第一章　　网络编程基础

1. 什么是线程，什么是进程？

答：进程是处于运行过程中的程序实例，是操作系统调度和分配资源的基本单位。　　一个进程实体由程序代码、数据和进程控制块三部分构成。线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。

2. 描述网络应用程序的一般组成。为什么说应用层协议是在应用程序中实现的　　？

答：从功能上，可以将网络应用程序分为两部分，一部分是专门负责网络通信的模块，它们与网络协议栈相连接，借助网络协议栈提供的服务完成网络上数据信息的交换。另一部分是面向用户或者作其他处理的模块，它们接收用户的命令，　或者对借助网络传输过来的数据进行加工，　　这两部分模块相互配合，来实现网络应用程序的功能。

2. 实现网间进程通信必须解决哪些问题？

答：（1）网间进程的标识问题；（2）如何与网络协议栈连接的问题；（3）协议的识别问题；（4）不同的通信服务的问题

4. 说明 TCP/IP 中，端口的概念和端口的分配机制。

答：端口是 TCP/IP 协议族中，应用层进程与传输层协议实体间的通信接口。　　TCP/IP 协议采用了全局分配（静态分配）和本地分配（动态分配）相结合的分配方法。

对于 TCP，或者 UDP，将它们的全部 65535 个端口号分为保留端口号和自由端口号两部分。保留端口的范围是 0—1023，又称为众所周知的端口或熟知端口　（well-known port ），其余的端口号，1024-65535，称为自由端口号，采用本地分配，又称为动态分配的方法。

总之，TCP或 UDP端口的分配规则是：

端口 0：不使用，或者作为特殊的使用；端口　1-255：保留给特定的服务，　TCP和 UDP均规定，小于 256 的端口号才能分配给网上著名的服务；端口　256-1023：保留给其他的服务，如路由；

端口 1024-4999：可以用作任意客户的端口；端口　5000-65535：可以用作用户的服务器端口。

5. 什么是网络应用进程的网络地址？说明三元组和五元组的概念。

答：应用层进程地址 =（传输层协议，主机的 IP 地址，传输层的端口号），它标识了因特网中，进程间通信的一个端点，也把它称为进程的网络地址。

（传输层协议，主机的 IP 地址，传输层的端口号），这样一个三元组，叫做一个半相关（half-association　）。

(传输层协议，本地机 IP 地址，本地机传输层端口，远地机　IP 地址，远地机传输层端口)，五元组称为一个全相关（ association　）。即两个协议相同的半相关才能组合成一个合适的全相关，或完全指定一对网间通信的进程。

6. 举例说明面向消息的协议与基于流动协议有什么不同。

答：例如甲向乙发送三个消息，分别是：　128、64 和 32 字节。

面向消息的协议中，如果接受缓冲区是　256 字节，足以接收 3 条消息，且这三条消息全部达到。乙仍然需要发送 3 条读取消息。分别返回 128、64、32 字节，而不用一次读取来调用者单个数据包。

基于流的消息协议中，在接收端乙的堆栈把所有进来的消息数据包聚集在一起，放入堆栈，等待应用进程读取。进程发送读取命令，指定了进程接收缓冲区，如果进程缓冲区有　　256 字节，系统马上返回全部 224 字节。

7.TCP提供的服务有哪些特征？

答：应用层进程提供一个面向连接的、端到端的、完全可靠的（无差错、无丢失、无重复或失序）全双工的流传输服务。　允许网络中的两个应用程序建立一个虚拟的链接，在任何一个方向上发送数据，把数据当作一个双向字节流进行交换，最后终止连接。

8. 简要说明三类网络编程。

答：基于 WWW应用的网络编程技术，　包括所见即所得的网页制作工具，　和动态服务器页面的制作技术。.NET 平台有四组产品　① 开发工具　② 专用服务器③ Web 服务。④ 设备。 MS.NET顺应了软件工业的趋势，包括 4 个方面：① 分布式计算②　组件化③ 企业级别的服务④ WEB 范型转移。 .NET 平台由三层软件构成①　顶层是全新的开发工具　VS.NET，② 中间层包括三部分： .NET 服务器、.NET 服务构件和 .NET 框架。③ 底层是 WINDOWS操作系统。 Web服务是松散耦合的可复用的软件模块，在　Internet　上发布后，

能通过标准的 Internet 协议在程序中访问，具有以下的特点： （1）可复用（ 2）松散耦合（ 3）封装了离散（4）Web服务可以在程序中访问（ 5）Web服务在 Internet 上发布

9. 说明 C/S 模式的概念、工作过程和特点。

答：C/S 模式即客户机 / 服务器模式，是应用程序最常用的通信模式。

服务器的工作过程是：（1）打开一通信通道，并告知服务器所在的主机，它愿意在某一公认的地址上接收客户请求。（2）等待客户的请求到达该端口。 （3）服务器接收到服务请求，处理该请求并发送应答信号。为了能并发地接收多个客户的服务请求， 要激活一个新进程或新线程来处理这个客户请求（如 UNIX 系统中用 fork 、exec）。服务完成后，关闭此新进程与客户的通信链路，并终止。 （4）返回第二步，等待并处理另一客户请求。 （5）在特定的情况下，关闭服务器。

客户方工作过程：

（1）打开一通信通道，并连接到服务器所在主机的特定监听端口。 （2）向服务器发送请求报文，等待并接收应答；继续提出请求，与服务器的会话按照应用协议进行。 （3）请求结束后，关闭通信通道并终止。

特点：（1）客户和服务器都是软件进程， C/S 模式是网络上通过进程通信建立分布式应用的常用模型。（2）非对称性：服务器通过网络提供服务，客户通过网络使用服务，这种不对称性体现在软件结构和工作过程上。（3）对等性：客户和服务器必有一套共识的约定，必与以某种应用层协议相联，并且协议必须在通信的两端实现。 （4）服务器的被动性：服务器必须先行启动，时刻监听，及时服务，只要有客户请求，就立即处理并响应，回传信息。但决不主动提供服务。 （5）客户机的主动性：客户机可以随时提出请求，通过网络得到服务，也可以关机走人，一次请求与服务的过程是由客户机首先激发的。（6）一对多：一个服务器可以为多个客户机服务，客户机也可以打开多个窗口，连接多个服务器。（7）分布性与共享性：资源在服务器端组织与存储，通过网络分散在多个客户端使用。

10. 说明用户和客户机，服务器和服务器类计算机的区别。

答：“ 客户 ”(client) 和服务器都指的是应用进程，即计算机软件。

“ 用户 ”(user) 指的是使用计算机的人。

服务器（ server ）这个术语来指那些运行着的服务程序。

服务器类计算机（ server-class computer ）这一术语来称呼那些运行服务器软件的强大的计算机。

第二章 UNIX 套接字编程接口

2. 实现套接字编程接口的两种方式是什么？

答：一种是在操作系统的内核中增加相应的软件来实现， 一种是通过开发操作系统之外的函数库来实现。

4. 什么是套接字？

答：是应用程序通过网络协议栈进行通信交互的接口。

5. 说明套接字特点。

答：（1）通信域。套接字通常只和同一域中的套接字交换数据。如果数据交换要穿越域的边界，就一定要执行某种解释程序。 （2）套接字有三种类型，流式、数据包和原始套接字。 （3）套接字由应用层创建，并为其服务，而后被释放。 （4）使用确定的 IP 地址和传输层端口号。

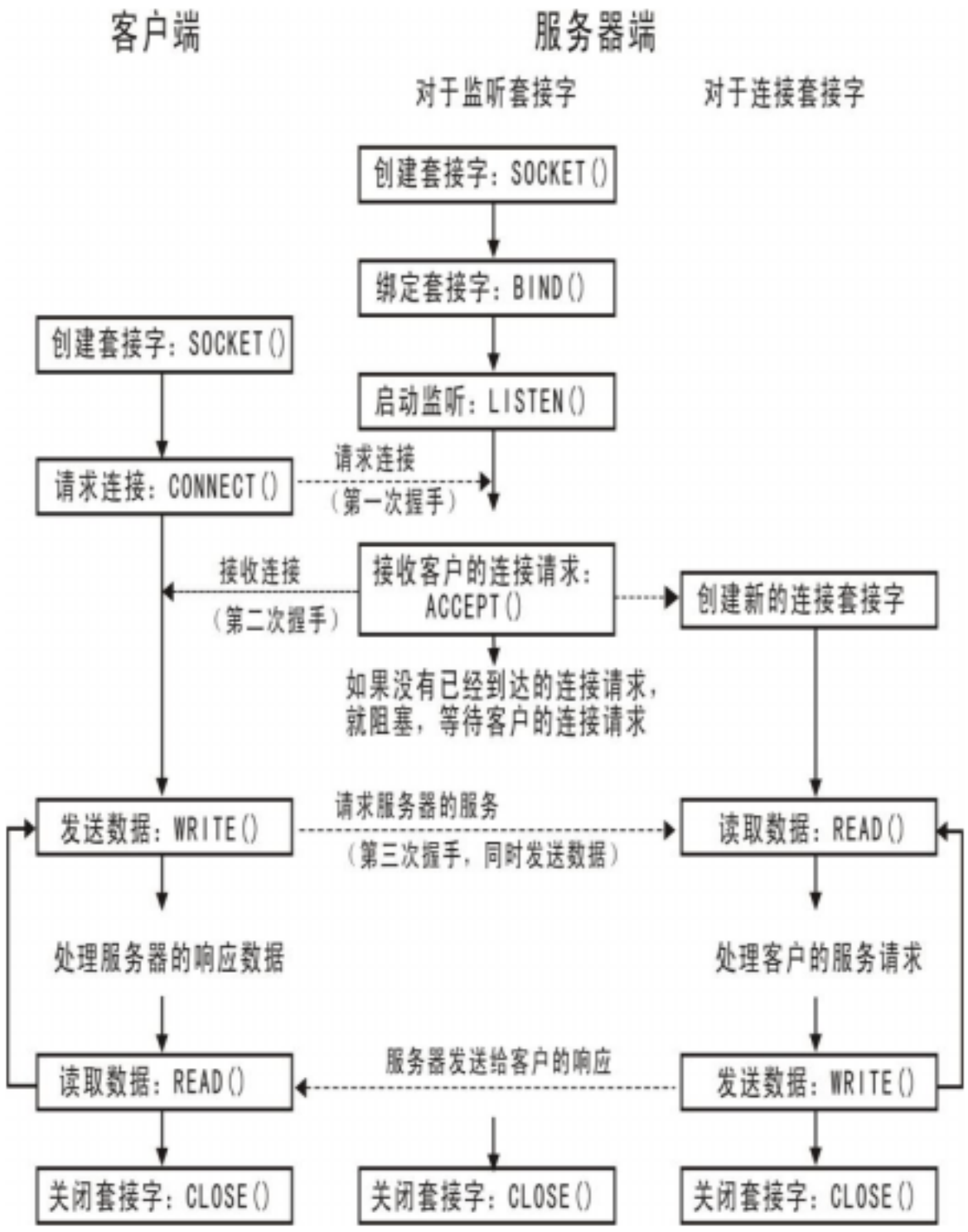
6. 说明套接字应用场合。

答：（1）不管是采用对等模式或者客户机 / 服务器模式，通信双方的应用程序都需要开发。 （2）双方所交换数据的结构和交换数据的顺序有特定的要求，不符合现在成熟的应用层协议，甚至需要自己去开发应用层协议，自己设计最适合的数据结构和信息交换规程。

7. 说明本机字节顺序和网络字节顺序的概念。

答：在具体计算机中的多字节数据的存储顺序，称为本机字节顺序。多字节数据在网络协议报头中的存储顺序，称为网络字节顺序。

8. 流式套接口的工作过程



9. 什么是阻塞问题？如何对应？

答:阻塞是指一个进程执行了一个函数或者系统调用，该函数由于某种原因不能立即完成，因而不能返回调用它的进程，导致进程受控于这个函数而处于等待的状态，进程的这种状态称为阻塞。利用 UNIX 操作系统的 FORK()系统调用，编制多进程并发执行的服务器程序。

第三章 Windows 环境的网络编程

1. 试述 WinSock 1.1 的特点。

答：（1）WinSock 1.1 全面继承了 Berkeley Sockets 规范

（2）数据库函数。其中六个采用 getXbyY() 的形式，大多要借助网络上的数据库来获得信息,

（3）WinSock 1.1 扩充了 Berkeley Sockets 规范

（4）WinSock 1.1 只支持 TCP/IP 协议栈

2.WinSock 规范与 Berkeley 套接口的区别是什么？

答： a. 套接口数据类型和该类型的错误返回值

b.select() 函数和 FD_*宏。在 Winsock 中，使用 select() 函数时，应用程序应坚持用 FD_XXX宏来设置，初始化，清除和检查 fd_set 结构。

c. 错误代码的获得 在 Winsock 中，错误代码可以使用 WSAGetLastError() 调用得到。

d. 指针所有应用程序与 Windows Sockets 使用的指针都必须是 FAR指针。

e. 重命名的函数。（1）close() 改变为 closesocket() （2）ioctl() 改变为 ioctlsocket()

f.Winsock 支持的最大套接口数目 在 WINSOCK.H中缺省值是 64，在编译时由常量 FD_SETSIZE决定。

g. 头文件 Berkeley 头文件被包含在 WINSOCK.H中。一个 Windows Sockets 应用程序只需简单地包含 WINSOCK.H就足够了。 h. Winsock 规范对于原始套接口的支持 .i.Winsock 规范对于原始套接口和消息驱动机制的支持。体现在异步选择机制、异步请求函数、阻塞处理方法、错误处理、启动和终止等方面。

3.WinSock 的注册和注销过程

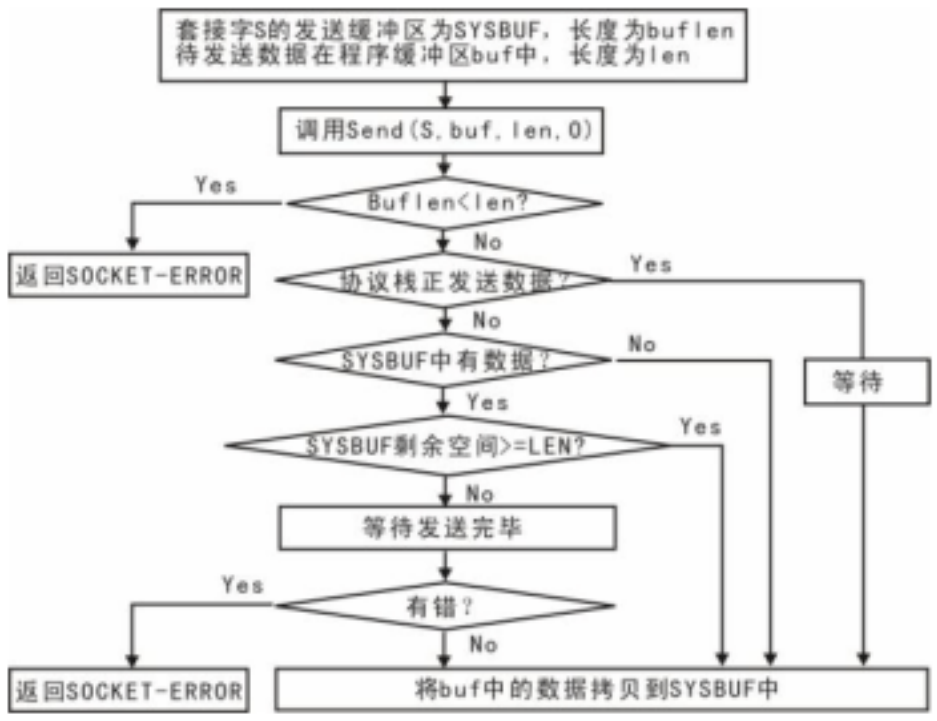
答：注册过程：调用 WSAStartup 的格式： int WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData); wVersionRequested ：指定要使用的 WinSock 的最高版本号, lpWSAData 用来返回 WinSockAPI实现细节的 WSAData结构变量指针。

注销过程：应用程序必须调用 WSACleanup()函数，来解除与 Winsock.DLL 库的绑定，释放 Winsock 实现分配给应用程序的系统资源，中止对 Windows Sockets DLL 的使用。 int WSACleanup (void);

4. 说明 WSAStartup 函数初始化过程。

（1）查找 WinSock.DLL 文件，如果有 WinSock 实现，则调入相关信息。若无，则初始化失败。返回错误信息。（2）。确认版本号。（3）建立 WinSock 同应用程序的联系。（4）函数创建成功，在 WSAData 结构中返回许多信息，否则，通知用户“ 初始化失败 ” 。

5. 画框图说明同步套接字的 send 函数的初始化过程。



：

第四章

1. 为什么说 MFC 是一个编程框架？它提供了哪些相应的工具？

MFC 应用程序框架，简称 MFC 框架，是由 MFC（Microsoft Foundation Class Library ）中的各种类结合起来构成的。 MFC 框架从总体上定义了应用程序的轮廓，并提供了用户接口的标准实现方法，程序员只须通过预定义的接口把具体应用程序特有的东西填入这个轮廓，就能建立 Windows 下的应用程序。

Microsoft Visual C++ 提供了相应的工具来完成这个工作：用应用程序向导（ AppWizard）可以生成应用程序的骨架文件（代码和资源等）;用资源编辑器可以直观地设计用户接口； 用类向导（ClassWizard ）可以将代码添加到骨架文件；用编译器可以通过类库实现应用程序特定的逻辑。 MFC 实现了对应用程序概念的封装，把类、类的继承、动态约束、类的关系和相互作用等封装起来。

2.MFC 类库封装了哪些内容？

1）对 Win32 应用程序编程接口的封装

2）对应用程序概念的封装

3）对 COM/OLE 特性的封装

4）对 ODBC 功能的封装

3. 典型的 MDI 应用程序 AppWizard 会自动创建一系列文件，如果工程的名字是 My,这些文件的名字是什么？

头文件 My.h，实现文件 My.cpp, 资源文件 My.rc 和模块定义文件 My.def

4. 说明构成应用程序的对象之间的关系。

这里，用图的形式可直观地表示所涉及的 MFC 类的继承或者派生关系，如图 4.2 所示意。

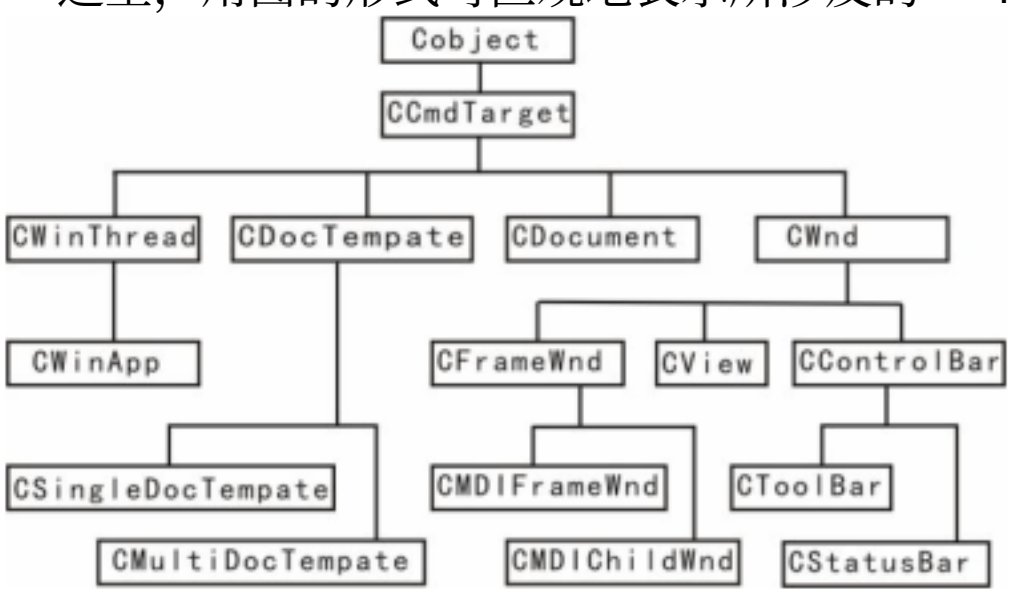


图 4.2 构成 MDI 应用程序的各对象之间的派生关系

5. 说明 MFC 对象和 Windows 对象的关系。

所谓 Windows 对象是 Win32 下用句柄表示的 Windows 操作系统对象；所谓 MFC 对象是 C++对象，是一个 C++类的实例。两者有很大的区别，但联系紧密。

以窗口对象为例：

图 4.3 MFC 的 CWnd 类窗口对象和 Windows的窗口对象的关系

6. 说明 MFC对象和 Windows对象的区别。

（1）对应的数据结构不同

（2）所处的层次不同

（3）创建的机制不同

（4）二者转换的方式不同

（5）使用的范围不同

（6）销毁的方法不同

7.CObject 类具有哪些特性？

CObject 类有很多有用的特性：对运行时类信息的支持，对动态创建的支持，对串行化的支持，对象诊断输出，等等。

8. 说明应用程序、文档模板、边框窗口、视图和文档等的创建关系。 P121

答：应用程序是全局对象，在启动程序之前构造；应用程序创建文档模板；文档模板创建文档和边框窗口；边框窗口创建试图。

9. 说明 WinMain 入口函数的流程。 P123

10. 消息循环的过程是什么？ P124

11. 应用程序的退出过程是什么？

下面以单击主窗口的关闭按钮为例，来说明应用程序退出的过程。

1）用户单击主窗口的关闭按钮，导致发送 MFC标准命令消息 ID_APP_EXIT，MFC调用 CwinApp::OnAppExit() 来完成对该命令消息的缺省处理，主要是向主窗口发送 WM_CLOSE消息。

2）主窗口处理 WM_CLOSE消息。

3）收到 WM_QUIT消息后，退出消息循环，进而退出整个应用程序。

第五章

一、 MFC提供的两个套接字类是什么？

答： CAsyncSocket 类， CSocket 类。

二、为什么说 CAsyncSocket 类是在很低的层次上对 Windows Sockets API 进行了封装？

答：它的成员函数和 Windows Sockets API 的函数调用直接对应。一个 CAsyncSocket 对象代表了一个Windows套接字。它是网络通信的端点。除了把套接字封装成 C++的面向对象的形式供程序员使用以外，这个类唯一所增加的抽象就是将那些与套接字相关的 Windows消息变为 CAsyncSocket 类的回调函数。

三、为什么说 Csocket 类是对 Windows Sockets API 的高级封装？

答：CSocket 类继承了 CAsyncSocket 类的许多成员函数， 用法一致。 CSocket 类的高级表现在三个方面：

（1）CSocket 结合 archive 类来使用套接字。

（2）CSocket 管理了通信的许多方面，如字节顺序问题和字符串转换问题。

（3）CSocket 类为 Windows消息的后台处理提供了阻塞的工作模式。

四、使用 CAsyncSocket 类的一般步骤是什么？

答：

序号	服务器（ Server ）	客户机（ Client ）
1	// 构建一个套接字 CAsyncSocket sockSrvr;	// 构建一个套接字 CAsyncSocket sockClient;

2	// 创建 SOCKET句柄，绑定到指定的端口
	sockSrvr.Create(nPort);
		// 创建 SOCKET句柄，使用默认参数
		sockClient.Create();

3	// 启动监听，时刻准备接受连接请求
	sockSrvr.Listen();	

4		// 请求连接到服务器
		sockClient.Connect(strAddr,nport);

5	// 构造一个新的空的套接字
	CasyncSocket sockRecv;
	// 接收连接
	sockSrvr.Accept(sockRecv);	

6	// 接收数据
	sockSrvr.Receive(pBuf,nLen);
		// 发送数据
		sockClient.Send(pBuf,nLen);

7	// 发送数据
	sockSrvr.Send(pBuf,nLen);
		// 接收数据
		sockClient.Receive(pBuf,nLen);

8	// 关闭套接字对象
	sockSrvr.Close();
		// 关闭套接字对象
		sockClient.Close();

五、CAsyncSocket 类可以接受并处理哪些消息事件？当这些网络事件发生时，MFC 框架作何处理？

答：MFC 套接字对象可以接受并处理的六种网络事件：

（1）FD_READ 事件通知：通知有数据可读。

（2）FD_WRITE 事件通知：通知可以写数据。

（3）FD_ACCEPT 事件通知：通知监听套接字有连接请求可以接受。

（4）FD_CONNECT 事件通知：通知请求连接的套接字，连接的要求已被处理。

（5）FD_CLOSE 事件通知：通知套接字已关闭。

（6）FD_OOB 事件通知：通知将有带外数据到达

处理：按照 Windows的消息驱动机制，MFC 框架应当把消息发送给相应的套接字对象，并调用作为该对象成员函数的事件处理函数。事件与处理函数是一一映射的。

在 afxSock.h 文件中的 CAsyncSocket 类的声明中，定义了与这六个网络事件对应的事件处理函数。

virtual void OnReceive(int nErrorCode); 对应 FD_READ 事件

virtual void OnSend(int nErrorCode); 对应 FD_WRITE 事件

virtual void OnAccept(int nErrorCode); 对应 FD_ACCEPT 事件

virtual void OnConnect(int nErrorCode); 对应 FD_CONNECT 事件

virtual void OnClose(int nErrorCode); 对应 FD_CLOSE 事件

virtual void OnOutOfBandData(int nErrorCode); 对应 FD_OOB 事件

六、CSocket 类如何通过 Carchive 对象来进行数据传输？

答：使用 CArchive 对象和套接字一起进行数据传输工作，必须使用流式套接字

七、说明 CSocket 类的编程模型。

答：下面给出针对流式套接字的 CSocket 类的编程模型

1. 服务器端

（1）CSocket sockServ; // 创建空的服务器端监听套接字对象。

// 用众所周知的端口，创建监听套接字对象的底层套接字句柄。

（2）sockServ.Create(nPort);

（3）sockServ.Listen(); // 启动对于客户端连接请求的监听。

（4）CSocket sockRecv; // 创建空的服务器端连接套接字对象。

sockServ.Accept(sockRecv); // 接收客户端的连接请求，并将其他的任务转交给连接套接字对象。

（5）CSockFile* file ;

file = new CSockFile(&sockRecv); // 创建文件对象并关联到连接套接字对象。

（6）CArchive* arIn, arOut;

arIn = CArchive(&file, CArchive::load); // 创建用于输入的归档对象，

arOut = CArchive(&file, CArchive::store); // 创建用于输出的归档对象。

// 归档对象必须关联到文件对象。

（7）arIn >> dwValue; // 进行数据输入。

adOut << dwValue; // 进行数据输出。输入或输出可以反复进行。

（8）sockRecv.Close();

sockServ.Close(); // 传输完毕，关闭套接字对象。

2. 客户端

（1）CSocket sockClient; // 创建空的客户机端套接字对象。

（2）sockClient.Create( ); // 创建套接字对象的底层套接字。

（3）sockClient.Connect(strAddr, nPort); // 请求连接到服务器。

（4）CSockFile* file ;

file = new CSockFile(&sockClent); // 创建文件对象，并关联到套接字对象。

（5）CArchive* arIn, arOut;

arIn = CArchive(&file, CArchive::load); // 创建用于输入的归档对象，

arOut = CArchive(&file, CArchive::store); // 创建用于输出的归档对象。

// 归档对象必须关联到文件对象。

（6）arIn >> dwValue; // 进行数据输入。

adOut << dwValue; // 进行数据输出。输入或输出可以反复进行。

（7）sockClient.Close(); // 传输完毕，关闭套接字对象。

八、说明使用 MFCAppWizard 创建客户机端应用程序框架具体的步骤？

答：1. 使用 MFC AppWizard 创建客户端应用程序框架。

创建的程序是一个基于对话框的 Win32 应用程序，将自动创建两个类，应用程序类 CTalkcApp，对应的文件是 talkc.h 和 talkc.cpp ；对话框类 CTalkcDlg，对应的文件是 talkcDlg.h 和 talkcDlg.cpp 。支持 Windows Socket，使用共享的 DLL实现 MFC42.DLL。

2. 为对话框界面添加控件对象

利用控件面板在程序的主对话框界面中添加相应的可视控件对象，然后修改控件的属性

3. 为对话框中的控件对象定义相应的成员变量

4. 创建从 CAsyncSocket 类继承的派生类

（1）选择菜单“ 插入 / 新建类 ”，进入“ New Class ”对话，选择或输入以下信息：

Class Type ：选择 MFC Class

Class Infoumation 下的 Name: 输入 CMySocket

Class Infoumation 下的 Base class ：选择 CAsyncSocket

点击“ OK” 按钮，系统会自动生成 CMySocket类对应的包含文件 MySocket.h 和 MySocket.cpp 文件，在 VC界面的 Class View 中就可以看到这个类。

（2）利用类向导 ClassWizard 为这个套接字类添加响应消息的事件处理成员函数，此程序中需要添加 OnConnect、OnClose 和 OnReceive 三个函数。

（3）为套接字类 CMySocket 类添加一般的成员函数和成员变量。对这个套接字类，添加一个私有的成员变量，是一个对话框类的指针。 private:CTalkcDlg * m_pDlg; 再添加一个成员函数： void SetParent(CTalkcDlg * pDlg);

（4）手工添加其他代码

对于 MySocket.h ，应在文件开头，添加对于此应用程序对话框类的声明： class CTalkcDlg;

对于 MySocket.cpp ，有四处添加：

① 应在文件开头，添加包含文件说明。这是因为此套接字类用到了对话框类的变量。

#include “ TalkcDlg.h ”

② 在构造函数中，添加对于对话框指针成员变量的初始化代码：

CMySocket:：CMySocket() { m_pDlg = NULL; }

③ 在析构函数中，添加对于对话框指针成员变量的初始化代码：

CMySocket:：~CMySocket() { m_pDlg = NULL; }

④ 为成员函数 setParent 和事件处理函数 OnConnect、OnClose 和 OnReceive 添加代码。

5. 为对话框类添加控件对象事件的响应函数

6. 为 CTalkcDlg 对话框类添加其它的成员函数和成员变量

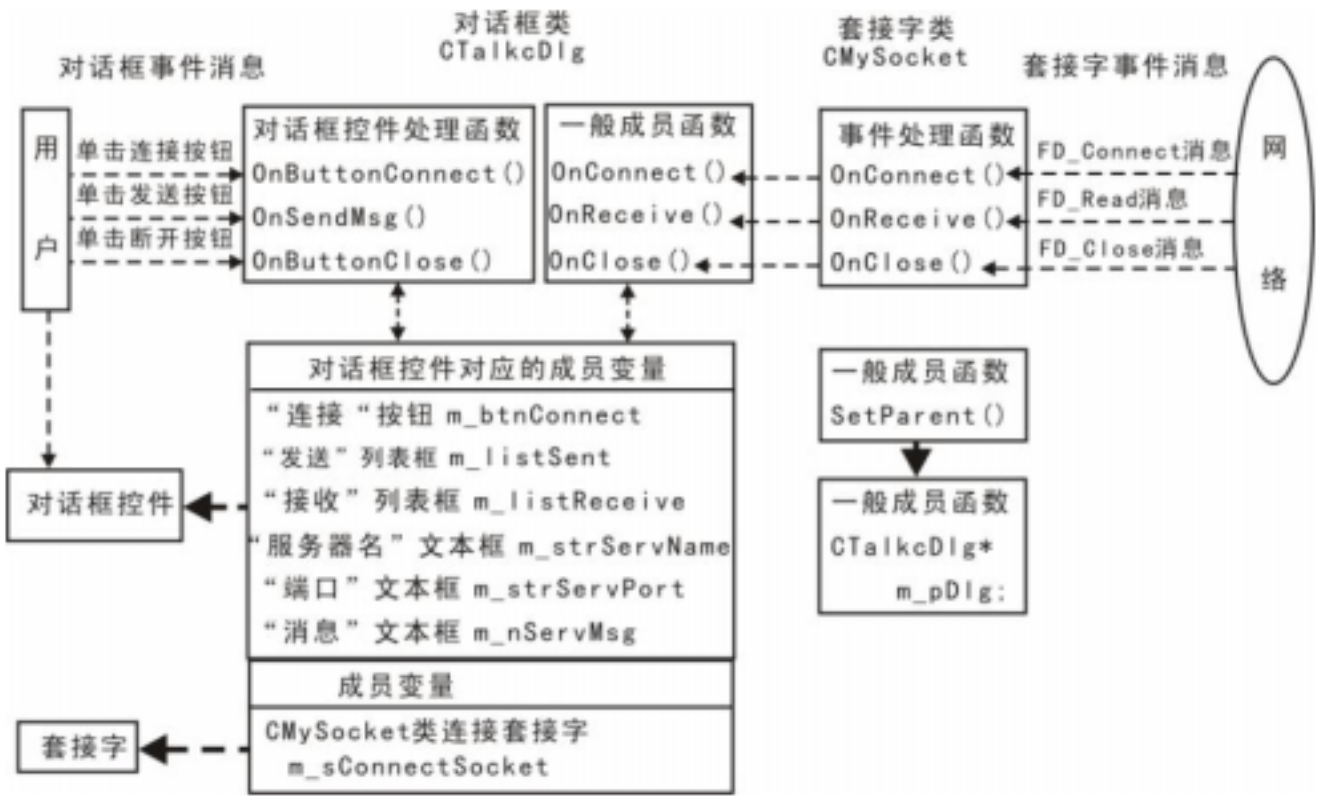
成员变量： CMySocket m_sConnectSocket; // 用来与服务器端连接的套接字。

成员函数： void OnClose(); void OnConnect(); void OnReceive();

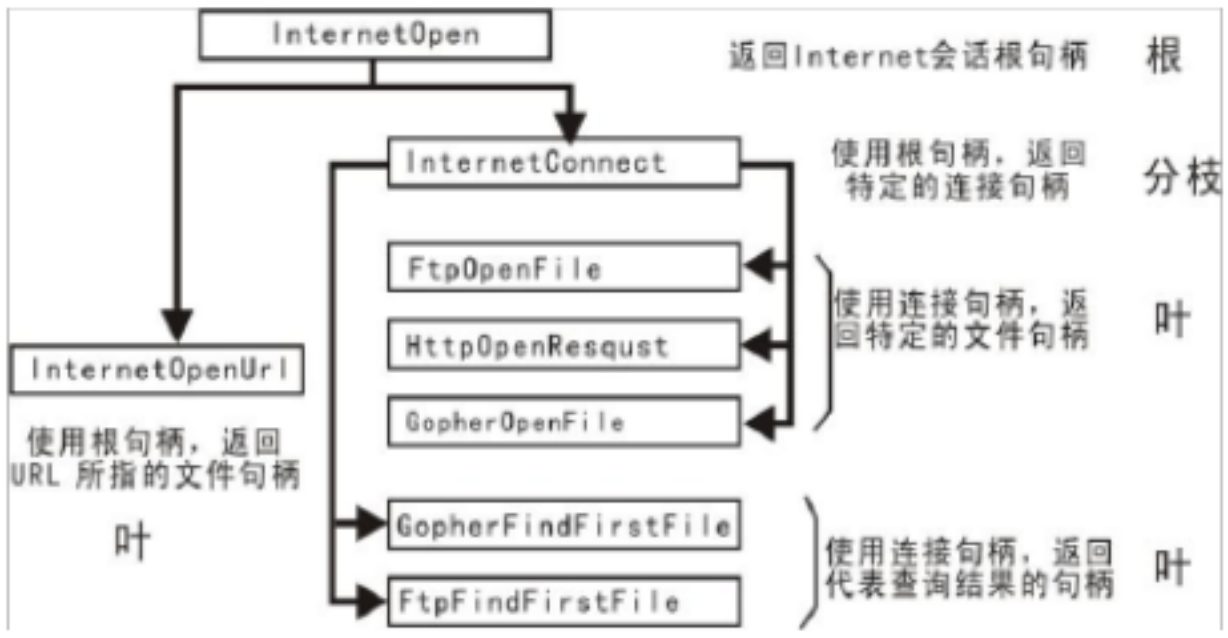
7. 手工添加的代码

在 CTalkcDlg 对话框类的 talkcDlg.h 中添加对于 MySocket.h 的包含命令，来获得对于套接字支持：

#include “ MySocket.h ”
在 CTalkcDlg 对话框类的 talkcDlg.cpp 中添加对于控件变量的初始化代码：
// TODO: Add extra initialization here
// 用户添加的控件变量的初始化代码
BOOL CTalkcDlg::OnInitDialog()
{m_strServName="localhost"; // 服务器名 = localhost
m_nServPort=1000; // 服务端口 = 1000
UpdateData(FALSE); // 更新用户界面
// 设置套接字类的对话框指针成员变量
m_sConnectSocket.SetParent(this);}
8. 添加事件函数和成员函数的代码
主要在 CTalkcDlg 对话框类的 talkcDlg.cpp 中和 CMySocket类的 Mysocket.cpp 中，添加用户自己的事件函数和成员函数的代码。
9. 进行测试。
九、说明点对点交谈的客户机端程序的类与消息驱动关系？
答: 下图说明了点对点交谈的客户端程序的类与消息驱动关系

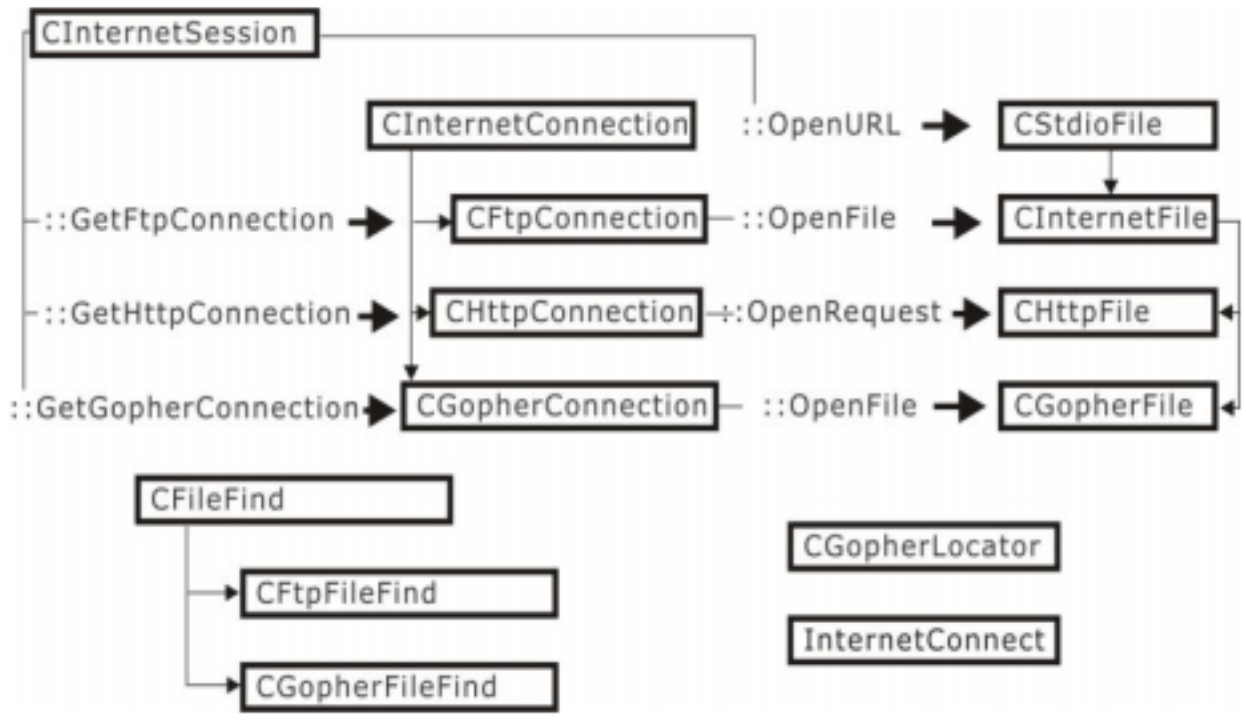


第六章
1. 说明了各种 HINTERNET句柄形成的树形体系
答：



2. 如何获取 WinInet API 函数执行的错误信息？
答：从函数的返回值来说， WinInet API 主要有两种类型。一种函数的返回值类型是 HINTERNET句柄型，一种函数的返回值类型是布尔型。应用程序可以根据函数执行后的返回值来判断函数的执行是否成功。
对于前一种函数，当函数执行成功时，会返回一个有效的句柄；当函数失败时，则返回 NULL。
对于后一种函数，当函数执行成功时，返回 TRUE；当函数失败时，返回 FALSE。
在函数调用失败后，用户往往需要了解出错的具体原因，应用程序可以随即调用 GetLastError 函数来获取更具体的错误信息。
3、说明使用 WinInet API 编制 FTP客户机端应用程序的一般步骤
答：（1）调用 InternetAttemptConnect 函数测试主机与 Internet 的连接状态，

（2）调用 InternetOpen 函数，创建 HINTERNE会话根句柄。
（3）创建 FTP会话句柄
4、 MFC WinInet 所包含的类有哪些？
答：1、CInternetSession 类，2. 连接类，包括 CInternetConnection 类和它的派生类 CFtpConnection 类、CHttpConnection 类、和 CGopherConnection 类；3 文件类，首先包括 CInternetFile 类和由它派生的 CHttpFile 类和 CGopherFile 类，另外，由 CFileFind 类派生的用于文件查找的 CFtpFileFind 类和 CGopherFileFind 类也应归入文件类的层次。 4. CInternetException 类
5、 MFC WinInet 各种类之间的关系
答：如下图，其中细线箭头从基类指向继承类，表示了类的派生关系；粗线箭头从函数指向它所创建的类对象



6、使用 WinInet 类编程的一般步骤是什么？
答：1）创建 CInternetSession 类对象，创建并初始化 Internet 会话。
（2）利用 CInternetSession 类的 QueryOption 或 SetOption 成员函数，可以查询或设置该类内含的 Internet 请求选项，这一步是可选。
（3）创建连接类对象，建立 CInternetSession 对象与网络服务器的连接，也就是应用程序与网络服务器的连接。
（4）创建文件检索类对象，对服务器进行检索
（5）如果需要使用异步操作模式，可以重载 CInternetSession 类的 OnStatusCallback 函数，并启动应用程序使用状态回调机制。重载相关函数，加入自己的代码。
（6）如果还想更紧密地控制对于服务器文件的访问，可以进一步创建文件类对象实例，完成文件查找或文件读写操作。
（7）创建 CInternetException 类对象实例，处理错误。
（8）关闭各种类，将资源释放给系统。
第七章
1、 WinSock的两种 I/O 模式是什么？各有什么优缺点，缺点如何解决？
答：WinSock的两种 I/O 模式是：“ 阻塞 ”模式（ Blocking Mode ）或“ 非阻塞 ”模式，又称为同步模式或异步模式。
阻塞模式的优点：阻塞套接字的 I/O 操作工作情况比较确定，无非是调用、等待、 返回。大部分情况下，I/O 操作都能成功地完成，不过就是花费了等待的时间。因而比较容易使用，容易编程。
缺点：在应付诸如需要建立多个套接字连接来为多个客户服务的时候， 或在数据的收发量不均匀的时候，或在输入输出的时间不确定的时候，却显得性能低下，甚至无能为力。
非阻塞模式的优点：能应付诸如需要建立多个套接字连接来为多个客户服务，可以处理数据的收发量不均匀、输入输出的时间不确定等问题。
非阻塞模式的缺点： 用非阻塞套接字， 需要编写更多的代码， 因为必须恰当地把握调用 I/O 函数的时机，尽量减少无功而返的调用，还必须详加分析每个 Winsock 调用中收到的 WSAEWOULDBLOCK 错误，采取相应的对策，这种 I/O 操作的随机性使得非阻塞套接字显得难于操作。
解决方法：对于非阻塞的套接字工作模式，进一步引入了五种“ 套接字 I/O 模型 ”，有助于应该程序通过一种异步方式，同时对一个或多个套接字上进行的通信加以管理。
对于阻塞的套接字工作模式，则进一步引入了多线程机制。

2、 简述 Win32 操作系统下的多进程多线程机制。
答：Win32 操作系统还支持同一进程的多线程。在一个 Windows进程内，可以包含多个线程。一个线程(thread) 是进程内的一条执行路径，具体地说，是一个应用程序中的一条可执行路径，往往是应用程序中的一个或多个函数。一个进程中至少要有一个线程，习惯将它称为主线程。任何一个应用程序进程都有一个主线程。
3、 多线程机制在网络编程中如何应用？
答：如果一个应用程序，有多个任务需要同时进行处理，那就最适合使用多线程机制。对于网络上客户机软件，采用多线程的编程技术，能克服在单线程的编程模式下，由于阻塞等待而产生的客户程序就不能及时响应用户的操作命令的问题。对于网络上服务器软件，采用多线程的编程技术，能更好地为多个客户服务。即便是对于一个客户，采用多线程机制也能大大提高应用程序的运行效率。
4、 说明用户接口线程和工作线程的概念和特点。
答：用户接口线程：通常用来处理用户输入产生的消息和事件，并独立地响应正在应用程序其它部分执行的线程们产生的消息和事件。用户接口线程包含一个消息处理的循环，以应对各种事件。
工作线程：适用于处理那些不要求用户输入并且比较消耗时间的其他任务。对用户来说，工作线程运行在后台。这就使得工作线程特别适合去等待一个事件的发生。
5、 简述创建 MFC的工作线程所必需的步骤。
答：（1）是编程实现控制函数。
（2）是创建并启动工作线程。
（3）创建工作线程的例子。
（4）创建工作线程的一般模式。
6、简述创建并启动用户界面线程所必需的步骤。
答：1. 从 CWinThread类派生出自己的线程类
2 ．改造自己的线程类
3. 创建并启动用户界面线程
7、如何正常终止线程？如何提前终止线程？
答：正常终止线程：执行完毕时退出控制函数，并返回三个用来表示终止原因的值即可。
提前终止线程：从线程内调用 AfxEndThread 函数，就可以强迫线程终止。
第八章
1、非阻塞套接字的五种“ 套接字 I/O 模型 ”：
select （选择）、
WSAAsyncSelect（异步选择）、
WSAEventSelect（事件选择）、
Overlapped I/O （重叠式 I/O ）
以及 Completion port （完成端口）。
2、 select 模型的操作步骤
select （选择）模型是 Winsock 中最常见的 I/O 模型。它的中心思想是利用 select 函数，实现对多个套接字 I/O 的管理。
用 select 操作一个或多个套接字句柄，一般采用下述步骤：
（1）使用 FD_ZERO宏，初始化自己感兴趣的每一个 fd_set 集合。
（2）使用 FD_SET宏，将要检查的套接字句柄添加到自己感兴趣的每个 fd_set 集合中，相当在指定的 fd_set 集合中，设置好要检查的 I/O 活动。
（3）调用 select 函数，然后等待。 select 完成返回后，会修改每个 fd_set 结构，删除那些不存在待决 I/O 操作的套接字句柄，在各个 fd_set 集合中返回符合条件的套接字。
（4）根据 select 的返回值，使用 FD_ISSET宏，对每个 fd_set 集合进行检查，判断一个特定的套接字是否仍在集合中，便可判断出哪些套接字存在着尚未完成（待决）的 I/O 操作。
（5）知道了每个集合中“ 待决 ”的 I/O 操作之后，对相应的套接字的 I/O 进行处理，然后返回步骤 1，继续进行 select 处理。
3、 WSAAsyncSelect异步 I/O 模型的编程步骤
⑴用 CreateWindow 函数创建一个窗口，并为该窗口提供一个窗口回调例程。
⑵调用 WSAAsyncSelect 函数创建套接字，指定关注的套接字、窗口句柄、打算接收的消息，以及程序

感兴趣的套接字事件。成功执行 WSAAsyncSelect函数，就打开了窗口的消息通知，并注册了事件。
⑶WSAAsyncSelect函数执行时，当注册的套接字事件之一发生时，指定的窗口会收到指定的消息，并自动执行该窗口的回调例程，可在其中添加自己的代码，处理相应的事件。

4、WSAEventSelect 事件选择模型的编程步骤

⑴创建事件对象句柄

事件选择模型要求应用程序针对每一个套接字，首先创建一个事件对象。创建方法是调用 WSACreateEvent函数，定义如下：

```
WSAEVENT  WSACreateEvent(void);
```

⑵关联套接字和事件对象，注册关心的网络事件

将事件对象句柄与某个套接字关联在一起，同时注册感兴趣的网络事件类型。调用 WSAEventSelect 函数，函数的定义为：

```
int WSAEventSelect(
SOCKET  s,
WSAEVENT  hEventObject,
long  lNetworkEvents
);
```

⑶等待网络事件触发事件对象句柄的工作状态

调用 WSAWaitForMultipleEvents 函数，等待网络事件触发事件对象句柄的工作状态。该函数的定义：

```
DWORD WSAWaitForMultipleEvents(
DWORD  cEvents,
const  WSAEVENT FAR *  lphEvents,
BOOL  fWaitAll,
DWORD  dwTimeout,
BOOL  fAlertable
);
```

⑷检查套接字上所发生的网络事件类型

调用 WSAEnumNetworkEvents函数，检查套接字上发生了什么类型的网络事件。该函数定义如下：

```
int  WSAEnumNetworkEvents(
SOCKET  s,
WSAEVENT  hEventObject,
LPWSANETWORKEVENTS  lpNetworkEvents
);
```

⑸处理网络事件

在确定了套接字上发生的网络事件类型后，可以根据不同的情况做出相应的处理。完成了对 WSANETWORKEVENTS结构中的事件的处理之后，应用程序应在所有可用的套接字上，继续等待更多的网络事件。

应用程序完成了对一个事件对象的处理后，便应调用 WSACloseEvent函数，释放由事件句柄使用的系统资源。函数的定义如下：

```
BOOL  WSACloseEvent(WSAEVENT  hEvent);
```

该函数也将一个事件句柄作为自己唯一的参数，并会在成功后返回 TRUE，失败后返回 FALSE。

第九章

1、什么是 HTTP会话？HTTP会话周期由哪些阶段组成？

HTTP是超文本传输协议（ Hypertext Transfer Protocol ）的简称， HTTP协议也是基于 TCP/IP 的客户 / 服务器协议。主要是用于传输文件的协议。

HTTP会话周期由连接、请求、响应和断开 4 个阶段组成。

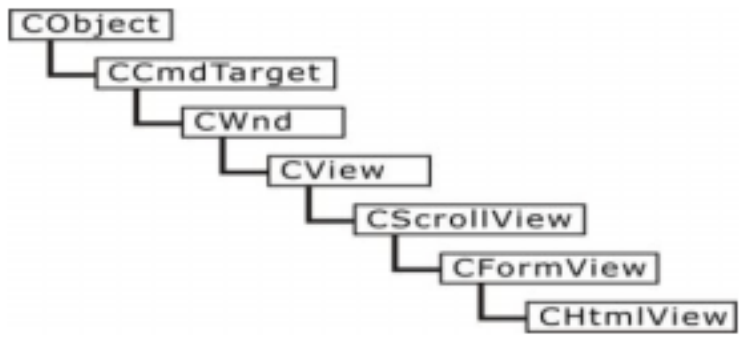
⑴建立 TCP/IP 连接

⑵Web客户机向服务器发送 HTTP请求

⑶服务器向客户机回送 HTTP响应

⑷断开 TCP/IP 连接

2、CHtmlView 类的继承关系



3、利用 MFC AppWizard创建 SDI 或 MDI的步骤

（1）第一步：选择应用程序结构

①为应用程序选择三种结构之一，单文档（ Single Document SDI ），多文档（Multiple DocumentMDI），基于对话框（ Dialog Based ）。Web浏览器型的应用程序一般选择 SDI 类型，以下的操作都假定选择了 SDI 类型。

②决定应用程序是否要支持 MFC的文档 / 视图结构（ document/view architecture ），实际上必须选择支持，因为不支持文档 / 视图结构的应用程序不能打开磁盘文件和从 CWnd类继承的窗口区，并且后面的步骤都是无效的。

③选择资源中的文本所使用的语言，应选择“ 中文 [中国] ”

作完这三件事以后，点击 Next 按钮

（2）第二步：选择应用程序支持的数据库

①为程序选择一种数据库支持，有四个选项。没有（ None），头文件支持（ Header file support ），带有文件支持的数据库视图 （Database View with file support ）或没有文件支持的数据库视图 （ Database View without file support ）。

②如果选择了数据库支持，点击数据源（ Data Source ），在外部 ODBC数据库、 DAO数据库和 OLE DB数据库中选择一个，然后选择相应的数据源和数据库表选项。

③点击 Next 按钮，出现第三步对话框

（3）第三步：选择应用程序支持的复合文档

①选择应用程序支持的复合文档（ compound document）类型。有五个选项：

None：不支持复合文档

Container ：容器，一个 OLE 2.0 风格的文档容器。

Mini-server ：最小服务器，一个 OLE服务器，但不能作为一个独立的程序来运行。

Full-server ：完全服务器，一个 OLE服务器，可以作为独立程序运行。

Both container and server ：容器和服务器两者。

②也可以选择选项来启用标准的 ActiveX 资源，增加额外的自动命令到应用程序的菜单条中。

③点击 Next 按钮，出现第四步对话框，

（4）第四步：选择应用程序的接口特性

①为程序选择基本的用户接口特性。 如快捷的工具条、 初始状态栏、 打印和打印预览、 内容敏感的帮助、3D控件、 Windows套接字等，都是复选框。

②要使用的工具条形式， IE4.0 ReBars 或者 MFC常规的工具条。

③最近打开的文件列表数目，默认值是 4。

④如果你想要修改你的程序的名字和扩展名，或者为你的程序调整用户接口窗口框架风格，点击 Advanced。

⑤点击 Next，出现第五步对话框。如图 9.9 所示。

（5）第 5 步：决定三个问题

①选择工程风格：

Windows Explorer ：左边是一个树形视图，右边是一个列表视图。

MFC Standard：为文件视图提供一个区域。

②选择是否希望应用程序向导在源文件中产生注释，以便指导你编写程序。

③选择使用 MFC库的选项

As a shared DLL ：库作为共享的 DLL

As a statically linked library ：库作为静态链节的 DLL

注意，静态链接到 MFC的库仅仅在 VC++的专业版和企业版中支持（ Professional and Enterprise Editions ）。

④点击 Next，出现 MFC AppWizard – Step 6 of 6 对话框。如图 9.10 。

（6）第 6 步：决定类名和基类

①如果你想要改变默认的由应用程序向导提供的类名，基类，头文件，或者实现文件的名字，输入新的名字，要改变基类，选择你的程序的视图类。

②点击 Finish 按钮，出现 New Project Information 对话框。如图 9.11。说明了应用程序的类型，创建的类、应用程序的特性和工程的目录。

③点击 OK按钮，工程创建成功，进入 VC++集成的开发环境。

4. 创建一个 Web浏览器型的应用程序的一般步骤

⑴利用 MFC AppWizard生成应用程序框架

生成的工程框架包含四个类：

应用程序类， CMyWebApp，对应 myWeb.h和 myWeb.cpp文件。

框架类， CMainFrame，对应 MainFrm.h 和 MainFrm.cpp 文件。

文档类， CMyWebDoc，对应 myWebDoc.h和 myWebDoc.cpp文件。

HtmlView 类， CMyWebView，对应 myWebView.h和 myWebView.cpp文件

⑵修改菜单

修改菜单，添加用户需要的功能条目。在工作区中选择 ResourceView 卡，选择 Menu，双击菜单控件的名字（ IDR_MAINFRAME）。

⑶修改工具栏

在工具栏上增加新的按钮， 以便快速执行所需的功能。 在工作区中选择 ResourceView 卡, 选择 ToolBar , 双击工具条控件的名字（ IDR_MAINFRAME），

⑷为控件添加事件处理函数

当用户点击菜单条目，或者点击相应的快捷按钮时，应能引起程序的反映，这就需要给这些控件添加事件处理函数。进入类向导，选择“ MessageMaps”卡。在“ Class Name”下拉框中选择 CMyWebView 类，

⑸添加事件处理函数的代码

⑹为应用程序添加 URL定位功能

⑺解决点击超链接时地址栏的同步问题

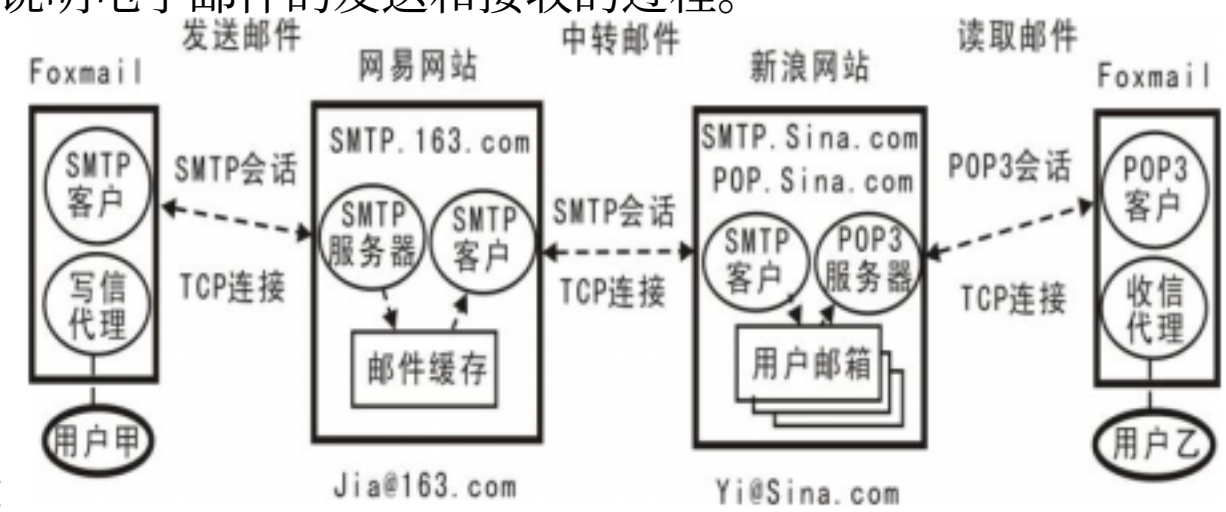
第十章

1、简述电子邮件系统的构成。

答：一个电子邮件系统包括三个主要的构件， 即用户代理， 邮件消息传输代理， 和电子邮件使用的协议。

2、说明电子邮件的发送和接收的过程。

答：



（1） 甲运行 Foxmail 软件，进入它的窗口用户界面，利用其中发信代理的相关功能，编辑要发送的邮件内容。然后，填好收信人的电子邮箱地址、邮件主题等信息，单击“ 发送 ”按钮。

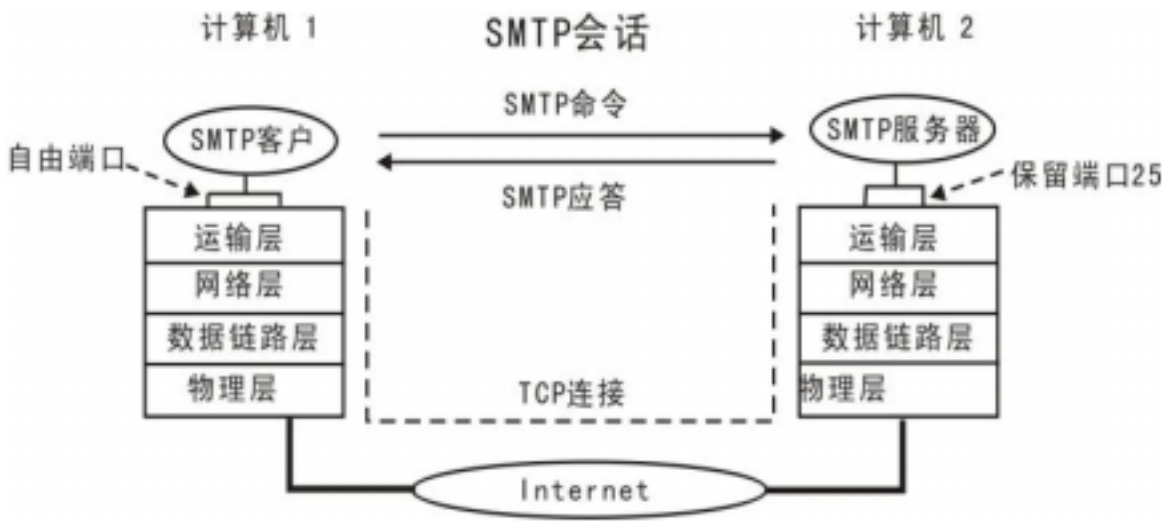
（2） Foxmail 软件中的 SMTP客户机进程向网易网站的 SMTP服务器发出连接请求，经过三次握手的过程建立了 TCP连接。然后进入 SMTP客户机与 SMTP服务器的会话过程，通过网络， smtp 客户机发出请求命令， SMTP服务器以响应作答。

（3） 一旦收到了邮件，就立即按照该邮件收件收信人的电子邮箱地址，向网站的 SMTP服务器进程发起建立 TCP连接的请求。经过 SMTP客户机与服务器的会话过程，将邮件从发送站点传到接收站点，接收站点的服务器将邮件存放到乙的电子邮箱中，等待乙来读取。

（4） 当乙读取邮件时， Foxmail 软件中的 POP3客户机进程立即向接收站点的 POP3服务器进程发出请求，建立 TCP连接，然后，按照 POP3，进入 POP3会话过程，将邮件或邮件的副本从位于接收的电子邮箱取出，发送到乙的 pc.

（5） 乙方 Foxmail 中的收信代理向用户显示邮件，乙可以利用它的相关功能对此邮件做进一步处理。

3、电子邮件系统有什么特点？
答：①是一种异步的通信系统，不像电话，通话的双方都必须在场。
②使用方便，传输迅速，费用低廉，不仅能传输文字信息，还能附上声音和图像。
③在电子邮件系统的实现中， ISP 的服务器必须 7X24小时地不间断地运行， 这样才能保证用户可以随时发送和接收信件，而发送或接收电子邮件的用户则随意。
4、简述 SMTP客户机与 SMTP服务器之间的会话过程，以及 SMTP会话具有的特点。
答：如下图：说明了 SMTP客户与 SMTP服务器之间的会话过程



SMTP会话具有以下特点：
（1）会话的过程采用交互式的请求应答模式，客户发送命令，服务器回送应答。
（2）客户发送的命令和服务器回送的应答都是纯文本形式，有一定格式。
（3）针对客户的每个命令，服务器总要返回一定的响应码，表示服务器是否接受或执行了客户端命令。
（4）会话过程有一定的顺序
5、使用 WinSock来实现电子邮件客户机程序与服务器的会话的步骤是什么？
答：（1）启动 SMTP服务器，在指定的传输层端口监听客户端的连接请求，为 SMTP服务器保留的端口是25。
（2）客户端设置 Winsock 连接的 IP 地址或域名，指定端口号，主动发出连接请求，连接到 SMTP服务器。比如，网易的 SMTP服务器的域名是 smtp.163.com，监听端口是 25。
（3）服务器接收客户端的连接请求， 并发回响应。客户端应收到类似 220 BigFox ESMTP Service ready 这样的信息，这就说明客户端已经与服务器建立 TCP/IP 连接，成功地实现了第一步。
（4）客户端和服务器分别向对方发送数据。
（5）客户端或服务器分别读取自己缓冲区中的数据。
（6）以上两步是 SMTP会话的主要部分，要按照 SMTP协议的规定，按照一定顺序，客户向服务器发送命令，服务器向客户发送应答，以上两步要多次重复。
（7）会话完毕，关闭客户端和服务器之间的连接。
6、说明电子邮件信件结构。
答：从组织上看， RFC822将电子邮件信件内容结构分为信头和信体两大部分，中间用一个空白行（只有CRLF符的行）来分隔。第一部分称为信件的头部，包括有关发送方、接收方和发送日期等信息。第二部分称为信件的体部，包括信件内容的正文文本。信头是必须的，信体是可选的，即信体可有可无。
7、建议的使用在信头中的字段顺序是什么？
答：一般来说，信头中的字段不要求任何特定的顺序， 但 Received、Return-Path 和 Resent-* 字段例外。Received 字段是在信件传送过程中，由经过的一系列的 MTA投递前增加的，必定放在信件的开始。Return-Path 字段只是在最后一个 MTA投递前增加的，也放在信件的开始。因为这两类字段是为诊断问题提供跟踪信息的，所以它们的位置必须保持不变。另外，给信件增加的任何重发字段，即以 Resent-前缀开头的字段也必须放在信件的开始。
8、说明 MIME的基本思想。
答：MIME的基本思想是：第一，不改动 SMTP和 POP3等电子邮件传输协议； 第二，仍然要继续使用 RFC822的格式来传输邮件。
9、说明 Base64编码算法。
答：在进行 Base64 的编码时，首先要将被编码的数据看成一个字节序列，不分行。然后按照以下三个步骤反复进行编码。
第一步：顺次从要编码的字节序列数据中，取出 3 个字节，作为一组，该组包含 24bit 。

第二步：将一组的 24 个 bit ，再顺次分为四个 6bit 的小组。并在每小组的前面补两个零，形成四个字节。显然这 4 个字节的值都在 0-63 范围内。
第三步：根据小组的每个字节的数值， 按照表 10.4 的对应关系，将它们分别转换为对应的可打印 ASCII 字符。
顺次对要编码的数据字节序列，重复进行以上处理，直到所有的数据编码完成为止。
10、说明 Quoted-printable 编码算法。
答：这种编码称为可打印的引用编码 (Quoted-printable encoding) ，被编码的数据以 8BIT 的字节为单位，这种编码方法的要点就是对于所有可打印的 ASCII 码，除了特殊符号“ = ”以外，都不改变。编码算法的要点是：
（1）如果被编码数据字节的值在 33 到 60 之间(字符 ! 至字符 <) ，或在 62 到 126 之间 (字符 >至字符 ~) ，这部分字符是可打印的，则该数据字节编码为 7bit 的对应 ASCII 字符，实际就是将最高位去掉。
（2）其他的数据，包括“ = ”字符，空格， ASCII 码在 0-32 的不可打印字符，以及非 ASCII 码的数据，都必须进行编码。被编码的数据以 8bit 的字节为单位，先将每个字节的二进制代码用两个 16 进制数字表示，然后在前面加上一个等号“ = ”，就是该字节的编码。
（3）如果要将编码后的数据分割成 76 字符的行，可以在分割处插入等号“ = ”和 CRLF。此等号也要计算在 76 个字符中。
11、简述 POP3会话的 3 个状态。
答：POP3会话一共有 3 个状态：验证状态，事务状态和更新状态。 每个状态都是会话过程中的特定阶段。
当连接服务器后， POP3会话首先进入验证状态， 在这个阶段里，可以使用 USER PassQuit 这三个 POP3 命令，客户段送交用户名和口令，服务器验证是否合法。
通过服务器验证后，服务器锁定该用户的信箱，从而防止多个 POP客户端同时对此邮箱进行邮件操作，比如删除，取信等。但是可以让新的邮件加入。这时会话过程转变为事务状态，在事务状态客户端可用的 POP3命令有： Noop Stat Quit List Retr Top Dele Rset Uidl 。使用这些命令进行各种邮件操作， POP对话的大部分时间都处在事务状态中。
当客户发出 Quit 命令后，结束事务状态， POP会话过程进入更新状态。在事务状态进行的一些操作，最终在更新状态中才得以体现。比如在事务状态使用 Dele 命令删除邮件，实际服务器并没有将邮件删除，只是做了一个删除标志；到了会话过程的更新状态，邮件才被删除。更新状态只是会话中的一个过程，该状态没有可使用的命令，目的是用户在事务状态后用以确认已经进行的操作。在进入该状态后，紧接着就完成了 POP3的会话过程，断开了与服务器的连接。要注意，由于异常原因导致的与服务器终止对话并没有进入更新状态。在事务状态删除的邮件没有被删除，下次进入信箱时邮件还是存在的。
12、接收电子邮件编程的一般步骤是什么。
答：首先利用 Winsock 连接上 POP3服务器，然后：
1. 使用 USER命令发送用户信箱名。
2. 使用 PASS命令发送信箱密码。如果密码和信箱不匹配，必须从上一步骤重新开始。
3. 对信箱邮件进行操作。
此阶段称为事务状态，在这一个阶段，有许多 POP3命令可以使用，大体分为下面几类：
（1）取得信箱及邮件状态的命令
Stat ：取得信箱大小信息
List ：取得邮件大小信息
Uidl ：取得邮件的唯一标识符。
（2）取得邮件内容的命令
Retr ：从服务器取回邮件
TOP: 取邮件信头和信体的前 N行。
（3）对邮件进行操作的命令
Dele：为邮件做删除标记。
Rset：复位 POP会话。
4. 接收邮件完毕，发送 QUIT命令，结束 POP3对话。