Windows, Fourth Edition* (ISBN: 1-57231-996-8).



**版权登记号：图字：01-1999-3114**

**图书在版编目（CIP）数据**

Windows核心编程（原书第4版）/（美）理查德（Richter, J.）著；黄陇，李虎译. —北京：机械工业出版社，2008.5

书名原文：Programming Applications for Microsoft Windows, Fourth Edition

（经典重读）

ISBN 978-7-111-23791-4

Ⅰ. W…　Ⅱ. ①理…　②黄…　③李…　Ⅲ. 窗口软件，Windows—程序设计　Ⅳ. TP316.7

中国版本图书馆CIP数据核字（2008）第040761号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码100037）

责任编辑：周茂辉

北京京北制版厂印刷 · 新华书店北京发行所发行

2008年5月第1 版第1次印刷

186mm × 240mm · 46.5印张

标准书号：ISBN 978-7-111-23791-4

ISBN 978-7-89482-613-8（光盘）

定价：99.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线电话（010）68326294

# 译 者 序

本书是讲解Windows操作系统内部机制的一本专著，作者从基本概念入手，全面系统地介绍了Windows的各种基本构件，如进程、线程、DLL和内存管理等，并列举了大量应用程序，精辟地分析了构件的使用方法，为掌握Windows编程技巧提供了一条有效的捷径。对于不同水平的Windows编程人员来说，本书都具有极好的参考价值。

侯捷先生在他的《Win32多线程程序设计》一书中说，搞Windows程序设计有两方面的资源是不可或缺的，一是MSDN，其次便是本书。可见这本书在Windows程序设计领域中的重要地位。本书的内容相当全面，加上作者亲自编写的十几个精巧程序来现身说法，因而成为Windows程序设计方面的一本圣典级书籍，任何想进行Windows系统级编程或者想提高Windows程序设计水平的人都应该好好研读。

全书由黄陇负责翻译，李虎参与了部分翻译工作并统校全书，此外，李晓丽、刘冬懿、许福、王晓博、宋淼、刘辉、贾荣飞、曹羽中、万钧、卞磊、张军、郭荣锋、曹剑挺、张家旗等对本书进行了检查，对译文给出重要的反馈意见，在此向他们表示感谢。这里还要特别感谢机械工业出版社华章分社的陈冀康主任和周茂辉编辑为本书所做的辛勤工作。需要说明的是，由于译者水平有限，译文中的不妥之处仍在所难免，欢迎广大读者批评指正。

黄陇　李虎

2008年4月17日

# 前 言

Microsoft Windows是一个复杂的操作系统。它提供了如此多的特性和功能，以至于任何人都无法充分地理解系统的每个细节。Windows操作系统的复杂性也让学习它的人很难决定应该从何处入手。我本人总是喜欢从最底层开始，逐步学习和掌握系统的**基本构件**（basic building block）。一旦理解了系统的基本构件，循序渐进地学习高级知识就不再是一件难事。

例如，我在本书中没有刻意讨论**组件对象模型**（Component Object Model，COM）。COM实质上是使用进程、线程、内存管理、DLL、线程本地存储、Unicode等基本构件构建而成的。如果对这些基本构件有所了解，那么理解COM实质上就是理解如何使用这些构件的问题。我非常同情那些企图一步登天来学习COM体系结构的人。他们学习的道路上有很长的路要走，并且受到一些知识漏洞的束缚，这些知识漏洞必定对他们编写代码和制定学习计划造成负面影响。

因此，本书的目的是：介绍每一个Windows操作系统平台上的开发者（至少我的观点如此）都应该非常熟悉的构件。在讨论每一个构件时，本书不仅将描述Windows操作系统如何使用它们，并且还将介绍应用程序如何最好地利用它们。这些构件典型的实现机制是利用范型函数或C++类，将这些Windows构件组合在一起所形成的效果要比它们各自形成的效果之和大得多。

## 今天的Windows平台

目前Microsoft公司销售3种不同的操作系统内核。每种内核都针对一个特定的计算环境进行了优化。Microsoft公司声称每种平台提供了相同的应用程序编程接口（API），以吸引软件开发人员使用Windows操作系统。这意味着，当你学习如何为一个内核编写Windows应用程序时，便了解了如何为任何其他内核编写Windows应用程序。

本书讲述了如何使用Windows API来编写应用程序，因此从本书中学习到的任何知识（在理论上）都适用于所有的内核。实际上，各个内核是有差异的，操作系统的功能是用不同的方法来实现的。这意味着不同的内核的底层概念是相同，但是具体细节可能不同。

下面首先介绍3种不同的Windows内核。

### Windows 2000内核

Window 2000是Microsoft公司推出的高端操作系统。它拥有很多特性，下面是它的一些特性（顺序不分先后）：

- 它可以作为工作站、服务器和数据中心来运行。
- 该系统非常健壮，它能避免不完善的应用程序导致的系统崩溃。
- 该系统非常安全，它能阻止对系统管理的资源（如文件和打印机）的未授权访问。
- 它拥有丰富的工具和实用程序，供组织中的管理员对操作系统进行管理。

- 内核大多是用C和C++编写的，因此该系统可以容易地移植到其他CPU架构。
- 系统本身支持Unicode，因此，系统的本地化和国际化变得更容易。
- 它的内存管理特性提供了极其丰富、高效的功能。
- 结构化异常处理（SEH）特性使得错误恢复更容易进行。
- 动态链接库（DLL）允许系统灵活地扩展。
- 多线程和对多处理器的支持，这使得系统具备很好的伸缩性，便于性能改进。
- 文件系统的特征提供了方便的途径用来跟踪用户如何在其机器上操纵数据。

### Windows 98内核

Windows 98是Microsoft公司推出的面向消费者的操作系统。它拥有Windows 2000的许多特性，但是却没有包含它的某些关键特性。例如，Windows 98不够健壮（一个应用程序可能导致系统崩溃），不够安全，是一个单处理器内核（这限制了它的伸缩性），并且它对Unicode的支持也不如Windows 2000。

Microsoft的目标是消除Windows 98，因为Windows 98内核不能提供Windows 2000内核的所有特性，并且修改Windows 98内核以支持这些特性显得过于困难。另外，如果要修改Windows 98内核来支持这些特性，那么该内核就要无论如何与Windows 2000内核相匹配。因此，Windows 2000内核将会在很长时间内伴随着我们，而Windows 98内核（如果真是那样的话）或许只有几年的生存时间。

为什么Windows 98仍然能够继续生存呢？原因是Windows 98比Windows 2000有更好的终端用户友好性。普通消费者不喜欢登录计算机，不喜欢管理计算机等。此外，消费者往往比企业员工更喜欢玩游戏（这是有理由的）。许多较老的游戏往往直接访问硬件，在Windows 98中这可能导致系统崩溃。Windows 2000是一种健壮的操作系统内核，它不允许这种情况的发生。Windows 2000将会立即终止试图直接访问硬件的应用程序，这些应用程序不会对计算机和其他应用程序产生不良影响。

由于上述原因，Windows 98仍然伴随着我们，并且它的消费者群体仍相当大。Microsoft公司正在积极努力，以使Windows 2000变得对终端用户更加友好，Windows 2000将很快推向消费市场。由于Windows 2000和Windows 98的内核拥有类似的特性集，而且这两种内核已经大量安装使用，因此本书将重点介绍这两种内核。

本书将介绍Windows的各种不同的特征。在适当的地方，将在文字中插入特定于内核的标记（如下所示），以引起读者对某一特定内核的实现细节的注意。

**Windows 98** *这是特定于Windows 98平台的实现细节。*

**Windows 2000** *这是特定于Windows 2000平台的实现细节。*

尽管本书没有明确提到Windows 95，但是关于Windows 98的信息也同样适用于Windows 95，因为这两种操作系统使用完全相同的内核。本书只提到Windows 98，而不是同时提到Windows 98和Windows 95，目的是让本书的正文有更好的可读性。

### Windows CE内核

Windows CE是Microsoft公司推出的最新Windows操作系统。这种新型操作系统的推出是为

了适应小型硬件设备的需要，这些设备包括手提式计算机、车用PC、智能终端、电烤箱、微波炉和自动售货机等。它们的耗电量通常较小，内存量不大，并且几乎没有（也可能有很少的）持久性存储器（如磁盘驱动器）。由于硬件的限制，Microsoft公司不得不开发一种新型操作系统内核，它的规模既小于Windows 2000，也小于Windows 98。

令人惊奇的是，Windows CE的功能非常强大，提供的特性非常多。由于Windows CE的机器主要是供个人使用的，因此，它的内核不需要许多对管理和伸缩性特性的支持。虽然本书并没有专门介绍Windows CE，但是书中介绍的许多概念也适用于该平台。该平台与其他平台之间确实存在差异，主要是Windows CE对不同的功能施加了一些限制。本书提供的资料应该视为对Windows CE的其他信息的补充。

### 未来的Windows平台（64位Windows 2000）

未来正向我们招手。Microsoft公司正在加紧开发Windows 2000内核，以使它成为一种真正的64位操作系统。预计它的名称是64位Windows 2000，上市时间预计为2000年。最初这种64位内核将在康柏公司的Alpha CPU（AXP64）和Intel的新型64位CPU（IA-64）架构上运行。

康柏的Alpha CPU一直采用64位的架构。因此，如果你已经拥有一台Alpha计算机，那么只需要安装64位的Windows 2000，便可运行完整的64位操作系统。Intel公司的Pentium系列（和更早的系列）CPU采用32位的架构（IA-32）。采用这种CPU的计算机不能运行64位的Windows 2000。目前Intel公司正在设计一种新的64位CPU架构。使用该结构的最早的芯片代号为Merced。采用Merced CPU的计算机将于2000年推向市场。

64位Windows 2000引起了我的极大兴趣，我一直在编写最新的代码来迎接这个时代的到来。目前Microsoft公司的Web站点包含了许多关于64位Windows 2000的文章以及它对软件开发人员的重大意义的评论。我很高兴能够向读者汇报一下这方面的情况：

- 64位Windows 2000内核是现有的32位Windows 2000内核的一个端口。这意味着关于32位Windows 2000的所有细节和复杂性也适用于64位的运行环境。实际上，Microsoft已经修改了32位Windows的源代码，以便它能够在编译后产生一个32位或64位的系统。两者共享一份基础源代码，因此，新的特性和错误调试能力可被同时应用于这两个系统。
- 由于这两种内核使用相同的代码和底层概念，因此，两种平台上的Windows API是相同的。这意味着不必重新设计或重新实现应用程序，便可以在64位Windows上运行。只需要对源代码稍加修改，然后重新创建应用程序。
- 由于移植32位应用程序非常容易，因此，我们很快就能看到支持64位应用程序开发的工具（如Microsoft公司的Developer Studio）。
- 为了实现向后兼容，64位Windows能够执行32位的应用程序。但是，如果创建了真正的64位应用程序，那么应用程序的运行性能肯定将得到大幅度提高。
- 从32位到64位，并没有太多要学习的新东西。大多数据类型的长度仍然是32位。这些数据类型包括`int`、`DWORD`、`LONG`和`BOOL`等。实际上在大多数情况下只需要注意指针和某些句柄，因为它们现在变成了64位的值。

由于Microsoft公司的Web站点提供了许多关于如何将现有的源代码改成64位的信息，因此本书不再对它们进行详细介绍。不过本书每一章都考虑到了64位Windows的问题。在适当的地

方，我会介绍针对64位Windows的具体内容。此外，我用64位编译器编译了本书中的所有应用程序示例，并且在一个Alpha CPU的早期64位Windows版本上测试了这些应用程序。因此，如果要编写本书的应用程序示例，并且按照要求的方法去做，那么，在创建单个基础源代码时将不会遇到太多困难，有了这个基础源代码，就能够很容易地为32位或64位Windows进行编译。

## 第四版的新内容

读过《Advanced Windows》这本书的读者会发现有一个大变化：书名变了。Microsoft出版社和我都认为这个名字并没有完全涵盖本书的内容。对于初学者，前一版的书名并不能表明这本书是面向开发者的。因此，书店可能会把这本书放错了位置。而且，我收到大量的读者来信，他们说本书过于“高级”，想让我推荐一本更初级（或中级）的Windows编程读物。

就内容而言，第四版（即本书）实际上是一本全新的著作。我重点对书中的一些章节进行压缩，并突出了某些内容，并对全书做了重新组织。我希望这些改进能够让这一版更容易阅读和理解。例如，Unicode那一章现在被安排在本书的开始，因为Unicode会在某种程度上影响其他大多数主题的介绍。

不仅如此，和前一版相比，这一版中每个主题的介绍深度都有所增加。具体地讲，本书介绍了更多的系统内部工作细节，以便让读者更准确地了解系统的后台是如何工作的。此外本书还更加详细地介绍了C/C++运行时库如何与操作系统交互——特别是关于过程、线程的创建与销毁，以及动态链接库等方面的内容。

除了内容组织和讲解深度进行了调整外，这一版还增加了许多新内容。以下是这些新内容的一部分：

- **Windows 2000的新特征**。如果不介绍Windows 2000提供的新特征，那么这本书就称不上真正的修订版。这一版在作业内核对象、线程合并函数、线程调度问题、地址窗口扩展、工具帮助和稀疏文件等许多主题上增加了新的内容。
- **64位Windows支持**。描述了某些64位Windows的具体问题，所有的应用程序示例都在64位Windows平台上经过创建和测试。
- **实用的应用程序示例**。新一版替换了旧版中的一些老的应用程序示例，以使它们能够更中肯地反映如何解决真实世界中的编程问题。
- **C++的使用**。新的应用程序示例现在用C++编写，因为许多读者提出了这样的要求。这样一来，应用程序示例只需要较少的代码行，它们的逻辑也比较容易学习和理解。
- **可重用的代码**。只要可能，我都尽量编写通用的和可重用的代码。你可以从中摘取出单独的函数或完整的C++类来编写你自己的应用程序，它们几乎不需要或只需要很少的修改。C++的使用，使得我们更容易地编写可重用的代码。
- **VMMap实用程序**。这个来自较早版本的应用程序示例得到了显著增强。新的VMMap应用程序示例现在能够遍历任何进程的地址空间，显示已经映射到内存地址空间的任何数据文件的路径名，刷新它的显示屏，将内存信息复制到剪贴板，并且可以（视情况而定）只显示区域或区域中的地址块。
- **ProcessInfo实用程序**。这是一个新增的应用程序示例。它能显示系统正在运行哪些进程，该模块正在使用哪些DLL。一旦你选择一个进程，本实用程序就能产生VMMap实用程序

以遍历其进程的整个地址空间。ProcessInfo实用程序还能显示系统中加载了哪些模块，哪些可执行程序正在使用该模块。它还能显示哪些模块由于不恰当基址重置而进行了重新定位。

- **LISWatch实用程序**。它也是一个新的应用程序示例。它为窗口监控系统级的和特定于线程的本地输入状态改变。该实用程序能够极大地帮助跟踪用户界面焦点的变更问题。
- **性能问题**。提供了许多关于如何编写更小和运行更快的代码的信息和提示。增加了对数据对齐、处理器亲缘性、CPU高速缓存行边界问题、线程同步问题、模块基址重置、模块绑定和延迟加载DLL等的介绍。
- **显著改进了有关线程同步的内容**。完全重新编写和组织了线程同步这一部分内容。新的内容首先介绍高性能的线程同步方法，然后再介绍低性能的线程同步方法。增加了“线程同步工具包”这一章，其中提供了一些常见线程同步场景下的可复用代码。
- **可执行文件的详细格式**。更详细地描述了可执行文件的格式和DLL模块文件的格式。介绍了这些模块的各个不同分区和特殊的链接程序开关，它们让你能够对该模块进行某些非常“酷”的操作。
- **更详细的DLL信息**。重新编写和组织了有关DLL的章节。有关DLL的第一章将回答两个基本DLL问题“什么是DLL”以及“如何创建DLL”。其他有关DLL的章节则深入挖掘DLL的高级特征（许多都是新特征），例如显示链接、延迟加载、函数转发器、DLL重定向（这是Windows 2000的新特征）、模块基址重置和绑定等。
- **API挂接**。多年来我收到了太多人发来的邮件，询问关于API挂接的问题，这是真的。最后我决定把这部分内容加到这本书中。我编写了几个C++类，它们能让你轻松地在进程的一个或多个模块中挂接API。其中的代码甚至能够捕获运行时对`LoadLibrary`和`GetProcAddress`的调用，以便强制实施API挂接。
- **改进了结构化异常处理部分**。本书对结构化异常处理部分的内容进行重新撰写和编排。提供了关于未处理的异常情况的详细信息，还提供了一个C++类，它包含了用结构化异常处理手段来处理虚拟内存的正确方法。此外还增加了对异常处理调试以及C++异常处理与结构化异常处理之间的关系介绍。
- **错误处理**。这是新增加的一章，介绍在调用Windows函数时应该如何恰当地检测错误。这一章还描述了几个些调试技巧以及如何让函数能够报告错误。
- **Windows Installer（Windows安装程序）**。不可思议，在前几版中我竟然把它忘记了：本书所附的光盘上的应用程序示例利用了Windows 2000中内置的新的安装程序。该安装程序可以让你更细粒度地控制想要安装的部件，还允许你使用Add/Remove程序控制面板来卸载书中的应用程序示例和可执行文件。如果你的机器运行的是Windows 95、Windows 98或Windows NT 4.0操作系统，那么光盘上的Setup程序运行后首先将自动安装Windows Installer。当然，如果愿意的话，随时可以直接使用光盘上的源文件和可执行文件。

## 本书没有错误

这一段的标题清楚地表明了我要说什么。我们大家都知道，一本书完全没有错误是根本不可能的。编辑人员和我一直在努力工作，为的是编写出一部准确的、最新的、详尽的、易读的、

容易理解的并且无错的专业著作。即使把一群天才聚在一起与我合作，我也知道有时候会不知不觉地出现差错。如果你发现本书有任何错误（特别是程序中的bug），我非常感谢您的宝贵意见，请将错误信息通过网站发送给我：http://www.JeffreyRichter.com。

## 关于光盘/系统需求

本书所附的光盘包含了书中提供的所有应用程序示例的源代码和可执行文件。所有应用程序示例都是用Microsoft Visual C++ 6.0编写和编译的。大多数应用程序可在Windows 95、Windows 98、Windows NT 4.0和Windows 2000平台上运行，但是其中有些应用程序需要的特征只存在于Windows NT 4.0或Windows 2000平台中。需要用Visual Studio工作区文件和常用的头文件。在根目录下，每个应用程序示例都有一个独立的目录。x86和Alpha32目录下包含所有应用程序示例的调试版本，这样就可以直接通过光盘来运行这些应用程序。

当将光盘插入驱动器时，就会自动显示Welcome屏幕。如果不出现这个屏幕，请进入驱动器的Welcome目录，并执行PressCD.exe应用程序。

## 支持

Microsoft出版社会提供对本书的修正信息，它的网址是http://mspress.Microsoft.com/mspress/support。

如果想对本书提出评论意见，或提出问题，请用书面邮件或电子邮件将它们传送给Microsoft出版社：

Microsoft Press
Attn:Progamming Applications for Microsoft Windows, 4th ed.,editor
One Microsoft Way
Redmond,WA 98052-6399
mspinput@microsoft.com

英文原书书号：ISBN 1-57231-996-8

# 作译者简介

## 作者简介

Jeffery Richter为软件开发者开设编程课程（http://www.SolSem.com）。它的客户来自许多知名公司，如Allen-Bradley、AT&T、Caterpillar、Digital、DreamWorks、GE Medical Systems、Hewlett-Packard、IBM、Intel、Intuit、Microsoft、Pitney Bowes、Sybase、Tandem和Unisys。

Jeff经常在工业界举行的一些会议上发表演讲，包括Software Development and COMDEX、Boston University's WinDev、Microsoft's Professional Developer's Conference和TechEd等会议。Jeff还是Microsoft Systems Journal的编委，是其中Win32 Q&A栏目的撰稿人，并且发表了多篇主题文章。

Jeff住在华盛顿州的Bellevue。他的业余爱好包括直升飞机、魔术和鼓乐。对经典的摇滚乐和爵士乐，Jeff也非常喜欢。

## 译者简介

**黄陇**：男，1975年生，北京航空航天大学出站博士后，发表论文近40篇，荣获军队科技进步奖等多项科研奖项。主要研究方向为C++程序设计、软件评测、软件质量与可靠性工程。

**李虎**：男，1974年生，北京航空航天大学计算机学院讲师，博士，在重要期刊及会议上发表论文10余篇，主持多项国家级科技项目，曾获国防科学技术三等奖，是多个计算机科学类学术期刊的审稿人，申请国家技术发明专利3件，先后翻译出版10余部计算机科学著作。

# 目　录


译者序
前言
作译者简介

## 第一部分　程序员必读

第1章　错误处理 ……1
1.1　自定义错误处理的实现 ……4
1.2　错误显示例程 ……5
第2章　Unicode ……11
2.1　字符集 ……11
2.1.1　单字节和双字节字符集 ……11
2.1.2　Unicode：宽字节字符集 ……12
2.2　为何需要Unicode ……13
2.3　Windows 2000和Unicode ……13
2.4　Windows 98和Unicode ……13
2.5　Windows CE和Unicode ……14
2.6　评论 ……14
2.7　关于COM ……15
2.8　如何编写Unicode源代码 ……15
2.8.1　C运行库的Unicode支持 ……15
2.8.2　Windows定义的Unicode数据类型 ……17
2.8.3　Windows系统中的Unicode函数和ANSI函数 ……17
2.8.4　Windows字符串函数 ……19
2.9　让应用程序符合ANSI和Unicode规范 ……19
2.9.1　Windows字符串函数 ……20
2.9.2　资源 ……22
2.9.3　确定文本是ANSI型还是Unicode型 ……22
2.9.4　在Unicode和ANSI间转换字符串 ……24
第3章　内核对象 ……27
3.1　内核对象的概念 ……27
3.1.1　使用计数 ……27
3.1.2　安全性 ……28
3.2　内核对象句柄表 ……30
3.2.1　创建内核对象 ……30
3.2.2　关闭内核对象 ……32
3.3　进程间内核对象的共享 ……32
3.3.1　对象句柄的继承性 ……33
3.3.2　改变句柄标志 ……35
3.3.3　命名对象 ……36
3.3.4　终端服务器命名空间 ……40
3.3.5　复制对象句柄 ……40

## 第二部分　完成编程任务

第4章　进程 ……45
4.1　编写第一个Windows应用程序 ……46
4.1.1　进程的实例句柄 ……49
4.1.2　进程的前一个实例句柄 ……50
4.1.3　进程的命令行 ……51
4.1.4　进程的环境变量 ……52
4.1.5　亲缘性 ……55
4.1.6　进程的错误模式 ……55
4.1.7　当前驱动器和目录 ……55
4.1.8　当前目录 ……56
4.1.9　系统版本 ……57
4.2　CreateProcess函数 ……60
4.2.1　pszApplicationName和pszCommandLine ……60
4.2.2　psaProcess、psaThread和bInheritHandles ……62
4.2.3　fdwCreate ……64
4.2.4　pvEnvironment ……65
4.2.5　pszCurDir ……66
4.2.6　psiStartInfo ……66

4.2.7 ppiProcInfo ……69
4.3 进程的终止 ……70
4.3.1 主线程的入口函数返回 ……71
4.3.2 ExitProcess函数 ……71
4.3.3 TerminateProcess函数 ……72
4.3.4 进程中所有线程的运行终止 ……73
4.3.5 进程的运行终止 ……73
4.4 子进程 ……74
4.5 枚举系统中运行的进程 ……76
第5章 作业 ……94
5.1 对作业进程的限制 ……96
5.2 把进程放入作业 ……102
5.3 终止作业中所有进程的运行 ……103
5.4 查询作业统计信息 ……103
5.5 作业通知信息 ……106
5.6 JobLab示例应用程序 ……108
第6章 线程的基本知识 ……126
6.1 创建线程的时机 ……126
6.2 何时不能创建线程 ……127
6.3 编写第一个线程函数 ……128
6.4 CreateThread函数 ……129
6.4.1 psa ……130
6.4.2 cbStack ……130
6.4.3 pfnStartAddr和pvParam ……130
6.4.4 fdwCreate ……131
6.4.5 pdwThreadID ……132
6.5 终止线程 ……132
6.5.1 线程函数返回 ……132
6.5.2 ExitThread函数 ……133
6.5.3 TerminateThread函数 ……133
6.5.4 在进程终止运行时终止线程 ……134
6.5.5 线程终止运行时发生的操作 ……134
6.6 线程的一些内部细节 ……134
6.7 对于C/C++运行时库的考虑 ……137
6.7.1 Oops——错误地调用了CreateThread …144
6.7.2 不该调用的C/C++运行时库函数 ……145
6.8 线程的身份标识 ……145
第7章 线程的调度、优先级和亲缘性 ……149
7.1 挂起和恢复线程的运行 ……150
7.2 进程的挂起和唤醒 ……151
7.3 睡眠 ……152
7.4 线程切换 ……153
7.5 线程的运行时间 ……153
7.6 上下文环境切换 ……156
7.7 线程优先级 ……160
7.8 优先级的抽象说明 ……161
7.9 编程优先级 ……164
7.9.1 动态提高线程的优先级等级 ……166
7.9.2 为前台进程调整调度程序 ……167
7.9.3 Scheduling Lab示例应用程序 ……168
7.10 亲缘性 ……176
第8章 用户模式下的线程同步 ……181
8.1 原子访问：互锁函数族 ……181
8.2 高速缓存行 ……186
8.3 高级线程同步 ……188
8.4 临界区 ……190
8.4.1 临界区准确的描述 ……192
8.4.2 临界区与循环锁 ……195
8.4.3 临界区与错误处理 ……195
8.4.4 有用的提示和技巧 ……196
第9章 线程与内核对象的同步 ……200
9.1 等待函数 ……201
9.2 成功等待的副作用 ……204
9.3 事件内核对象 ……205
9.4 等待定时器内核对象 ……215
9.4.1 用等待定时器给APC项排队 ……218
9.4.2 定时器的松散特性 ……220
9.5 信号量内核对象 ……220
9.6 互斥内核对象 ……222
9.6.1 释放问题 ……223
9.6.2 互斥对象与临界区的比较 ……224
9.6.3 Queue应用程序示例 ……225

9.7 线程同步对象表 ……234
9.8 其他线程同步函数 ……235
9.8.1 异步设备I/O ……235
9.8.2 WaitForInputIdle ……235
9.8.3 MsgWaitForMultipleObjects(Ex) …236
9.8.4 WaitForDebugEvent ……236
9.8.5 SignalObjectAndWait ……237
第10章 线程同步工具包 ……239
10.1 临界区的实现：Optex ……239
10.2 创建线程安全的数据类型和反信号量 …251
10.3 单写入多读出程序的保护 ……263
10.4 WaitForMultipleExpressions函数的实现 ……272
第11章 线程池 ……288
11.1 场景1：异步调用函数 ……289
11.2 场景2：按规定的时间间隔调用函数 …291
11.3 场景3：在某个内核对象变为已通知状态时调用函数 ……297
11.4 场景4：异步I/O请求运行完成时调用函数 ……299
第12章 纤程 ……301
12.1 使用纤程 ……301
12.2 Counter示例应用程序……303

## 第三部分 内存管理

第13章 Windows内存结构……313
13.1 进程的虚拟地址空间 ……313
13.2 虚拟地址空间分区 ……313
13.2.1 无效断点分配分区（适于Windows 2000和Windows 98）……314
13.2.2 MS-DOS/16位Windows应用程序兼容分区（仅适于Windows 98）……315
13.2.3 用户模式分区（适用Windows 2000和Windows 98）……315
13.2.4 64KB禁止进入分区（仅适用于Windows 2000）……316
13.2.5 共享的MMF分区（仅适用于Windows 98）……317
13.2.6 内核模式分区（使用于Windows 2000和Windows 98）……317
13.3 地址空间区域 ……317
13.4 在地址空间区域中提交物理存储器 ……318
13.5 物理存储器和页面文件 ……319
13.6 保护属性 ……321
13.6.1 Copy-On-Write访问 ……322
13.6.2 特殊访问保护属性标志 ……323
13.7 综合使用所有元素 ……323
13.7.1 区域的内部详情 ……326
13.7.2 Windows 98上地址空间的差异 ……330
13.8 数据对齐的重要性 ……333
第14章 虚拟内存 ……338
14.1 系统信息 ……338
14.2 虚拟内存的状态 ……346
14.3 确定地址空间状态 ……352
14.3.1 VMQery函数 ……353
14.3.2 虚拟内存表应用程序示例 ……360
第15章 应用程序中虚拟内存的使用 ……371
15.1 地址空间中保留区域 ……371
15.2 在保留区域中提交存储器 ……373
15.3 同时进行保留区域并提交内存 ……373
15.4 何时提交物理存储器 ……374
15.5 物理存储器的回收和地址空间区域的释放 ……375
15.5.1 何时回收物理存储器 ……376
15.5.2 虚拟内存分配示例应用程序 ……377
15.6 改变保护属性 ……386
15.7 清除物理存储器内容 ……387
15.8 地址窗口扩展（仅使用于Windows 2000） ……390
第16章 线程栈 ……404
16.1 Windows 98下的线程栈 ……406
16.2 C/C++运行时库中的栈检测函数 ……408

16.3 Summation示例应用程序 ……409
第17章 内存映射文件 ……416
17.1 内存映射的可执行文件和DLL文件 ……416
17.1.1 可执行文件或DLL的多个实例之间无法共享的静态数据 ……417
17.1.2 在可执行文件或DLL的多个实例之间共享静态数据 ……419
17.1.3 AppInst示例应用程序 ……424
17.2 内存映射数据文件 ……429
17.2.1 方法1：一个文件，一个缓存 ……429
17.2.2 方法2：两个文件，一个缓存 ……429
17.2.3 方法3：一个文件，两个缓存 ……430
17.2.4 方法4：一个文件，零个缓存 ……430
17.3 使用内存映射文件 ……430
17.3.1 步骤1：创建或打开文件内核对象 …431
17.3.2 步骤2：创建文件映射内核对象 ……432
17.3.3 步骤3：将文件数据映射到进程地址空间 ……434
17.3.4 步骤4：进程地址空间中撤销文件数据的映像 ……437
17.3.5 步骤5和步骤6：关闭文件映射对象和文件对象 ……438
17.3.6 文件倒序示例应用程序 ……439
17.4 使用内存映射文件处理大文件 ……447
17.5 内存映射文件的一致性 ……448
17.6 设定内存映射文件的基地址 ……449
17.7 实现内存映射文件的具体细节 ……450
17.8 使用内存映射文件在进程之间实现数据共享 ……452
17.9 受页面文件支持的内存映射文件 ……452
17.10 稀疏提交的内存映射文件 ……459
第18章 堆 ……473
18.1 进程的默认堆 ……473
18.2 创建辅助堆的原因 ……474
18.2.1 保护组件 ……474
18.2.2 更有效地管理内存 ……475
18.2.3 进行本地访问 ……475
18.2.4 减少线程同步开销 ……475
18.2.5 快速释放 ……475
18.3 创建辅助堆的方法 ……476
18.3.1 分配堆中的内存块 ……477
18.3.2 改变内存块的大小 ……478
18.3.3 获取内存块的大小 ……479
18.3.4 释放内存块 ……479
18.3.5 销毁堆 ……479
18.3.6 用C++程序使用堆 ……479
18.4 其他堆函数 ……482

## 第四部分 动态链接库

第19章 DLL基础 ……485
19.1 DLL与进程的地址空间 ……486
19.2 DLL的总体运行情况 ……487
19.3 创建DLL模块 ……490
19.3.1 导出的真正含义 ……492
19.3.2 使用非Visual C++工具创建DLL ……493
19.4 创建可执行模块 ……494
19.5 运行可执行模块 ……497
第20章 DLL高级技术 ……499
20.1 显式加载DLL模块和符号链接 ……499
20.1.1 显式加载DLL模块 ……499
20.1.2 显式卸载DLL模块 ……501
20.1.3 显式链接到导出符号 ……503
20.2 DLL的入口函数 ……504
20.2.1 DLL_PROCESS_ATTACH通知 ……505
20.2.2 DLL_PROCESS_DETACH通知 ……506
20.2.3 DLL_THREAD_ATTACH通知 ……508
20.2.4 DLL_THREAD_DETACH通知 ……509
20.2.5 顺序调用DllMain ……509
20.2.6 DllMain和C/C++运行时库 ……512
20.3 延迟加载DLL ……513

20.4 函数转发器 ……523
20.5 已知的DLL ……523
20.6 DLL重定向 ……525
20.7 模块的基址重置 ……525
20.8 绑定模块 ……531
第21章 线程本地存储 ……533
21.1 动态TLS ……533
21.2 静态TLS ……537
第22章 DLL注入以及API挂接 ……539
22.1 DLL注入：一个例子 ……539
22.2 使用注册表注入DLL ……541
22.3 使用Windows钩子注入DLL ……542
22.4 使用远程线程注入DLL ……555
22.4.1 Inject Library示例应用程序 ……559
22.4.2 Image Walk DLL ……567
22.5 使用特洛伊DLL注入DLL ……570
22.6 将DLL作为调试程序注入 ……570
22.7 在Windows 98平台上使用内存映射文件注入代码 ……570
22.8 使用CreateProcess来注入代码 ……571
22.9 API挂接：一个例子 ……571
22.9.1 通过覆写代码实现API挂接 ……572
22.9.2 通过操作模块的导入部分来实现API挂接 ……573
22.9.3 LastMsgBoxInfo示例应用程序 ……576

## 第五部分 结构化异常处理

第23章 终止处理例程 ……593
23.1 Funcenstein1 ……594
23.2 Funcenstein2 ……595
23.3 Funcenstein3 ……596
23.4 Funcfurter1 ……597
23.5 小测验：FuncaDoodleDoo ……598
23.6 Funcenstein4 ……599
23.7 Funcarama1 ……600
23.8 Funcarama2 ……600
23.9 Funcarama3 ……601
23.10 Funcarama4:最终的边界 ……602
23.11 有关finally块的说明 ……603
23.12 Funcfurter2 ……604
23.13 SEH终止示例应用程序 ……605
第24章 异常处理程序和软件异常 ……608
24.1 通过例子理解异常过滤器和异常处理程序 ……608
24.1.1 Funcmeister1 ……608
24.1.2 Funcmeister2 ……609
24.2 EXCEPTION_EXECUTE_HANDLER ……611
24.2.1 一些有用的例子 ……611
24.2.2 全局展开 ……614
24.2.3 暂停全局展开 ……616
24.3 EXCEPTION_CONTINUE_EXECUTION ……617
24.4 EXCEPTION_CONTINUE_SEARCH ……619
24.5 GetExceptionCode ……621
24.5.1 与内存相关的异常 ……621
24.5.2 与异常相关的异常 ……622
24.5.3 与调试相关的异常 ……622
24.5.4 与整数相关的异常 ……622
24.5.5 与浮点数相关的异常 ……622
24.6 GetExceptionInformation ……624
24.7 软件异常 ……628
第25章 未处理异常和C++异常 ……631
25.1 即时调试 ……633
25.2 关闭异常消息框 ……634
25.2.1 强制进程终止运行 ……634
25.2.2 包装一个线程函数 ……634
25.2.3 包装所有的线程函数 ……635
25.2.4 自动调用调试器 ……635
25.3 自己调用UnhandledExceptionFilter ……636
25.4 UnhandledExceptionFilter函数的内部细节 ……636
25.5 异常和调试程序 ……638
25.6 C++异常与结构化异常比较 ……652

## 第六部分 窗口

第26章 窗口消息 ……657
26.1 线程的消息队列 ……657
26.2 将消息投送到一个线程的消息队列中 …658
26.3 向窗口发送消息 ……660
26.4 唤醒一个线程 ……664
26.4.1 队列状态标志 ……665
26.4.2 从线程队列中提取消息的算法 ……666
26.4.3 使用内核对象或者队列状态标志来唤醒一个线程 ……669
26.5 使用消息发送数据 ……671
26.6 Windows处理ANSI/Unicode字符和字符串的方法 ……678
第27章 硬件输入模型与本地输入状态 ……681
27.1 原始输入线程 ……681
27.2 本地输入状态 ……682
27.2.1 键盘输入和焦点 ……683
27.2.2 鼠标光标管理 ……686
27.3 将虚拟输入队列和本地输入状态相关联 ……687
27.3.1 LISLab示例应用程序 ……689
27.3.2 LISWatch示例应用程序 ……704

## 附 录

附录A 构建环境 ……713
附录B 消息解析器、子控件宏以及API宏…725