TURING 图灵程序设计丛书 微软技术系列

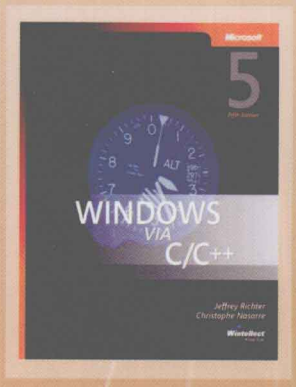
Microsoft

Windows via C/C++ Fifth Edition

Windows核心编程

（第5版·英文版）

[美] Jeffrey Richter
[法] Christophe Nasarre 著



- Windows程序设计巅峰之作
- 新版涵盖Windows Vista和Windows Server 2008最新核心特性
- Windows程序员迈向卓越的必由之路

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 微软技术系列

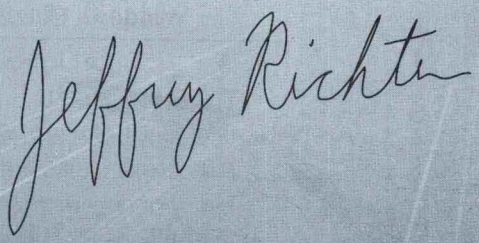
Windows via C/C++

Fifth Edition

Windows核心编程

（第5版·英文版）

[美] Jeffrey Richter
[法] Christophe Nasarre 著



人民邮电出版社
北 京

图书在版编目（CIP）数据

Windows 核心编程：第 5 版：英文 /（美）里克特（Richter，J.），（法）纳萨尔（Nasarre，C.）著；—北京：人民邮电出版社，2008.8
（图灵程序设计丛书）
ISBN 978-7-115-18212-8

Ⅰ. W… Ⅱ. ①里…②纳… Ⅲ. 窗口软件，Windows—程序设计—英文 Ⅳ. TP316.7

中国版本图书馆 CIP 数据核字（2008）第 077689 号

内 容 提 要

本书是 Windows 程序设计领域的名著，涵盖了 Windows 的最新版本 Vista 以及 Windows XP 的最新内容。书中全面深入地介绍了 Windows 的各种基本要素，如进程、线程池、虚拟内存、DLL、设备 I/O 和 SEH 等，并列举了大量应用程序，精辟地分析了要素的使用方法。

本书适于各层次 Windows 编程人员阅读。

图灵程序设计丛书

Windows 核心编程（第 5 版·英文版）

◆ 著 [美] Jeffrey Richter [法] Christophe Nasarre
责任编辑 傅志红

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
三河市潮河印业有限公司印刷

◆ 开本：800×1000 1/16
印张：52.75
字数：1013 千字 2008 年 8 月第 1 版
印数：1 – 3 000 册 2008 年 8 月河北第 1 次印刷

著作权合同登记号 图字：01-2008-1720 号

ISBN 978-7-115-18212-8/TP

定价：128.00 元

读者服务热线：(010)88593802 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

Dedication

To Kristin, words cannot express how I feel about our life together.
I cherish our family and all our adventures.
I'm filled each day with love for you.

To Aidan, you have been an inspiration to me and have taught me
to play and have fun. Watching you grow up has been so rewarding
and enjoyable for me. I feel lucky to be able to
partake in your life; it has made me a better person.

To My New Baby Boy (shipping Q1 2008),
you have been wanted for so long it's hard to believe
that you're almost here. You bring completeness and balance
to our family. I look forward to playing with you,
learning who you are, and enjoying our time together.

– Jeffrey Richter

To my wife Florence, au moins cette fois c'est écrit: je t'aime Flo.

To my parents who cannot believe that learning English
with Dungeons & Dragons rules could have been so efficient.

– Christophe Nasarre

Introduction

Microsoft Windows is a complex operating system. It offers so many features and does so much that it's impossible for any one person to fully understand the entire system. This complexity also makes it difficult for someone to decide where to start concentrating the learning effort. Well, I always like to start at the lowest level by gaining a solid understanding of the system's basic building blocks. Once you understand the basics, it's easy to incrementally add any higher-level aspects of the system to your knowledge. So this book focuses on Windows' basic building blocks and the fundamental concepts that you must know when architecting and implementing software targeting the Windows operating system. In short, this book teaches the reader about various Windows features and how to access them via the C and C++ programming languages.

Although this book does not cover some Windows concepts—such as the Component Object Model (COM)—COM is built on top of basic building blocks such as processes, threads, memory management, DLLs, thread local storage, Unicode, and so on. If you know these basic building blocks, understanding COM is just a matter of understanding how the building blocks are used. I have great sympathy for people who attempt to jump ahead in learning COM's architecture. They have a long road ahead and are bound to have gaping holes in their knowledge, which is bound to negatively affect their code and their software development schedules.

The Microsoft .NET Framework's common language runtime (CLR) is another technology not specifically addressed in this book. (However, it is addressed in my other book: *CLR via C#*, Jeffrey Richter, Microsoft Press, 2006). However, the CLR is implemented as a COM object in a dynamic-link library (DLL) that loads in a process and uses threads to execute code that manipulates Unicode strings that are managed in memory. So again, the basic building blocks presented in this book will help developers writing managed code. In addition, by way of the CLR's Platform Invocation (P/Invoke) technology, you can call into the various Windows' APIs presented throughout this book.

So that's what this book is all about: the basic Windows building blocks that every Windows developer (at least in my opinion) should be intimately aware of. As each block is discussed, I also describe how the system uses these blocks and how your own applications can best take advantage of these blocks. In many chapters, I show you how to create building blocks of your own. These building blocks, typically implemented as generic functions or C++ classes, group a set of Windows building blocks together to create a whole that is much greater than the sum of its parts.

64-Bit Windows

Microsoft has been shipping 32-bit versions of Windows that support the *x86* CPU architecture for many years. Today, Microsoft also offers 64-bit versions of Windows that support the *x64* and IA-64 CPU architectures. Machines based on these 64-bit CPU architectures are fast gaining acceptance. In fact, in the very near future, it is expected that all desktop and server machines will contain 64-bit CPUs. Because of this, Microsoft has stated that Windows Server 2008 will be the last 32-bit version of Windows ever! For developers, now is the time to focus on making sure your applications run correctly on 64-bit Windows. To this end, this book includes solid coverage of what you need to know to have your applications run on 64-bit Windows (as well as 32-bit Windows).

The biggest advantage your application gets from a 64-bit address space is the ability to easily manipulate large amounts of data, because your process is no longer constrained to a 2-GB usable address space. Even if your application doesn't need all this address space, Windows itself takes advantage of the significantly larger address space (about 8 terabytes), allowing it to run faster.

Here is a quick look at what you need to know about 64-bit Windows:

- The 64-bit Windows kernel is a port of the 32-bit Windows kernel. This means that all the details and intricacies that you've learned about 32-bit Windows still apply in the 64-bit world. In fact, Microsoft has modified the 32-bit Windows source code so that it can be compiled to produce a 32-bit or a 64-bit system. They have just one source-code base, so new features and bug fixes are simultaneously applied to both systems.
- Because the kernels use the same code and underlying concepts, the Windows API is identical on both platforms. This means that you do not have to redesign or reimplement your application to work on 64-bit Windows. You can simply make slight modifications to your source code and then rebuild.
- For backward compatibility, 64-bit Windows can execute 32-bit applications. However, your application's performance will improve if the application is built as a true 64-bit application.
- Because it is so easy to port 32-bit code, there are already device drivers, tools, and applications available for 64-bit Windows. Unfortunately, Visual Studio is a native 32-bit application and Microsoft seems to be in no hurry to port it to be a native 64-bit application. However, the good news is that 32-bit Visual Studio does run quite well on 64-bit Windows; it just has a limited address space for its own data structures. And Visual Studio does allow you to debug a 64-bit application.
- There is little new for you to learn. You'll be happy to know that most data types remain 32 bits wide. These include **`int`**s, **DWORD**s, **LONG**s, **BOOL**s, and so on. In fact, you mostly just need to worry about pointers and handles, since they are now 64-bit values.

Because Microsoft offers so much information on how to modify your existing source code to be 64-bit ready, I will not go into those details in this book. However, I thought about 64-bit Windows as I wrote each chapter. Where appropriate, I have included information specific to 64-bit Windows. In addition, I have compiled and tested all the sample applications in this book for 64-bit Windows. So, if you follow the sample applications in this book and do as I've done, you should have no trouble creating a single source-code base that you can easily compile for 32-bit or 64-bit Windows.

What's New in the Fifth Edition

In the past, this book has been titled *Advanced Windows NT*, *Advanced Windows*, and *Programming Applications for Microsoft Windows*. In keeping with tradition, this edition of the book has gotten a new title: *Windows via C/C++*. This new title indicates that the book is for C and C++ programmers wanting to understand Windows. This new edition covers more than 170 new functions and Windows features that have been introduced in Windows XP, Windows Vista, and Windows Server 2008.

Some chapters have been completely rewritten—such as Chapter 11, which explains how the new thread pool API should be used. Existing chapters have been greatly enhanced to present new features. For example, Chapter 4 now includes coverage of User Account Control and Chapter 8 now covers new synchronization mechanisms (Interlocked Singly-Linked List, Slim Reader-Writer Locks, and condition variables).

I also give much more coverage of how the C/C++ run-time library interacts with the operating system—particularly on enhancing security as well as exception handling. Last but not least, two new chapters have been added to explain how I/O operations work and to dig into the new Windows Error Reporting system that changes the way you must think about application error reporting and application recovery.

In addition to the new organization and greater depth, I added a ton of new content. Here is a partial list of enhancements made for this edition:

New Windows Vista and Windows Server 2008 features Of course, the book would not be a true revision unless it covered new features offered in Windows XP, Windows Vista, Windows Server 2008, and the C/C++ run-time library. This edition has new information on the secure string functions, the kernel object changes (such as namespaces and boundary descriptors), thread and process attribute lists, thread and I/O priority scheduling, synchronous I/O cancellation, vectored exception handling, and more.

64-bit Windows support The text addresses 64-bit Windows-specific issues; all sample applications have been built and tested on 64-bit Windows.

Use of C++ The sample applications use C++ and require fewer lines of code, and their logic is easier to follow and understand.

Reusable code Whenever possible, I created the source code to be generic and reusable. This should allow you to take individual functions or entire C++ classes and drop them into your own applications with little or no modification. The use of C++ made reusability much easier.

The ProcessInfo utility This particular sample application from the earlier editions has been enhanced to show the process owner, command line, and UAC-related details.

The LockCop utility This sample application is new. It shows which processes are running on the system. Once you select a process, this utility lists the threads of the process and, for each, on which kind of synchronization mechanism it is blocked—with deadlocks explicitly pointed out.

API hooking I present updated C++ classes that make it trivial to hook APIs in one or all modules of a process. My code even traps run-time calls to **`LoadLibrary`** and **`GetProcAddress`** so that your API hooks are enforced.

Structured exception handling improvements I have rewritten and reorganized much of the structured exception handling material. I have more information on unhandled exceptions, and I've added coverage on customizing Windows Error Reporting to fulfill your needs.

Code Samples and System Requirements

The sample applications presented throughout this book can be downloaded from the book's companion content Web page at

http://www.Wintellect.com/Books.aspx

To build the applications, you'll need Visual Studio 2005 (or later), the Microsoft Platform SDK for Windows Vista and Windows Server 2008 (which comes with some versions of Visual Studio). In addition, to run the applications, you'll need a computer (or virtual machine) with Windows Vista (or later) installed.

Support for This Book

Every effort has been made to ensure the accuracy of this book and the companion content. As corrections or changes are collected, they will be added to an Errata document downloadable at the following Web site:

http://www.Wintellect.com/Books.aspx

Questions and Comments

If you have comments, questions, or ideas regarding the book or the companion content, or questions that are not answered by visiting the site just mentioned, please send them to Microsoft Press via e-mail to

mspinput@microsoft.com

Or via postal mail to

Microsoft Press
Attn: *Windows via C/C++* Editor
One Microsoft Way
Redmond, WA 98052-6399

Please note that Microsoft software product support is not offered through the above addresses.

Acknowledgments

We could not have written this book without the help and technical assistance of several people. In particular, we'd like to thank the following people:

Jeffrey's Family

Jeffrey would like to thank Kristin (his wife) and Aidan (his son) for their never ending love and support.

Christophe's Family

Christophe would not have been able to write the fifth edition of this book without the love and support of Florence (his wife), the never ending curiosity of Celia (his daughter), and the purring sneak attacks of Canelle and Nougat (his cats). Now, I don't have any good excuse to not take care of you!

Technical Assistance

For writing a book like this one, personal research is not enough. We were owe great thanks to various Microsoft employees who helped us. Specifically, we'd like to thank Arun Kishan, who was able to either instantly answer weird and complicated questions or find the right person on the Windows team to provide more detailed explanations. We would also like to thank Kinshuman Kinshumann, Stephan Doll, Wedson Almeida Filho, Eric Li, Jean-Yves Poublan, Sandeep Ranade, Alan Chan, Ale Contenti, Kang Su Gatlin, Kai Hsu, Mehmet Iyigun, Ken Jung, Pavel Lebedynskiy, Paul Sliwowicz, and Landy Wang. In addition, there are those who listened to questions posted on Microsoft internal forums and shared their extensive knowledge, such as Raymond Chen, Sunggook Chue, Chris Corio, Larry Osterman, Richard Russell, Mark Russinovich, Mike Sheldon, Damien Watkins, and Junfeng Zhang. Last but not least, we would like to warmly thank John "Bugslayer" Robbins and Kenny Kerr who were kind enough to provide great feedback on chapters of this book.

Microsoft Press Editorial Team

We would like to thank Ben Ryan (acquisitions editor) for trusting a crazy French guy like Christophe, managers Lynn Finnel and Curtis Philips for their patience, Scott Seely for his constant search for technical accuracy, Roger LeBlanc for his talent in transforming Christophe's French-like English into something understandable, and Andrea Fox for her meticulous proofreading. In addition to the Redmond team, Joyanta Sen spent a lot of his personal time supporting us.

Mutual Admiration

Christophe sincerely thanks Jeffrey Richter for trusting him not to spoil the fifth edition of Jeff's book.

Jeffrey also thanks Christophe for his tireless efforts in researching, reorganizing, rewriting, and reworking the content in an attempt to reach Jeff's idea of perfection.

Table of Contents