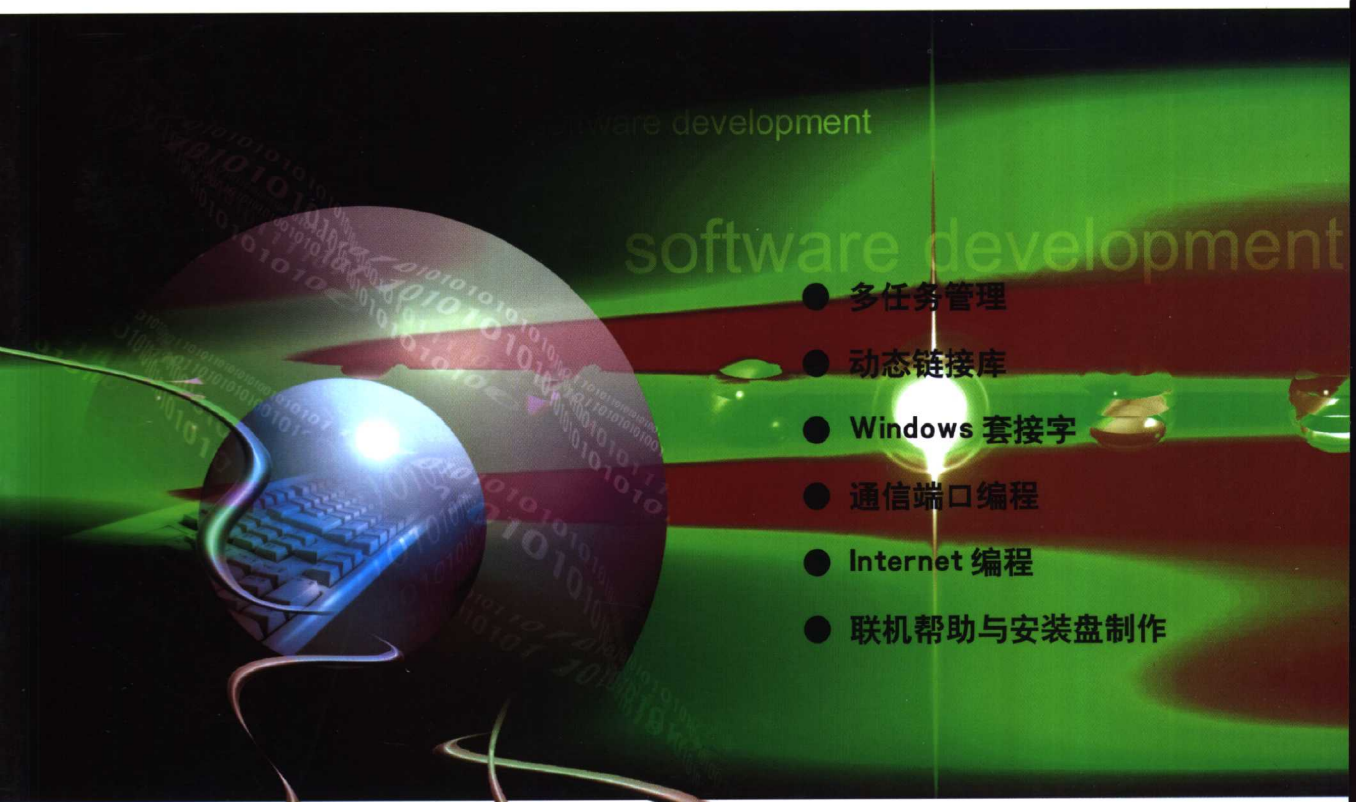
信息科学与技术丛书

程序设计系列

● 郎　锐　罗发根　编著

Visual C++ 网络通信程序开发指南



机械工业出版社
CHINA MACHINE PRESS

信息科学与技术丛书
程序设计系列

Visual C++网络通信程序开发指南

郎　锐　罗发根　编著

机械工业出版社

本书以 Visual C++ 开发环境为背景，对 Windows 套接字、邮槽、管道、MAPI、WinInet 以及通信端口等主要的网络通信编程技术作了较详细的介绍。为使读者能够系统地掌握 Visual C++ 对网络通信程序的开发，本书还对 Windows 编程基础、MFC 应用程序设计基础等基础知识作了简要介绍，同时也对与网络通信编程密切相关的技术如多任务管理、动态链接库和钩子等作了阐述。本书最后介绍的联机帮助和安装盘的制作方法，使读者能够开发出具有专业水准的网络通信程序。

本书可供各大专院校电子类专业及其相近专业师生、从事 IT 业的工程技术人员及所有编程爱好者参考使用。

图书在版编目（CIP）数据

Visual C++ 网络通信程序开发指南/郎锐，罗发根编著.—北京：机械工业出版社，2004.3
（信息科学与技术丛书　程序设计系列）
ISBN 7-111-13969-0

Ⅰ.V...　Ⅱ.①郎...②罗...　Ⅲ.C 语言—程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2004）第 008655 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策　　划：胡毓坚
责任编辑：孙　业
责任印制：施　红
三河市宏达印刷有限公司印刷·新华书店北京发行所发行
2004 年 3 月第 1 版·第 1 次印刷
787mm×1092mm 1/16 · 20.25 印张 · 501 千字
0 001—5 000 册
定价：30.00 元

出 版 说 明

随着信息科学与技术的迅速发展，人类每时每刻都会面对层出不穷的新技术、新概念。毫无疑问，在节奏越来越快的工作和生活中，人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书，来获取新知识和新技能，从而不断提高自身素质，紧跟信息化时代发展的步伐。

众所周知，在计算机硬件方面，高性价比的解决方案和新型技术的应用一直备受青睐；在软件技术方面，随着计算机软件的规模和复杂性与日俱增，软件技术受到不断挑战，人们一直在为寻求更先进的软件技术而奋斗不止。目前，计算机在社会生活中日益普及，随着因特网延伸到人类世界的层层面面，掌握计算机网络技术和理论已成为大众的文化需求。也正是由于信息科学与技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用，所以这些专业领域中的研究人员和工程技术人员将越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们对了解和掌握新知识、新技能的热切期待，以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状，机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络、工程应用等内容，注重理论与实践相结合，内容实用，层次分明，语言流畅，是信息科学与技术领域专业人员不可或缺的图书。

现今，信息科学与技术的发展可谓一日千里，机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作，为推进我国的信息化建设作出贡献。

机械工业出版社

前　言

网络通信编程是目前非常热门的一类编程技术，广泛应用于工程、科研、金融和教育等诸多领域。本书以 Microsoft Visual C++ 6.0 开发环境为背景，对 Windows 套接字、邮槽、管道、MAPI、WinInet 以及端口通信等主流的网络通信编程技术作了详细介绍。在结构编排上，本书从基本原理和相关概念入手，逐步对各种编程技术的概况、基本实现流程和具体编程细节作了比较清晰的描述，并根据各种编程技术之间的相互关系对各章节进行安排。

虽然本书是针对网络通信编程技术的，但由于在具体编程时往往需要用到与之联系密切的其他一些编程技术，所以从内容的完整性和连贯性方面考虑，本书还对多任务管理、动态链接库、内存管理和钩子等比较高级的编程技术作了较为深入的阐述。为了使读者能够系统地掌握 Visual C++ 对网络通信程序的开发，本书对 Windows 编程基础、MFC 应用程序设计基础等基础编程知识作了扼要的介绍。在本书最后讲述的联机帮助和安装盘的制作方法，使读者能够轻松开发出具有专业水准的网络通信程序。

阅读本书的读者应具有基本的计算机网络知识和 C++ 编程能力。本书各章节出现的所有程序源代码都可从机械工业出版社网站（http://www.cmpbook.com）中与本书相关网页上免费下载，这些示例代码在 Windows 2000 Professional 下由 Microsoft Visual C++ 6.0 调试、编译通过。

最后，要特别感谢我的父母郎益青先生和张连霞女士在我写作过程中所给予的全力支持与鼓励；感谢山东大学云昌钦先生、刘殿秀副教授和鲁能积成电子系统实验所（IESLab）梁成辉硕士在笔者求学时所给予的教诲；感谢中国电波传播研究所（CRIRP）赵振维研究员、原普研究员、郑名源研究员、姜凤荣高工、总装备部工程兵科研一所徐玉清高工、张国进高工等对笔者的支持与鼓励；对金燕波硕士、刘玉梅硕士等所有为本书写作提供过帮助的人士表示感谢！成书仓促，不妥之处在所难免，诚请读者提出宝贵意见。

郎　锐

目　录

第 1 章　Windows 编程基础

本章主要介绍 Windows 编程环境、Visual C++ 集成开发环境、SDK 编程技术等有关 Windows 程序开发的基本概念和基础理论知识，这些是学习本书后续内容的理论基础。

1.1　Windows 操作系统及编程环境

1.1.1　Windows 操作系统

自 Microsoft 于 1985 年推出第一个 Windows 系列产品 Windows 1.0 至今，Microsoft 公司已先后开发出几代不同的操作系统，并由此形成了一个单一、一致的操作系统家族——Microsoft Windows 操作系统家族。

Windows 系列的第一个产品 Windows 1.0 只是一个运行于 MS-DOS 下的具有简单多任务和图形化用户界面(GUI)的操作系统，虽然相比于 MS-DOS 是一个很大的突破，但是不论从哪个方面而言都是非常简陋的。1987 年推出的 Windows 2.0 已经开始在用户界面上对其进行改进，而且提供了供开发人员使用的软件开发工具包(Software Development Kit，SDK)，在很大程度上提高了操作系统的编程能力。真正显示出 Windows 操作系统强大潜力的是 1990 年发布的 Windows 3.0 操作系统，该版本主要对内存的管理方法进行了较大的改动，使之可以运行在 80386/486 的保护方式下，从而取得了巨大的成功。1992 年发布的 Windows 3.1 和 1993 年发布的 Windows for Workgroup 3.1(WFW 3.1)进一步增加了操作系统的功能，后者甚至将网络软件 Lan Manager 的部分功能嵌入到系统，使之可以作为部门计算机系统。Windows 3.x 提供了三种工作模式：实模式、标准模式和增强模式。其中，增强模式实际已经是一个 32 位保护模式的操作系统。

Microsoft 相继于 1993 年和 1995 年推出了 Windows NT 3.1 和 Windows NT 3.51 操作系统，Windows NT 是一种全新的操作系统，采用了强占式优先级多任务调度、客户/服务器计算、32 位以及多线程等先进技术。各个版本的 Windows NT 均分网络服务器版本(Windows NT Server)和工作站版本(Windows NT Workstation)，可以运行在 Intel x86、MIPS R4x00、DEC Alpha、PowerPC 等硬件平台上。在发展到 Windows NT 4.0 以后，通过 SP4 和 SP6 补丁程序的安装，使其安全可靠性得到了进一步的提升，成为企业和工程应用的首选操作系统之一。

Microsoft 在推出 Windows NT 操作系统的同时，还发布了 Win 32s 操作系统，该系统实际是对 Windows 3.1 的扩展，具备除多线程和优先级多任务调度外的 Windows NT 的大多数功能，因此也可以看作是 Windows NT 的一个子集。由于其他因素的影响，Win 32s 并没有得到推广。1995 年推出的 Windows 95 包括了 Win 32s 的大部分功能，同 Windows NT 一样也具备多线程和优先级多任务调度功能，尤其在 GUI 方面表现比较出色，但在网络管理方面仍有所欠缺。1997 年发布的 Windows 95 OEM 版本同 Windows 95 相比并没有太大的改进，直到

1998 年及其后续发布的 Windows 98 和 Windows 98 SE 才在稳定性以及用户友好特性等方面作了较大的改善，尤其是 Windows 98 SE 修正了 Windows 98 存在的一些设计缺陷和漏洞，因此运行更加稳定。2000 年发布的 Windows ME 是基于 Windows 9x 内核的最后一版消费型操作系统，虽然它已经将 Windows 9x 内核发挥到了极致，但是仍没有什么实质性的改进，只是在多媒体、易用性方面的改善表现比较突出。Microsoft 于同年发布的基于 Windows NT 架构的 Windows 2000，它是微软公司产品研发迄今为止投入最大的一个产品。它结合了 Windows 98 和 Windows NT 4.0 的很多优良的功能、性能于一身，远远超越了 Windows NT 的原来含义，它的出现被 IT 业界认为是“一个软件新世纪的开端”。Windows 2000 系列共有四个版本：Windows 2000 Professional，Windows 2000 Server，Windows 2000 Advanced Server，Windows 2000 Datacenter Server。其中 Windows 2000 Professional 是一个商业用户的桌面操作系统，也适合移动用户，是 Windows NT Workstation 4.0 的升级版。Windows 2000 Server 和 Advanced Server 则分别是 Windows NT Server 4.0 及其企业版的升级产品。Windows 2000 Datacenter Server 是一个新的品种，主要通过 OEM 的方式销售，是一个 64 位的产品，支持 8 个以上的 CPU 和 64GB 的内存，以及 4 个节点的集群服务。总的来说，Windows 2000 平台在系统稳定性、简单化以及对新设备的支持等方面均突破了以往的设计思想。

除此之外，Microsoft 为了满足小型硬件设备的需要还开发了一种规模远小于 Windows 2000 和 Windows 98 的操作系统 Windows CE，它的功能仍比较强大，主要面向个人用户。2001 年 Microsoft 发布了最新的 32 位操作系统 Windows XP 也是基于 NT 技术的。相比而言，Windows XP 更健壮、更稳定，而且在外观、人机交互功能、多媒体等方面均有不俗的表现。

目前，Microsoft 正在开发新一代的 Windows 操作系统，使其成为真正 64 位的 Windows 操作系统。这种 64 位的操作系统显然将在运行性能上得到大幅度的提高，而且仍然可以使用以前的 Win32 API。为向下兼容，64 位的 Windows 仍可以运行 32 位的应用程序，但要确保数据模型变更的影响只涉及到指针数据类型，并且要定义一些新的数据类型以调整与指针有关的数据的长度。

1.1.2 Windows 的编程环境

Windows 操作系统无论是从界面上还是从操作上都是非常人性化的，而且人机界面尽可能的友好，但是这只是针对普通用户而言。对于那些在 Windows 操作平台上进行程序设计的开发人员，进行 Windows 编程则是一件非常困难的事情，因为他们面对的将是数以百计的 Win32 API 函数、面向对象的模式、数百种对象和消息以及众多窗口的风格等等。由此可见，进行 Windows 程序设计时一个好的程序开发环境将起到至关重要的作用。

最基本的 Windows 编程是 SDK-C 编程，即在 SDK 的开发环境中使用 C 语言进行程序设计，开发人员要完全通过自己编码来实现和构造整个应用程序的结构和所有的界面元素，因此这种 SDK-C 的编程方式无论是从难度还是从编程工作量上看都是比较大的。Borland 公司和 Microsoft 公司为 Windows 程序设计开发出了“所见即所得”的可视化编程工具 Delphi 和 Visual Basic，虽然这两种编程环境极大降低了编程难度，但由于过分强调界面的美观和编程难度的降低，使得对系统内核的编程效果并不是太好。

为了既能降低编程难度，又可以拥有对系统强大的编程控制功能，Microsoft 开发出了基本类库 MFC(Microsoft Foundation Class)，其中包含了主要的 SDK 函数和结构等。为了能使

程序开发人员更好地把精力放在对算法的控制上，而不被界面等程序框架的创建所影响，Microsoft 进一步开发出了 Microsoft Visual C++ 开发环境，设计人员可以通过 VisualWorkbench、AppStudio、AppWizard 以及 ClassWizard 等可视化向导来自动生成所需要的程序框架和其他一些程序元素，大大提高了编程效率和程序的可靠性。本书即以 Microsoft Visual C++ 6.0 为开发工具在 Windows 环境下进行网络编程。

1.1.3 Microsoft Visual C++ 6.0 集成开发环境

前面已经简单提过 Microsoft Visual C++ 6.0 同其他编程工具相比，Visual C++ 在提供可视化编程方法的同时，也适用于编写直接对系统进行底层操作的程序。尤其是 MFC 对底层 API 函数进行了彻底的封装，在程序设计时可以完全用面向对象的方法来进行，极大提高了程序的编码效率。图 1-1 给出了 Visual C++ 进行 Windows 编程时编辑、编译与链接的过程示意，从中可以比较清楚地了解代码和资源从编辑、编译到链接产生可执行应用程序的全过程。

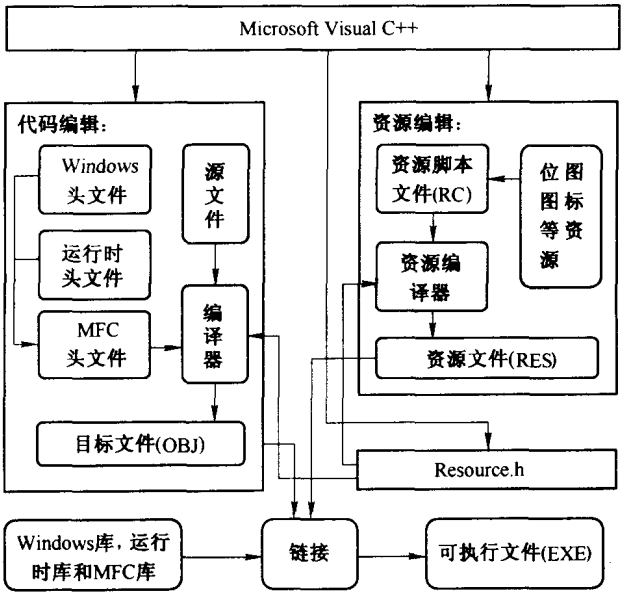


图 1-1 Visual C++ 应用程序编译、创建过程示意

Visual C++ 6.0 的集成开发环境(Integrated Developing Environment，IDE)被称作 Developer Studio，从中可以创建工程、源文件、各种资源以及其他一些文档。对项目的组织是通过工作空间(Workspace)和工程(Project)来完成的，一个工作空间可以包含一个或多个相关的工程。在操作界面上，通过 Workspace 浮动窗口上的三个选项卡 ClassView、ResourceView 和 FileView 等可分别用不同的方式来管理、操作工作空间中的各个工程及其内部文档和资源见图 1-2。

在用 Visual C++ 6.0 集成开发环境去创建一个新的工作空间时，可通过“File”菜单下的“New...”来完成，这时将弹出如图 1-3 所示的“New”对话框，可以通过“Workspace”选项卡来创建一个新的工作空间，通过“Project”选项卡来创建新的工程。“Project”选项卡所列举出来的各个工程项目是针对不同目的的项目开发而设置的，在使用时应根据需求灵活选择创建相应的工程项目。下面是这些项目的功能说明。

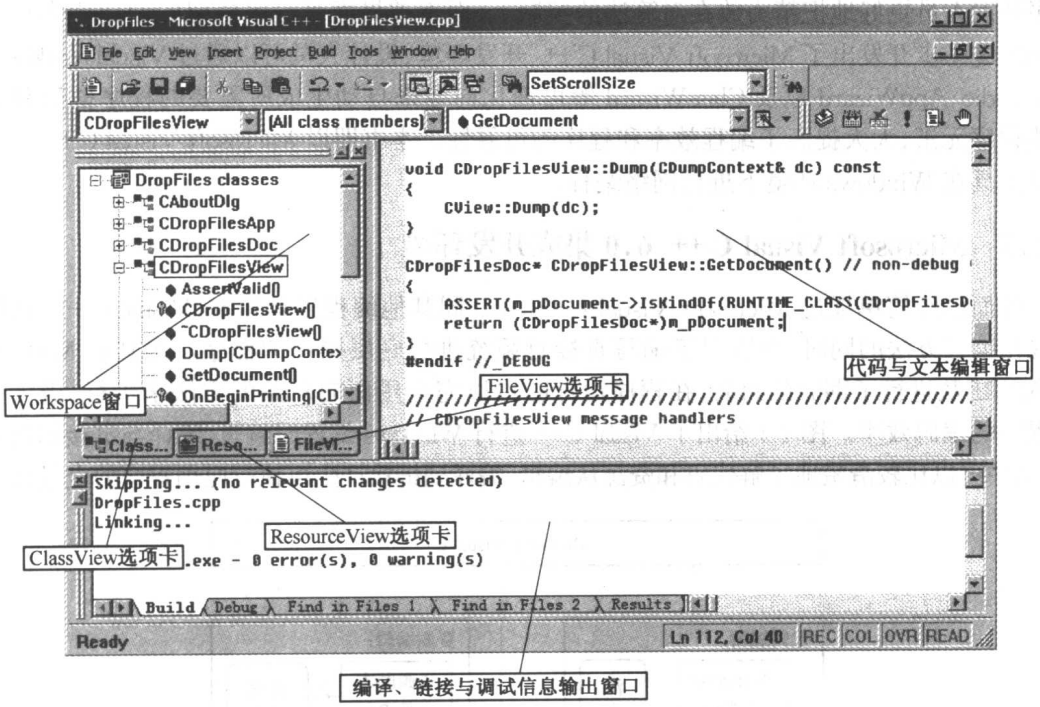


图 1-2　Visual C++ 的集成开发环境

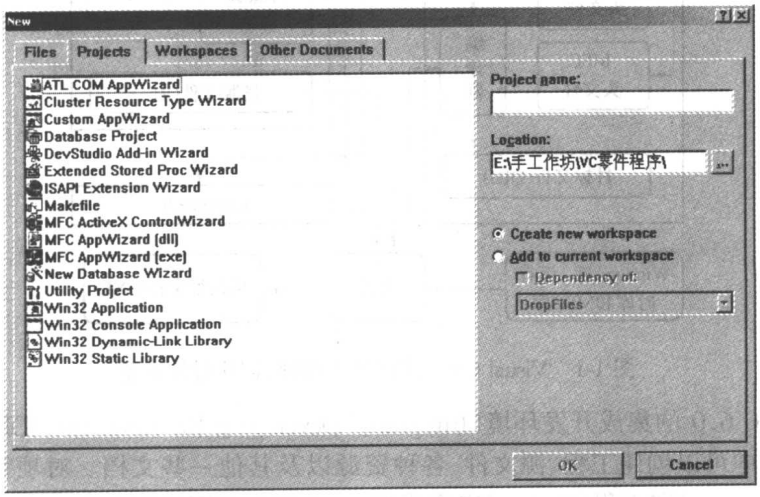


图 1-3　Visual C++ 6.0 的工程向导

- ATL COM AppWizard：该向导用于创建不具有任何初始 COM 对象的 ATL（Active Template Library，活动模板库）工程。所建立的工程一般用来编写小的 ActiveX 控件，主要供熟练掌握 ActiveX 控件编写方法的开发人员使用。
- Cluster Resource Type Wizard：该向导用于创建一个 Windows NT 下的簇群管理器扩展 DLL 框架工程。
- Custom AppWizard：开发人员可以通过该向导插入自己的模板，将自己常用的工程模板通过该向导插入后，将同其他标准工程模板一样出现在列表框中。这种通用向导框架

在大规模程序设计中将有效提高程序的编写效率。

- Database Project:该向导将用于生成一个数据库工程。
- DevStudio Add-in Wizard:该向导将创建一个 Developer Studio Add-in 的框架工程。
- Extended Stored Proc Wizard:该向导用于创建一个 SQL 服务器扩展存储程序。
- ISAPI Extension Wizard :该向导主要用于简化 Internet Server 扩展和过滤器的创建工作。扩展器是由 ISAPI(Internet Server API)开发出的由用户通过 Web 页激活的 DLL,而过滤器则是由 ISAPI 开发的与 HTTP 服务器同时运行的 DLL,负责查看或改变服务器上往来的数据。
- Makefile:该向导主要供那些用可独立应用的工具来取代一部分 Developer Studio 的开发人员使用,用以创建一个 makefile 工程。
- MFC ActiveX ControlWizard :该向导负责创建一个基于 MFC 的 ActiveX 控件框架工程。
- MFC AppWizard(dll):该向导用于创建一个可使用 MFC 类的动态链接库。
- MFC AppWizard(exe):该向导用于生成普通的单文档、多文档或对话框模式的应用程序。
- New Database Wizard:主要供安装了 Visual InterDev 的开发人员使用,通过此向导可以简化 Web 页到 SQL 数据库的连接。
- Utility Project:该向导将创建一个空的工程。
- Win32 Application:用于生成不需要 MFC 支持的空的工程,全部由开发人员负责编写、维护程序代码和资源,一般多通过该向导编写一些 SDK 程序。
- Win32 Console Application:该向导用于生成一个简单的控制台应用程序工程。
- Win32 Dynamic-Link Library:该向导将创建一个不需要模板和 MFC 支持的空的动态链接库工程。
- Win32 Static Library:该向导用于创建静态库工程。

在由向导创建出工程后,由 Visual C++ 6.0 生成的程序源码除了通常的扩展名为 h 和 cpp 的头文件和 C++ 源文件外,通常还存在一些其他格式的文件,具体功能说明如表 1-1 所示。

表 1-1 Visual C++ 工程文件说明

文件扩展名	功 能 说 明
APS	该文件内容供 ResourceView 使用
BSC	浏览信息文件
CLW	该文件内容供 ClassWizard 使用
DEP	从属文件
DSP	工程文件(禁止删除或用编辑软件对其进行编辑)
DSW	工作空间文件(禁止删除或用编辑软件对其进行编辑)
MAK	外部产生文件
NCB	该文件内容供 ClassView 使用
OPT	该文件保存有工作空间的配置信息
PLG	编译日志文件

1.2 认识 Windows 环境框架

1.2.1 Windows 系统结构

Windows 操作系统是由众多模块组成的一个综合体，这些模块包含可执行应用程序、设备驱动程序以及动态链接库等多种形式的数据。这些模块根据各自的功能，可以划分为若干个层次，图 1-4 对 Windows 的系统结构做了较为清晰的展示。从 Windows 系统结构图可以看出，Windows 操作系统是非常容易进行扩充升级的，而且 Windows 的大部分模块采用的是动态链接库形式，使升级变得更加简单方便。在开发 Windows 应用程序时，是通过对 Windows API 函数的调用实现对 Windows 系统的编程的。虽然这些 API 函数是由随系统启动而被载入内存的几个不同模块所提供的，但是在编写程序时并不需要知道哪个 API 函数存在于哪一个模块，更不必知道在模块内部的工作模式，而只需关心开发者所明确涉及的模块。可见，这种分层结构所带来的独立性大大降低了控制系统和编程的复杂性。

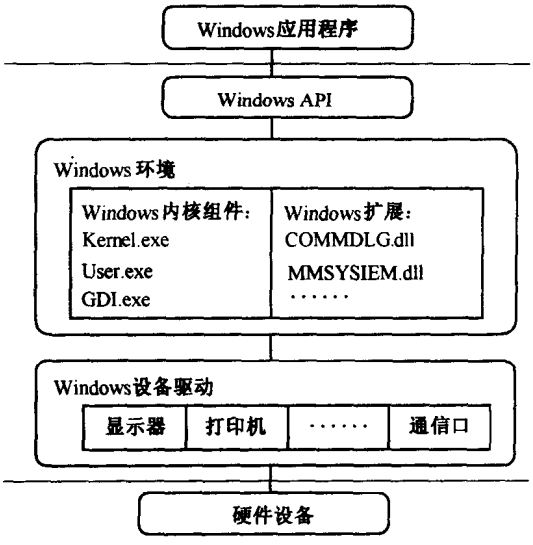


图 1-4 Windows 系统结构示意

1.2.2 虚拟机与虚拟设备驱动程序

虚拟机是 Windows 操作系统的一个重要概念，是包括应用程序、支持软件、内存以及 CPU 寄存器所组成的可执行任务，由 CPU 直接支持。Windows 的核心模块和所有的应用程序都是在一个称为系统 VM 的虚拟机上运行的。在 386 增强模式下 CPU 可以运行多个虚拟 8086 模式(V86)应用程序，这些应用程序虽然是运行于同一个 CPU 上的多个并发任务，但效果却象是在各自的计算机上运行。CPU 把这些 V86 模式的任务分别放入其自己的虚拟机(VM)中。这些虚拟机由虚拟机管理器(VMM)负责管理和调度。

虚拟设备是那些可以使用但却并不真实存在的设备，通常多以设备驱动程序(VxD)来实现。这些 VxD 实际是一 32 位保护模式的动态链接库，对某种硬件设备或系统资源进行管理，使其可以同时为多个应用程序所共用。虚拟机管理器为 VxD 提供了许多重要的服务，并且通过使用 CPU 的内存调页能力来为系统虚拟机创建一个 32 位的虚地址空间。

从图 1-5 可以大致看出虚拟机、虚拟机管理器以及虚拟设备驱动程序之间的组织关系，每一个 V86 模式任务分别在自己的虚拟机中运行，物理内存的分配也是先 System VM 然后依次是 VM2、VM3……。Windows 操作系统是一种保护模式的操作系统，共分 Ring0、Ring1、Ring2 和 Ring3 四个级别，通常的应用程序都是运行在级别最低的 Ring3 层上的，而作为系统核心组成部分的虚拟设备驱动程序和虚拟机管理器则处于最高保护级别的 Ring0 层，除有特

殊要求，一般情况下 Windows 操作系统是不允许对 Ring0 层进行操作处理的。

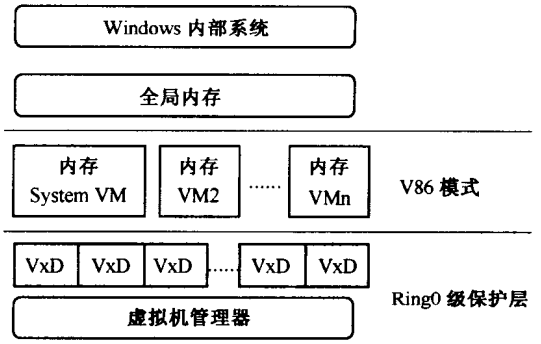


图 1-5　虚拟机、虚拟机管理器与虚拟设备驱动程序关系示意

1.2.3　多任务管理

前面提到，在 Windows 环境中可以在多个虚拟机上同时运行多个程序，这些程序按照分时方式进行多任务操作，每一个虚拟机都被赋予一定的 CPU 时间片，并在此分配的时间片内使用 CPU。这种有别于 DOS 独占运行的方式为 Windows 应用程序的设计带来了一定的限制。为了实现对 Windows 应用多任务的管理，每个应用程序必须至少具备一个能被任务管理程序调用的窗口函数，而且为了能将控制交还给任务管理程序，还需要具备消息循环功能。虽然没有上述处理对该程序自身的运行影响并不是很大，但却破坏了整个系统的多任务管理机制。Windows 所采用的多任务是一种合作的多任务方法，主要依靠应用程序自己来交替分享 CPU。

1.2.4　窗口与消息

窗口是 Windows 操作系统中一种非常基本也是非常重要的对象。狭义上讲，窗口是指能进行图形处理的视觉上可见的窗口；而从广义上讲，窗口包括可见或不可见的所有能进行消息处理的单元。而实际上，应用程序创建的这些可接收、处理消息的不可见窗口除了在视觉上不可见外，在消息处理等方面同可视窗口是完全一致的。窗口对象的数据是系统为每个窗口和每个窗口类保存的信息，由 Windows 发送给窗口的消息由 Windows 调用的窗口函数负责处理。虽然 Windows 应用程序的实质性工作都是在窗口函数中完成的，但是却看不到主程序对各窗口函数的显式调用，这是由于 Windows 操作系统采用了消息驱动机制。虽然借助于 Visual C++ 的集成开发环境可以在不了解 Windows 内部运行机制的情况下也能进行程序开发，但是如果要进行更加深入的编程就非常有必要对 Windows 操作系统的内部运行机制尤其是要对 Windows 的消息驱动机制有一个清楚的认识。

消息是消息驱动机制的核心，是报告某件事情发生的通知。Windows 系统下消息的产生有多种途径，可以将其归结为输入消息、控制消息、系统消息与用户消息等四大类。为接收消息，Windows 在应用程序执行后将为其创建一个消息队列，此消息队列用来存放发送给程序各窗口的消息，见图 1-6。这些消息在发送过程中是通过结构 MSG 来对其各个元素进行组织的，该结构定义如下：

```
typedef struct tagMSG{
```

```
    HWND hwnd;
    UINT message;
    WPARAM wParam;
    LPARAM lParam;
    DWORD time;
    POINT pt;
} MSG;
```

成员 hwnd 为表示接收消息的窗口句柄；message 为发送的消息号；wParam 和 lParam 为消息参数，具体意义同发送的消息有关；time 和 pt 分别为发送消息时的时间和光标位置。可见，该结构比较全面地对消息进行了表述。

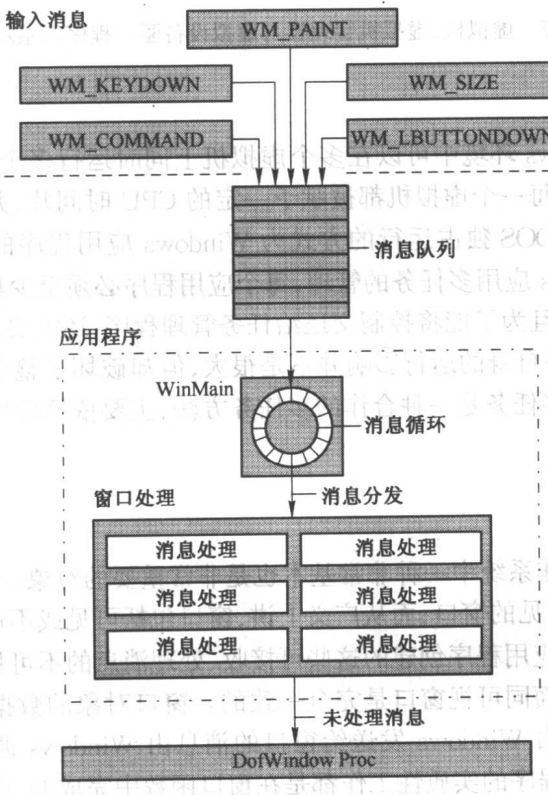


图 1-6　Windows 的消息驱动机制

将消息传送到应用程序有两种具体的方法，即使用 SendMessage()函数的发送消息方法和使用 PostMessage()函数的邮寄消息方法。这两种消息传递方式的目的都是一致的，都是要把消息传递给应用程序，但是这两种消息传递方式在使用时仍存在一些细微差别的。发送消息时，系统不经由消息队列而直接调用窗口函数，并立即将消息发送给应用程序的窗口函数。这种通过发送方式进行传输的消息是一种“不进队消息”，应用程序直到窗口进程为调用函数返回一个结果后才得以继续执行。邮寄消息则先把消息放置到应用程序的消息队列中存储起来，应用程序将在空闲的时侯搜寻消息队列，如果在消息队列中搜寻到了消息，将负责把消息发送到既定窗口，同时在消息队列中将其删除。这种邮寄消息的方式在把消息放入到消息队列后就即刻返回，但是返回值只能说明消息是否邮寄成功，而并不能表示被调用窗口进程的结

果。由邮寄方式发出的消息属于“进队消息”,通常多为由用户输入而产生的消息。

1.2.5 句柄

句柄同消息一样,也是 Windows 编程中的一个重要概念。句柄是 Windows 使用的用于标识应用程序一个对象的一种无重复整数,也可以将其看作是赋予给对象的惟一名称,在给一个对象赋予句柄之后,就可以通过此句柄来完成对该对象的引用了。这些对象包括模块、任务、实例、文件、内存块、菜单、控制、字体、各种资源以及各种 GDI 对象。Windows 为这些对象分配确定的句柄,而不是通过物理内存地址来对其进行标识。在将句柄返回给创建该项目的应用程序后,应用程序就可以通过 API 函数对句柄对应的 Windows 对象进行处理。

1.2.6 资源

Windows 编程的另一个主要特点是在程序设计过程中大量使用了资源。资源主要包括图标、位图、对话框、工具条以及菜单等等,这些资源集中在一个资源文件中定义,这样的组织形式可以使开发人员能以更加结构化的方式进行编程。在处理资源时,一般是把它当作一个库来对待,即遵循装入、使用、放弃的处理流程。资源在装载时可以有两种加载方式:加载到本地堆和加载到全局内存。不同资源的加载函数也是不一样的,API 提供了一些资源加载函数如 LoadIcon()、LoadString()和 LoadBitmap()等,来加载图标、字符串和位图等资源,具体加载哪一个资源是通过一个与之对应的全局惟一的资源 ID 号来指定的。

1.2.7 内存管理

Windows 在内存管理上也提供了比较丰富的处理方法,由内存管理程序控制着系统的所有可用内存,同时还提供一些用以进行内存管理的 API 函数来进行内存块的分配与释放等工作。

从应用程序的角度,内存管理程序将内存分成局部堆和全局堆两种。局部堆是局限于一个默认数据段中的内存,大小不超过一个物理段(64 KB),在其内分配的内存可以使用近指针寻址。相比而言,全局堆的功能要强大许多,而且全局堆也是 Windows 内存管理模式的一个最重要的特性:允许为应用程序分配多个内存块,包含所有可用的内存,因此使用全局堆进行信息的存放可以处理局部堆所无法处理的大容量数据,而不必再使用大内存模式。除了在处理大数据量时必须使用全局堆外,如果需要在两个模块中共享某些数据也必须在内存管理上采取全局堆的方式进行管理。

虽然全局堆在使用时要比局部堆功能强大,但是以下几点是必须要注意的:首先,对全局堆的访问总是用远指针来寻址的;其次,一定要考虑到全局内存位置是可以随时改动的,而不能认为它会始终占据某段固定的内存。除此之外,不论是局部堆还是全局堆,都是应用程序保留的可用于内存分配的一块区域,在使用时必须确保内存已分配成功,而且在使用完毕后应及时将在堆上分配的内存释放掉。

1.2.8 图形设备接口

图形设备接口(Graphics Device Interface,GDI)是通向 Windows 可视界面的入口,它提供一套非常丰富的、面向图形的函数库,通过该函数库所提供的接口函数可以实现对窗口用户区