基于 MFC 封装的 Windows 通信 API 的研究

李 丽 蒋洪睿 刘亚军 (桂林电子工业学院 541004)

摘要:本文介绍 Windows 环境下网络编程的多种通信 API 及 MFC(微软基础类库)对它们的封装类,重点阐述封装 WinSock API 的 MFC 类 CASyncSocket 和 Csocket 的编程模型。

关键词:WinSock CAsyncSocket MFC API

一、引言

近年来,如何利用 Internet 进行网际间通信,不论是 WWW 浏览、FTP、Gopher 这些常规服务,还是网络电话、多媒体会议等这些对实时性要求严格的应用,已经成为研究的热点,而且已经是必不可少的了。Windows 环境下进行通信程序设计的最基本方法是应用 Windows Sockets 实现进程间的通信。为此微软提供了大量基于 Windows Sockets 的通信 API,如 WinSock API、WinInet API 和 ISAPI,微软一直致力于开发更快、更容易的通信 API 并和 MFC 集成在一起,以使通信编程越来越容易,MFC 作为微软所提供的面向对象编程方法的解决方案,是 VC++编程环境最重要的组成部分,MFC 为用户提供了一大批预先定义的类和成员函数,封装了大量的 Windows API,同时 VC++环境提供了与 MFC 对象和代码一起工作的专用工具:Visuall C++中的 AppStudio 源程序编辑器、AppWizard 和 Class Wizard,这些彻底改变了 Windows C++编程技术,并且最终成为一个工业标准。

应用 MFC,可以使 Windows 程序员用很少的时间和精力就可开发出复杂的通信应用程序。

结合笔者自己在开发实时网络音频工具 FreeTalk 过程中的一些经验,介绍 Windows 环境下常用 API 和封装它们的 MFC 类。重点介绍使用 MFC 的 CAsyncsocket 和 CSocket 类编写网络通信程序的方法,这两个类封装了 WinSock API,并使他们更容易使用,更适应于 MFC 编程环境。

二、Windows 环境下的通信 API 和相应 MFC 类

1. Windows Socket(WinSock) API

Windows Sockets 定义了微软 Windows 的网络编程接口。Windows Sockets 基于加利福利亚大学伯克利分校的伯克利 UNIX sockets。它既包括 BSD 风格的例程,还加入了 Windows 的扩展部分,例如用于消息驱动的扩展函数。Windows Sockets 可以运行在许多网络协议之上,这包括 TCP/IP, XNS, DECNet, IPX/SPX 以及其他。在 Win32 环境下, Windows Sockets 提供线程安全。通过微软与标准组织的努力,为 WinSock 定义了应用程序设计接口(WinSock API),可以非常方便的利用下层的网络协议(如 TCP/IP)进行网络间通信。

通过提供两个类 CAsyncSocket 和 CSocket,MFC 支持使用 WinSock API 通信程序设计。MFC 把复杂的 WinSock API 封装到类里,这使得编写应用程序更容易。CAsyncSocket 类逐个封装了 Win Sock API,为高级网络程序员提供了更有力更灵活的方法。这个类基于程序员了解网络通信的假设,目的是为了在 MFC 中使用 WinSock,程序员有责任处理诸如阻塞、字节顺序和在 Unicode 与 MBCS 间的字符转换的任务。为了给程序员提供更方便的接口自动处理这些任务,MFC 给出了 CSocket 类,这个类是由 CAsyncSocket 类继承下来的,它提供了比 CAsyncSocket 更高层的 WinSock API 接口。CSocket 和类 CSocketFile 以及 Carchive 一起合作,管理发送和接收的数据,因此,管理数据收发更加便利。CSocket 对象提供阻塞模式,这对于 Carchive 的同步操作是至关重要的,阻塞函数,比如 Receive(), Send(), ReceiveFrom(), SendTo()和 Accept()直到操作完成后才返回控制权。因此,如果需要低层控制和高效率,使用 CasyncSock 类,需要方便,则可使用 Csocket 类。

2. Win32 Internet(WinInet) API

微软公布了一些使因特网应用程序设计比以前更快、更容易的 API:WinInet API。这个 API 提供了中高层通信函数,这使得访问主要的因特网协议变得相当容

易。这些函数在程序员和 WinSock 驱动之间提供了隔离层。有四类 WinInet API 函数：

(1)通用 WinInet 函数；

(2)WinInet 文件传输协议(FTP)函数；

(3)WinInet Gopher 函数；

(4)WinInet 超文本传输协议(HTTP)函数。

事实上，MFC 把 WinInet API 和 ActiveX 技术封装进类，使因特网编程更加容易，这些类包括 CInternetSession、CInternetConnection、CInternetFile、CHttpConnection、CHttpFile、CGopherFile、CFtpConnection、CGopherConnection、CFileFind、CFtpFileFind、CGopherFileFind、CGopherLocator 和 CInternetException。

3. 因特网服务器 API(ISAPI)

微软的 IIS(因特网信息服务器)是唯一的与 Windows NT Server 操作系统紧密集成的 WWW 服务器，它作为因特网服务器和企业内部网(Intranet)服务器的应用范围很广。IIS 允许扩展功能，这是通过 ISAPI(因特网服务器 API)来实现的，ISAPI 描述了与因特网服务器之间的接口。用 ISAPI 提供的工具，可建立高性能、高效率，满足商业安全，符合新的 IIS 标准的因特网服务器。同样，ISAPI 在 MFC 由典型的类所封装，包括 CHttpFilter、CHttpFilterContext、CHttpServer、CHttpServerContext、Related Classes 和 CHtmlStream。

三、WinSock API 的 MFC 封装类

一些有特殊要求的网络应用程序，如网络电话、多媒体会议工具，实时性要求非常强，程序要求能够直接应用 WinSock 发送和接收数据。这时设计者应该选择直接应用 WinSock API 或者由 MFC 封装的 WinSock API。新开发的应用程序中，为了充分利用 MFC 的优势，首选方案应当是 MFC 中的 CAsyncSocket 和 CSocket 类，这两个类完全封装了 WinSock API，并提供更多的便利。现有的资料和书籍中，少有关于这方面的介绍以及应用的例子。因此，本文介绍应用这两个类的编程模型，并引出相关的成员函数与一些概念的解释。在此基础上，具体给出一个使用 CAsyncSocket 类产生数据报套接字通信的聊天程序例子。

1. CAsyncSocket 和 CSocket 类简述

CAsyncSocket 和 CSocket 类的继承关系由图 1 给出。CSocket 类是由 CAsyncSocket 继承而来的，事实上，在 MFC 中 CAsyncSocket 逐个封装了 WinSock API，每个 CAsyncSocket 对象代表一个 Windows Socket，使用 CAsyncSocket 类，要求程序员对网络编程较为熟悉，它实际上是为高级编程人员 Windows Socket 提供了面向对象的抽象。相比起来，CSocket 类是 CAsyncSocket 的派生类，继承了它封装的 WinSock API。一个 CSocket 对象代表了一个比 CAsyncSocket 对象更高层次的 Windows Socket 抽象，CSocket 类与 CSocketFile 和 Carchive 类一起工作发送和接收数据，因此它使用更加容易，对设计者的网络编程经验要求较少一些。CSocket 对象提供阻塞模式，因为阻塞功能对于 CArchive 的同步操作是至关重要的。在这里有必要对阻塞的概念作一解释：一个 socket 可以处于“阻塞模式”或“非阻塞模式”，当一个套接字处于阻塞模式(即同步操作)时，它的阻塞函数直到操作完成才会返回控制权，之所以称为阻塞是因为在此套接字的阻塞函数完成操作返回之前什么也不能做。如果一个 socket 处于非阻塞模式(即异步操作)，则被调用函数会立即返回，在 CAsyncSocket 类中可以用 GetLastError 成员函数查询最后的错误，如果错误是 WSAEWOULDBLOCK 则说明有阻塞发生，而 CSocket 绝不会返回 WSAEWOULDBLOCK，因为它自己管理阻塞。微软建议尽量使用非阻塞模式，通过网络事件的发生通知应用程序进行相应的处理。但在 CSocket 类中，为了利用 CArchive 处理通信中的许多方面问题，简化编程，它的一些成员函数总是具有阻塞性质的，这是因为 CArchive 类需要同步的操作。在 Win32 环境下，如果要使用具有阻塞性质的套接字，应该放在独立的工人线程中处理，利用多线程的方法使阻塞不至于干扰其他线程，也不会把 CPU 时间浪费在阻塞上。多线程的方法既可以使程序员享受 CSocket 带来的简化编程的便利，也不会影响用户界面对用户的反应。

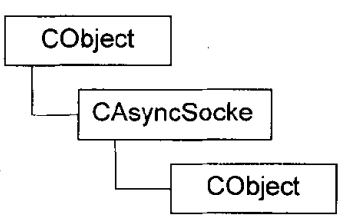

图 1

2. CAsyncsocket 类编程模型

在一个 MFC 应用程序中,想要轻松处理多个网络协议,又不想牺牲灵活性时,可以考虑使用 CAsyncSocket 类,它的效率比 CSocket 要高。CAsyncSocket 类针对字节流型套接字的编程模型简述如下:

(1)构造一个 CAsyncSocket 对象,并用这个对象的 create 成员函数产生一个 SOCKET 句柄。可以如下构造:

```
CAsyncSocket sock;
sock.Create( ); //使用默认参数产生一个字节流套接字
```

或者

```
CAsyncSocket * pSocket = new CAsyncSocket;
int nPort = 27;
pSocket->Create( nPort, SOCK-DGRAM ); //用指定端口号产生一个数据报套接字
```

第一种方法在栈上产生一个 CAsyncSocket 对象,而第二种方法在堆上产生 CAsyncSocket 对象。第一种 Create 成员函数用缺省参数产生一个字节流套接字,第二种 Create 成员函数用指定的端口和地址产生一个数据报套接字。

Create 的参数有:

①端口,UINT 类型,注意:如果是服务器方,则使用一个众所周知的端口供服务方连接,如果是客户方,典型做法是接受默认参数,使套接字可以自主选择一个可用端口;

②socket 类型,SOCK-STREAM (默认值)或 SOCK_DGRAM;

③socket 地址,例如"ftp.gliet.edu.cn"或"202.193.64.33"。

(2)如是客户方程序,用 CAsyncSocket::Connect 成员函数连接到服务器方;如是服务器方程序,用 CAsyncSocket::Listen 成员函数开始监听,一旦收到连接请求,则调用 CAsyncSocket::Accept 成员函数开始接收。注意:CAsyncSocket::Accept 成员函数要用一个新的并且为空的 CSocket 对象作为它的参数,这里所说的"空的"指的是这个新对象还没有调用 Create 成员函数。

(3)调用其他的 CAsyncSocket 类成员函数进行通信管理。

(4)通信结束后,销毁 CAsyncSocket 对象。如果是在栈上产生的 CAsyncSocket 对象,则对象超出定义的范围时自动被析构,如果是在堆上产生,也就是用了 new 这个操作符,则必须使用 delete 操作符销毁 CAsyncSocket 对象。

3. CSocket 类编程模型

使用 CSocket 对象涉及 CArichive 和 CSocketFile 类对象。以下介绍的针对字节流型套接字的操作步骤中,只有第 3 步对于客户方和服务器方操作是不同的,其他步骤都相同。

(1)构造一个 CSocket 对象。

(2)使用这个对象的 Create 成员函数产生一个 SOCKET 句柄。在客户方程序中,除非需要数据报套接字,Create 一般情况下应该使用默认参数。而对于服务器方程序,必须在调用 Create 时指定一个端口。请注意,CArchive 不能与数据报(UDP)套接字一起工作,因此对于数据报套接字,CAsyncSocket 和 CSocket 的使用方法是一样的。

(3)如果是客户方套接字,则调用 CAsyncSocket::Connect 与服务方套接字连接;如果是服务方套接字,则调用 CAsyncSocket::Listen 开始监听来自客户方的连接请求,收到连接请求后,调用 CAsyncSocket::Accept 接受请求,建立连接。请注意,Accept 成员函数需要一个新的并且为空的 CSocket 对象作为它的参数,解释同上。

(4)产生一个 CSocketFile 对象,并把它与 CSocket 对象关联起来。

(5)为接收和发送数据各产生一个 CArchive 对象,把它们与 CSocketFile 对象关联起来。切记 CArchive 是不能和数据报套接字一起工作的。

(6)使用 CArchive 对象在客户与服务方传送数据。

(7)通信完毕后,销毁 CArchive、CSocketFile、和 CSocket 对象。

参考文献

[1] Microsoft Developer Network, Microsoft Corporation, 1999

[2] Peter Norton, Rob McGregor 著,孙凤英等译,MFC 开发 Window95/NT4 应用程序,清华大学出版社,1998

[3] Robert D. Thompson 著,MFC 开发人员参考手册,机械工业出版社,1997

[4] Clayton Walnum 著,Windows 98 编程核心技术精解,机械工业出版社,1999

[5] Richard C. Leinecker 著,Visual C++5 开发人员参考手册,机械工业出版社,1998

(来稿时间:1999 年 4 月)