



Windows 2000 Systems
Programming Black Book

Windows
技术丛书



Windows

Windows

2000

Windows
2000
系统编程

(美) Al Williams 著
钮文良 姜余祥 申功迈 尤克 译



机械工业出版社
China Machine Press

CORIOLIS

Windows 技术丛书

Windows 2000

系统编程

(美) Al Williams 著

钮文良 姜余祥 申功迈 尤克 译



机械工业出版社
China Machine Press

本书用专业的语言详细地讲解了Windows 2000系统编程的基本概念、技术及最新的相关内容。通过实例研究,进一步向读者讲述了Windows 2000的多线程操作、同步、进程间通信、安全性、虚拟存储管理及连网操作。使读者能够从实践中掌握Windows 2000的特性,编写更有效和更有用的程序。

本书提供了直接解答及编程和管理难题的深入分析,有助于用户执行特殊任务,尤其是关键性的任务。本书适用于编写系统层次应用程序和用户应用程序的编程人员,以及编写Windows 2000系统程序的开发人员。

Al Williams: Windows 2000 Systems Programming Black Book.

Original English language edition published by The Coriolis Group LLC, 14455 N. Hayden Drive, Suite 220, Scottsdale, Arizona 85260 USA, telephone (602)483-0192, fax (602)483-0193.

Copyright © 2000 by The Coriolis Group. All rights reserved.

Simplified Chinese language edition copyright © 2000 by China Machine Press. All rights reserved.

本书中文版由美国Coriolis公司授权机械工业出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,侵权必究。

本书版权登记号: 图字: 01-2001-0479

图书在版编目(CIP)数据

Windows 2000系统编程 / (美)威廉姆斯 (Williams, A.) 著; 钮文良等译. - 北京: 机械工业出版社, 2001.1

(Windows 技术丛书)

书名原文: Windows 2000 Systems Programming Black Book

ISBN 7-111-08616-3

I. W… II. ①威…②钮… III. 窗口软件, Windows 2000 IV. TP316.7

中国版本图书馆CIP数据核字 (2000) 第79974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: 温丹丹

北京昌平第二印刷厂印刷·新华书店北京发行所发行

2001年1月第1版第1次印刷

787mm × 1092mm 1/16 · 29.5印张

印数: 0 001-5 000册

定价: 55.00元(附光盘)

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换

译者序

本书对Microsoft公司推出的新一代操作系统——Windows 2000的编程技术进行广泛、深入的研究，并利用大量实用的程序进一步阐述这种技术。本书作者Al Williams是一位富有经验的Windows编程专家，有许多优秀的编程书籍出版，其中《MFC Black Book》和《Developing ActiveX Web Control》是非常流行的两本专门编程的书籍。而本书专门针对Windows 2000系统编程而且将成为非常流行的书。本书作者还是著名杂志Visual Developer and Web Techniques的专栏作家及技术顾问，在全美国开办培训讲座。

阅读本书，可以使读者在了解一些Windows的基本编程技术的基础上，从Windows提供的服务程序入手，跟着作者的指引，一步一步地进入Windows编程技术的核心。当读完本书时，读者就会了解并使用Windows的多任务处理、存储器管理及进程间的通信等方面的技术，特别是Windows 2000操作系统在这些方面的最新技术。此外，读者还会了解有关Windows 2000编程技术方面的一些新概念。值得注意的是，本书作者列举了大量反映Windows 2000编程技术的实用程序。这是作者长期从事操作系统的研究和编程的结晶。通过这些具体、实用且简单的应用程序，可以更好地理解Windows 2000的新技术。

本书内容丰富、技术新颖、涉及面宽。参与翻译工作的教师在繁忙的教学工作之余，为广大读者奉献了这样一本好书。参加本书翻译工作的教师是钮文良（翻译第1章至第4章），姜余祥（翻译第5章至第8章），申功迈（翻译第9章至第10章、附录、前言和封底），尤克（翻译第11章至第12章）。本书全文由申功迈审校。由于时间有限，译文难免有这样或那样的缺憾，因此，我们恳切希望专家和广大读者批评指正。然而，瑕不掩瑜，不管怎样，这都是一本值得一读的好书。

2000.8.11 于北京

前言

如果你是编程人员，那么你一定了解许多事情已经改变了。我经历了从穿孔卡、纸带、到CD-ROM和兆字节的RAM；从冷藏库大小的计算机到运行冷藏库的计算机；从对在线打印机的Snoopy日历表到运行在掌上电脑上的日历表。

在办公方面的飞速进步常常造成编程人员陷入老一套。第一种老套表现在墨守成规。有一天你会发现，世界（至少你的公司）终将发生改变。例如，当在公司工作时，你从使用UNIX转换到使用Windows，许多使用UNIX的老家伙却大声地抱怨。我喜欢UNIX，但我总是愿意学习一些新东西。

第二种老套更加险恶。当操作系统（如Windows）升级和改变的时候，保持不变并忽略新技术及其特性较为容易。如果你早早开始用一个特别的系统的话，这种情况尤其不好。开始使用Windows 3.1的编程人员应当忘却老习惯，但有时他们却不能忘却老习惯。以旧的方式编写代码是容易的，因为，过去Windows的新版本还支持老的方式。

旧式的编程也许轻松自在，但对编程常常是有害的。使用旧的编程技术会引起程序消耗了过多的存储器，影响到多任务处理性能，且使你享受新的操作系统特性更困难。

用Windows 2000，就不必了解一些旧习惯。如果你熟悉UNIX或其他操作系统，你就会发现Windows 2000是一个具有完整功能的环境。它比任何现在的操作系统做同样的事更漂亮，当然你也许不得不了解实现旧诀窍的新方法。如果你是Windows编程的老手，你就会发现在Windows 2000中的新特性和微妙的差别。不管怎么说，使用Windows 2000可以有更多的时间了解Windows。

本书的内容

本书叙述了如何使用Windows 2000以获得更大的利益。从存储器管理到联网的广泛主题包括你能够检查的说明性程序，你甚至可以“借用”一些代码。

许多示例程序使用不加修饰的控制台界面，以使你能够直接钻研有关的细节。虽然这些程序不需要更多的标准的Windows编程的知识，其他一些使用MFC并且至少需要了解一些基本的Windows应用程序编程的知识。

使用现代工具，你能够轻松地创建Windows程序。但若没有使用特殊的Windows特性，你就不会真正得到完全的成功。毕竟，如果只使用标准的C或C++，你可以编写很好的UNIX或DOS程序。如果你不利用特殊的Windows特性，为什么使用Windows呢？

作为一个例子，考虑这个方案：假设你在编写分析视频信号的程序。任何一个现在的操作系统都将需要分配大块的存储区（即，用malloc）来存储这些信号。但使用Windows，你能够使用特殊的操作系统特性，以便更有效地分配需要的存储器。

本书的目的就是介绍如何利用Windows 2000的优势。如何使编程人员编写更快、更小或更

好的程序。

本书的目标

为了从本书获得更多的知识，就应当首先了解一些Windows 编程技术。本书首先着重介绍由操作系统所提供的服务程序。虽然它包含一些传统的GUI程序，多数程序强调存储器管理、多任务处理和连网的功能。

如果你是一个C++编程人员，本书介绍了如何更好地使用Windows所提供的工具。使用其他语言的编程人员，也能够从Windows提供的服务程序的有关信息获益。

当读完本书时，你就能够使用Windows的多任务处理、存储器管理和进程间通信。你还将了解服务程序、管理控制台和重叠的I/O。另外，本书还介绍有关安全、注册表、登录和外壳程序的编程。

本书涉及内容很广，涉及到Windows要提供的许多内容。因此，在每章末尾的“直接解答”小节里提供了各章的细节和实际的劝告，你将能够更快地使用好Windows。

系统需求

任何版本的Windows 2000都可以运行本书提供的多数示例程序，但我们是使用Windows 2000服务器运行的，某些程序可以运行在Windows NT 4上，少量程序也能运行在Windows 98上（也许会降低其功能）。

本书里的所有程序都使用Microsoft的Visual C++编写（Visual Studio 98包括的版本）。如果你使用其他语言，就会在第1章得到一些指导。然而，你必须自己建立本书提供的材料和你所使用语言之间的联系。

阅读方法

千里之行，始于足下。在你的需求基础上，迈开你的第一步。如果你想了解全部内容，就按顺序阅读每一章。也可以浏览目录，并选择有直接需要的章节。

每章结尾都有“直接解答”小节，介绍相关主题要点。如果你十分忙，就可以考虑从“直接解答”开始，并且参考相关章节的主要部分。

每一章包括了说明其主题的代码。完整的代码则包括在附带的CD-ROM中。

不管你如何阅读这本书，只读本前言，将不会了解得更多。挑选主题，去开始深入Windows 2000的陌生旅途吧！

英文原书书号：1-57610-280-7

目 录

译者序

前言

第1章 纵览Windows 20001

1.1 新的语言1

1.2 Windows简史2

1.2.1 Windows NT3

1.2.2 Windows 953

1.2.3 其他方面4

1.3 Windows版本4

1.4 Windows体系结构5

1.4.1 实际情况6

1.4.2 Win95与Win986

1.5 Windows的特性与差别7

1.5.1 多任务处理与线程处理7

1.5.2 UNICODE9

1.5.3 文件系统问题9

1.5.4 DLL10

1.6 开发工具10

1.7 直接解答11

1.7.1 开发工具的选择11

1.7.2 Windows 的体系结构11

1.7.3 理解进程11

1.7.4 从C++中调用API12

1.7.5 从VB中调用API12

1.7.6 从Visual J++中调用API12

1.7.7 Internet资源13

第2章 ActiveX 配套工具14

2.1 ActiveX与Java15

2.2 定义15

2.3 ActiveX对象的结构16

2.4 对象17

2.5 代码的重用18

2.6 多态性19

2.7 几个其他的ActiveX特性20

2.7.1 HRESULT和SCODE20

2.7.2 GUID/UUID/IID20

2.8 关于IUnknown21

2.9 创建对象和寻找界面22

2.10 关于索引的计数23

2.11 关于聚集23

2.12 奇妙的特性24

2.12.1 预定义界面25

2.12.2 类型库25

2.12.3 代理程序、承接程序、调度程序25

2.12.4 关于多线程27

2.13 ActiveX/C++的连接27

2.14 为什么不直接使用C++28

2.15 系统注册表28

2.16 注册表：在近亲的和个人的之上28

2.17 注册表的奇妙之处30

2.17.1 RegEnumValue30

2.17.2 RegDeleteKey30

2.17.3 错误返回31

2.17.4 数据类型31

2.17.5 检查注册表代码31

2.18 操作注册表32

2.19 注册对象32

2.20 类的安装34

2.21 使用REGEDIT34

2.22 自注册35

2.22.1 自注册EXE文件36

2.22.2 自注册DLL文件36

2.23 人格化对象36

2.24 类型库37

2.24.1 类型库在哪里	38
2.24.2 创建类型库	39
2.24.3 类型库的隐蔽	41
2.25 宏	42
2.26 了解客户的基础	42
2.27 简单的服务器程序设计	45
2.27.1 一般服务器程序的发布	46
2.27.2 服务器程序的用法	46
2.27.3 类站	46
2.28 简单的客户程序	46
2.29 EXE服务程序	48
2.29.1 如何运作	53
2.29.2 没有取值	54
2.29.3 调度程序	54
2.29.4 编写Script	55
2.29.5 现在就运行	56
2.29.6 关于服务器程序	56
2.30 DLL服务器程序	56
2.31 支持多界面	60
2.32 MFC技术	61
2.32.1 MFC的EXE服务器程序	61
2.32.2 其他考虑	70
2.32.3 MFC的DLL服务器程序	70
2.33 MFC聚合	71
2.34 IDispatch	72
2.34.1 BSTR、SAFEARRAY及 VARIANT	73
2.34.2 返回Invoke	76
2.34.3 简化IDispatch	77
2.34.4 绑定时间和效率	85
2.35 新领域: COM+	85
2.36 直接解答	87
2.36.1 在命令解释程序中的ActiveX	87
2.36.2 了解注册表	88
2.36.3 公用界面指南	89
2.36.4 对IDispatch使用MFC	89
2.36.5 对最初的界面使用MFC	89

2.36.6 关于ATL	89
第3章 进程、线程和纤程	90
3.1 其他操作系统	90
3.2 进程	91
3.3 CreateProcess	93
3.4 作业与工作区	97
3.5 线程	97
3.6 MFC 和线程	99
3.7 线程的局部存储	100
3.8 纤程	103
3.9 APC	104
3.10 直接解答	106
3.10.1 运行新程序——最容易的方式	106
3.10.2 运行新程序——有点难度的方 式	106
3.10.3 运行新程序——最难的方式	106
3.10.4 等待程序完成	107
3.10.5 用Windows API创建新线程	107
3.10.6 用C++库创建新线程	107
3.10.7 使用线程与MFC	107
3.10.8 创建MFC工作线程	108
3.10.9 创建MFC 的UI线程	108
3.10.10 使窗口和消息对话框显示在上 面	109
3.10.11 获得进程和线程的句柄	109
3.10.12 等待线结束, 设置 (或读取) 线程退出代码	109
3.10.13 操纵MFC线程	110
3.10.14 了解MFC返回值	111
3.10.15 完整的MFC示例	111
3.10.16 使用线程局部存储	117
3.10.17 使用纤程	117
3.10.18 有选择地使用线程	118
第4章 同步	119
4.1 定义问题	119
4.1.1 情况一	120
4.1.2 情况二	120

4.1.3 情况三	121	5.7 完成端口示例	158
4.2 关于同步	121	5.8 访问文件的另外方式	162
4.3 同步的细节	121	5.9 示例	162
4.4 互锁命令	122	5.10 直接解答	166
4.5 细说事件	123	5.10.1 打开文件	166
4.6 细说多用户终端执行程序	124	5.10.2 同步读/写文件	166
4.7 细说信号量程序	126	5.10.3 检测EOF(同步)	166
4.8 安全同步	127	5.10.4 复制文件句柄	167
4.9 使用WaitForMultipleObjects	127	5.10.5 关闭文件	167
4.10 能报警的等待	128	5.10.6 使用异步I/O的方法	167
4.11 可等待的计时器	128	5.10.7 利用线程使用异步I/O	167
4.12 临界区	128	5.10.8 启动重叠I/O	169
4.13 主要示例	130	5.10.9 结束异步I/O	170
4.14 其他可等待的句柄	137	5.10.10 检测EOF(异步)	170
4.14.1 改变标志信息	137	5.10.11 使用ReadFileEx和WriteFileEx	170
4.14.2 控制平台句柄	137	5.10.12 使用完成端口	171
4.15 直接解答	138	5.10.13 存储器映像文件	171
4.15.1 选择同步的方法	138	第6章 进程间的通信	172
4.15.2 避免死锁	139	6.1 为什么重要	173
4.15.3 使用事件	140	6.2 Windows 2000的IPC机制综述	174
4.15.4 使用多用户终端执行程序	140	6.3 简单的IPC	176
4.15.5 使用临界区	140	6.4 将DLL用于共享存储器	182
4.15.6 使用信号量程序	141	6.5 其他共享存储器技术	185
4.15.7 等待多个对象	141	6.6 匿名管道	186
4.15.8 使用互锁的变量	141	6.7 命名管道	189
4.15.9 了解MFC和同步	142	6.7.1 使用管道	189
第5章 文件I/O	147	6.7.2 消息模式管道	189
5.1 为什么异步I/O	148	6.7.3 其他管道命令	189
5.2 回顾文件I/O	148	6.8 邮件通道	190
5.3 使用线程	151	6.9 关于网络接口	198
5.4 重叠I/O	152	6.9.1 网络接口	199
5.4.1 完成I/O	152	6.9.2 启动服务器程序	200
5.4.2 使用事件	153	6.9.3 启动客户程序	201
5.4.3 使用ReadFileEx和WriteFileEx	153	6.9.4 Windows网络接口	201
5.4.4 EOF检测	153	6.9.5 更多的网络接口	202
5.5 I/O完成端口	154	6.9.6 关于跳棋程序	203
5.6 应用程序的示例	154		

6.9.7 特殊考虑	217	7.3 使用VirtualAlloc	252
6.9.8 改进	218	7.4 实际上的VirtualAlloc	253
6.10 MFC网络接口	218	7.5 使用页面属性	255
6.10.1 关于CSocket	219	7.6 使用堆栈	259
6.10.2 使用有CSocket的档案文件	220	7.7 直接解答	261
6.10.3 更进一步的CAsyncSocket	220	7.7.1 何时使用VirtualAlloc	261
6.10.4 封锁命令	220	7.7.2 使用VirtualAlloc与VirtualFree	261
6.10.5 示例	221	7.7.3 设置页面保护	262
6.10.6 基本架构	221	7.7.4 读取页面保护	262
6.10.7 加入客户网络接口	234	7.7.5 响应异常事件	262
6.10.8 其他考虑	235	7.7.6 创建新堆栈	263
6.10.9 网络接口的包装	236	7.7.7 查找默认堆栈	263
6.11 关于RPC	236	7.7.8 分配与释放堆栈数据	263
6.11.1 RPC理论	236	7.7.9 压缩堆栈	264
6.11.2 实践中的RPC	237	7.7.10 调试堆栈	264
6.11.3 一步一步地编写RPC	237	7.7.11 堆栈性能的考虑	264
6.11.4 在IDL文件内	241	第8章 Windows 的安全	265
6.11.5 编写客户程序	242	8.1 安全的目标	265
6.11.6 编写服务器程序	242	8.2 特权与授权	266
6.11.7 实现连接	243	8.3 用NULL填充	267
6.11.8 更多内容	243	8.4 SID	268
6.12 Microsoft消息队列	243	8.5 ACE和ACL	268
6.13 直接解答	244	8.6 安全对象的类型	269
6.13.1 IPC方法的综述	244	8.7 使用默认值	269
6.13.2 使用 WM_COPYDATA	245	8.8 建立SD	270
6.13.3 使用共享存储器的DLL	246	8.9 令牌与人格化	271
6.13.4 文件映像共享存储器	246	8.10 建立专断的ACL	271
6.13.5 使用匿名管道	247	8.11 使用AccessCheck	275
6.13.6 使用命名管道	247	8.12 专用安全性	275
6.13.7 用管道替代标准句柄	248	8.13 特权	279
6.13.8 使用邮件通道	248	8.14 Kerberos	279
6.13.9 使用网络接口	248	8.15 直接解答	280
6.13.10 使用RPC	249	8.15.1 了解对象的拥有者	280
6.13.11 了解何时舍弃或少用IPC方法	250	8.15.2 将SID变换为名字	281
第7章 存储器管理	251	8.15.3 将名字变换为SID	281
7.1 为什么存储器管理会带来麻烦	251	8.15.4 创建描述符	281
7.2 存储器分页	252	8.15.5 在绝对与自相对描述符间转换	282

8.15.6 使用默认的安全性描述符	282	10.3.1 逐步地建立MFC的ActiveX对象	320
8.15.7 使用通用安全性描述符	282	10.3.2 图标句柄细节	321
8.15.8 使用特定的安全性描述符	283	10.4 比较ATL与MFC	325
8.15.9 使用专用安全性	283	10.5 图标托盘程序	325
第9章 注册表与登录	284	10.6 关于智能指针	329
9.1 放弃INI文件	284	10.7 快捷键	329
9.2 注册表: 封闭的和个人的	285	10.7.1 创建快捷键	330
9.3 注册表的奇异特性	287	10.7.2 消除快捷键	331
9.3.1 RegEnumValue	287	10.8 MMC	334
9.3.2 RegDeleteKey	287	10.8.1 ATL Wizard可实现的	335
9.3.3 错误的返回值	288	10.8.2 Wizard不能实现的	335
9.3.4 键入的数据	288	10.8.3 增加代码	336
9.3.5 MFC程序示例	288	10.8.4 ATL MMC快捷按钮	337
9.4 那是注册表吗	299	10.9 直接解答	354
9.5 注册表中有什么	299	10.9.1 关于扩展命令解释程序	354
9.6 建立REG文件	299	10.9.2 撤销快捷键	355
9.7 在文件类型内部	301	10.9.3 创建快捷键	355
9.8 登录	302	10.9.4 编写图标托盘程序	355
9.9 事件资源	302	10.9.5 其他命令解释程序的命令	356
9.10 创建消息文件	303	10.9.6 关于MMC	356
9.11 系统消息	304	10.9.7 编写MMC快捷按钮	357
9.12 操作记录	305	第11章 Internet与网络编程	358
9.13 直接解答	308	11.1 Internet快捷键	358
9.13.1 打开注册表主键	308	11.1.1 开始运行	359
9.13.2 查找子键	308	11.1.2 使用CEditView	366
9.13.3 查找键值	309	11.2 使用Internet资源管理器	366
9.13.4 使用注册表替代INI文件	309	11.2.1 自动操作	367
9.13.5 创建REG文件	310	11.2.2 进一步控制	372
9.13.6 创建INF文件	310	11.2.3 使用CHtmlView	373
9.13.7 创建消息DLL	311	11.2.4 资源的URL	375
9.13.8 注册消息DLL	311	11.2.5 HTML对话框	376
9.13.9 登录事件	311	11.3 Web发送API	379
9.13.10 将错误转换为消息	312	11.3.1 定制上载程序	381
第10章 用ATL的命令解释程序技巧	313	11.3.2 其他容易的方法	385
10.1 扩展命令解释程序	313	11.3.3 ActiveX的连接	386
10.2 基本的ATL	315	11.3.4 读取Web	386
10.3 使用MFC代替ATL	319	11.4 MFC Internet的支持	391

11.5 活动目录服务概述	401	12.2.1 安装服务程序	434
11.5.1 术语	401	12.2.2 调试服务程序	440
11.5.2 使用ADSI	401	12.2.3 其他服务程序的考虑	440
11.6 直接解答	402	12.3 面向对象的服务程序	441
11.6.1 运行Internet快捷键	402	12.4 直接解答	447
11.6.2 Internet通信的方法	403	12.4.1 启动控制台程序	447
11.6.3 使用资源URL	403	12.4.2 使用控制台程序的MFC	447
11.6.4 在Internet Explorer内	404	12.4.3 访问控制台的方法	447
11.6.5 使用Scripting对象模型	404	12.4.4 创建GUI程序的主控制台	448
11.6.6 使用HTML对话框	405	12.4.5 创建和使用辅助控制台	448
11.6.7 使用MFC的Internet支持	405	12.4.6 处理控制台事件	448
第12章 控制台应用程序与服务程序	406	12.4.7 查找控制台的窗口句柄	449
12.1 创建简单的控制台应用程序	406	12.4.8 在服务程序内	449
12.1.1 一些内容	406	12.4.9 访问服务程序	450
12.1.2 AVC 细节	424	12.4.10 C++服务的基类	450
12.1.3 学习控制台窗口	425	12.4.11 调试服务程序	450
12.1.4 特殊的控制台命令	427	附录A 超越新领域	452
12.1.5 处理事件	429	附录B Windows 2000: 编程的冒险	
12.1.6 创建有用的新的控制台	429	旅行	456
12.2 服务程序	433	附录C 随书光盘的内容	458

第1章 纵览Windows 2000

毫不隐瞒地说，我是一个科幻迷。但是，我认为现实中的科幻描述很荒谬可笑。虽然我能接受存在生物圈和星系帝国的事实，甚至可以接受超光速的星际飞行器，但是我不能接受的是那么容易地与外星人进行交流。当然，在Star Trek中，所有外星人都讲英语。事实上，极少数的科幻故事涉及到第一次接触外星文化会如何。通常都是外星人通过收音机和电视学英语，要么它们依靠一台功能强大的翻译机。不管哪种方法都在人们看不见的地方进行，因而不会妨碍故事的进展。

考虑人们之间的交流。不论与谁交流，不论要讨论什么话题，两人的交流大部分都是不言而喻的，并不需要表达出来，举个什么例子呢？看下面古怪却正确的句子：

“I saw the Grand canyon while flying to New York.”（在飞往纽约时，我见到了Grand Canyon。）

你我都知道，说话者那时正飞往纽约，并看见了Grand Canyon（大峡谷）。但从语法上理解，这句话意指，Grand Canyon大峡谷正飞往纽约，且我看见了它！当然是废话，因为大家都知道Grand Canyon大峡谷是不会飞的。而外星人（或电脑，就此而言）并不知道Grand Canyon是什么。之所以人们能理解这句话，是因为有每个人了解一些常识。

当你穿越世界上的文化界线时，问题则更糟。多年前，英国报纸上刊载在头条的真实新闻如下：

“British Left Waffles on Falklands”（British Left议论Falkland群岛的局势。）

通常，美国人会感到惊讶，英国人忘记吃早餐为什么值得报道（他们认为英国人在Falklands忘记吃华夫饼干）。然而，如果你了解在英国发生的事件，就会明白British Left是一个政治党派。头条新闻在说，他们不能接受Falkland群岛的局势。

英国的报纸编辑是不是疯了？当然不是。他们简单地认为，读者已经有了这些方面的知识，而多数美国人却没有。

同样的概念在任何复杂的程序设计中也是正确的。当你与其他人讨论编程问题时，你也应设想他们了解这些问题。若在大街上拦住一个人，就问为什么Printf连接到代码时会失败。除非你碰巧遇到C或C++程序员，不然对方绝对不知所云。

1.1 新的语言

假定此刻你是C++程序员。你与其他C++程序员一样知道某些基础知识。如果老板指派你解决Cobol语言中的2000年问题，没人指望你能顺利地开始这项工作。Cobol语言的程序员具有和C++程序员不同的知识基础。Cobol程序员和C++程序员的交流通常十分困难，每个人都认为如此。

然而，由于某些原因，操作系统并不能共享这种奢侈。老板们似乎期望你轻易地、不绕弯路地在UNIX、VMS、MVS、MS-DOS和Windows间转换。如果你擅长Windows的某个版本，这格外正确。但事实上，这些任务大相径庭。如果你学习过Windows 3.0下的程序设计，那你就可以慢慢掌握Windows NT或Windows 2000，但这也是努力的结果。

如果你从其他操作系统转而使用Windows 2000，那情形会相当痛苦。作为一个UNIX的老手，我目睹了许多朋友从UNIX转向Windows的经历。他们通常抱怨Windows怎么如此糟糕。

事实上，32位版本的Windows 和 UNIX并不是完全不同。但确有差异。丢弃原有的基础知识并重新开始，确实非常困难。如果有人要求“创建一个新进程”，那么UNIX的权威，就会考虑fork 而不是CreateProcess。当然，你能在帮助中查到CreateProcess，但就像出国旅行并使用一本愚蠢的外语常用手册，这会有几分效果，但不舒服也不方便。所以你需要一些基础知识。

本章的目的是要使每个人从相同的起点开始。这里将给大家展示应该了解的（或许已经知道的）有关Windows的一些内容。这样，你就增加了需要了解的书中其余部分的基本信息。

1.2 Windows简史

如果我们回去创建公司并开辟Windows操作系统市场，不久之后，我们也许会另谋职业。Windows刚面世时，是有争议的。只有Microsoft这样的公司才能支撑Windows，直到开发出实用的东西。

Windows的问题不全是Microsoft的过失。不要忘记，那时的计算机只有缓慢的处理器、带宽窄的总线及容量小得可怜的存储器。Windows能运行就很幸运了。只是时机尚未成熟。

Intel386是第一个PC兼容的处理器，它能处理真正像Windows这样的系统。Microsoft悄悄地推出命名为Windows/386的产品，且显示出它的优点。这就是Windows的第一个版本，运行良好，深受大多数人的喜爱。

Windows/386之后不久，Microsoft推出Windows 3.0。这是第一个Windows 的商业化版本。该版本并不是没有共享问题，而是问题没有足以影响到人们购买该产品。由于潜在购买者众多，软件公司急急忙忙进行Windows的程序设计（通常，称作应用程序）。

而后不久，Microsoft 又发布Windows 3.1。这个版本不仅修补了Windows 3.0中的许多程序错误，而且还引入了许多重要的新特性。许多编程人员首先学习了基于Windows 3.0或3.1的Windows编程。

尽管用户们非常喜欢Windows的这些版本，但多数编程人员都不喜欢它们。Windows的这些版本还保留了以前的版本遗留的许多包袱。如内存管理是有缺陷的、分段式存储模式常常令人生厌、多任务处理还很粗糙并需要涉及到的所有程序协同合作完成。如果任一个程序不合作，整个系统便会崩溃。16位系统的根使其对来自UNIX或其他32位操作系统的端口代码相当困难。

另外，市场压力要求Windows的应用程序，即编程人员提供的应用程序。不论好坏，这就是大多数程序员的起点。

Microsoft也曾发布过不同版本的Windows 3.1，包括Windows for Workgroups 3.1和3.11、Windows 3.11。从开发角度讲，这些并不重要。

1.2.1 Windows NT

然而，真正的举动是Microsoft决定何时开发Windows NT。因为Microsoft深信，如果不是受以前处理器的限制，Windows会更加出色。同时也知道，在连网和分布处理时，Windows无法与UNIX这样的操作系统竞争。

那时，使Microsoft感到焦虑的事情是：其他操作系统或微处理器结构的问世会挤垮Windows和Intel硬件平台。Microsoft确信只有借助NT的灵活性，使其仿效其他操作系统运行于不同硬件平台，才不致受到影响。

NT的发布可以说毫无生气。它对硬件的需求，远远超过当时人们所具有的硬件环境。在16MB内存的情况下（而那时很少有内存超过8MB），仅仅得到最小的性能。同时，还要有CD-ROM去安装NT，而那时极少数人有光驱（事实上，我买光驱完全是出于安装NT的需要）。

即使已经具有NT运行所需要的硬件环境，这里仍存在两个主要的问题：缺少设备支持和缺少应用程序。因此，当时世界上只有少数PC机运行NT，而多数仍运行Windows 3.1。硬件和软件的开发商缺少支持NT的动力。当然，NT还可运行多数Windows 3.1的程序，而不是所有的程序都可以运行。为什么要买NT，难道只是为了运行Windows 3.1的程序吗？尤其，NT需要更多的费用，并需要昂贵的硬件升级。

然而，Microsoft仍支撑着NT（就像当初开发Windows产品那样），并默默地在几个版本上改进NT。另外，Microsoft知道必须要吸引第三方开发商来编制NT的应用程序。

1.2.2 Windows 95

解决上述问题的答案是Windows 95。Microsoft推出新产品Windows 95替代Windows 4.0。Microsoft希望将Windows 95看作 Windows NT Lite（体会其特性，而不是其替代品）。然而，实际上Windows 95更像Windows 3.1。

Windows 95的关键是其名义上支持如同Windows NT一样的编程界面。那么，为什么不编写运行于Windows 95和NT的程序代码呢？这样会使能运行于NT上的软件大量出现，不是吗？为达到这个目的，Microsoft所要做的是使Windows 3.1的每个用户立即购买Windows 95。这形成一些设计选择，它们在其他方面是难于调整的。

这不仅是一个运作得非常好的战略性工作，32位应用程序的数量激增。那么，已经有了NT产品的经销商们会如何呢？这不正是为NT打开Windows 3.1市场的大好时机吗？然而事实并非如此。Windows 95应用程序接口（Application Programming Intertace, API）只是Windows NT应用程序接口的一个子集，如果了解了这一点，就能编写在两个平台都能运行的代码，但是除非从开始就有打算，否则编写的代码是靠不住的。在技术上，Windows 3.1 API称作Win16的API，而其他版本则使用Win32的API。不过，在Windows 95下的Win32与在Windows NT下的Win32并不完全是一回事。

然而，从Windows 3.1程序转到新的Windows 95/NT API显得突然。许多程序员都趋之若鹜。对于那些仍执迷于Windows 3.1的用户，Microsoft为开发者开发了称作Win32S的产品。如果在程序中使用了Win32S的库，则会形成一个很小的Windows NT命令子集。Win32S库会自动地将其

转换成Windows 3.1式的命令。理论上, 代码的一部分可以运行在Windows 3.1、Windows 95和Windows NT上。

事实上, Win32S失败了。部分原因是其运行不稳定, 另外的原因是实际上构成的命令子集非常小, 并对开发者的限制太多。虽然仍有少数开发者在改善Win32S, 最终Microsoft还是放弃了它。

总而言之, Microsoft的战略是非常成功的。也能运行于NT的Windows 95的应用程序在各地发展起来。突然发现, 人们可以在Windows NT上运行真正的32位程序了。

1.2.3 其他方面

与此同时, 硬件方面的技术发展的也相当快。奔腾(Pentium)级的处理器已成为标准, 内存的价格迅速下跌。高性能视频卡和打印机也已具有8MB的随机存储器(Random Access Memory, RAM), 许多人的PC机装上64MB或更大的RAM。这些PC机足以满足运行NT的要求, 因此许多人开始转而使用 NT 3.5或NT 3.51。多处理器的主板价格的下跌, 更刺激人们转向使用NT。(在多处理器系统上, 运行Windows 95是一种浪费, 因为Win95任何时候都只使用一个处理器)

Internet的蓬勃发展更激发了人们使用NT的兴趣。NT是一个相当不错的网络服务器平台, 最重要的是Microsoft在NT服务器软件包中包含了Internet工具。

Windows NT 3.51的唯一问题是, 从用户角度看, 它的外观和运作太象Windows 3.1了。许多用户更喜欢Windows 95的界面风格、即插即用、及仅适用于Windows 95的其他特点。Windows NT 4.0的发布, 则带给NT用户更多这方面的东西(和一些新功能)。

对于Windows 2000, Microsoft带来更多的用户所希望的Windows特性(如支持多台监视器), 并为编程人员新增了不少新工具。时至今日, 大多数(并不是所有)的家庭用户使用Windows 95或Windows 98。大多数公司用户, 开发者和超级用户(也不是所有的)在运行Windows NT, 并可能转而使用Windows 2000。

1.3 Windows版本

Windows的每个版本都有不同的风格。多数情况, 编程人员并不在意他们使用哪个版本。不同风格间的主要区别是Microsoft在磁盘上放置的额外软件和许可证号(像协作用户所允许的号码)。

单桌面用户的首选是专业版(Professional Edition)。它包含作为个人操作系统而使用Windows所需的全部功能。甚至可以用专业版构成一个小型的网络。

如果要使Windows运行在网络或Web服务器上, 则应选择服务器版(Server Edition)。根据需要, 服务器产品有几个版本(例如, Site Server提供Microsoft个人化系统和用于大容量Web站点的工具)。

最后, Windows Advanced Server和Datacenter Server使程序潜在地寻址更大的虚拟空间。Windows的这些特殊版本都提供了先进的联网特性和支持附加的多进程处理。

然而，作为编程人员，这些版本的Windows之间没有真正的差别。可以用同样的方法，使用同样的工具，做相同的调用。如果依靠额外的组件（或3GB虚拟内存的限制），则也许需要特殊的版本，但通常情况，不必在意程序必须运行于哪一种Win32的平台上。

1.4 Windows体系结构

从编程人员的观点，Windows 2000提供了相当简单的环境。每个程序都有一个地址范围从0到2GB的平面（非分段式的存储结构）存储空间。当然，大部分程序并没有全部使用此空间，的确，今天并不是所有系统都有那么充裕的硬盘空间和物理存储空间。但这不是问题，因为Windows只有在程序使用时才分配其真正的存储空间。

地址是从0到4GB的32位整数值（尽管只有0到2GB有效）。你不必担心使用老版本的Windows的分段结构，跟踪内存或其他古怪的事情。

顺便指出，Pentium处理器允许寻址到4GB的空间。然而，操作系统保留了2GB的最高限制。操作系统真能用到那么大的空间吗？不，能够运行Windows的处理器有2GB的用户程序空间和2GB的操作系统监控程序空间。因此这种模式与微处理器匹配得非常好。

Windows API是系统的标准接口。我们可以构成针对操作系统的调用命令，而操作系统实际上寄存于系统动态链接库（Dynamic Link Libraries, DLL）。如果使用其他操作系统，这些命令则与之不同，但是功能相同。这些功能包括分配存储空间、打开文件等等。这些特性主要被人们用来编写程序设计语言（像C++）。你将经常使用程序语言中的结构去实现上述功能。例如，你大概不会调用Windows API中的CreateFile，而会调用C++库中的fopen。另一方面，CreateFile可以实现fopen不能处理的某些功能。

如果你有先验的Windows编程经验，就会发现这些函数的命令有几分相似。许多函数与Windows 3.1及更早的版本中的调用函数相同。而另一些有些细微的差别。有时，Microsoft稍微改变命令的名字，以表示其参数或返回值存在不同。对名字的常见改动是在其尾部加上Ex。这样，在Windows 3.1中使用GetCurrentPosition，而在Windows 2000（或NT）中则使用GetCurrentPositionEx。它们有什么不同呢？当两个16位字表示的32位字返回值时，GetCurrentPosition返回当前描述的位置。而GetCurrentPositionEx则返回作为变元传送的指针结构中的两个32位字。这些函数有相同的目的，而Win32变量则以32位运作。

另外的改变发生在接收字符串的任一函数中。NT和Win2000内部不使用ANSI字符串。取而代之，它们使用称作UNICODE的16位的字符（且后者使用的更多）。然而，大多数程序使用ANSI（或考虑8位的ASCII字符）。所以接收字符串每个函数有两种情况：其一给出ANSI字符串（并将其转换为UNICODE），另一种则直接接收UNICODE字符串。

例如，考虑Windows 3.1的函数MessageBox。大家知道，这个函数弹出带文本字符串的对话框，Win32没有此函数。但它有MessageBoxA给出ANSI变元及MessageBoxW处理UNICODE。然而，文档也引用MessageBox。这是因为C++编译器和许多其他工具根据涉及UNICODE的优先权，来选择正确的结果。在C++程序中，当引用MessageBox时，根据对编译器的指令，使用的扩展为MessageBoxA和MessageBoxW宏。当然，这意味着编译器的错误信息涉及到哪个函数会被拒绝，例如MessageBoxA。