科海电脑技术丛书

C++/C# Programmer's Guide for Windows 2000

基于 Windows 2000 的 C++/C# 程序员指南

- 充分利用 Windows 2000 的功能进行应用开发
- 面向 C++与 C# 编程人员
- Windows 2000 的并行结构、线程、进程和异常处理
- .NET 框架：体系结构、公共语言运行环境(CLR)、元数据等
- 光盘提供书中所用示例的全部代码

[美] Ronald D. Reeves 著
李 路 译

PH PTR

科学出版社

科海电脑技术丛书

基于 Windows 2000 的 C++/C#程序员指南

(C++/C# Programmer's Guide for Windows 2000)

[美] Ronald D. Reeves 著

李 路 译

科学出版社

2002

著作权合同登记号：01-2002-1514

内 容 提 要

本书是作者集 40 多年的计算机系统设计与开发经验，精心编写的一本 Windows 2000 应用程序编程指南。书中重点讲述微软的.NET 框架、Visual C++ 7.0 和 C#编译器，展示如何通过这两种编译器，充分利用.NET 框架构建强大的分布式 Web 应用程序。全书共 5 章，并附有 15 个附录。附录部分详细列举了本书的所有素材，包括所有的 Win32 API 和.NET 框架基类，以及软件优先级列表等。

本书实例丰富，附带的光盘提供了很多 C++与 C#的程序范例，帮助用户理解整个开发环境的配合与协调工作。本书适用于有经验的 Windows 2000 程序开发人员，也适用于从其他开发环境转向 Windows 2000 的开发人员。

图书在版编目（CIP）数据

基于 Windows 2000 的 C++/C#程序员指南/（美）理维斯（Reeves,R.D.）著；
李路译.－北京：科学出版社，2002.8
ISBN 7-03-010533-8

Ⅰ. 基… Ⅱ.① 理…② 李… Ⅲ.C 语言—程序设计 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2002）第 041271 号

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717

北京市耀华印刷有限公司印刷

科学出版社发行 各地新华书店经销

*

2002 年 8 月第一版	开本：异 16
2002 年 8 月第一次印刷	印张：20.875
印数：1-5 000	字数：481 962

定价：42.00 元（含盘）

译 者 的 话

近几年来，.NET 框架的概念呈现出了风起云涌之势，因为它将类和面向对象程序设计的丰富内涵与通信紧密地结合在一起，更加适合新型应用设计的需要。同时，微软将 Windows 2000 操作系统的 Win32 API 封装在.NET 框架基类中，大大减轻了开发人员的负担。基类和 CLR 的出现，也使开发人员可以更轻松地控制很多包含在 Windows 2000 应用开发和执行中的函数。在此基础上，微软又相继推出了 Visual C++ 7.0 和 Visual C#编译器，为使用.NET 框架和基础类库提供了更加简单的工具。

本书的作者是一位资深的 Windows 程序开发者，同时也是一个经验丰富的教师，两种身份的结合，使他在本书中所传达的知识非常丰富且易于理解。首先，他分析了 Windows 2000 操作系统的体系结构；接下来讲述了如何使用 Win32 API 开发 Windows 2000 的应用程序；然后，他又分析了.NET 框架的体系结构，向大家展示了这种框架与 Windows 2000 的 Win32 API 之间的关系；最后则阐述了 Visual C++编译器、新型的 C#编译器以及它们在新环境下的构成。

本书适用于有经验的 Windows 程序员，特别是从其他版本的 Windows 转向 Windows 2000 的程序员，同时也适用于那些从其他环境（比如 UNIX）转向 Windows 2000 的程序员。它从体系结构和应用开发的角度对 Windows 2000、Visual Studio.NET、Visual C++ 7.0 以及 C#进行了原理性和功能性的介绍，是一本很好的程序员指南。对于广大急切需要了解 Windows 2000、Visual C++ 7.0 与 C#功能和特性的程序员来说，此书是不可多得的好书。

本人希望通过自己的工作，使读者可以全面地领会作者的意图，尽可能地掌握本书所传授的知识。但是由于经验和能力的限制，可能会出现一些错误，希望读者能够谅解，并欢迎批评指正。如果大家阅读本书后感觉有一些收获的话，本人将倍感欣慰。

序

Visual Studio.NET 代表着软件应用开发环境持续发展过程中的一次显著进步，而.NET 框架更是一个伟大的进步，因为它在通信时结合了类和面向对象程序设计的丰富内涵，并充分利用了 Windows 2000 操作系统的强大功能。自此，在开发人员与编译器、.NET 框架基类、CLR（Common Language Runtime，通用语言运行时间）以及 Windows 2000 操作系统之间，一种强大的融合力正在生成。由于.NET 框架基类封装了 Windows 2000 操作系统的 Win32 API，所以可大大减轻开发人员的负担，同时，基类和 CLR 智能化的协作，也可以自动控制很多包含在 Windows 2000 应用开发和执行中的函数功能。

在过去的 7 年中，Ron Reeves 博士一直担任着 UCI 软件技术培训中心（UCI Software Technical Training）的顾问和教师。作为一名教师，Ron 做出了巨大的努力，以确保学生们能够清楚地理解所有的知识，并使他们的学习经历更加轻松愉快。同时，作为一名图书创作人员，他也表现出了卓越的才能。他还是一位优秀的天才作家，大家在读这本书的时候，一定会喜欢上他所传达的知识和他的创作风格。

在这本书中，Ron 运用他 40 多年从事计算机系统设计和应用设计的经验，讨论了一种创建应用软件的革命性方法。他采用自下而上的顺序详细解释了这个新型的体系结构是如何协调工作的。首先，他分析了 Windows 2000 操作系统的体系结构及其重要的组件；接下来讲述了如何使用 Win32 API 开发在 Windows 2000 下运行的应用程序，当然这是 Visual C++不受管理模式，也是该编译器的默认模式。然后，他又分析了.NET 框架的体系结构及其所包含的重要组件，这些内容向大家展示了.NET 框架与 Windows 2000 的 Win32 API 之间的关系；下一步的阐述则上升到 Visual C++编译器，以及它在新环境下是如何工作的；最后一章则分析了一种新型的 C#编译器以及它在新环境下的构成。本书同时指出如何在开发人员与硬件引擎之间开发出更多且更加灵活的 META 层，以及这些 META 层将如何影响开发者对编译器结构的理解。

Andrew Scoppa
UCI 软件技术培训中心

前 言

Windows 2000 是一个重要的大型系统，同时也是微软更具包容性的体系结构 Windows DNA 2000 的核心。在这里 DNA 表示分布式互连网应用（Distributed interNet Applications），它代表了微软建立分布式系统的观点。这种体系结构的重点在于为企业开发新型的“数字神经系统（digital nervous system）”。“数字神经系统”其实是人类神经系统的合成数字替代品，它是一个重要的信息系统，可以在正确的时间提供综合的信息流，并把它们送到组织中的正确位置。设计此类系统可以从不同的等级入手，最低的等级是使用提供访问特权指令的设备驱动程序，最高的等级则是使用功能强大的应用软件开发工具。本书的讨论重点是使用 C++/C#和 Visual Studio.NET 开发环境进行 Windows 2000 应用程序设计。C++/C#和 Visual Studio.NET 的讨论和范例都基于 Windows 2000 所要求的 BETA 1 版 Win32 程序设计。本书适用于从其他环境转向 Windows 2000 的程序员，比如 UNIX 和大型主机（mainframe），也适用于那些从早期的 Windows 版本升级到 Windows 2000 的程序员。本书的大部分内容都关注于构成.NET 框架和 Windows 2000 操作系统的组件。大家必须意识到，在所有的组件中都存在着无数的约束，同时也应该从开始就努力理解这些组件是如何在整个.NET 框架和 Windows 2000 操作系统中协调工作的。

学习如此复杂的技术可能会有相当大的挑战性，因为文献资料的数量是巨大的，而且一直都在发生着改变。你也许会参与互连网上的讨论小组，这样会使你每天都收到几百个电子邮件。同时，对于这项技术的各个不同方面，都存在着太多太多的专业书籍。那么究竟怎样才能了解这项技术的全貌呢？本书瞄准技术的整体性，向 C++/C#程序员提供了应用指南。它不是技术文献和各种专业书籍的替代品，也不是帮助大家学习不同类型 API 的“圣经”。相反，本书只是一个教程，用以向大家提供创建有效的 Windows 2000 应用系统所需要的一切基础信息。本书和附带的光盘中，有很多 C++与 C#的程序范例，可以帮助大家快速理解整个环境是如何配合在一起协调工作的。

第 1 章“概论”是对 Windows 2000 体系结构的概述性介绍。在这里展示了 Windows 2000 的整体结构框图，并概括地讨论了 Windows 2000 的一些关键组件。这一章还包括了对不同版本 Windows 2000 的一般性描述。

第 2 章分析了 C++/C#程序员进行 Windows 2000 程序设计时需要了解的最基本的原理概念，这一章从 Windows 2000 的结构概述开始，用足够的细节帮助那些有经验但却刚接触 Windows 2000 的 C++程序员充分理解 Windows 2000。这一章还包括了进程、线程和作业的概念，以及对错误和异常情况的处理。软件优先级结构的内容也包含在本章里。这一章还详细解释了如何使用 Win32 API 而不涉及.NET 框架的基类进行程序设计。但是大家可以看到，.NET 框架基类几乎完全封装了这些 Win32 API。作为一个 C++程序员，如果你愿意的话，

仍然可以选择使用本机模式（native mode），你也可以在应用组件中将本机模式和受管理代码模式（managed code mode）混合在一起使用，但应该知道的是，C#只工作在受管理代码模式下。然而，C#代码可以利用一些关键字而使用本机模式的部分代码。在第 4 章将会看到，Visual Studio.NET 有一种标准的方法来处理错误与异常情况。以上这些主题都直接影响到是否能够创建一个可靠的应用。最近由 Ron Reeves 和 Marshall Brain 共同推出的一本书《Win32 系统服务——Windows 98 和 Windows 2000 的核心》，已经详细讲述了如何使用 Win32 API 进行应用开发。

第 3 章主要从体系结构的角度，分析了.NET 框架的基本特性，其中所使用的素材是本书的关键。.NET 应该会成为现代 Windows 应用体系结构和程序开发的绝对中心。因此，理解一些基本的内容是非常重要的。这一章将向大家提供学习第 4、第 5 章 C++和 C#知识的基础。在多级应用系统的开发中，COM+将会继续扮演重要的角色，关于这个主题有很多书可以参考，包括 Robert Oberg 的《理解 COM+程序设计——Windows 2000 DNA 指南》。

第 4 章分析了 Visual C++ 7.0 编译器，以及使用这个编译器创建应用时所涉及到的内容。其中的讨论主要基于使用受管理代码模式的 C++，因为这是使用 CLR 组件时的惟一模式，它可以充分发挥 CLR 新型的管理特性的所有优点。这一章还详细讨论了帮助程序员使用.NET 框架体系结构的 Managed Extensions for C++。这一章内容的出发点是考虑如何在应用开发中使用编译器，而不是单纯的语法分析。

第 5 章分析了 C# 编译器，以及使用这个编译器创建应用时所涉及到的内容。Windows 2000 和.NET 框架都将依赖于 C#进行企业级的系统开发。而且，对于使用 C#的分布式处理，这种新的方法不要求对其行为进行系统注册，用单纯的语法就可以实现。

附录部分详细列举了本书的所有素材。某些附录的内容可能会像一个章节那样多，因为它包含了所有的 Win32 API 和.NET 框架的基类。本书附录还包括软件优先级的列表等等。

目 录

第1章 概 论

本章将讨论Windows 2000操作系统的主要性能，这些性能对于Windows 2000程序设计都是最基本和最重要的。显然本章还可以包括更多的内容，但是我们相信这些内容对于理解Windows 2000程序设计才是最重要的。这种最基础的知识，以及书中所包含的其他主题，将有助于对很多Windows 2000服务进行设计，这些服务包括基本服务、组件服务、数据访问服务、图形与多媒体服务、管理服务、消息与协作服务、网络服务、安全、工具和语言、用户接口服务，以及Web服务。书中所包含的所有主题可以帮助观察这些Windows 2000服务，并提供一种对它们进行程序设计的方法。掌握了这些以后，将进入Visual Studio.NET软件开发环境，了解如何在这种新的开发环境下进行应用开发。最后，我们还会介绍如何在这种新型的开发环境下控制并利用Windows 2000引擎的强大功能。

复杂操作系统的一个重要体系结构特征是软件优先方案，因为它控制着如何对多个并发事件进行处理。一般来说，这是使用操作系统（OS，Operating System）开始开发应用程序时，要首先了解的问题。OS也有一个硬件优先方案，但它不属于本书的讨论范畴。接下来要介绍的是接口通信API（Application Programming Interface，应用程序编程接口），它使开发者可以控制OS并与之进行通信。当然，与优先方案紧密相连的是OS部分如何工作，如何向不同的应用软件行为分配资源，特别是内存的分配。软件同步机制可以用来控制应用和系统资源分配，它也与软件优先方案有着紧密的关联。在Windows 2000这种伪实时操作系统中，依据线程的优先级来同步不同的事件是非常重要的。下面各小节将详细讨论这些问题。这些基础知识适用于所有的Windows 2000平台。Windows 2000平台包括以下4个产品：

- Windows 2000专业版
- Windows 2000服务器
- Windows 2000高级服务器
- Windows 2000数据中心

微软做了很多卓越的工作，使得Windows 2000的编程接口Win32 API家族成为所有Windows 2000平台的标准编程接口。Win32 API代表着接口通信能力，允许程序员控制Windows 2000的各个特性功能并与之进行通信。微软还提供了一个内容丰富的WFC（Windows Foundation Classes，Windows基础类库）。.NET框架结合Windows 2000，为使用

Windows 2000操作系统的功能提供了一个珍贵的语义接口。Visual C++编译器、C#编译器以及IDE（Integrated Development Environment，集成开发环境）的相关工具，允许在IDE中开发、执行和调试程序。这个环境向程序员提供了一种简单但却强大的工具，使他们可以使用Windows 2000 API开发应用软件和相关的服务。如果一个程序员正在使用结合了Win32 API的Web服务API或WinForm进行程序设计，那么他将会大大减少应用设计时软件开发的步骤。MFC（Microsoft Foundation Class，微软基础类库）和ATL（Active Template Library，活动模板库）对应用开发也是非常有用的。这些库大大简化了对COM+组件、图形用户界面、数据库接口的创建，以及其他方面的应用开发。有趣的是，ATL已经被集成进了MFC，从而允许开发者为GUI和小型高效的软件组件开发选择最佳设计方案。

下一步将讨论如何使用C++编译器和C#编译器，以及新型的Visual Studio.NET集成开发环境中的相关工具（Visual Studio.NET是Visual Studio 6.0的下一个版本）。虽然Visual Studio.NET的BETA 1版也许会和最终的版本有所不同，然而，软件开发环境不会影响Windows 2000操作系统的基本原理，所以本书开始的几章将讨论Windows 2000的基本原理，我们会从API调用和使用它们进行应用开发的角度对基本原理进行分析。Visual Studio.NET引入了另一种包装了Windows 2000操作系统的框架，即.NET框架。分析了Windows 2000之后，将研究这种新的框架。其实它就是一个位于应用和Windows 2000操作系统之间的中间件。

1.1 Windows 2000 操作系统体系结构

图1-1是Windows 2000操作系统的主要部件的结构框图。正如这个结构框图所展示的一样，应用是独立于操作系统的。操作系统代码运行在一种有特权的处理器模式下，叫做内核（kernel）模式，并允许访问系统数据和硬件。应用则运行在无特权的处理器模式下，叫做用户模式，只能通过一系列严格控制的API 有限地访问系统数据和硬件。Windows 2000操作系统的设计思想之一就是保持基本的操作系统尽可能地小型并高效。这样就只能允许那些在别的地方无法正确运行的功能保留在基本操作系统中。那些被排除在内核之外，而在一系列无特权的服务器上运行的功能，被称作受保护子系统（protected subsystems）。受保护子系统通过一系列功能丰富的API向应用提供传统的操作系统支持。

1.1.1 执行程序

Windows 2000操作系统的内核模式部分叫作执行程序（Executive），除了一个用户接口以外，它就是一个完整的操作系统。执行程序不能被系统管理员改变或重新编译。从本质上讲，执行程序是一个软件组件的集合，向受保护子系统提供基本的操作系统服务。从图1-1

可以看出，执行程序组件包括：

- I/O（输入/输出）管理器
- 对象管理器
- 安全参考监视器
- 进程管理器
- 本地过程调用设备
- 虚拟内存管理器
- 窗口管理器
- 图形设备接口
- 图形设备驱动程序

执行程序组件相互之间是完全独立的，它们通过精心控制的接口进行通信。这种模块化的设计允许删除已存在的组件，并可以用采用新技术特性的组件代替它。只要能够维持存在接口的完整性，操作系统就可以像修改组件前一样正确运行。执行程序的顶层叫作系统服务（System Services），它们是用户模式的受保护子系统和内核模式之间的接口。每个执行程序组件都有一系列API，这些向用户空间开放的API被统一称作“系统服务”，就像图1-1所表现的一样。后面将会看到，LPC（Local Procedure Call，本地过程调用）工具与这些系统服务API之间的通信是经由LPC操作处理的。

附录D列出了所有API的完整列表，但不包括它们的相关参数。附录D按照字母顺序，一共列出了1 939个Win32 API函数。这1 939个函数分布在95个Win32函数类别中。通过分析这些功能性的类别，你可以找到对自己有用的Win32 API。它还可以帮助你了解哪些执行程序组件能提供需要的功能，同时使我们更加容易理解，为什么说有一系列功能丰富的API可供使用。

1.1.2 受保护子系统

受保护子系统是用户模式的服务器，随着Windows 2000的启动而启动，它包括两种不同的类型：整体（integral）和环境（environment）。整体子系统是执行重要的操作系统功能的服务器，比如执行安全服务。而环境子系统则向不同操作系统环境下的应用提供支持，比如OS/2。Windows 2000装载了3种环境子系统：Win32子系统、POSIX子系统和OS/2子系统。

Win32（或者说32位Windows）子系统是Windows 2000的固有子系统，它向其应用提供了最多的功能和最高的效率，是新型软件开发的最佳选择。POSIX和OS/2子系统向其各自的应用提供兼容性环境，但是不能像Win32子系统一样拥有丰富的功能。

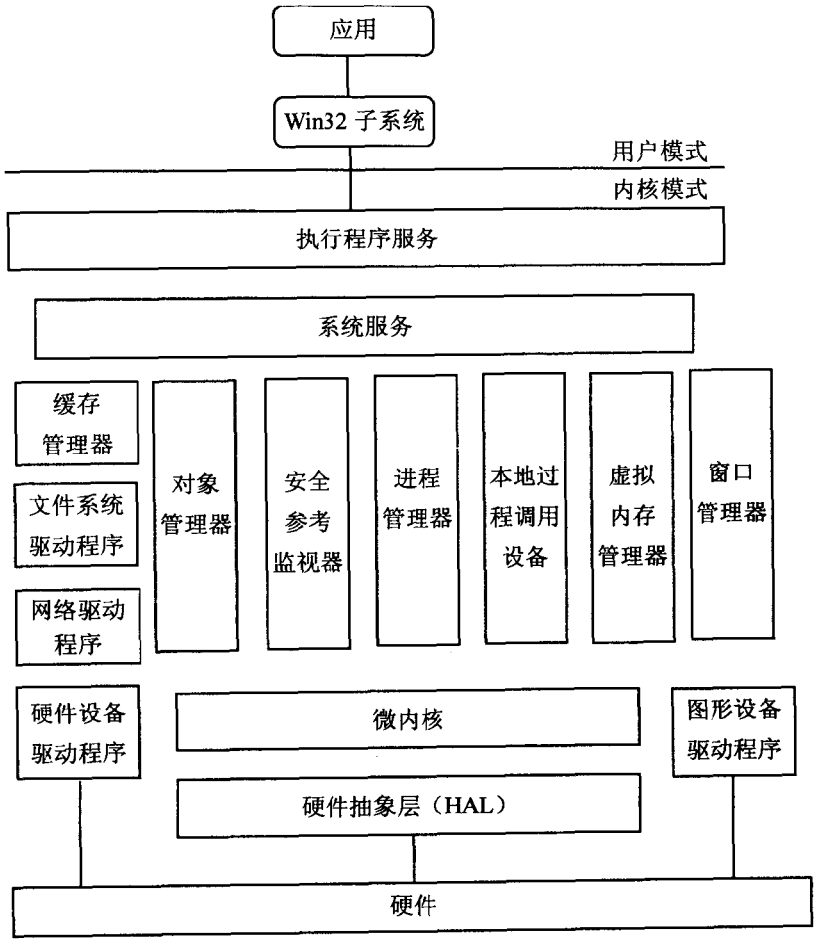

图 1-1　Windows 2000 功能结构图

1.1.3　本地过程调用设备

应用和环境子系统是一种客户机与服务器的关系。这就是说，客户机（一个应用）调用环境服务器(一个子系统)，请求某些类型的系统服务。考虑到它们之间的这种关系，Windows 2000为它们提供了一种通信机制。执行程序支持本地过程调用（Local Procedure Call，LPC）设备，以实现二者之间的消息传递功能。该设备的工作方式类似于用于网络处理的远程过程调用（Remote Procedure Call，RPC）设备。不同的是，LPC对运行在同一台计算机上的两个进程进行了最优化处理。

应用与环境子系统的通信是通过LPC传递消息来实现的。对客户机而言，消息传递进程通过以专用动态链接库（Dynamic-link Library，DLL）形式表现的函数存根（来自服务器环

境调用所使用的非执行占位符），把自己隐藏了起来。当一个应用对环境子系统进行API调用时，客户机（应用）进程的存根包装了调用的参数，并把它们传送给服务器（子系统）进程以执行调用，如图1-2所示。正是LPC设备允许存根过程向服务器进程传递数据并等待响应，Win32子系统中进程的工作方式就是一个例子。当调入一个Win32应用运行时，它就会链接一个包含有Win32 API中所有函数存根的DLL。应用调用一个Win32函数（在这个例子中，调用CreateWindow( )函数）时，调用是按照以下步骤进行的：

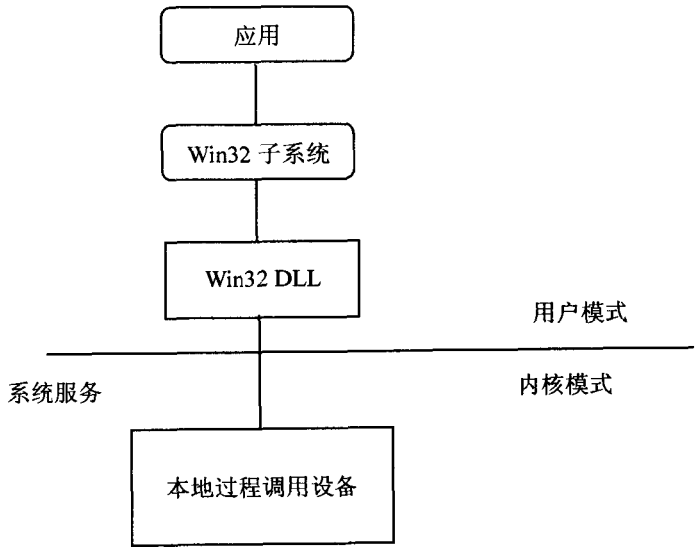

图 1-2 本地过程调用设备

（1）DLL中的CreateWindow( )存根函数被客户机的Win32应用调用。

（2）存根函数构造一个消息，包含了创建一个窗口所需要的所有数据，然后把消息传递给Win32服务器进程（就是Win32子系统）。

（3）Win32子系统接收到这个消息，并调用真正的CreateWindow( )函数，创建窗口。

（4）Win32子系统将包含了CreateWindow( )函数结果信息的消息传回DLL中的存根函数。

（5）存根函数将来自子系统的服务器消息解包，将结果返回客户机的Win32应用。

从应用的角度来看，好像是DLL中的CreateWindow( )函数创建了窗口。它并不知道工作实际上是由Win32服务器进程（Win32子系统）完成的，是消息的传递促使了这一切的发生，它甚至根本不知道Win32服务器进程的存在。应用更不会知道是子系统调用了一个或更多的执行程序的系统服务器，从而支持了它对函数的调用。这种功能性的关键在于向应用提供的操作系统环境，下一章将对其进行详细讨论。