

Windows 高级编程指南

(第三版)

[美] Jeffrey Richter 著
王书洪 刘光明 译
郑海明 审校

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书以详尽的资料和大量的范例深入讨论了有关 32 位 Windows 编程的高级问题, 包括进程和线程的管理、Win32 的内存管理、消息处理、动态链接库、文件系统和设备输入输出、结构化异常处理、Unicode 等, 并重点讨论了 Windows 95 和 Windows NT 在实现上的不同之处, 其中包含了大量的编程技巧。

本书可供高等院校计算机专业的师生和广大的计算机编程人员使用。

Windows 高级编程指南(第三版)

Advanced Windows, 3rd Edition

Jeffrey Richter

Copyright 1997 by Jeffrey Richter.

Original English language Edition Copyright 1997 by Jeffrey Richter.

Published by arrangement with the original publisher, Microsoft Press,

a division of Microsoft Corporation, Redmond, Washington, U. S. A.

本书中文版由 Microsoft Press 授权清华大学出版社出版。

北京市版权局著作权合同登记号 图字 01-97-1659 号

版权所有, 翻印必究。

本书封面贴有 Microsoft Press 激光防伪标签, 无标签者不得销售。

图书在版编目(CIP)数据

Windows 高级编程指南 / (美) 里克特(Richter, J.) 著; 王书洪, 刘光明译 .- 3 版 .—北京: 清华大学出版社, 1999

ISBN 7-302-03328-5

I . W… II . ① 里… ② 王… ③ 刘… III . 窗口软件, Windows-程序设计 IV . TP316

中国版本图书馆 CIP 数据核字(1999)第 03107 号

出 版 者: 清华大学出版社 (北京清华大学校内, 邮编 100084)

http: / / www.tup.tsinghua.edu.cn

责任编辑: 张善余

印 刷 者: 清华大学印刷厂

发 行 者: 新华书店总店北京发行所

开 本: 78× 960 1 /16 印张: 51.5 字数: 1120 千字

版 次: 1999 年 6 月 第 1 版 1999 年 6 月 第 1 次印刷

书 号: ISBN 7-302-03328-5 /TP · 1792

印 数: 0001~5000

定 价：99.00 元

译者序

随着 Microsoft Windows 95 和 Windows NT 4.0 的相继推出, 32 位 Windows 已经成为个人计算机操作系统事实上的标准。使用 Windows 95 作为客户机系统, Windows NT 作为服务器系统已经成为标准的配置。而作为技术的前沿, 它们都有很多知识值得我们认真学习。本书就是能够满足大家需求的一本不可多得的好教材。

不论你是一位刚接触 32 位 Windows 的新手, 还是一位从其他操作系统转向 Windows 的开发人员, 或者已经有了相当多的编程经验, 本书都会对你大有裨益。本书的作者 Jeffery Richter 是一位富有经验的 Win32 编程和培训人员, 他培训过的客户来自 AT&T、Digital、HP、IBM、Intel、Microsoft 和 Sybase 等知名的大公司。本书是他的《Advanced Windows》的第 3 版。

本书以详尽的资料和大量的范例应用程序, 深入讨论了有关 32 位 Windows 编程的高级问题, 包括进程和线程的管理、Win32 的内存管理、消息处理、动态链接库、文件系统和设备 I/O、结构化异常处理、Unicode 等, 并且重点讨论了 Windows 95 和 Windows NT 在实现上的不同。作者还以多年的编程经验在书中提供了大量的编程技巧。相信读者一定会从中受益匪浅。

全书共分 18 章, 并有 3 个附录。

由于译者水平所限, 书中难免有错误和不妥之处, 欢迎批评指正。

目 录

第 0 章	绪论	1
0.1	关于应用程序范例	1
0.1.1	用 C 语言编写程序	2
0.1.2	Message Cracker Macros(消息拆析宏)	2
0.1.3	有关 16 位 Windows 编程的知识	2
0.1.4	在 Windows 95 下运行应用程序范例	2
0.1.5	不相关代码	2
0.1.6	范例应用程序的独立性	3
0.1.7	STRICT 编译	3
0.1.8	错误检查	3
0.1.9	没有错误	3
0.1.10	测试平台和环境	4
0.1.11	Unicode	4
0.2	安装范例程序	5
0.2.1	配套的 CD-ROM 盘	5
第 1 章	WIN 32 API 和支持它的平台	7
1.1	梦想一下: Win 32 API	7
1.2	Win 32s	8
1.3	Windows NT	8
1.4	Windows 95	10
1.5	Windows CE	10
1.6	现实: Win32 API	11
第 2 章	内核对象	13
2.1	什么是内核对象?	13
2.1.1	使用计数	14
2.1.2	安全	14
2.2	进程的内核对象句柄表	16
2.2.1	创建内核对象	17

2.2.2	关闭内核对象	18
2.3	在进程间共享内核对象	18
2.3.1	对象句柄继承	19
2.3.2	改变句柄标志	21
2.3.3	命名对象	23
2.3.4	复制对象句柄	25
第 3 章	进程	30
3.1	编写第一个 Win32 应用程序	31
3.1.1	进程的实例句柄	33
3.1.2	进程的前一个实例的句柄	35
3.1.3	进程的命令行	36
3.1.4	进程的环境变量	37
3.1.5	进程的错误模式	40
3.1.6	进程的当前驱动器 and 目录	40
3.1.7	系统版本	42
3.2	CreateProcess 函数	43
3.2.1	lpzApplicationName and lpzCommandline	44
3.2.2	lpzProcess, lpzThread and fInheritHandles	45
3.2.3	fdwCreate	46
3.2.4	lpvEnvironment	48
3.2.5	lpzCurDir	49
3.2.6	lpzStartInfo	49
3.2.7	lpzProcInfo	53
3.3	终止进程	54
3.3.1	ExitProcess 函数	54
3.3.2	TerminateProcess 函数	54
3.3.3	进程中的所有线程结束了	55
3.3.4	进程终结时发生的事情	55
3.4	子进程	56
3.4.1	运行分离的子进程	57
第 4 章	线程	58
4.1	何时创建线程	58
4.2	何时不用创建线程	59

4.3	编写第一个线程函数	61
4.3.1	线程的栈	62
4.3.2	线程的 CONTEXT 结构	62
4.3.3	线程的执行时间	62
4.4	CreateThread 函数	64
4.4.1	lpSa	65
4.4.2	cbStack	65
4.4.3	lpStartAddr 和 lpvThreadParm	66
4.4.4	fdwCreate	66
4.4.5	lpIDThread	67
4.5	终止线程	68
4.5.1	ExitThread 函数	68
4.5.2	TerminateThread 函数	68
4.5.3	进程终结	69
4.5.4	线程终结时的情况	69
4.6	识别自己的身份	69
4.7	系统如何调度线程	72
4.7.1	使用 Win32 API 怎样赋优先级	74
4.7.2	改变进程的优先级类	76
4.7.3	设定线程的相对优先级	77
4.7.4	挂起和恢复线程	80
4.8	系统内部情况	80
4.9	进程、线程和 C 运行时库	84
4.9.1	应避免使用的 C 运行时函数	88
第 5 章	Win32 内存结构	89
5.1	虚地址空间	89
5.1.1	Windows 95 如何划分进程的地址空间	90
5.1.2	Windows NT 如何划分进程的地址空间	92
5.2	地址空间中的区域	94
5.3	在区域内提交物理存储	95
5.4	物理存储	96
5.4.1	不包含在页面文件中的物理存储	98
5.5	保护属性	99
5.5.1	写拷贝访问	99

5.5.2	特殊的访问保护属性标志	100
5.6	小结	100
5.6.1	区域的内部	103
5.7	Windows 95 的地址空间的不同	108
第 6 章	虚拟内存	115
6.1	系统信息	115
6.1.1	系统信息范例应用程序	116
6.2	虚拟内存状态	125
6.2.1	虚拟内存状态范例应用程序	126
6.3	确定地址空间的状态	132
6.3.1	VMQuery 函数	132
6.3.2	虚拟内存映射范例应用程序	142
第 7 章	在应用程序中使用虚拟内存	152
7.1	在地址空间中保留区域	152
7.2	在保留区域中提交存储	154
7.3	同时保留区域和提交存储	155
7.4	何时提交物理存储	155
7.5	释放物理存储和释放区域	157
7.5.1	何时释放物理存储	157
7.5.2	虚拟内存分配范例应用程序	158
7.6	修改保护属性	169
7.7	在 RAM 中锁定物理存储	170
7.8	重设物理存储的内容	172
7.9	线程的栈	175
7.9.1	Windows 95 下的线程的栈	177
7.9.2	C 运行时库的栈检查函数	180
第 8 章	内存映射文件	183
8.1	内存映射 EXE 和 DLL	183
8.1.1	不被 EXE 或 DLL 的多个实例共享的静态数据	184
8.2	内存映射数据文件	187
8.2.1	方法 1: 一个文件, 一个缓冲区	187
8.2.2	方法 2: 两个文件, 一个缓冲区	187

8.2.3	方法 3: 一个文件, 两个缓冲区	188
8.2.4	方法 4: 一个文件, 零个缓冲区	188
8.3	使用内存映射文件	188
8.3.1	第 1 步: 创建或打开文件内核对象	189
8.3.2	第 2 步: 创建文件映射内核对象	190
8.3.3	第 3 步: 将文件数据映射入进程地址空间	193
8.3.4	第 4 步: 从进程地址空间中解除文件数据映射	195
8.3.5	第 5 步和第 6 步: 关闭文件映射对象和文件对象	196
8.3.6	用内存映射文件处理大文件	197
8.4	内存映射文件和一致性	199
8.4.1	文件倒置范例应用程序	200
8.4.2	指定内存映射文件的基本地址	210
8.5	内存映射文件和 Win32 实现	211
8.5.1	使用内存映射文件在进程间共享数据	214
8.5.2	由页面文件支持的内存映射文件	217
8.5.3	内存映射文件共享范例应用程序	218
8.6	稀疏提交的内存映射文件	224
第 9 章	堆	226
9.1	什么是 Win32 堆?	226
9.1.1	进程的缺省堆	227
9.1.2	创建自己的 Win32 堆	227
9.1.3	创建 Win32 堆	229
9.1.4	释放 Win32 堆	232
9.1.5	在 C++ 时使用堆	233
9.1.6	其他一些堆函数	236
9.2	16 位 Windows 堆函数	238
第 10 章	线程同步	243
10.1	线程同步概述	243
10.1.1	最坏的事情	243
10.2	临界区	244
10.2.1	创建临界区	246
10.2.2	使用临界区	247
10.2.3	临界区范例应用程序	254

10.3	用内核对象同步线程·····	268
10.3.1	互斥量·····	272
10.3.2	互斥量范例应用程序·····	277
10.3.3	信号量·····	287
10.3.4	超市范例应用程序·····	289
10.3.5	事件·····	314
10.3.6	Bucket of Balls 范例应用程序 ·····	316
10.3.7	SWMRG 复合同步对象 ·····	320
10.3.8	Bucket 范例源代码 ·····	322
10.3.9	Document Statistics 范例应用程序 ·····	344
10.3.10	可等的计时器 ·····	354
10.4	线程挂起·····	360
10.4.1	Sleep ·····	360
10.4.2	异步文件 I / O ·····	360
10.4.3	WaitForInputIdle ·····	361
10.4.4	MsgWaitForMultipleObjects ·····	362
10.4.5	WaitForDebugEvent ·····	362
10.4.6	SignalObjectAndWait ·····	363
10.4.7	Interlocked 函数族 ·····	364
第 11 章	窗口消息和异步输入 ·····	367
11.1	多任务·····	367
11.1.1	抢先式调度·····	368
11.2	线程队列和消息处理·····	370
11.2.1	Win32 消息队列结构 ·····	370
11.2.2	向线程的消息队列投递消息·····	370
11.2.3	发送消息给窗口·····	373
11.2.4	唤醒线程·····	376
11.3	用消息发送数据·····	381
11.3.1	CopyData 范例应用程序 ·····	383
11.4	非序列化输入·····	390
11.4.1	输入如何被非序列化·····	391
11.5	局部输入状态·····	394
11.5.1	键盘输入和焦点·····	395
11.5.2	鼠标光标管理·····	398

11.5.3	局部输入状态实验范例应用程序	399
第 12 章	动态链接库	418
12.1	创建动态链接库	418
12.1.1	映射 DLL 到进程的地址空间	420
12.2	DLL 的进入 / 退出函数	426
12.2.1	DLL- PROCESS- ATTACH	427
12.2.2	DLL- PROCESS- DETACH	428
12.2.3	DLL- THREAD- ATTACH	430
12.2.4	DLL- THREAD- DETACH	430
12.2.5	系统如何序列化对 DllMain 的调用	433
12.2.6	DllMain 和 C 运行时库	436
12.3	从 DLL 中输出函数和变量	436
12.4	从 DLL 中引入函数和变量	439
12.4.1	DLL 的头文件	444
12.5	创建 DLL 供非 Visual C++ 工具使用	445
12.6	在 EXE 或 DLL 映射之间共享数据	446
12.6.1	EXE 或 DLL 中的节	447
12.6.2	ModUse 范例应用程序	450
12.6.3	MultInst 范例应用程序	458
第 13 章	线程局部存储	462
13.1	动态线程局部存储	463
13.1.1	使用动态线程局部存储	465
13.1.2	动态线程局部存储范例应用程序	467
13.2	静态线程局部存储	479
13.2.1	静态线程局部存储范例应用程序	480
第 14 章	文件系统	490
14.1	Win32 的文件名约定	491
14.2	系统和卷操作	492
14.2.1	得到卷的特定信息	494
14.2.2	磁盘信息查看范例应用程序	497
14.3	目录操作	507
14.3.1	得到当前目录	507

14.3.2	改变当前目录	508
14.3.3	得到系统目录	508
14.3.4	得到 Windows 目录	508
14.3.5	创建和删除目录	509
14.4	拷贝、删除、移动和重命名文件	509
14.4.1	拷贝文件	509
14.4.2	删除文件	511
14.4.3	移动文件	511
14.4.4	重命名文件	514
14.5	查找文件	514
14.5.1	目录漫游范例应用程序	519
14.6	文件系统变化通知	529
14.6.1	文件变化范例应用程序	533
14.7	操纵文件属性	549
14.7.1	可执行文件类型	549
14.7.2	文件属性	549
14.7.3	文件大小	551
14.7.4	文件时间戳	551
14.8	创建临时文件	554
第 15 章	设备 I / O	556
15.1	打开和关闭设备	556
15.1.1	CreateFile	558
15.2	使用文件设备	563
15.2.1	定位文件指针	564
15.2.2	设置文件尾	565
15.2.3	加锁和解锁文件的区域	566
15.3	进行同步 I / O	568
15.3.1	向设备强制刷新数据	569
15.4	进行异步 I / O	569
15.5	使设备内核对象有信号	570
15.6	使事件内核对象有信号	576
15.7	告警 I / O	577
15.7.1	告警 I / O 范例应用程序	580
15.8	I / O 完成端口	593

15.8.1	模拟完成 I / O 请求	601
15.8.2	I / O 完成端口范例应用程序	602
15.9	判断 I / O 请求是否完成	614
15.10	取消未完成的 I / O 请求	615
第 16 章	结构化异常处理	616
16.1	终止处理程序	617
16.1.1	理解终止处理程序	618
16.1.2	关于 finally 块的一些注解	629
16.1.3	SEH 终止范例应用程序	631
16.2	异常过滤程序和异常处理程序	641
16.2.1	理解异常过滤程序和异常处理程序	642
16.2.2	EXCEPTION- EXECUTE- HANDLER	643
16.2.3	EXCEPTION- CONTINUE- EXECUTION	644
16.2.4	EXCEPTION- CONTINUE- SEARCH	646
16.2.5	全局展开	648
16.2.6	停止全局展开	651
16.2.7	关于异常过滤程序的进一步讨论	652
16.2.8	GetExceptionInformation	657
16.2.9	SEH 异常范例应用程序	662
16.2.10	SEH Sum 范例应用程序	672
16.3	软件异常	680
16.3.1	SEH 软件异常范例应用程序	682
16.3.2	未处理的异常	694
16.3.3	没有附属调试器的未处理的异常	694
16.3.4	不显示异常消息框	697
16.3.5	自己调用 UnhandledExceptionFilter	698
16.3.6	Windows NT 的未处理的内核模式异常	699
第 17 章	Unicode	700
17.1	字符集	700
17.1.1	单字节和双字节字符集	700
17.1.2	Unicode: 宽字节字符集	701
17.2	为什么要使用 Unicode	702
17.3	如何编写 Unicode 源代码	702

17.3.1	Windows NT 和 Unicode	703
17.3.2	Windows 95 和 Unicode	703
17.3.3	C 运行时库对 Unicode 的支持	703
17.3.4	Win32 定义的 Unioode 数据类型	708
17.3.5	Win32 中的 Unicode 和 ANSI 宏数	709
17.4	使你的程序能识别 ANSI 和 Unicode	710
17.4.1	Win32 中的字符串函数	711
17.4.2	资源	713
17.4.3	文本文件	713
17.4.4	在 Unicode 和 ANSI 之间转换字符串	714
17.4.5	Windows NT 的窗口类和过程	717
第 18 章	打开进程边界	720
18.1	为什么要打开进程边界: 一个例子	720
18.2	使用注册表来注入 DLL	722
18.3	使用 Windows 钩来注入 DLL	724
18.3.1	Desktop Item Position Savev 工具	725
18.4	使用远程线程注入 DLL	738
18.4.1	如何装入一个 DLL	738
18.5	影响其他进程的 Win32 函数	739
18.5.1	CreateRemoteThread	740
18.5.2	GetThreadContext 和 SetThreadContext	741
18.5.3	VirtualQueryEx 和 VirtualProtectEx	745
18.5.4	VirtualAllocEx 和 VirtualFreeEx	745
18.5.5	ReadProcessMemory 和 WriteProcessMemory	746
18.6	创建一个函数向任意进程的地址空间注入 DLL	746
18.6.1	版本 0: 为什么显然的方法不能工作	746
18.6.2	版本 1: 手编的机器语言	747
18.6.3	版本 2: AllocProcessMemory 和 CreateRemoteThread	750
18.6.4	ProcMem 工具函数	753
18.6.5	InjectLib 函数	757
18.7	测试 InjectLib 函数	766
18.7.1	注入库范例应用程序	766
18.7.2	映象遍历动态链接库	771
18.8	小结	774

附录 A	FIBERS	776
A.1	例程：计算器应用程序	779
附录 B	MESSAGE CRACKERC	789
B.1	Message Crackers	790
B.2	Child Control 宏	793
B.3	API 宏	794
附录 C	编程环境	795
C.1	CMNHDR.H 头文件	795
C.1.1	4 级警告	795
C.1.2	Windows 版本	796
C.1.3	STRICT 宏	796
C.1.4	CPU Protability 宏	796
C.1.5	标准编码(Unicode)	796
C.1.6	chDIMOF 宏	797
C.1.7	chBEGINTHREADEX 宏	797
C.1.8	chASSERT 宏和 chVERIFY 宏	798
C.1.9	chHANDLE- DLGMSG 宏	798
C.1.10	chMB 宏	798
C.1.11	chINITSTRUCT 宏	798
C.1.12	chSETDLGICONS 宏	799
C.1.13	chWARNIFUNICODEUNDERWIN95 宏	799
C.1.14	Pragma Message Helper 宏	799
C.1.15	链接指令	800
C.2	在源文件中不能对项目进行设置	805

绪 论

我很高兴能够编写这本书。比起其他事情来,我更喜欢处于科技的前沿并能够学到新东西。Microsoft Windows 95 和 Microsoft Windows NT 就显然位于科技的前沿,而且这里有很多新知识可学。不过,不要被新知识之多吓住。

读者如果已经是一位 16 位 Windows 程序设计人员,那么只要学到一点儿移植已有代码的技术,就会发现自己可以开始编写 Win32 应用程序了。但是,这些移植的程序不能利用 Windows 95 和 Windows NT 这些 Win32 环境提供的强大的、令人兴奋的功能。

如果读者正在从其他的操作系统(如 UNIX)转移到 Windows NT,那么这本书将解释如何最好地利用 NT 这种丰富的操作系统所提供的功能。让已有的应用程序运行在 Windows NT 上是一回事;创建专为 Windows NT 而设计的应用程序是另一回事。Windows NT 能提供很多功能。读者如果只是简单地让已有的应用程序运行在 Windows NT 上,那就不会处在强有力的竞争位置上。但是,如果修改一些应用程序来真正利用 Windows NT 的功能,就可能会发现在使应用程序运行得更快的同时,可以删去原有程序中的很大一部分代码,而且减少对内存的需求。

在开始利用 Win32 工作后,就能在应用程序中加入它的越来越多的功能。Win32 的很多功能使编程变得极为简单。而且,正如我很快发现的那样,在移植自己的一些代码时,能够删去原有程序中的很大一部分代码,而通过调用 Win32 提供的功能来代替它们。

利用这些功能是如此的方便,以致于我现在专门编写 Win32 程序,而且经常在公司里或会议上解释如何开发有效的 Win32 应用程序。本书将帮助读者为注定要成为工业标准的环境开发应用程序而做好准备。

0.1 关于应用程序范例

这些应用程序范例的作用是用实际的代码来解释如何利用 Win32 的高级功能。读再多的书籍也不能代替通过编写应用程序而获得的知识和经验。我在学习 Win32 时就有这样的经历。本书所提供的很多应用范例是我在了解了 Win32 如何工作的情况下经过努力编写的。