21世纪计算机系列教材

计算机网络实验指导

杨金生　等编著

JISUANJI WANGLUO SHIYAN ZHIDAO

上海交通大学出版社

21世纪计算机系列教材

计算机网络实验指导

杨金生　等编著

上海交通大学出版社

内 容 简 介

本书是配合学习计算机网络原理课程的实验指导书，共有九个实验：实验 1，聊天程序；实验 2，网络文件传输；实验 3，FTP 协议的实现；实验 4，跟踪路由；实验 5，广播通信技术；实验 6，多播通信技术；实验 7，在网络中传输图像；实验 8，串口通信；实验 9，连续 ARQ 协议。

全书九个实验循序渐进，对设备要求不高，很多实验在单机上模拟网络环境就可以完成，有助于学生在学习计算机网络基本原理的基础上掌握实际的开发能力。

本书可作为高等院校相关专业讲授《计算机网络》时的实验指导书，也可作为从事网络编程开发人员的参考用书，同时也适合读者自学。

图书在版编目(CIP)数据

计算机网络实验指导/杨金生等编著.
—上海:上海交通大学出版社,2002.9
21 世纪计算机系列教材
ISBN7-313-03164-5
Ⅰ.计... Ⅱ.杨... Ⅲ.计算机网络
实验—高等学校—教材
Ⅳ.TP393-33
中国版本图书馆 CIP 数据核字(2002)第 070463 号

计算机网络实验指导
杨金生 等编著
上海交通大学出版社出版发行
(上海市番禺路 877 号 邮政编码 200030)
电话:64071208 出版人:张天蔚
常熟市华通印刷有限公司 印刷 全国新华书店经销
开本:787mm×1092mm 1/16 印张:11.75 字数:285 千字
2002 年 10 月第 1 版 2002 年 10 月第 1 次印刷
印数:1～2 050
ISBN7-313-03164-5/TP·523 定价:20.00 元

序

计算机技术的发展推动了经济的发展、社会的发展。如果说 1946 年诞生的电子数字计算机显示出它的速度和记忆能力的优势，那么 1958 年出现的高级语言则把计算机从少数熟悉机器语言的专业人员那里解放出来，交给了广大的计算机用户；而 1971 年开始经历了近 20 年发展的个人计算机，则将计算机从企业、单位等少数拥有者那里推向了每个家庭、个人；始于 1969 年发展于 20 世纪 90 年代的计算机网络，它的互联性、开放性和共享性得到了淋漓尽致的发挥。坐在终端前的你，就可以查到想要的信息，可以办好的事愈来愈多，仿佛整个世界就在你的眼前。计算机改变了世界。当今的社会也好像受摩尔定律的支配一样，每 18 个月翻一番，形成了知识爆炸的年代，新经济的年代。

社会的发展回过头来对计算机教育提出了更新的要求。1991 年 ACM 和 IEEE 所提出的计算 91 教程，计算机网络还没有成为一个单独的知识领域，而 2001 年所推出的 2001 教程，出现了网络及其计算的知识领域，因此计算机网络的课程建设意义非同一般；计算机学科是门实践性很强的学科，提升学生的动手实践能力，是计算机课程建设的重要课题；这就需要有教师致力于这个项目的建设，使教学实践活动提高到一个新的水平上。

上海交通大学计算机系杨金生副教授总结了他多年的教学经验，设计了计算机网络具有代表性的九个实验，编写的同时也考虑到各个院校的不同实验环境，大部分实验在单机上模拟网络环境就可以完成。他以规范的形式撰写了实验指导书，规定了实验目的、实验要求，详细阐述了实验原理和实验步骤，并给出了思考题。在上海交通大学出版社的支持下，《计算机网络实验指导》一书将付梓出版，这是有利于教学，有利于学生的好事，是值得推荐的事。

侯文永

2002 年 5 月

前　言

现今，学生中对计算机网络的学习热情很高，但其动手实践能力却比较差，究其原因，是因为缺少一本规范的循序渐进的实验指导书，而《计算机网络》这门课程仅讲述理论知识，同学们在学习时会觉得抽象和难以理解。本书编写的目的就是为了让学生亲自动手实现一个网络程序，体会一下基于网络的程序是如何工作的，各种通信方式的差别，IP 地址和端口号的真正含义，协议中数据帧的格式如何用程序语言表达等。通过这些实验，进一步理解计算机网络的基本原理。

全书共有九个实验：实验 1，聊天程序；实验 2，网络文件传输；实验 3，FTP 协议的实现；实验 4，跟踪路由；实验 5，广播通信技术；实验 6，多播通信技术；实验 7，在网络中传输图像；实验 8，串口通信；实验 9，连续 ARQ 协议。

Microsoft 公司的 Visual C++ MFC 对 Windows Socket API 的封装使我们能轻松地实现网络编程。当然，即便如此，学习网络编程也不是一件容易的事情，它需要多方面的知识。因此，读者在学习本书之前应该已具备了网络和 C++的基本知识。

本书既适合教学之用，又适合于自学，对设备要求不高，很多实验在单机上模拟网络环境就可以完成，有助于学生在学习计算机网络基本原理的基础上，掌握实际的开发能力。实验步骤给出每一个具体的编程及操作方法，边讲原理边分析程序，讲练结合，具有典型性、实用性和指导性的特点，不但可作为大专院校相关专业讲授《计算机网络》时的实验指导书，同时也可作为从事网络编程开发人员的参考用书。

参加本书编写的有：杨金生、黄毅峰、刘敦、魏珂、林显春、许春、杨斯奇等。

由于作者（yang-js@cs.sjtu.edu.cn）水平有限，书中难免存在错误和疏漏之处，恳请广大同行和读者提出批评和建议。

编　者

2002 年 5 月

实 验 环 境

实验 1~实验 7:

硬件环境：586 以上的兼容机，装有 TCP/IP 协议的单机、局域网或 Internet 网。
操作系统：Windows 98 或以上版本。
软　　件：Visual C++

实验八:

硬件环境：586 以上的兼容机两台，通过 RS232C 接口连接，接线方法见实验 9 原理。
操作系统：Windows 98 或以上版本。
软　　件：Visual C++

实验九:

硬件环境：586 以上的兼容机两台，通过 RS232C 接口连接，接线方法见实验 9 原理。
操作系统：Windows 98 或以上版本。
软　　件：Turbo C

注意：RS232C 接口的驱动芯片很容易烧毁，不可接错或带电插拔。

说明：书中的程序代码可从上海交通大学出版社网站“资料下载”栏目中下载。
http://www.jiaodapress.com.cn
http://press.sjtu.edu.cn

目　录

实验 1　聊天程序

【实验目的】

(1) 熟悉 Visual C++的基本操作。
(2) 基本了解基于对话框的 windows 应用程序的编写过程。
(3) 对于 Windows Socket 编程建立初步概念。

【实验要求】

(1) 应用 Visual C++中 MFC CSocket 类，实现网络数据传输。
(2) 仿照本实验步骤，制作实用的局域网一对一聊天程序。

【实验原理】

一、Windows Socket 和套接口的基本概念

网际协议（Internet Protocol，IP）是一种用于互联网的网络协议，已广为人知。它可广泛用于大多数计算机操作系统上，也可用于大多数局域网 LAN（比如办公室小型网络）和广域网 WAN（比如说互联网）。从它的设计看来，IP 是一个无连接的协议，并不能保证数据投递万无一失。两个上层协议（TCP 和 UDP）依赖 IP 协议进行数据通信。

如果希望在 Microsoft Windows 下通过 TCP 和 UDP 协议建立网络应用程序，则需要使用 Winsock 套接口编程技术。

套接口，就是一个指向传输提供者的句柄。Win32 中，套接口不同于文件描述符，所以它是一个独立的类型——SOCKET。Windows Sockets 描述定义了一个 Microsoft Windows 的网络编程界面，它是从 Unix Socket 的基础上发展而来的，为 Windows TCP/IP 提供了一个 BSD 型的套接字规范，除与 4.3BSD Unix Sockets 完全兼容外，还包括一个扩充文件，通过一组附加的 API 实现 Windows 式(即事件驱动)的编程风格；而 Winsock 则是在 Microsoft Windows 中进行网络应用程序设计的接口。Windows 在 Internet 支配域中的 TCP /IP 协议定义了 Winsock 网络编程规范，融入了许多新特点。使用 Socket 的目的是使用户在网络协议上工作而不必对该网络协议有非常深入的了解。此外，编写的程序还可被迅速地移植到任何支持 Socket 的网络系统中去。

Winsock 提供了一种可为指定传输协议打开、计算和关闭会话的能力。在 Windows 下，TCP/IP 上层模型在很大程度上与用户的 Winsock 应用有关；换言之，用户的 Winsock 应用控制了会话的方方面面，必要时，还会根据程序的需要格式化数据。

套接口有三种类型：流式套接口、数据报套接口及原始套接口。

流式套接口定义了一种可靠的面向连接的服务（利用 TCP 协议），实现了无差错无重复的顺序数据传输。数据报套接口定义了一种无连接的服务（UDP 协议），数据通过相互独立

的报文进行传输，是无序的，并且不保证可靠和无差错。原始套接口允许对低层协议如 IP 或 ICMP 直接访问，主要用于新的网络协议实现的测试等。

面向连接服务器处理的请求往往比较复杂，不是一来一去的请求应答所能解决的，而且往往是并发服务器。使用面向连接的套接口编程，可以通过图 1.1 来表示。

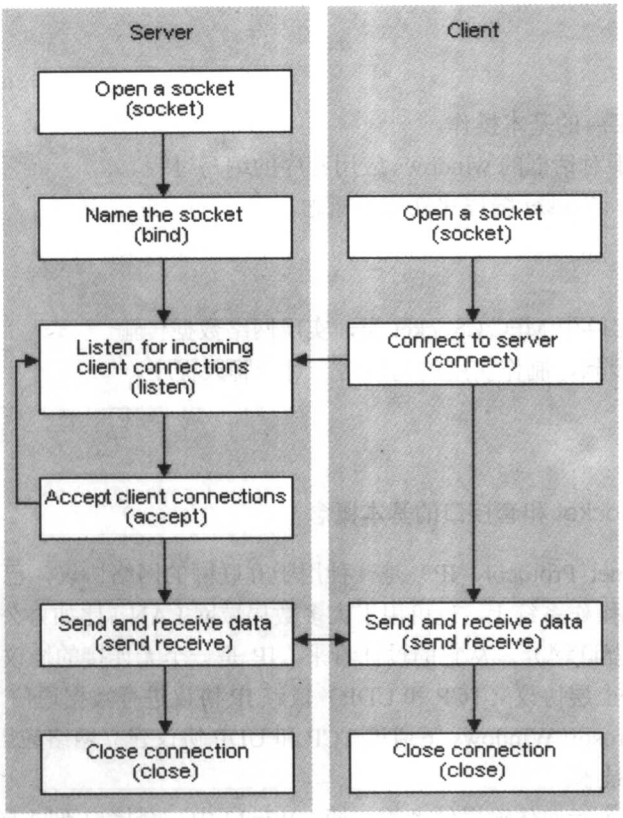

图 1.1

无连接服务器一般都是面向事务处理的，一个请求、一个应答就完成了客户程序与服务程序之间的相互作用。若使用无连接的套接口编程，程序的流程可以用图 1.2 表示。

原始套接口的介绍和使用将放在实验 4 中讲解。

套接口工作过程如下：服务器首先启动，通过调用 socket()建立一个套接口，然后调用 bind()将该套接口和本地网络地址联系在一起，再调用 listen()使套接口做好侦听的准备，并规定它的请求队列的长度，之后，调用 accept()来接收连接。客户在建立套接口后就可调用 connect()和服务器建立连接。连接一旦建立，客户机和服务器之间就可以通过调用 read()和 write()来发送和接收数据。最后，待数据传送结束后，双方调用 close()关闭套接口。

在网络编程中，掌握端口的概念十分重要。

端口：基于 TCP/IP 协议的网络中，计算机都分配有一个 IP 地址，用一个 32 位二进制数来表示，正式的称呼是“Ipv4 地址”。客户机需要通过 TCP 或 UDP 和服务器通信时，必须指定服务器的 IP 地址和服务端口号。另外，服务器打算侦听接入客户机请求时，也必须指定

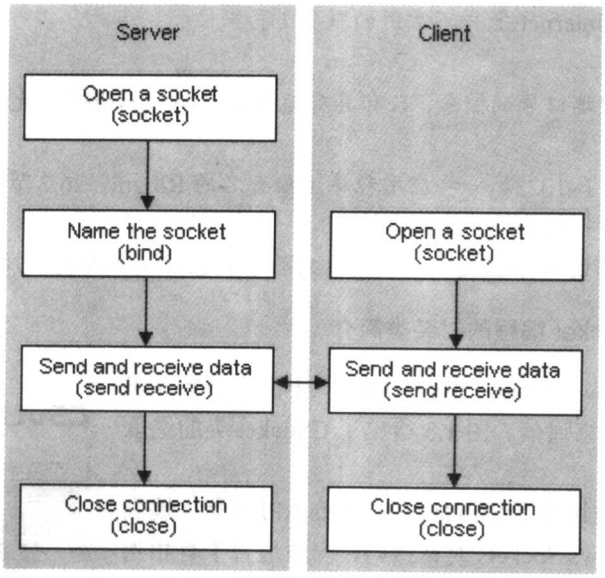

图 1.2

一个 IP 地址和一个端口号。在选择端口时，应特别小心，因为有些可用端口号是为“已知的”（即固定的）服务保留的，如文件传输协议和超文本传输协议，即 FTP（21 号端口）和 HTTP（一般为 8080 端口）。“已知的协议”，即固定协议，采用的端口由“互联网编号分配认证（IANA）”控制和分配，RFC 1700 中说明的编号。

从本质上说，端口号可分成 3 类：“已知”端口、已注册端口、动态和（或）私用端口。

⑴ “已知”端口 0~1023，由 IANA 控制，是在 UNIX 中为固定服务保留的。

⑵ 已注册的端口 1024~49151，由 IANA 列出来的，供普通用户的普通用户进程或程序使用。

⑶ 动态和（或）私用端口 49152~65535。

普通用户应用应选择 1024~49151 之间的已注册端口，从而避免端口号已被另一个应用或系统服务所用。此外，49152~65535 间的端口可自由使用，因为 IANA 这些端口上没有注册服务。

到 1992 年为止，人们所熟知的端口号介于 1~255，而 256~1023 的端口号通常由 Unix 系统占用，以提供一些特定的 Unix 服务。现在 IANA 管理 1~1023 之间所有的端口号。

Internet 扩展服务与 Unix 特定服务之间的一个差别就是 Telnet 和 Rlogin。它们二者都允许通过计算机网络登录到其他主机上。Telnet 是采用端口号为 23 的 TCP/IP 标准且几乎可以在所有操作系统上进行实现。相反，Rlogin 最开始时只是为 Unix 系统设计的（尽管许多非 Unix 系统现在也提供该服务），因此在 20 世纪 80 年代初，它的有名端口号为 513。

客户端通常对它所使用的端口号并不关心，只需保证该端口号在本机上是惟一的就可以了。客户端口号又被称作临时端口号（即存在时间很短暂）。这是因为它通常只是在用户运行该客户程序时才存在，而服务器则只要主机开着，其服务就运行。

大多数 TCP/IP 实现给临时端口分配 1024~5000 之间的端口号。大于 5000 的端口号是为

其他服务器预留的（Internet 上并不常用的服务）。①

保留端口号

Unix 系统有保留端口号的概念。只有具有超级用户特权的进程才允许给它自己分配一个保留端口号。

这些端口号介于 1~1023，一些应用程序（如有名的 Rlogin，26.2 节）将它作为客户与服务器之间身份认证的一部分。②

而 Windows 中的端口分配又有所不同，只要不和已知端口冲突，原则上可自由使用。

二、MFC 对 Socket 编程的封装类简介

Microsoft Windows Class Library(MFC) 中提供了较高级封装的类用来实现网络通信。图 1.3 给出了 CSocket 类的继承关系。

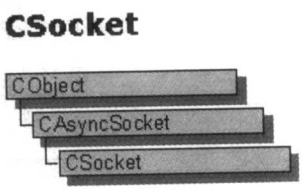

图 1.3

CAsyncSocket 类封装了 Windows Sockets API 函数，提供了较低层的与 Windows Sockets 对话接口，一般适合于有相当水平的网络编程者使用，可方便地进行底层的网络事件通知及信息回叫控制等操作。

CSocket 类派生于 CAsyncSocket，它继承了父类中一些常用易懂的 Windows Sockets API 函数，并对 CAsyncSocket 中底层的较难控制的一些 API 函数或成员函数进行了处理，使得网络传输简捷易用，同时它支持模块化的后台信息处理，解决了 CAsyncSocket 中较难克服的多线程处理。

下面介绍用 Visual C++在 Windows 中实现 Socket 的 CSocket 类型成员函数(这些成员函数实际上是从 CAsyncSocket 类继承来的)。

成员函数和参数说明：

(1) **BOOL Create** (**UINT** *nSocketPort* = 0， **int** *nSocketType* = SOCK_STREAM，**long** *lEvent* = **FD_READ |FD_WRITE|FD_OOB|FD_ACCEPT|FD_CONNECT| FD_CLOSE**，**LPCTSTR** *lpszSocketAddress* = NULL)

该函数用来建立 Socket，如果函数成功，则返回非零值；否则返回值为 0。其中：

nSocketPort ：为所选择的 Socket 端口，一般要大于 1023，如果该参数为 0，则由系统选定一端口，默认值为 0。

nSocketType ：为套接字类型——SOCK_STREAM 或 SOCK_DGRAM。SOCK_STREAM 表示为流套接字（本实验使用基于 TCP 连接的流套接字编程），SOCK_DGRAM 表示为数据报套接字（将在以后实验中讲述），默认值为 **SOCK_STREAM** 。

lEvent ：标识该 Socket 要完成哪种工作，默认值为 **FD_READ|FD_WRITE|FD_OOB|FD_ACCEPT|FD_CONNECT|FD_CLOSE** 。

lpszSockAddress：一个指向字符串的指针，该字符串包含了被连接套接口的网络地址。一个带点的数字，如“128.56.22.8”，默认值为 **NULL**。

---

① Richard Stevens. 《TCP/IP 协议详解》，p.13①的译文。

② Richard Stevens. 《TCP/IP 协议详解》，p.13③的译文。

注意：CSocket 中，WinsockAPI 的初始化(socket)和绑定(bind)两部分工作都完成了。

(2) **BOOL Listen (int** *nConnectionBacklog* = 5)

该函数的作用是等待 Socket 请求，如果调用成功，则返回非零值；否则返回值为 0。Listen 仅对那些支持连接的套接字起作用，也就是 **SOCK_STREAM** 类型的套接字。在进程应答连接并把它放到等待队列时，套接字被置成被动模式（passive mode）。本函数一般由哪些一次可以有多个连接的服务器使用（或任何需要接收连接的应用）。

nConnectionBacklog ：表示等待队列的长度，默认值为最大值 5，有效值为 1~5。

(3) **BOOL Connect (LPCTSTR** *lpszHostAddress*，**UINT** *nHostPort*)

该函数的作用是提出连接请求。其中：

lpszHostAddress ：对象连接的套接字的网络地址、机器名，如 ftp.sjtu.edu.cn，或以句点分隔的数字，如“211.80.43.100”。

nHostPort ：为接受请求进程的网络地址和 Socket 端口号。

注意：Connect 函数还有另一个版本：**BOOL Connect(const SOCKADDR*** *lpSockAddr*，**int** *nSockAddrLen*)；具体用法可以参阅 MSDN Library 中关于 **CAnyscSocket** 类的阐述。

(4) **virtual void Close( )**

该函数的作用是关闭该 Socket。

三、利用 CSocket 进行传输的辅助类简介

1. CSocketFile 类

CSocketFile 继承了 **CFile** 类，见图 1.4，它可以很自如地用来在基于 Windows Socket 网络上传输数据。首先，将一个建立连接 CSocket 对象实例作为参数进行初始化，然后，将已经初始化的 CSocketFile 对象连接到 **CArchive** 对象上，接着将数据串行化，以使用 MFC 系列来简化发送和接收数据，最终实现利用网络的 Socket 传输和本机上的流传输一样简单。

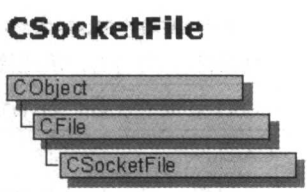

图 1.4

成员函数：CSocketFile 用到的成员函数只有构造函数。

CSocketFile::CSocketFile(CSocket* *pSocket*, **BOOL** *bArchiveCompatible* = TRUE);**其中：**

pSocket ：连接到 CSocketFile 对象的套接口。

bArchiveCompatible ：指示该文件对象是否与一个 CArchive 对象一起使用。只有当希望在单机方式下来使用这个 CSocketFile 对象时，才传递 FALSE。因为仅有 CSocketFile 类实例本身并没有什么意义，所以通常将其置为 TRUE。

说明：此成员函数用来构造一个 CSocketFile 对象。当此对象超出范围或被删除时，它的析构函数将使它自己从插槽对象上分离。

注意：一个 CSocketFile 对象也可以在没有 CArchive 对象的情况下作为一个（受限制的）文件来使用。缺省情况下，CSocketFile 构造函数的 bArchiveCompatible 参数是 TRUE，它表明此文件对象是与一个档案一起使用的。要在没有档案的情况下使用该文件对象，则给 bArchiveCompatible 参数传递 FALSE。在“档案兼容”模式下，一个 CSocketFile 对象可以提供更好的表现，并减少出现“死锁”的几率。

2. CArchive 类

CArchive 类没有基类。CArchive 允许以一个永久二进制（通常为磁盘存储）的形式保存一个对象的复杂网络，它可以从永久存储中装载对象，并在内存中重新构造它们。使数据永久保留的过程就叫作“串行化”。一般可以把一个 CArchive 对象看作一个二进制流，可以将它和输入输出流 iostream 类的用法进行比较。CArchive 对象一般和一个文件类关联（CFile 类或 CSocketFile 类）。输入输出流是加工处理 ASCII 字符，而 CArchive 类的用处是高效、无冗余地处理二进制数据。

在 CArchive 类中，重载了提取（>>）和插入（<<）运算符，它是方便的归档编程接口，主要支持 CObject 派生类。

四、MFC CSocket 类的通信流程

使用 CSocket 类进行网络二进制数据通信的连接流程，如下表所示。

	服 务 器 端	注 释		客 户 端
1	CSocket m_server;	构造一个 socket 对象		CSocket m_client;
2	m_server.create(nport);	创建 socket		m_client.create(nport);
3	m_server.listen();	听等 连接	与服务器 建立连接	m_client.connect(straddr，nport); 此时阻塞，等待服务器端侦听。
4	CSocket m_receive; m_server.accept(m_receive); 此时阻塞，等待客户机连接。	构造新的 socket 对象用以接收客户端的连接		
5	CSocketFile file(&m_server);	构造一文件对象		CSocketFile file(&m_server);
6	CArchive arin(&file，CArchive::load); CArchive arout(&file，CArchive::store);	构造流对象		CArchive arin(&file，CArchive::load); CArchive arout(&file，CArchive::store);
7	arin >>value; arout<<value;	用流进行数据的传输概念和 cin，cout 相似		arin >>value; arout<<value;

注意事项：利用 CArchive 类进行网络数据传输的操作固然方便直观，但是如果编写的程序是和别人的程序进行通信的话，就要注意对方的程序是否也用了 CArchive 类，否则会造成数据相互不能识别。

五、使用 CSocket 类的同步问题和解决方法

有了以上的基础，就可动手进行网络数据通信了，可以做到基于阻塞发送和接收二进制数据。

比如，可以 Client 端发送，Server 端接收：

Server 端：m_receive. Receive(void* *lpBuf*， int *nBufLen*， int *nFlags* = 0);

Client 端：m_client. Send(const void* *lpBuf*， int *nBufLen*， int *nFlags* = 0);

接着，再分析一下各个类中提到的常用方法的同步特性：

Listen(...)：执行后不管有没有连接，立即返回。

Connect(...)：如果服务器端有端口正在侦听，则立即成功返回；如果没有，则过几秒钟将显示无法连接。

Accept(...)：Listen 函数返回后可以执行此函数，但是此函数是基于阻塞的，只要客户机 connect 连接并且端口正确，则立即成功返回，建立连接；如果迟迟侦测不到连接，则不断阻塞，直到连接成功或者强行关闭。

Send(...)：调用后就将数据保存在 socket 缓冲区中，立即返回。

Receive(...)：和 Accept 一样的阻塞，直到能从 socket 缓冲区成功读取到 nBufLen 长度的数据。

按上分析，读者可能会想到这样两个问题：

(1) 在侦听的时候，如果客户端迟迟没有连接，则侦听方执行到 Accept 则阻塞不能响应。

(2) 在用 Receive 接收数据，如果迟迟得不到发送的数据，也阻塞不能响应。

显然，带有这两个问题的软件是不能接受的。幸亏 CSocket 类里可以使用继承自 CAsyncSocket 里的 OnReceive 和 OnAccept 消息处理函数，其原理分别是：

(1) **OnAccept()**：Listen 过后，如果侦测到客户机有连接，则产生消息调用 OnAccept()，一般可以在此函数里面调用 Accept 便可避免侦听时的阻塞。

virtual void OnAccept(int *nErrorCode* **)**；其中：

nErrorCode：套接字上最近的错误代码。此成员函数可用的错误代码有：

0：函数成功地执行并返回；

WSAENETDOWN：Windows Sockets 检测到网络系统故障。说明由框架调用，通知监听套接字现在可以调用 Accept 成员函数来接收挂起的连接请求（有 connect 请求进入）。

(2) **OnReceive()**：建立连接后，如果侦测到 Socket 缓冲区里有数据到达，便自动调用 OnReceive()，在此函数里面使用 Receive 接收就可避免接收数据的阻塞。

virtual void OnReceive(int *nErrorCode* **)**；其中：

nErrorCode：套接字上最近的错误代码。此成员函数可用的错误代码有：

0：函数成功地执行并返回；

WSAENETDOWN：Windows Sockets 检测到了网络故障。说明本函数由框架调用，通知套接字缓冲中有数据，可以调用 Receive 成员函数取出。

【实验步骤】

下面以一个最简单的点对点通信的聊天程序为例：

客户机/服务器模式是 socket 点对点网络程序典型的模式。它用到的方法也是面向连接 TCP 连接的套接字 MFC 典型方式。其工作过程是：服务器首先启动，创建套接字后等待客户的连接；客户启动以后，创建套接字，然后和服务器建立连接；连接建立后，客户机和服务器可以通过建立的套接字连接进行信息通信。

先建立一个 MFC，选 dialogBased，工程名为 ***LX2*** ，如图 1.5 所示。

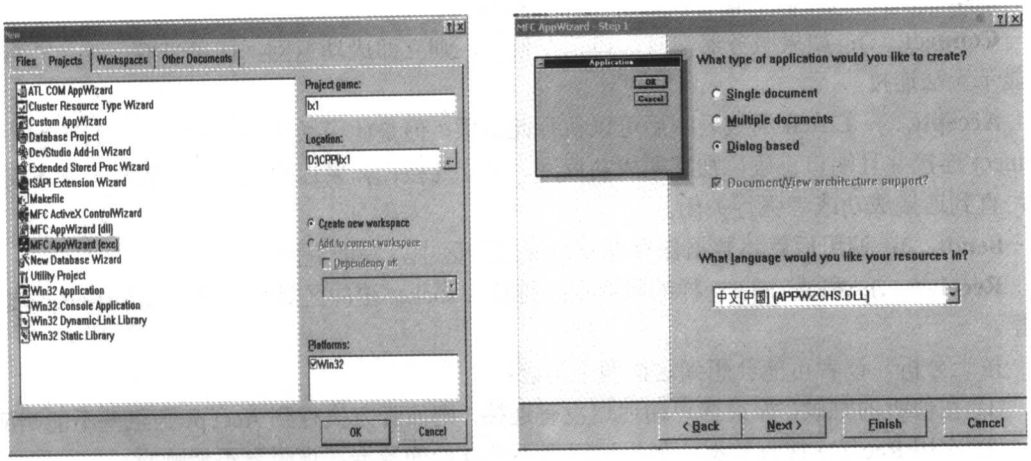

图 1.5

下一个对话框选择 Window Sockets，后面的选项酌情考虑，或者连续选择默认的即可，如图 1.6 所示。

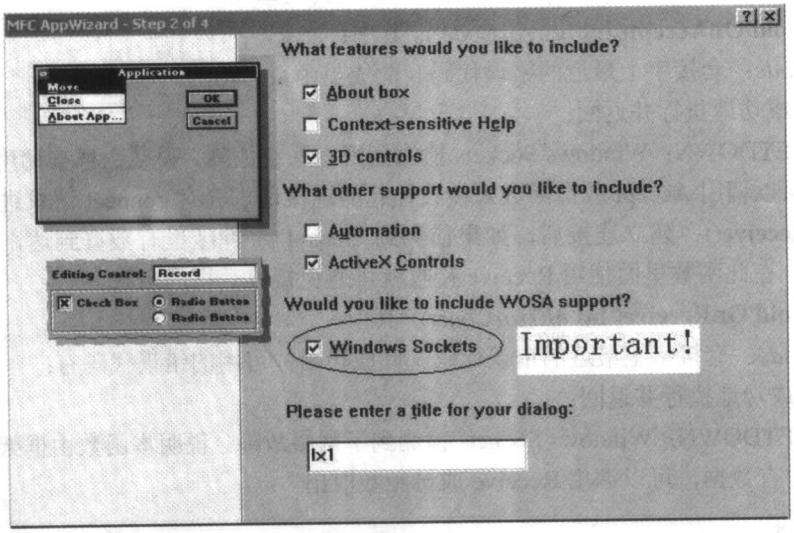

图 1.6

如果忘了添加 Windows Sockets 选项，可以在文件头部添加下列语句进行补救：

```
#include   "Winsock.h"
#include   "Ws2tcpip.h"
#pragma   comment( lib，"Ws2_32.lib" )
```

注：这些语句支持 winsock2。

出现 Dialog 以后，编辑界面，使其如图 1.7 所示并且对控件点击右键，选择属性选项，把每个控件的 ID 改掉（控件 ID 就是每个控件的名字，要改成有意义的，以便将来管理）。

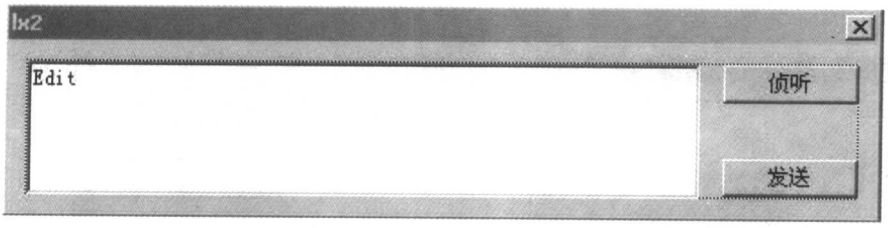

图 1.7

各个控件的 ID 如下表，并且在对话框视图中点击右键，选择 Class Wizard 选项，用该工具对控件添加变量，使其如图 1.8 所示。

控　件　ID	变量名	绑定变量类型	对应界面上的控件
IDC_CONTENT	m_msg	CString	输入发送内容的文本框
IDC_CONTENT	m_ctrl	CEdit	输入发送内容的文本框
IDC_LISTEN	m_listen	CButton	侦听按钮
IDC_SEND	m_send	CButton	发送按钮

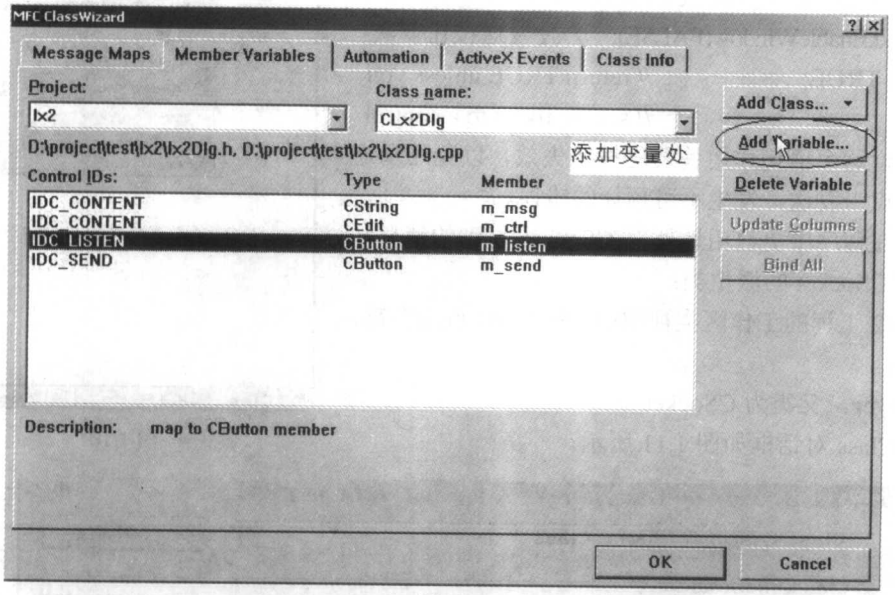

图 1.8

接着，再打开一个 VC，建立客户机工程，工程名称为 ***LX1***，各个控件的 ID 如下表，界面如图 1.9 所示。

控件 ID	变量名	绑定变量类型	对应界面上的控件
IDC_CONTENT	m_msg	CStringt	输入发送内容的文本框
IDC_CONTENT	m_ctrl	CEdit	输入发送内容的文本框
IDC_CONNECT	m_connect	CButton	连接按钮
IDC_SEND	m_send	CButton	发送按钮
IDC_IP	m_ip	CString	输入连接目的 IP 的文本框

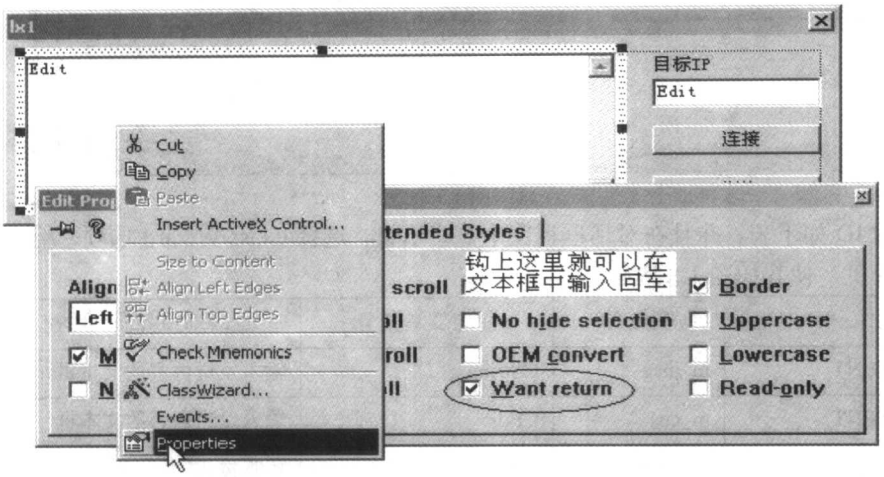

图 1.9

然后，在 BOOL CLx1Dlg::OnInitDialog()和 BOOL CLx2Dlg::OnInitDialog()末尾添加语句，使其如下所示：

```
m_send.EnableWindow(FALSE);  // 使发送按钮变灰
return TRUE;                 // return TRUE unless you
                             // set the focus to a control
```

注意：这个语句作用使发送按钮失效，以免还未连接用户就点击发送，发生不可预计的错误。

为了在程序中更自由地处理 CSocket 得到的消息，必须新建 CSocket 的派生类：

在 Lx2 工程的工作区类视图中（图 1.10）点击右键，添加新类：

CServer，父类为 CSocket 。

NewClass 对话框如图 1.11 所示。

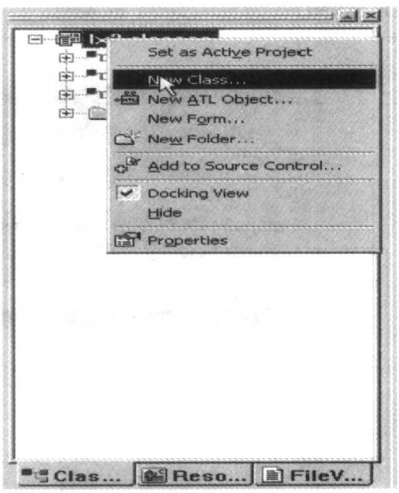

图 1.10

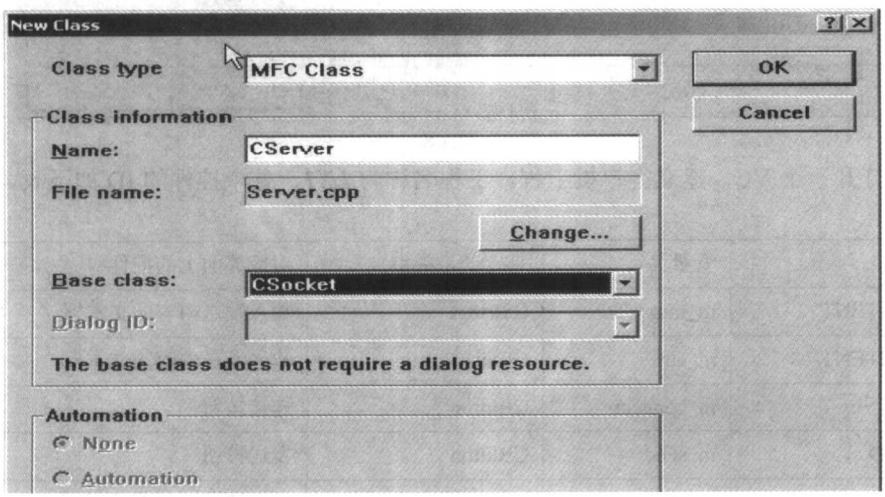

图 1.11