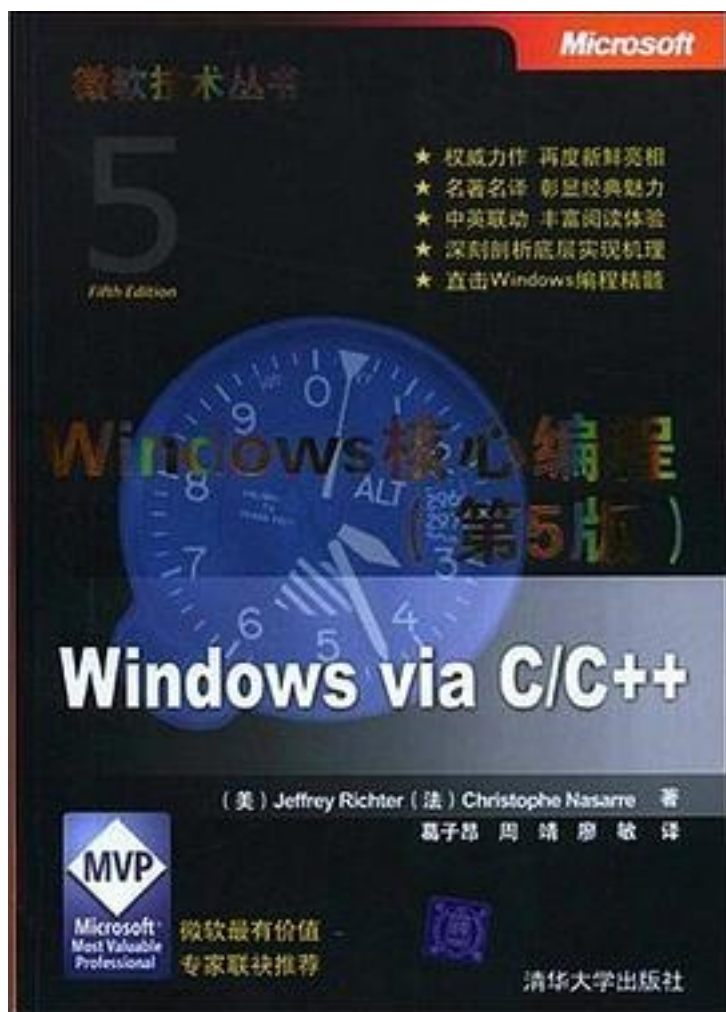


# Windows核心编程(第5版)



[Windows核心编程\(第5版\) 下载链接1](#)

著者:Jeffrey Richter

出版者:清华大学出版社

出版时间:2008-9

装帧:16开

isbn:9787302184003

这是一本经典的Windows核心编程指南，从第1版到第5版，引领着数十万程序员走入W

indows开发阵营，培养了大批精英。.

作为Windows开发人员的必备参考，本书是为打算理解Windows的C和C++程序员精心设计的。第5版全面覆盖Windows XP，Windows Vista和Windows Server 2008中的170个新增函数和Windows特性。书中还讲解了Windows系统如何使用这些特性，我们开发的应用程序又如何充分使用这些特性，如何自行创建新的特性。...

作者介绍:

Jeffrey

Richter是一位在全球享有盛誉的技术作家，尤其在Windows/.NET领域有着杰出的贡献。他的第一本Windows著作Windows 3: A Developer's Guide大获好评，从而声名远扬。之后，他又推出了经典著作《Windows高级编程指南》和《Windows核心编程》。如今这两本书早已成为Windows程序设计领域的颠峰之作，培育了几代软件开发设计人员。他的每一本新作问世，我们都有理由相信这是一本巨著，我们想要的一切尽在其中。Jeffery是Wintellect公司的创始人之一，也是MSDN杂志.NET专栏的特邀编辑。现在他正领导开发该公司的.NET程序设计课程，向大众推广.NET技术。因为他自1999年开始就参与了微软.NET框架开发组的咨询工作，与这些一线人员一起经历了.NET的孕育与诞生，所以他对.NET思想的领悟、对.NET的细节熟稔，是其他任何作家难以企及的。他是.NET著作领域中当之无愧的一面旗帜。

Christophe Nasarre是Business

Objects的软件架构师和开发部门领导，该公司致力于帮助其他企业更好地专注于其主营业务，通过商业智能方案来提升决策能力和业绩。他为Addison-Wesley，APress和Microsoft Press出版的许多图书担任过技术审校，此外还是MSDN Magazine的撰稿人。

目录: 第1部分 必备知识

第1章 错误处理

1.1 定义自己的错误代码

1.2 ErrorShow示例程序

第2章 字符和字符串处理

2.1 字符编码

2.2 ANSI字符和Unicode字符与字符串数据类型

2.3 Windows中的Unicode函数和ANSI函数

2.4 C运行库中的Unicode函数和ANSI函数

2.5 C运行库中的安全字符串函数

2.5.1 初识新的安全字符串函数

2.5.2 在处理字符串时如何获得更多控制

2.5.3 Windows字符串函数

2.6 为何要用Unicode

2.7 推荐的字符和字符串处理方式

2.8 Unicode与ANSI字符串转换

2.8.1 导出ANSI和Unicode DLL函数

2.8.2 判断文本是ANSI还是Unicode

第3章 内核对象

3.1 何为内核对象

3.1.1 使用计数

3.1.2 内核对象的安全性

3.2 进程内核对象句柄表

3.2.1 创建一个内核对象

3.2.2 关闭内核对象

### 3.3 跨进程边界共享内核对象

#### 3.3.1 使用对象句柄继承

#### 3.3.2 改变句柄的标志

#### 3.3.3 为对象命名

#### 3.3.4 终端服务命名空间

#### 3.3.5 专有命名空间

#### 3.3.5 复制对象句柄

### 第 II 部分 工作机制

## 第4章 进程

### 4.1 编写第一个Windows应用程序

#### 4.1.1 进程实例句柄

#### 4.1.2 进程前一个实例的句柄

#### 4.1.3 进程的命令行

#### 4.1.4 进程的环境变量

#### 4.1.5 进程的关联性

#### 4.1.6 进程的错误模式

#### 4.1.7 进程当前所在的驱动器和目录

#### 4.1.8 进程的当前目录

#### 4.1.9 系统版本

### 4.2 CreateProcess函数

#### 4.2.1 pszApplicationName和pszCommandLine参数

#### 4.2.2 psaProcess, psaThread和bInheritHandles参数

#### 4.2.3 fdwCreate参数

#### 4.2.4 pvEnvironment参数

#### 4.2.5 pszCurDir参数

#### 4.2.6 psiStartInfo参数

#### 4.2.7 ppiProcInfo参数

### 4.3 终止进程

#### 4.3.1 主线程的入口点函数返回

#### 4.3.2 ExitProcess函数

#### 4.3.3 TerminateProcess函数

#### 4.3.4 当进程中的所有线程终止时

#### 4.3.5 当进程终止运行时

### 4.4 子进程

### 4.5 管理员以标准用户权限运行时

#### 4.5.1 自动提升进程的权限

#### 4.5.2 手动提升进程的权限

#### 4.5.3 何为当前权限上下文

#### 4.5.4 枚举系统中正在运行的进程

#### 4.5.5 Process Information示例程序

## 第5章 作业

### 5.1 对作业中的进程施加限制

### 5.2 将进程放入作业中

### 5.3 终止作业中的所有线程查询作业统计信息

### 5.4 作业通知

### 5.6 Job Lab示例程序

## 第6章 线程基础

### 6.1 何时创建线程

### 6.2 何时不应该创建线程

### 6.3 编写第一个线程函数

### 6.4 CreateThread函数

#### 6.4.1 psa参数

#### 6.4.2 cbStackSize参数

#### 6.4.3 pfnStartAddr和pvParam参数

- 6.4.4 dwCreateFlags
- 6.4.5 pdwThreadId
- 6.5 终止运行线程
  - 6.5.1 线程函数返回
  - 6.5.2 ExitThread函数
  - 6.5.3 TerminateThread函数
  - 6.5.4 进程终止运行时
  - 6.5.5 线程终止运行时
- 6.6 线程内幕
- 6.7 C/C++运行库注意事项
  - 6.7.1 用\_beginthreadex而不要用CreateThread创建线程
  - 6.7.2 绝对不应该调用的C/C++运行库函数
- 6.8 了解自己的身份
  - 6.8.1 将伪句柄转换为真正的句柄
- 第7章 线程调度、优先级和关联性
  - 7.1 线程的挂起和恢复
  - 7.2 进程的挂起和恢复
  - 7.3 睡眠
  - 7.4 切换到另一个线程
  - 7.5 在超线程CPU上切换到另一个线程
  - 7.6 线程的执行时间
  - 7.7 在实际上下文中谈CONTEXT结构
  - 7.8 线程优先级
  - 7.9 从抽象角度看优先级
  - 7.10 优先级编程
    - 7.10.1 动态提升线程优先级
    - 7.10.2 为前台进程微调调度程序
    - 7.10.3 调度I/O请求优先级
    - 7.10.4 Scheduling Lab 示例程序
  - 7.11 关联性
- 第8章 用户模式下的线程同步
  - 8.1 原子访问：Interlocked系列函数
  - 8.2 高速缓存行
  - 8.3 高级线程同步需要避免使用的一种方法
  - 8.4 关键段
    - 8.4.1 关键段：细节
    - 8.4.2 关键段和旋转锁
    - 8.4.3 关键段和错误处理
  - 8.5 Slim读/写锁
  - 8.6 条件变量
    - 8.6.1 Queue示例程序
    - 8.6.2 在停止线程时的死锁问题
    - 8.6.3 一些有用的窍门和技巧
- 第9章 用内核对象进行线程同步
  - 9.1 等待函数
  - 9.2 等待成功所引起的副作用
  - 9.3 事件内核对象
  - 9.4 可等待的计时器内核对象
    - 9.4.1 让可等待的计时器添加APC调用
    - 9.4.2 计时器的剩余问题
  - 9.5 信号量内核对象
  - 9.6 互斥量内核对象
    - 9.6.1 遗弃问题
    - 9.6.2 互斥量与关键段的比较

- 9.6.3 Queue示例程序
- 9.7 线程同步对象速查表
- 9.8 其他的线程同步函数
  - 9.8.1 异步设备I/O
  - 9.8.2 WaitForInputIdle函数
  - 9.8.3 MsgWaitForMultipleObjects (Ex) 函数
  - 9.8.4 WaitForDebugEvent函数
  - 9.8.5 SignalObjectAndWait函数
  - 9.8.6 使用等待链遍历API来检测死锁
- 第10章 同步设备I/O与异步设备I/O
  - 10.1 打开和关闭设备细看CreateFile函数
  - 10.2 使用文件设备
    - 10.2.1 取得文件的大小
    - 10.2.2 设置文件指针的位置
    - 10.2.3 设置文件尾
  - 10.3 执行同步设备I/O
    - 10.3.1 将数据刷新至设备
    - 10.3.2 同步I/O的取消
  - 10.4 异步设备I/O基础
    - 10.4.1 OVERLAPPED结构
    - 10.4.2 异步设备I/O的注意事项
    - 10.4.3 取消队列中的设备I/O请求
  - 10.5 接收I/O请求完成通知
    - 10.5.1 触发设备内核对象
    - 10.5.2 触发事件内核对象
    - 10.5.3 可提醒I/O
    - 10.5.4 I/O完成端口
    - 10.5.5 模拟已完成的I/O请求
- 第11章 Windows线程池
  - 11.1 情形1：以异步方式调用函数
    - 11.1.1 显式地控制工作项
    - 11.1.2 Batch示例程序
  - 11.2 情形2：每隔一段时间调用一个函数
  - 11.3 情形3：在内核对象触发时调用一个函数
  - 11.4 情形4：在异步I/O请求完成时调用一个函数
  - 11.5 回调函数的终止操作
    - 11.5.1 对线程池进行定制
    - 11.5.2 得体地销毁线程池：清理组
- 第12章 纤程
- 第III部分 内存管理
- 第13章 Windows内存体系结构
  - 13.1 进程的虚拟地址空间
  - 13.2 虚拟地址空间的分区
    - 13.2.1 空指针赋值分区
    - 13.2.2 用户模式分区
  - 13.3 地址空间中的区域
  - 13.4 给区域调拨物理存储器
  - 13.5 物理存储器和页交换文件
  - 13.6 页面保护属性
    - 13.6.1 写时复制
    - 13.6.2 一些特殊的访问保护属性标志
  - 13.7 实例分析
  - 13.8 数据对齐的重要性
- 第14章 探索虚拟内存

- 14.1 系统信息
- 14.2 虚拟内存状态
- 14.3 NUMA机器中的内存管理
- 14.4 确定地址空间的状态
  - 14.4.1 VMQuery函数
  - 14.4.2 示例程序：虚拟内存映射
- 第15章 在应用程序中使用虚拟内存
  - 15.1 预订地址空间区域
  - 15.2 给区域调拨物理存储器
  - 15.3 同时预订和调拨物理存储器
  - 15.4 何时调拨物理存储器
  - 15.5 撤销调拨物理存储器及释放区
    - 15.5.1 何时撤销调拨物理存储器
    - 15.5.2 虚拟内存分配示例程序
  - 15.6 改变保护属性
  - 15.7 重置物理存储器的内容
  - 15.8 地址窗口扩展
- 第16章 线程栈
  - 16.1 C/C++运行库的栈检查函数
  - 16.2 Summation示例程序
- 第17章 内存映射文件
  - 17.1 映射到内存的可执行文件和DLL
    - 17.1.1 同一个可执行文件或DLL的多个实例不会共享静态数据
    - 17.1.2 在同一个可执行文件或DLL的多个实例间共享静态数据
    - 17.1.3 Application Instances示例程序
  - 17.2 映射到内存的数据文件
    - 17.2.1 方法1：一个文件，一块缓存
    - 17.2.2 方法2：两个文件，一块缓存
    - 17.2.3 方法3：一个文件，两块缓存
    - 17.2.4 方法4：一个文件，零个缓存
  - 17.3 使用内存映射文件
    - 17.3.1 第1步：创建或打开文件内核对象
    - 17.3.2 第2步：创建文件映射内核对象
    - 17.3.3 第3步：将文件的数据映射到进程的地址空间
    - 17.3.4 第4步：从进程的地址空间撤销对文件数据的映射
    - 17.3.5 第5步和第6步：关闭文件映射对象和文件对象
  - 17.6 File Reverse示例程序
  - 17.7 用内存映射文件来处理大文件
  - 17.8 内存映射文件和一致性
  - 17.9 给内存映射文件指定基地址
  - 17.10 内存映射文件的实现细节
- 第18章 堆
  - 18.1 进程的默认堆
  - 18.2 为什么要创建额外的堆
    - 18.2.1 对组件进行保护
    - 18.2.2 更有效的内存管理
    - 18.2.3 使内存访问局部化
    - 18.2.4 避免线程同步的开销
    - 18.2.5 快速释放
  - 18.3 如何创建额外的堆
    - 18.3.1 从堆中分配内存块
    - 18.3.2 调整内存块的大小
    - 18.3.3 获得内存块的大小
    - 18.3.4 释放内存块

- 18.3.5 销毁堆
- 18.3.6 在C++中使用堆
- 18.4 其他堆函数
- 第IV部分 动态链接库
- 第19章 DLL基础
  - 19.1 DLL和进程的地址空间
  - 19.2 纵观全局
    - 19.2.1 构建DLL模块
    - 19.2.2 构建可执行模块
    - 19.2.3 运行可执行模块
- 第20章 DLL高级技术
  - 20.1 DLL模块的显式载入和符号链接
    - 20.1.1 显式地载入DLL模块
    - 20.1.2 显式地卸载DLL模块
    - 20.1.3 显式地链接到导出符号
  - 20.2 DLL的入口点函数
    - 20.2.1 DLL\_PROCESS\_ATTACH通知
    - 20.2.2 DLL\_PROCESS\_DETACH通知
    - 20.2.3 DLL\_THREAD\_ATTACH通知
    - 20.2.4 DLL\_THREAD\_DETACH通知
    - 20.2.5 DllMain的序列化调用
    - 20.2.6 DllMain和C/C++运行库
  - 20.3 延迟载入DLL
  - 20.4 函数转发器
  - 20.5 已知的DLL
  - 20.6 DLL重定向
  - 20.7 模块的基地址重定位
  - 20.8 模块的绑定
- 第21章 线程局部存储区
  - 21.1 动态TLS
  - 21.2 静态TLS0
- 第22章 DLL注入和API拦截
  - 22.1 DLL注入的一个例子
  - 22.2 使用注册表来注入DLL
  - 22.3 使用Windows挂钩来注入DLL
  - 22.4 使用远程线程来注入DLL
    - 22.4.1 Inject Library示例程序
    - 22.4.2 Image Walk DLL
  - 22.5 使用木马DLL来注入DLL
  - 22.6 把DLL作为调试器来注入
  - 22.7 使用CreateProcess来注入代码
  - 22.8 API拦截的一个例子9
    - 22.8.1 通过覆盖代码来拦截API0
    - 22.8.2 通过修改模块的导入段来拦截API
    - 22.8.3 Last MessageBox Info示例程序
- 第V部分 结构化异常处理
- 第23章 终止处理程序
- 第24章 异常处理程序与软件异常
  - 24.1 通过实例理解异常过滤程序和异常处理程序
    - 24.1.1 Funcmeister1函数
    - 24.1.2 Funcmeister2函数
  - 24.2 EXCEPTION\_EXECUTE\_HANDLER1
    - 24.2.1 一些有用的例子
    - 24.2.2 全局展开

- 24.2.3 停止全局展开
- 24.3 EXCEPTION\_CONTINUE\_EXECUTION
- 24.4 EXCEPTION\_CONTINUE\_SEARCH0
- 24.5 GetExceptionCode2
- 24.6 GetExceptionInformation6
- 24.7 软件异常
- 第25章 未处理异常、向量化异常处理与C++异常
- 25.1 UnhandledExceptionFilter函数详解
- 25.2 即时调试
- 25.3 电子表格示例程序
- 25.4 向量化异常和继续处理程序
- 25.5 C++异常与结构化异常的比较
- 25.6 异常与调试器
- 第26章 错误报告与应用程序恢复
- 26.1 Windows错误报告控制台
- 26.2 可编程的Windows错误报告
- 26.3 对进程中所有的问题报告进行定制
- 26.4 问题报告的创建与定制
- 26.4.1 创建一个自定义的问题报告
- 26.4.2 设置报告参数：WerReportSetParameter
- 26.4.3 将小型转储文件放入报告：WerReportAddDump8
- 26.4.4 将任意文件放入报告：WerReportAddFile9
- 26.4.5 修改对话框文本：WerReportSetUIOption0
- 26.4.6 提交错误报告：WerReportSubmit0
- 26.4.7 关闭问题报告：WerReportCloseHandle
- 26.4.8 Customized WER示例程序
- 26.5 应用程序的自动重启与恢复
- 26.5.1 应用程序的自动重启
- 26.5.2 对应用程序恢复的支持
- 第VI部分
- 附录A 构建环境
- 附录B 消息处理宏、子控件宏和API宏
- 索引
- • • • • ([收起](#))

[Windows核心编程\(第5版\)\\_下载链接1\\_](#)

## 标签

Windows编程

Windows

编程

操作系统



计算机

C++

程序设计

Programming

## 评论

怎么评价这本书呢?书买了好几年,我承认很多内容就一直没有看懂过。只是每当遇到难以解决的问题时,我就会翻开这本书--问题于是迎刃而解! 对于Windows编程中的很多概念,也是从这本书开始澄清的,比如线程和进程的实质,Windows消息循环的原理等等

-----  
Win的APUE对应物。看了同步机制的部分,先放一边了

-----  
可惜Jeffrey Richter不是linux程序员..

-----  
扉页的购买日期是2009年..

-----  
还是读新版的好,因为有系统新特性的讲解,也有以前版本知识不足的填充。这书真是经典。而且也实用,对于了解Windows系统也有帮助。

-----  
虽然过时了,没读过的就没必要看了,对工作有很大帮助的几本书之一

-----

要有点OS的基础知识才能看，最好懂点汇编，要有一定的基础。非常好。

---

累觉仍爱。

---

在前台开发混了一年多，终于对windows编程有了点理解，回头再看这本书，已经觉得不那么难懂了。书中的知识点都狠精彩，不过还是感慨一下：windows编程好难。

---

和APUE对照着看效果好，觉得APUE写得更专业，更结合实际，只得还不够深入（只怪Windows是商业软件，有着太多的商业秘密）。

---

东西讲得很细，有一些底层的知识目前还看不懂，我也不打算现在就深究。我终究还是对win下的dev缺乏兴趣阿……个人感觉Windows下的接口设计没有\*nix简洁好用阿

---

受益匪浅

---

太深。

---

大体扫了扫

---

去买本崭新的回来收藏!

---

学习windows编程的必经之路

-----  
这个版本追加了很多针对vista的篇幅，还是那么的经典~~

-----  
范例均采用C++和winAPI编写，如要深入理解，更需了解winAPI

-----  
翻译得很烂...

-----  
没什么新内容，相当失望，还不如去看windows system programming，名声果然是吹出来的。。

-----  
[Windows核心编程\(第5版\)\\_下载链接1](#)

## 书评

怎么说呢，这是我下狠心的第一本打算看完的英文技术书籍  
以前只是小说和一些资料。我得到的最大的好处恰恰是我看书的节奏，以前看书速度有些快，是不自觉的。但是这次我很自觉的就慢下来了，毕竟看过一次中文版本。发现自己吃的还是不透，已经工作了，只能一点一点的看了。个人...

-----  
第一遍读真的是很晕，代码能看懂但是不清楚干什么  
很多概念直接就是用了，作者认为读者已经了解很多 所以需要之前把《windows internals》等多读读  
我个人建议读读毛教授最近的《windwos情景分析》以及NTFSI的第一部分  
这本书真的不错，特别是了解win原理以后，这本书的威力就...

-----  
怎么评价这本书呢?书买了好几年,我承认很多内容就一直没有看懂过。只是每当遇到难以解决的问题时,我就会翻开这本书--问题于是迎刃而解! 对于Windows编程中的很多概念，也是从这本书开始澄清的，比如线程和进程的实质，Windows消息循环的原理等等

。这不是一本Windows编程的入...

---

我总觉的这本书是我看过最好的书，对我的触动很大。我看的时候刚好大学讲完操作系统。OS课程理论性太强了，一大堆东西都是停留在概念上，我总觉得，从OS理论到OS实现，总是少了一层知识，很多知识都连接不起来。比如，上个OS课的人基本都可以说出内核态是什么概念，但是...

---

第一遍读过去肯定要忘，东西太多了。至少要读两遍。 btw，第4版有关于window messaging的一章，不知道为啥第5版里删掉了。

---

刚刚看Part1。之前读过上一版的书，帮助很大。这本书里加了不少新东西，内容也非常有深度。

---

大二买了这本书，一直到现在，已经有四年了吧！ 内容很经典。书上介绍的内容是一些WINDOWS编程基础的内容。如开始介绍的内核对象，内存分配管理，WINDOWS消息机制等问题。我拿WINDOWS消息机制来说，虽然介绍了WINDOWS消息的一些内部机制，但是绝对没有彻底讲消息到底是怎...

---

在WINDOWS编程时，遇到一些操作系统限制的问题时，你总能在这本书上找到答案，例如究竟一个进程能分配多少个线程，一个进程能分配多少可用内存。由于WINDOWS代码是不可见的，而这本书已经尽最大努力让你在API层上窥探一些系统底层的一些知识 对于一名WINDOWS程序员来说，平台知...

---

看到线程调度那章，我坚信了这是一本靠谱的书，它符合我对线程调度的认知。看到虚拟内存和内存映射文件那章，我坚信了这是一本好书，它解答了我对虚拟内存的一些疑惑。当我看到剖析dll那章，我被惊艳了，我坚信这是一本神书，很有深度，还很有趣。这是一本神书，不管你信不...

这本书并不适合初学者，初学者最好先把windows下各种东西先用一遍，碰到很多问题之后带着问题来阅读这本书才会发现这本书的优秀。此书以精炼的文笔说明了windows下程序开发与操作系统打交道时的方方面面，如果你仅仅是算法工程师，那么你并不需要读这本书，而如果你是windows...

由于时间的关系，这本书只看了80%，但已让我受益无穷。这本书给我什么感觉呢，它将每一个知识点的来龙去脉讲的清清楚楚，读后让人有一通百通的爽快。事实上，第一遍是看不懂的，这里的“看不懂”并非难以理解，相反，这本书的让你一读就知道在讲什么，但因其涉及到...

[illegible]

如果有英文纸书，肯定不会读中文版。书肯定是好书，不过没有达到APUE的高度。对于想学习windows编程的人还是有些帮助的，对于想了解windows系统的人也不错。就写这么多了，天不早了，该洗洗睡了。

如果你被多线程困扰着，这本书一定会让你爱不释手！  
看有人说这本书很难，估计是没有从头看起。只要一章一章挨着看，这本书其实很好懂的。

原著是好书，中文翻译出来成了鸡肋的计算机类书比比皆是。在这个毁书不倦的出版界，能如此精确的将原书作者的愿意精确的表达，实在让人惊讶！好书当赞。而对于英语仅过四级水准的读者来说，好的译本更应赞赏！

大二的时候就仰慕这本书了，直到现在才有时间好好地细读，感觉不错，很优秀，特别是学习windows的线程模型，win32多线程编程这本书已经是经典了，但是这本书关于线程方面觉得更加优秀

如果你去读原版书的话，会发现这些翻译的人完全糟蹋了经典的好书，前后章节的专业

术语都翻译地不一样。 错误更是多的一塌糊涂

-----  
刚看完《Windows系统编程（原书第三版）》里面有推荐到这本书，看了《Windows系统编程（原书第三版）》里面很多问题不够深入，再看这本刚好。 不过刚拿到书～～

-----  
[Windows核心编程\(第5版\)\\_下载链接1\\_](#)