Visual Basic 网络通信协议分析与应用实现

汪晓平 钟军 等编著

人民邮电出版社

前　言

Internet 有强大的通信功能，如文件传送、远程登录、E-mail、Internet Phone 和 Internet Fax 等，它使传统的电信产业发生了巨大变化。同时，它是一个大型信息资源库，所含信息不仅包罗万象，而且日新月异。尤其是 WWW（World Wide Web）的出现使全球信息联成一体，并使千家万户可以随时共享这一人类伟大的资源。对于任何一位软件开发者来说，学习开发 Internet 应用程序已经是一件刻不容缓的事情了。串口通信虽然历史悠久，但是它在目前的通信领域，仍然占据着重要的地位，在许多的工业领域广为使用。

Visual Basic 简单易懂，只要写少量的代码，就可以实现特定功能。不仅是对于专业程序员而且是对于那些特别希望能够尽早看到自己的编程成果的业余程序员来说，这无疑是个好消息。同时可以通过 Visual Basic 进行程序的快速开发，可以迅速增加对新知识和内容的体验，而不是把大部分的精力放在怎么优化程序代码，以及解决编程过程中出现问题等细节上。

对于网络和通信编程而言，使用 Visual Basic 无疑是一个能够快速开发网络应用程序的选择。本书主要内容针对 Visual Basic 进行网络高级开发，本书的特点体现在以下几个方面：

- 突出网络高级编程，重点讲解网络编程中的使用技巧及难点；
- 针对比较流行的协议如 Telnet、FTP、HTTP、E-mail 等进行详细的理论讲解；
- 每一章都有丰富的实例讲解，使读者通过实例深入了解各个协议的内容。

本书共包含 10 章，各章内容的介绍如下：

第 1 章：网络编程基本知识

本章主要介绍的是有关 TCP/IP 协议、Winsock 编程的基础知识，安排本章的内容是为了让读者能对后面的程序更容易理解。

第 2 章：网络与通信控件

本章介绍了 Visaul Basic 中进行网络通信开发的各种控件的属性、方法和事件，为后面进行各种网络通信应用的开发提供坚实的基础。

第 3 章：实现网络基本应用

本章介绍几种读者比较感兴趣的网络基础应用，这些都是进行网络编程不可缺少的，如端口扫描程序、Ping 程序的实现、根据域名获得计算机的 IP 地址、获取网卡地址、超级链接和发送 E-mail 等。

第 4 章：TCP/UDP 编程实例

本章是对第 2 章的各种网络技术的综合，通过各种聊天程序的开发，让读者对各种网络技术的开发有一个升华，弄清楚了这些编程的原理，也是进行后面各种网络协议开发的基础。

第 5 章：E-mail 协议及高级编程

本章首先通过介绍 SMTP 或 POP3 了解收发 E-mail 的工作原理，结合程序实现收发 E-mail，接着详细分析 E-mail 的内容结构及格式，在这一章中还详细讲解了发送附件的 E-mail 格式以及各种编码解码的算法和相应的源程序，并且对 E-mail 的乱码进行了分析。最后讲解

设计 E-mail 程序的另一种方法——MAPI。对于每个理论（SMTP 发送 E-mail，POP3 接收 E-mail，E-mail 的编码解码，发送 E-mail 附件，MAPI）都有相应的实例源程序介绍。

第 6 章：Telnet 协议及高级编程

本章不但详细讲解了 Telnet 的基本理论，而且给出了 BBS 客户端的源程序。

第 7 章：HTTP 协议及高级编程

本章提供了类似网络蚂蚁功能的源程序、WWW 服务器源程序、下载整个站点的应用程序的源代码等。同时向读者揭示了断点续传、代理服务器的工作原理。

第 8 章：FTP 高级编程

本章介绍了从使用 Internet Transfer 控件，到 FTP API 函数，甚至是使用 Winsock 进行较底层的通信应用开发。

第 9 章：RAS 高级编程

本章介绍了 RAS 的编程方法，包括对拨号网络的电话簿、拨号连接、挂断、以及获取拨号网络的动态 IP 等内容。

第 10 章：串口通信高级编程

本章首先介绍了如何利用 MSComm 控件进行数据通信、文件传输，从基本理论分析入手，最后提供功能丰富的串口通信程序，能进行文件传输、自动监控和应答等等。接着介绍了如何利用 Windows API 进行串口高级编程，介绍了各种 API 的含义、以及开发步骤，并在最后提供了一个串口通信程序。相信通过最后两章的学习，读者能够开发出各种功能的串口通信程序。

本书主要由钟军、汪晓平编写而成，此外参与资料收集与编写工作的还有张宏林、肖洪伟、李廷文、张增强、王洪涛、吴继刚、周学明、李闽溟、黄沙、宣小平、但正刚、张文毅、张小磊、胡昱、范国平、陈晓鹏、王凯封、潘邦传、王锐、闫卫东、赵明华、许福、施新刚、郑刚、李现勇、谭思亮、邹超群、郭瑞军、杨枝灵、彭坷珂、赵苏琦、徐建军、胡伟、刘江、王茹、阎海荣、刘理、谭春华、张益贞、刘韬、刘鹏、黄良大、周金萍、黄超、王非、冉光志、金海、赵振、孙越、纪慧波、杨茂林、董晓宇、王三暖、刘星等，在此一并表示感谢。

限于笔者的能力，书中错误、浅陋之处在所难免，恳请广大读者批评指正。本书责任编辑的 E-mail：zhanglike@ptpress.com.cn，作者的 E-mail：busywxp@163.com，欢迎联系讨论。

编者

图书在版编目（CIP）数据

Visual Basic 网络通信协议分析与应用实现/汪晓平，钟军等编著．—北京：人民邮电出版社，2003.1
ISBN 7-115-11004-2

Ⅰ．V… Ⅱ．①汪… ②钟… Ⅲ．BASIC 语言—程序设计 Ⅳ．TP312

中国版本图书馆 CIP 数据核字（2002）第 109071 号

内容提要

本书主要针对目前流行的 FTP、HTTP、E-mail、Telnet、ICMP、Modem 串口通信编程、拨号网络编程等内容进行详细的讲解，并结合大量的实例使读者能够深入地了解各种网络应用程序的开发技巧。除了深入剖析各种网络协议之外，本书还介绍了 Visual Basic 6.0 中各种开发网络通信的方法和技巧，以及各种网络通信的基础应用。

本书适合中高级 Visual Basic 程序员阅读、参考。

Visual Basic 网络通信协议分析与应用实现

◆ 编　　著　汪晓平　钟　军等
责任编辑　张立科

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ pptph.com.cn
网址　http://www.pptph.com.cn
读者热线　010-67180876
北京汉魂图文设计有限公司制作
北京　　印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：42.5
字数：1045 千字　　2003 年 1 月第 1 版
印数：1-　000 册　　2003 年 1 月北京第 1 次印刷

ISBN 7-115-11004-2/TP・3304

定价：62.00 元（附光盘）

目　　录

第 1 章　网络编程基础知识 1

1.1　网络的基本应用 1
1.2　TCP/IP 模型 1
1.3　UDP 数据包 5
1.4　TCP 数据包 5
1.5　小结 11

第 2 章　网络与通信控件 12

2.1　Winsock 规范 12
2.2　Winsock 控件 14
2.2.1　TCP 协议基础知识 14
2.2.2　UDP 基础知识 14
2.2.3　Winsock 控件属性 15
2.2.4　Winsock 控件方法 18
2.2.5　Winsock 控件事件 21
2.2.6　Winsock 控件的使用 24
2.3　Internet Transfer 控件 30
2.3.1　Internet Transfer 控件属性 30
2.3.2　Internet Transfer 控件方法 35
2.3.3　Internet Transfer 控件事件 39
2.3.4　Internet Transfer 控件的使用 40
2.4　MSComm 控件 45
2.4.1　MSComm 控件的属性 45
2.4.2　MSComm 控件的事件 49
2.4.3　利用 MSComm 控件通信步骤 49
2.5　Winsock API 49
2.5.1　Winsock API 的函数声明 49
2.5.2　WinsockAP 的函数使用 69
2.6　串口通信 API 73
2.6.1　打开和关闭串口 73
2.6.2　串口配置和串口属性 76
2.6.3　读写串口 86
2.6.4　通信事件 94
2.6.5　设备控制命令 96
2.7　小结 97

第 3 章　实现网络基本应用 98

3.1　端口扫描程序 98
3.2　Ping 程序的实现 102
3.3　根据域名或者计算机名获取 IP 地址 112
3.3.1　获取本机 IP 地址 113
3.3.2　根据域名或者计算机名获取 IP 地址 114
3.4　获取网卡地址 118
3.5　增加超级链接和发送 E-mail 122
3.6　小结 124

第 4 章　TCP/UDP 应用开发 125

4.1　Winsock API 实现 TCP 聊天 125
4.1.1　建立工程项目 125
4.1.2　代码分析 127
4.2　Winsock API 实现 UDP 聊天 134
4.2.1　建立工程项目 134
4.2.2　代码分析 136
4.3　Winsock 控件实现 TCP 聊天 141
4.3.1　建立工程项目 141
4.3.2　代码分析 142
4.4　Winsock 控件实现 UDP 聊天 148
4.4.1　建立工程项目 149
4.4.2　代码分析 150
4.5　小结 152

第 5 章　E-mail 协议及高级编程 153

5.1　Foxmail 发送接收 E-mail 153
5.2　SMTP、POP3 与 E-mail 157
5.3　SMTP 及发送电子邮件 157
5.3.1　SMTP 的模型描述 157
5.3.2　SMTP 的会话过程 157
5.4　发送无附件 E-mail 程序 166
5.4.1　建立工程项目 166
5.4.2　代码分析 167
5.5　POP3 与接收电子邮件 171
5.5.1　POP3 的模型描述 171
5.5.2　POP3 的会话过程 171
5.6　接收 E-mail 的程序 179
5.6.1　建立工程项目 179
5.6.2　代码分析 180

5.7 信件结构详述 192
5.7.1 RFC822 信件的格式和内容 192
5.7.2 构造符合 RFC822 的信件 200
5.7.3 RFC822 信件的语法分析 202
5.8 MIME 编码解码与发送附件 204
5.8.1 RFC822 的局限性 204
5.8.2 Uuencode 编码与解码 204
5.8.3 MIME 及其编码 209
5.8.4 构造 MIME 信件 232
5.8.5 MIME 信件的语法分析 234
5.9 E-mail 客户端高级编程 235
5.9.1 建立工程项目 235
5.10 E-mail 乱码 239
5.10.1 乱码的常见形式及形成原因 239
5.10.2 避免乱码的方法 240
5.11 MAPI 概述 241
5.11.1 Windows 的 MAPI 介绍 241
5.11.2 在 VB 中使用 MAPI 241
5.12 MAPI 高级编程 247
5.12.1 建立工程项目 247
5.12.2 代码分析 248

第 6 章 Telnet 协议及高级编程 262

6.1 Telnet 简介 262
6.2 使用 Windows 的 Telnet 程序登录远程服务器 263
6.3 深入 Telnet 协议 264
6.3.1 NVT ASCII 字符集 264
6.3.2 Telnet 命令 264
6.3.3 协商选项 266
6.3.4 子协商选项 267
6.3.5 Telnet 操作方式 267
6.4 BBS 客户端高级开发 268
6.4.1 建立工程项目 269
6.4.2 关键代码分析 269

第 7 章 HTTP 协议及高级编程 296

7.1 HTTP 协议介绍 296
7.1.1 HTTP 背景 296
7.1.2 HTTP 的内容 299
7.1.3 消息（Message） 300

7.1.4 请求（Request）.....301
7.1.5 响应（Response）.....305
7.1.6 访问认证.....308
7.1.7 URL 编码.....310
7.1.8 HTTP 协议的应用.....311
7.2 断点续传.....312
7.2.1 建立工程项目.....312
7.2.2 代码分析.....313
7.3 网页服务器高级开发.....342
7.3.1 Web Server 的一些理论.....342
7.3.2 建立工程项目.....343
7.3.3 代码分析.....345
7.4 网站下载程序高级开发.....372
7.4.1 实例介绍.....372
7.4.2 WinInet HTTP API 实现文件下载的使用方法.....373
7.4.3 代码分析.....375
7.5 HTTP API 高级开发.....399
7.5.1 实例介绍.....400
7.5.2 WinInet HTTP API 实现断点续传.....400
7.5.3 关键代码分析.....404
7.6 HTTP 代理服务器高级开发.....425
7.6.1 建立工程项目.....426
7.6.2 代码分析.....429

第 8 章 FTP 协议及高级编程.....433

8.1 FTP 简介.....433
8.2 安装设置 FTP 服务器.....434
8.3 使用 Windows 内置 FTP 程序.....439
8.4 深入 FTP 协议.....441
8.4.1 FTP 命令大全.....441
8.4.2 FTP 工作模式.....460
8.5 Internet Transfer 控件实现 FTP 程序.....461
8.5.1 建立工程项目.....461
8.5.2 关键代码分析.....462
8.6 Winsock 开发高级 FTP 客户端程序.....475
8.6.1 建立工程项目.....475
8.6.2 关键代码分析.....477
8.7 API 开发高级 FTP 客户端程序.....521
8.7.1 建立工程项目.....521
8.7.2 关键代码分析.....522

8.8 3 种 FTP 客户端程序开发方法的比较.....542

第 9 章 RAS 高级编程.....543

9.1 RAS 客户机.....543
9.2 建立拨号连接.....544
9.3 RAS 简单拨号程序.....548
9.4 RAS 重要函数说明.....549
9.4.1 连接函数.....549
9.4.2 连接管理函数.....554
9.4.3 电话簿和用户凭证管理.....557
9.4.4 拨号方式.....558
9.5 RAS 高级程序开发实例.....560
9.5.1 建立工程项目.....560
9.5.2 程序运行结果图.....561
9.5.3 关键代码分析.....565
9.5.4 RAS 编程小结.....611
9.6 RAS 应用实例——远程文件共享.....612

第 10 章 串口通信高级编程.....616

10.1 串口通信中字符传输.....616
10.1.1 ASCII 控制字符.....616
10.1.2 通信中的字符和字节.....618
10.2 MSComm 控件编程实例.....619
10.2.1 建立工程项目.....619
10.2.2 代码分析.....619
10.3 Windows API 串口通信高级实例.....637
10.3.1 VB 中调用 Windows API.....637
10.3.2 建立工程项目.....638
10.3.3 代码分析.....639

第 1 章　网络编程基础知识

1.1　网络的基本应用

目前网络的应用有许多方面，而对于广大的编程人员来说，在应用程序开发方面，应用最为广泛的就是以下几种类型。

- HTTP 协议的开发
- FTP 协议的开发
- Telnet 协议的开发
- E-mail 协议开发
- 新闻组协议的开发

当然还有其他的许多协议，以上列举的只是目前比较通用的一些协议。其实在实际的开发过程中，完全可以根据需要，自定义开发各种不同的协议以满足自己的需要。在本书中所涉及到的网络编程，基本上是网络的高级应用，而不涉及到网络的底层编程。网络程序的开发是包括许多方面的，在后面读者会知道，ISO 把网络结构划分为 7 层，每一层都可以进行程序开发。通常说的网络编程也就是最上层即应用层程序的开发。

上面列举出了 5 种开发协议，从这 5 种开发协议可以引申出许多的网络开发。其中较为复杂的就是 HTTP 协议的程序开发。目前，在网络上，最流行的也莫过于 WWW 应用，而它正是基于 HTTP 协议的。通过 HTTP 协议，可以进行许多方面的编程，比如 WWW 服务器的开发、网络浏览器的开发、代理服务器开发、CGI 的开发、ASP 开发（PHP、JSP 等）、网络搜索引擎开发等。

1.2　TCP/IP 模型

TCP/IP 是协议的集合，其名称代表它的两个基本协议，TCP（Transmission Control Protocol，传输控制协议）和 IP（Internet Protocol，互联网协议）。虽然从名称上来说只有两个协议，其实它包含了 4 层协议。

1. TCP/IP 协议的分层

TCP/IP 通常被认为是一个由 4 层协议系统，由以下 4 层组成：

- 应用层
- 传输层
- 网络层
- 链路层

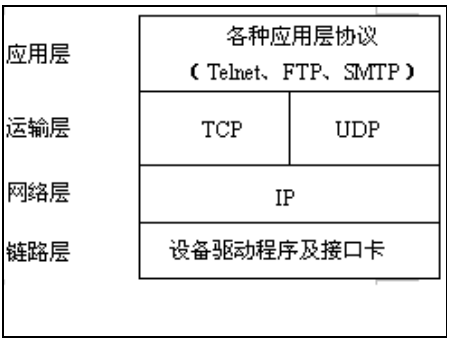


图 1-1　TCP/IP 协议族的体系结构

注意这里只有 4 层，它同 ISO 规定 7 层协议 OSI（开放系统互联）是有所区别的，稍后再介绍 OSI 系统。上面提到的每一层都有不同的功能。

■　链路层

有时候也称为数据链路层或者网络接口层，通常包括操作系统中的设备驱动程序和计算机中对应的网络接口卡。它们一起处理电缆（或者其他任何传输媒介）的物理接口细节。

■　网络层

有时候也被称为互联网层，处理分组在网络中的活动，例如分组的选录，在 TCP/IP 协议族中，网络层包括 IP 协议（网际协议）、ICMP 协议（Internet 互联网控制报文协议），以及 IGMP 协议（Internet 组管理协议）。

■　传输层

传输层主要为两台主机上的应用程序提供端到端的通信。在 TCP/IP 协议族中，有两个互不相同的传输协议：TCP（传输控制协议）和 UDP（用户数据包协议）。TCP 为两台主机提供可靠的数据通信。它所作的工作包括把应用程序交给它的数据分成合适的小块交给下面的网络层，确认接收到的分组，设置发送最后确定分组的超时时钟等。由于传输层提供了高可靠性的端到端的通信，因此应用层可以忽略所有的细节。而另一方面，UDP 则为应用层提供一种非常简单的服务。它只是把称作数据包的分组从一台主机发送到另外一台主机，但并不保证该数据包能到达另一端。任何必需的可靠性必须由应用层来提供。

■　应用层

应用层负责处理特定的程序细节，在本书中的所有应用程序全部是在应用层进行的。

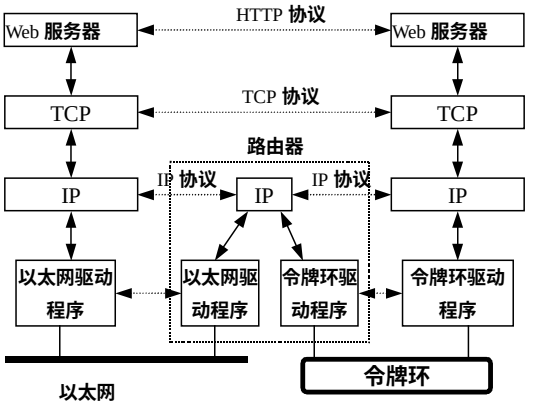


图 1-2　通过 TCP/IP 和路由器连接的两个网络

2. OSI 模型

OSI 网络模型主要是由 ISO（国际标准化组织）规定的一套协议。OSI 为 Open System Interconnection 的简写。实际上该模型是源自 TCP/IP 协议，但比 TCP/IP 多了 3 层，一共有 7 层协议。具体分层如下：

- 应用层（使用网络应用程序）
- 表示层（应用程序的标准化数据表示）
- 会话层（管理应用程序之间的会话）
- 传输层（提供端对端的错误检测和校正）
- 网络层（为上一层管理网络之间的连接）
- 数据链路层（提供跨越物理链路的可靠数据传输）
- 物理层（定义网络媒体的物理特性）

3. 网际地址

互联网上的每一个接口必须有唯一的 Internet 地址也就是 IP 地址。一个 IP 地址通常由 4 个用点隔开的 8 个二进制位的数字（从 0 到 255）组成的，地址总长为 32 比特。目前 IP 地址共分为 5 类即 A、B、C、D、E。其中具体的分布情况如下：

A 类：从 0.0.0.0 到 127.255.555.255

B 类：从 128.0.0.0 到 191.255.255.255

C 类：从 192.0.0.0 到 223.255.255.255

D 类：从 224.0.0.0 到 239.255.255.255

E 类：从 240.0.0.0 到 247.255.255.255

注意，多接口的主机具有多个 IP 地址，其中每个接口都对应一个 IP 地址。由于互联网上的每一个接口必须有一个唯一的 IP 地址，因此必须要由一个管理机构为介入互联网的网络分配 IP 地址。这个管理机构就是互联网络信息中心，称作 InterNIC。

每个 32 位 IP 地址被分割成两部份：前缀和后缀。前缀用于确定计算机从属的物理网络，后缀则用于确定网络上一台单独的计算机。互联网中每一个物理网络都有一个唯一的值作为网络号（Network Number）。IP 地址的层次性保证了以下两个重要性质：

- 每台计算机分配一个唯一的地址
- 虽然网络号分配必须全球一致，但后缀可本地分配，不须全球一致

此外，需要特别注意以下几个特殊的 IP 地址：

- 网络地址：IP 中主机地址为 0 的地址表示网络地址。如 128.211.0.0
- 广播地址：网络号后跟一个所有位全是 1 的后缀，就是直接广播地址
- 回送地址：127.0.0.1 用于测试

除了每个主机分配一个 IP 地址外，IP 协议规定也应给路由器分配一个 IP 地址。事实上，每个路由器被分配了两个或更多的 IP 地址。一个路由器连接到多个物理网络， 每一个 IP 地址包含一个特定物理网络的网络号。一个 IP 地址并不标识一台特定的计算机，而是标识一台计算机和一个网络间的一个连接。

现在所有的主机都要求支持子网编址（RFC 950 [Mogul and Postel 1985）。不是把 IP 地址看成由单纯的一个网络号和一个主机号组成，而是把主机号再分成一个子网号和一个主机号。

这样做的原因是因为 A 类和 B 类地址为主机号分配了太多的空间，可分别容纳的主机数为 2^{24}-2 和 2^{16}-2，而事实上，在一个网络中人们并不安排这么多的主机。在 InterNIC 获得某类 IP 网络号后，就由当地的系统管理员来进行分配，由他（或她）来决定是否建立子网，以及分配多少比特给子网号和主机号。例如，这里有一个 B 类网络地址（140.252），在剩下的 16 位中，8 位用于子网号，8 位用于主机号，其格式如图 1-3 所示。这样就允许有 254 个子网，每个子网可以有 254 台主机。

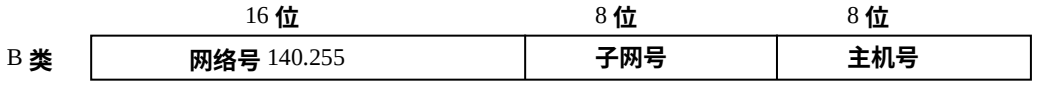


图 1-3　B 类地址的一种子网编址

除了地址以外，主机还需要知道有多少比特用于子网号及多少位用于主机号。这是在引导过程中通过子网掩码来确定的。这个掩码是一个 32 位的值，其中值为 1 的比特留给网络号和子网号，为 0 的比特留给主机号。在上面的例子中，子网掩码就是 255.255.255.0。

通常在记忆 IP 地址的时候比较困难，因此在实际的应用中，会经常使用到域名。在进行网络程序开发的时候，必须要给出远程主机的地址，一般可以使用 IP 地址或者使用域名。

这里要注意，因为域名通常要经过 DNS（域名服务器）的解析，所以比直接使用 IP 地址会慢一些。因为网络不理解域名，只能理解 IP 地址。将名称解析为地址的方法通常有两种。古老的方法是将所有的名称与地址存放在一个名为 HOSTS 的文本文件中。在 UNIX 中，这个表格可从一个共享目录中访问。在 Windows 95 中，这个文件位于网络上的每一台机器的“Windows”目录中。在 Windows NT 中，它驻留在“WINNT\System32\Drivers\etc”目录下。如果需要，可以由用户来编辑这个文件。

这个系统的缺点是非常明显的。因为如果每个人都使用一个 HOSTS 文件作为域名解析，就不得不把自己的 IP 地址和名称给别人，而其他人都必须更新，因此该系统只能在局域网内实现。目前实现名称解析的方法是使用 DNS（域名服务）系统。DNS 是一个分布式的数据库，它包含了所有注册过的 Internet 主机的 IP 地址，域名也是由 InterNIC 来管理的。

4. 端口

在 TCP/IP 连接过程中，都是通过采用 16 位端口号来识别的。因为 IP 地址只是标志了一台机器在网络中的位置。而 IP 端口是对应了一个主机上运行的应用程序。服务器一般都是通过知名端口号来识别的。例如对于每个 TCP/IP 实现来说，FTP 服务器的 TCP/IP 端口号是 21、Telnet 的端口号是 23、TFTP（简单文件传送协议）服务器的 UDP 端口号是 69。任何 TCP/IP 实现所能提供的服务都用知名的 1~1023 之间的端口号。这些端口号由 Internet 号分配机构（IANA）来管理。

实际上，客户端对它所使用的端口号并不关心，只需要保证该端口号在本机上是唯一的就可以了。客户端口号又称作临时端口号（即存在时间很短）。这是因为它通常只是在用户运行该客户程序时才存在。而服务器只要没有停止服务，就一直运行并侦听连接。

大多数的 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预留的。

1.3　UDP 数据包

1.　UDP 数据包格式

UDP 数据包的格式如图 1-4 所示，伪首部是为了计算检验而设置的；伪首部包含 IP 首部一些字段，其目的是让 UDP 两次检查数据是否已经正确到达目的地（例如，IP 没有接收地址不是本主机的数据包，以及 IP 没有把应传给另一高层的数据包传给 UDP）。使用 UDP 协议时，“ 协议 ”字为 17。源端口是可选的，不用时时置 0。报文长度是包括头部和数据区的总长度，最小 8 个字节。校验和是以 16 位为单位，各位求补（首位为符号位）将和相加，之后再求补。

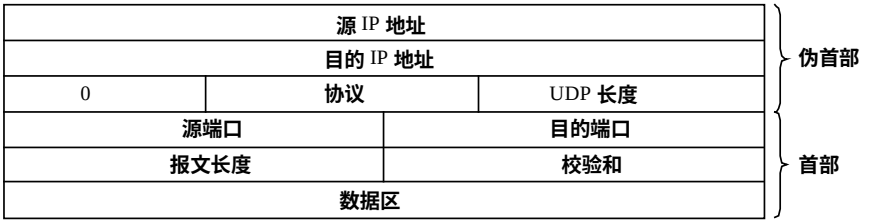


图 1-4　UDP 数据包格式

2.　UDP 数据包的传输

理论上，IP 数据包的最大长度是 65 535 字节，这是由 IP 首部 16 比特总长度字段所限制的。去除 20 字节的 IP 首部和 8 个字节的 UDP 首部，UDP 数据包中用户数据的最长长度为 65 507 字节。但是，大多数实现所提供的长度比这个最大值小。

这样，可能会遇到两个限制因素。一个是因为应用程序可能会受到其程序接口的限制，Socket API 提供了一个可供应用程序调用的函数，以设置接收和发送缓存的长度；对于 UDP 套接字这个长度与应用程序可以读写的最大 UDP 数据包的长度直接相关，现在的大部分系统都默认提供了可读写大于 8192 字节的 UDP 数据包（使用这个默认值是因为 8192 是 NFS 读写用户数据数的默认值）。另一个限制来自于 TCP/IP 的内核实现，可能存在一些实现特性（或差错），使 IP 数据包长度小于 65 535 字节。

因为 UDP 协议是无差错控制的，所以发送过程与 IP 协议类似，就是先 IP 分组，然后再用 ARP 协议来解析物理地址，然后发送。

1.4　TCP 数据包

1.　TCP 数据包格式

TCP 数据包格式如图 1-5 所示。其中，序列号字段是指该报头段在发送方字节流中的位置，确认序号是指接收方希望接收的下一个报文段序号，窗口字段是指接收缓冲区的大小，

码元比特字段是一个 6 位的标志字段，该字段每一比特（从左到右）置 1 时的意义如下：

- URG：紧急指针可用
- ACK：确认字段可用
- PSH：本报文段请求急迫 PUSH 操作
- RST：连接复位
- SYN：序号同步
- FIN：发送方字节流结束

紧急指针字段是用来处理带外数据（DOOB）而设的；最大段长度用来选择适当的 MSS，目的是与 MTU 相符合，如果取值过小会导致带宽利用率太低，而如果取值过大又会造成 IP 数据包分段，若分段丢失导致重发整个报文，这样会浪费网络资源。最佳长度是使携带了长度 S 的 TCP 报文段的 IP 数据包在从源至目的地的路径上不被分段。

图 1-5　TCP 报头

2. 可靠的数据传输

可靠传输服务软件所应具有的特征如下：

- 面向数据流：数据流（stream）就是两个应用程序间传输的数据
- 虚电路连接：包括连接的建立，通信的开始，连接的结束要求所建立的连接是可靠的，连接的结束要完美（在连接终止前传送的所有数据为可靠的）
- 带缓冲的传送
- 无结构的数据流，即不考虑数据内容
- 全双工连接：包含两个独立且方向相反的连接，即 PIGGYBACKING（捎带应答）

TCP 提供了一个完全可靠的、面向连接的、全双工的、流传输服务。允许两个应用程序建立一个连接，并在全体一个方向上发送数据，然后终止连接，每一个 TCP 连接可靠地建立完善的终止，在终止发生前，所有数据都会被可靠的传送。

3. 滑动窗口

提供可靠性的一般方法是采用带重发的肯定确认（PAR）机制，接收方在收到数据包后给源主机发回确认信息（ACK）发送方对送出的每一个数据包均存有一份记录，在发送下一个数据包时，等待确认同时启动计时器，若超时而无 ACK 返回，则重发同一数据包。为了解决同一帧数据发两次的都收到的情况，必须对 ACK 编号，一般的编号方法就是滑动窗口

方法。

实现数据确认的最简单的方法是使用停止等待协议，该方法编号只用一位，有两种状态，每发完一帧，都必须等到确认帧后，发下一帧，如果同一帧发了两次，因为第二次收到的帧编号不符合将被舍弃。这种方法的缺点是需要等待很长的时间，优点是实现简单。另一种可能是连续发送，但这样有两个缺点，一个是如果有帧丢失，将不得不重发所有的从丢失帧开始的所有帧，另一个是这样，编号位数必然增加，导致位浪费。所以通常会对未被确认的帧的个数加以限制。这就是本节要讨论的滑动窗口协议的基础。

设发送序号用 3 个比特来编码，同时设发送窗口为 5，也就是说帧编号可以从有 0～7 共 8 个，而允许发送的未被确认的最大帧数目是 5。如果把发送窗口也设成 8 的话，就只有等到 8 个确认帧全都收到后才能继续发送了，这样的话该方法就与只用一位的停止等待协议一样了，所以通常发送窗口要小于帧编号数。

发送的过程如图 1-6 所示，当发送端发完 0~4 号帧后，如果没有收到确认帧，就必须停止等待，如果收到一个确认帧，发送窗口就沿顺时针方向旋转一个号，使窗口后沿再次与一个为被确认的帧号相邻。这时 5 号帧落在了发送窗口之内，就可以发送了，以后又设有三个帧收到确认信息，发送窗口再次旋转 3 格，这时就可以依次发送 6、7 和 0 号帧了，如果超时没有收到任何确认帧，就把发送窗口里的帧全部重发。

下面来讨论为什么发送窗口不能为 8，因为刚才的讨论都忽略了一个重要的东西：确认帧丢失的情形。一旦 0、1 号确认帧丢失，发送端以为接受端没有收到任何帧，因此重发所有的帧（0~7），而接收端不知道发送端没有收到确认帧，以为发送端发送的是 3-10 号，这样就导致数据协调错误。

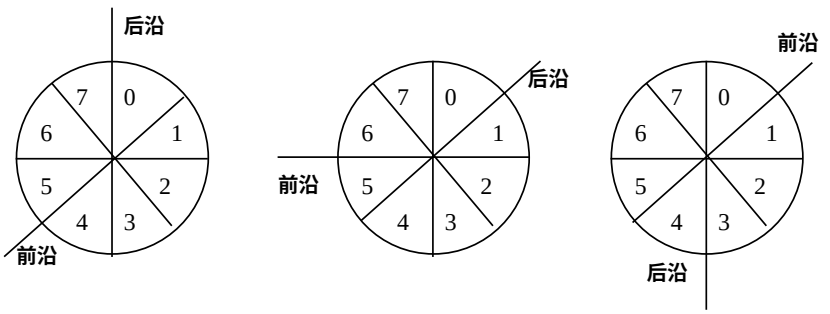


图 1-6　滑动窗口示意图

在滑动窗口协议的基础上又产生了选择重传协议等一些改进版本，但是它们的使用都不如滑动窗口广泛，有兴趣的读者可以参考相关书籍。

4. 超时与重发

TCP 针对互联网环境而设计的协议。由于互联网中具有多个网络和路由器，因此很难预测确认信息何时返回。TCP 采用自适应重发算法：

- 收集每一个报文段的往返时间，得来均值 RTT（ROUND TMP TIME）。
- 将该 RTT 做加权平均值存储利用新的样本逐渐修改其值。
- 保留一个变化量利用该变化量与 RTT 的线性组合，计算出新的 RTT：

RTT= α × OLD_RTT+（1- α）×NEW_RTT_SAMPLE

式中，α：变化量∈[0，1)，α=1，对延迟不敏感，α=0，对延迟敏感；定时器的时限为 TIMEOUT： TIMEOUT=β×RTT β>1 （β推荐取 2)。

TCP 的超时重发机制会导致网络的拥塞崩溃，过程如下：拥塞→延迟→重发→崩溃为了解决该问题，TCP 假设大多数据包的丢失就是来源于拥塞，采用两种技术来解决该问题。

- 加速递减

对于每一个连接，TCP 保留两个窗口，一个窗口称通告窗口，另一个称为拥塞窗口。TCP 取两者的最小值作为发送字节的标准，初始时拥塞窗口=MSS，如果该段及时得到 ACK 则拥塞窗口为 2×MSS，否则拥塞窗口=min{MSS/2，通告窗口}。

- 慢启动

在开始新的连接时，或在拥塞后，增加流量时仅以一个段大小，作为拥塞窗口的初始值每得到一个 ACK 后大小增加一个段。

5. 三次握手

三次握手是指通信双方彼此交换三次信息。三次握手的目的是在存在包丢失、重复和延迟的情况下，确保通信双方信息交换确定性的充分必要条件。下面先解释一下几个常用的名词：

- CR：请求连接
- ACC：接受连接请求
- SEQ：信息序列号
- DR：终止连接

三次握手的操作过程如下：

（1）建立连接时的三次握手

① 正常情况（如图 1-7 所示）：

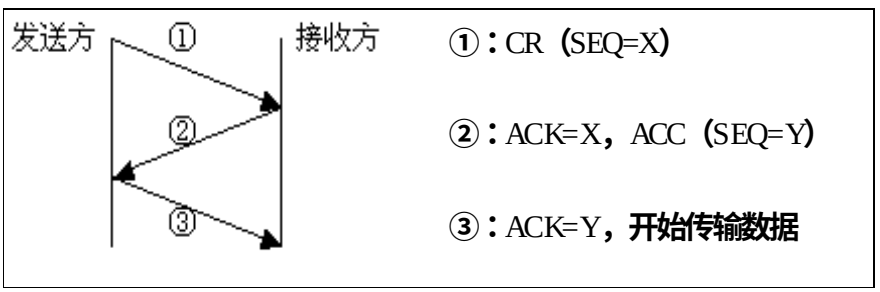
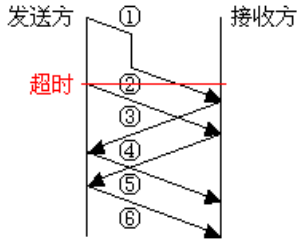


图 1-7 建立连接的正常情况下三次握手

② CR 出现重复（如图 1-8 所示）：

✍ 注意：在 CR 由于延迟而造成的计时器超时后，该连接请求变为无效的连接请求。对于发送方而言，该连接请求已经不存在了！当接收方连续收到两个 CR 时，并不认为第一个 CR 或第二个 CR 是无效的，仅仅认为第二个 CR 为另一个新的连接请求，并对其回应。



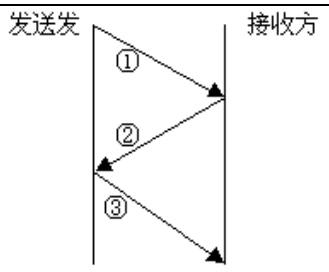
- 延迟的 CR（SEQ=X）
- 在规定时间内未收到 ACK 而导致计时器超时，重发 CR（SEQ=Y）
- 接收方收到延迟的 CR（SEQ=X），发 ACK=X，ACC（SEQ=Y）
- 接收方收到重发的 CR（SEQ=Z），认为是新的连接请求，发 ACK=Z，ACC（SEQ=R）
- 发送方收到一个对超时 CR 的确认，发 REJEST=Y
- 发送方收到一个重发的 CR 的确认，发 DATA（SEQ=Y），ACK=R

图 1-8　建立连接的 CR 出现反复的三次握手

（2）释放连接

释放连接可以分成两种，对称释放和非对称释放。对称释放就是指双方均同意的释放连接。非对称释放是指单方同意后的强行释放连接。由于非对称释放会造成数据的丢失，因而采取对称释放的策略。在释放连接时，为避免产生两军问题（Two Arrmy Problem），我们采用三次握手加计时器的解决方案。我们分 4 种情况来讨论。

① 正常情况（如图 1-9 所示）：



- 发送方发 DR（SEQ=X）
- 接收方同意，发 DR（SEQ=Y），ACK=X
- 发送方收到 DR（SEQ=Y），发 ACK=Y 后释放连接接收方收到 ACK=Y 后释放连接。

图 1-9　释放连接的正常情况下的三次握手

② 异常情况 1：最后的 ACK 丢失（如图 1-10 所示）。

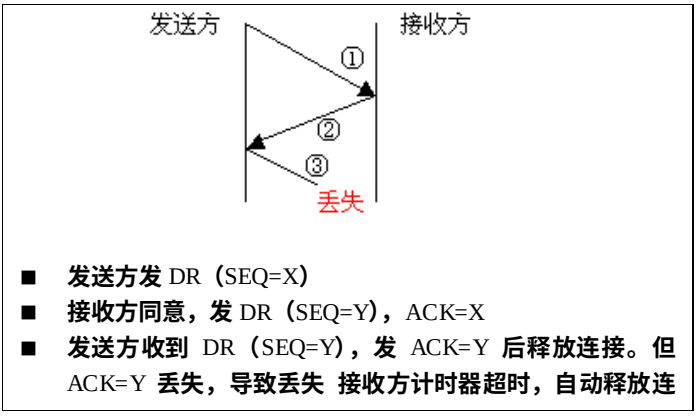


图 1-10　释放连接的异常情况 1 下的三次握手

③ 异常情况 2：第二个 DR 丢失（如图 1-11 所示）。

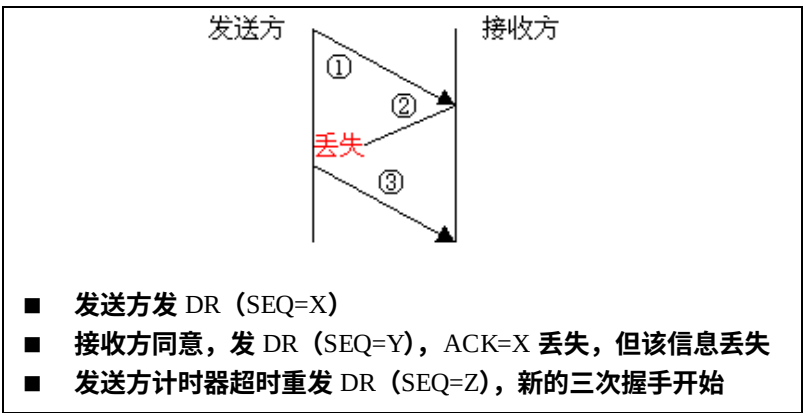


图 1-11　释放连接的异常情况 2 下的三次握手

④ 异常情况 3：第一个 DR 以后的数据均丢失，通信双方在一定时间内没有任何信息交换，连接自动释放。

6. 紧急方式

TCP 提供了“紧急方式（Urgent Mode）”，它使一端可以告诉另一端有些具有某种方式的“紧急数据”已经放置在普通的数据流中。另一端被通知这个紧急数据已被放置在普通数据流中，由接收方决定如何处理。可以通过设置 TCP 首部中的两个字段来发出这种从一端到另一端的紧急数据已经被放置在数据流中的通知。URG 比特被置 1，并且一个 16bit 的紧急指针被置为一个正的偏移量，该偏移量必须与 TCP 首部中的序号字段相加，以便得出紧急数据的最后一个字节的序号。

仍有许多关于紧急指针是指向紧急数据的最后一个字节还是指向紧急数据最后一个字节的下一个字节的争论。最初的 TCP 规范给出了两种解释，但 Host Requirements RFC 确定指向最后一个字节是正确的。

然而，问题在于大多数的实现（包括源自伯克利的实现）继续使用错误的解释。所有符合 Host Requirements RFC 的实现都是可兼容的，但很有可能无法与其他大多数主机正确通

信。

TCP 必须通知接收进程，何时已接收到一个紧急数据指针以及何时某个紧急数据指针还不在此连接上，或者紧急指针是否在数据流中向前移动。接着接收进程可以读取数据流，并必须能够被告知何时碰到了紧急数据指针。只要从接收方当前读取位置到紧急数据指针之间有数据存在，就认为应用程序处于“ 紧急方式 ”。在紧急指针通过之后，应用程序便转回到正常方式。

TCP 本身对紧急数据知之甚少。没有办法指明紧急数据从数据流的何处开始。TCP 通过连接传送的唯一信息就是紧急方式已经开始（TCP 首部中的 URGbit）和指向紧急数据最后一个字节的指针。其他的事情留给应用程序去处理。

许多实现不正确地称 TCP 的紧急方式为带外数据（Out-of-band Data）。如果一个应用程序确实需要一个独立的带外信道，另一个 TCP 连接是达到这个目的的最简单的方法，许多传输层确实提供许多人认为的那种真正的带外数据（使用同一个连接的独立的逻辑数据通道作为正常的数据通道），这是 TCP 所没有提供的。

TCP 的紧急方式与带外数据之间的混淆，也是因为主要的编程接口（接口 API）将 TCP 的紧急方式映射为称为带外数据的插口。

紧急方式有什么作用呢？最常见的例子是 Telnet。Telnet 从服务器到客户使用紧急方式是因为在这个方向上的数据流很可能要被客户的 TCP 停止（也即，它通告了一个大小为 0 的窗口）。但是如果服务器进程进入了紧急方式，尽管它不能够发送任何数据，服务器 TCP 也会立即发送紧急指针和 URG 标志。当客户 TCP 接收到这个通知时就会通知客户进程，于是客户可以从服务器读取其输入、打开窗口并使数据流动。

如果在接收方处理第一个紧急指针之前，发送方多次进入紧急方式会发生什么情况呢？在数据流中的紧急指针会向前移动，而其在接收方的前一个位置将丢失。接收方只有一个紧急指针，每当对方有新的值到达时它将被覆盖。这意味着如果发送方进入紧急方式时所写的内容对接收方非常重要，那么这些字节数据必须被发送方用某种方式特别标记。后面将会看到 Telnet 通过在数据流中加入一个值为 255 的字节作为前缀来标记它所有的命令。

1.5　小结

本章并没有介绍具体的编程细节，而是介绍了一些编程的基本概念。因为本书主要是在 TCP/IP 协议的基础上进行 Winsock 编程，所以着重介绍了与 TCP/IP 相关的一些知识。比如地址、端口等基础知识。这样对于理解后面的程序会有一定的帮助。

第 2 章　网络与通信控件

在 Visual Basic（以下简称为 VB）中，进行网络通信开发是非常方便的。Visual Basic 除了提供了丰富的控件以外，还能够提供了各种 API 来进行更为高级的应用程序的开发。本章着重介绍进行网络开发的 Winsock 控件、Internet Transfer 控件、Winsock API 函数以及进行串口通信的 MSComm 控件和串口开发的 API 函数。

2.1　Winsock 规范

我们目前绝大部分的应用层程序都是利用 Socket 来进行开发。在 Windows 操作系统中，又称为 Winsock 开发。对于众多的基层网络协议，Winsock 是访问它们的首选接口。注意在每个 Win32 平台上，Winsock 都以不同的形式存在着。Winsock 是网络编程的接口，而不是协议，这一点读者要搞清楚。通常用应用层的协议就是基于 Winsock 接口而实现的。

Windows Sockets 规范以 U.C.Berkeley 大学 BSD UNIX 中流行的 Socket 接口为范例定义了一套 Micosoft Windows 下网络编程接口。它不仅包含了人们所熟悉的 Berkeley Socket 风格的库函数；也包含了一组针对 Windows 的扩展库函数，以使程序员能充分地利用 Windows 消息驱动机制进行编程。在 Win32 环境中，Winsock 接口最终成为一个真正与协议无关的接口，尤其是在 Winsock 2 发布之后。

Windows Sockets 规范本意在于提供给应用程序开发者一套简单的 API，并让各家网络软件供应商共同遵守。此外，在一个特定版本 Windows 的基础上，Windows Sockets 也定义了一个二进制接口（ABI），以此来保证应用 Windows Sockets API 的应用程序能够在任何网络软件供应商的符合 Windows Sockets 协议的实现上工作。因此这份规范定义了应用程序开发者能够使用，并且网络软件供应商能够实现的一套库函数调用和相关语义。

遵守这套 Windows Sockets 规范的网络软件，我们称之为 Windows Sockets 兼容的，而 Windows Sockets 兼容实现的提供者，我们称之为 Windows Sockets 提供者。一个网络软件供应商必须百分之百地实现 Windows Sockets 规范才能做到与 Windows Sockets 兼容。

任何能够与 Windows Sockets 兼容实现协同工作的应用程序就被认为是具有 Windows Sockets 接口。我们称这种应用程序为 Windows Sockets 应用程序。

应用程序调用 Windows Sockets 的 API 实现相互之间的通信。Windows Sockets 又利用下层的网络通信协议功能和操作系统调用实现实际的通信工作。

建立 Winsock 2 规范的主要目的是提供一个与协议无关的传送接口。本书基本上都是利用 Winsock 对象来实现各种应用程序的开发，而并不需要知道数据在网络中是如何传输的。

应用程序调用 Windows Sockets 的 API 实现相互之间的通信。Windows Sockets 又利用下层的网络通信协议功能和操作系统调用实现实际的通信工作。它们之间的关系如图 2-1 所示。

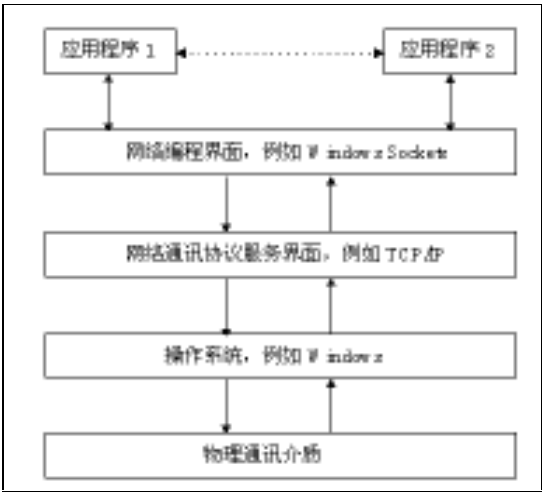


图 2-1　应用程序与 Winsock 的关系

实际上通信的基础是套接字，一个套接字是通信的一端。在这一端上可以找到与其对应的一个名字。每一个正在被使用的套接字都有它的类型和与其相关的进程。套接字存在于通信域中。通信域是为了处理一般的线程通过套接字通信而引进的一种抽象概念。套接字通常和同一个域中的套接字交换数据（数据交换也可能穿越域的界限，但这时一定要执行某种解释程序）。Windows Sockets 规范支持单一的通信域，即 Internet 域。各种进程使用这个域互相之间用 Internet 协议族来进行通信（Windows Sockets 1.1 以上的版本支持其他的域，例如 Windows Sockets 2）。

套接字可以根据通信性质分类；这种性质对于用户是可见的。应用程序一般仅在同一类的套接字间通信。不过只要底层的通信协议允许，不同类型的套接字间也照样可以通信。

用户目前可以使用两种套接口，即流套接字和数据报套接字。流套接字提供了双向的。有序的，无重复并且无记录边界的数据流服务。数据报套接字支持双向的数据流，但并不保证是可靠、有序，无重复的。也就是说，一个从数据报套接字接收信息的进程有可能发现信息重复了，或者和发出时的顺序不同。数据报套接字的一个重要特点是它保留了记录边界。对于这一特点，数据报套接字采用了与现在许多包交换网络（例如以太网）非常类似的模型。

套接字有两种不同的类型：流套接字和数据报套接字。

1. 流套接字

流套接字提供双向的、有序的、无重复并且无记录边界的数据流服务，它适用于处理大量数据。网络传输层可以将数据分散或集中到合适尺寸的数据包中。

流套接字是面向连接的，通信双方进行数据交换之前，必须建立一条路径，这样既确定了它们之间存在的路由，又保证了双方都是活动的、可彼此响应的，但在通信双方之间建立一个通信信道需要很多开支。除此以外，大部分面向连接的协议为保证发送无误，可能会需要执行额外的计算来验证正确性，因此会进一步增加开支。

2. 数据报套接字

数据报套接字支持双向的数据流，但并不保证数据传输的可靠性、有序性和无重复性。

也就是说，一个从数据报套接字接收信息的进程有可能发现信息重复，或者和发出时的顺序不同的情况。此外，数据报套接字的一个重要特点是它保留了记录边界。

数据报套接字是无连接的，它不保证接收端是否正在侦听，这一点类似于邮政服务：发信人把信装入邮箱即可，至于收信人是否想收到这封信或邮局是否按时将信件投递到收信人处等等，发信人都不得而知。因此，数据报并不十分可靠，需有程序员负责管理数据报的排序和可靠性。

2.2 Winsock 控件

利用 VB 开发网络应用程序，更多的是利用 Winsock 控件来实现，而很少使用 Winsock API。对于许多中初级读者来说，直接使用 API 比较难，同时效率也比较低。而直接使用 Winsock 控件的效率高很多，而且通俗易懂，所以下面着重介绍 Winsock 控件的属性、方法和事件。

Winsock 控件对用户来说是不可见的，它提供了访问 TCP 和 UDP 网络服务的方便途径。Microsoft Access、Visual Basic、Visual C++或 Visual FoxPro 的开发人员都可使用它。为编写客户或服务器应用程序，不必了解 TCP 的细节或调用低级的 Winsock API。通过设置控件的属性并调用其方法就可轻易地连接到一台远程计算机上去，并且还可双向交换数据。

2.2.1 TCP 协议基础知识

数据传输协议允许创建和维护与远程计算机的连接，连接两台计算机就可彼此进行数据传输。

如果创建客户应用程序，就必须知道服务器计算机名或者 IP 地址（RemoteHost 属性），还要知道进行“侦听”的端口（RemotePort 属性），然后调用 Connect 方法。

如果创建服务器应用程序，就应设置一个收听端口（LocalPort 属性）并调用 Listen 方法。当客户计算机需要连接时就会发生 ConnectionRequest 事件。为了完成连接，可调用 ConnectionRequest 事件内的 Accept 方法。

建立连接后，任何一方计算机都可以收发数据。为了发送数据，可调用 SendData 方法。当接收数据时会发生 DataArrival 事件。调用 DataArrival 事件内的 GetData 方法就可获取数据。

2.2.2 UDP 基础知识

用户数据文报协议（UDP）是一个无连接协议。跟 TCP 的操作不同，计算机并不建立连接。另外 UDP 应用程序可以是客户机，也可以是服务器。

为了传输数据，首先要设置客户计算机的 LocalPort 属性。然后，服务器计算机只需将 RemoteHost 设置为客户计算机的 Internet 地址，并将 RemotePort 属性设置为跟客户计算机的 LocalPort 属性相同的端口，并调用 SendData 方法来着手发送信息。于是，客户计算机使用 DataArrival 事件内的 GetData 方法来获取已发送的信息。

2.2.3　Winsock 控件属性

首先介绍 Winsock 控件的属性，了解这些属性是熟练使用 VB 进行网络开发的基础。

1. BytesReceived 属性

该属性返回接收到的（当前在接收端缓冲区内的）数据的数量，使用 GetData 方法来获取数据，GetData 方法将在后面的方法中介绍。

在设计时是只读的，而且是不可用的。

使用语法如下：

```
Winsock1.BytesReceived
返回值：Long
```

2. LocalHostName 属性

该属性返回本地计算机名。在设计时是只读的，而且是不可用的。

使用语法如下：

```
Winsock1.LocalHostName
返回值：String
```

3. LocalIP 属性

返回本地计算机的 IP 地址，格式是 IP 地址加点字符串 (xxx.xxx.xxx.xxx)。在设计时是只读的，而且是不可用的。

使用语法如下：

```
Winsock1.LocalIP
数据类型：String
```

4. LocalPort 属性

返回或者设置所用到的本地端口。在设计时是可读/写的，而且是可用的。

对客户来说，该属性指定发送数据的本地端口。如果应用程序不需要特定端口，则指定 0 为端口号。在这种情况下，控件将选择一个随机端口。在建立起连接之后，这就是用于 TCP 连接的本地端口。

对于服务器来说，这是用于侦听的本地端口。如果指定的是端口 0，就使用一个随机端口。在调用了 Listen 方法后，属性就包含了已选定的实际端口。

使用语法如下：

```
Winsock1.LocalPort=long
返回数据类型：Long
```

✍ 在计算机之间常用端口 0 来动态地建立连接。例如，一个客户希望服务器给他“回电话”，它就可用端口 0 获得新的（随机）端口号，然后将该端口号交给远程计算机，从而达到目的。

5. Protocol 属性（Winsock 控件）

返回或设置 Winsock 控件所使用的协议——或者是 TCP，或者是 UDP。

使用语法如下：

```
Winsock1.Protocol[=protocol]
```

返回值：Void

protocol 的设置值如表 2-1 所示。

表 2-1 protocol 设置值

常数	值	描述
sckTCPProtocol	0	缺省的 TCP 协议
sckUDPProtocol	1	UDP 协议

✍ 在能够重新设置属性之前必须（用 Close 方法）关闭控件。

6. RemoteHost 属性

返回或设置远程计算机，控件向它发送数据或从它那里接收数据。既可提供主机名，比如“FTP://ftp.microsoft.com”，也可提供点格式下的 IP 地址字符串，比如“100.0.1.1”。

使用语法如下：

```
object.RemoteHost=string
```

RemoteHost 属性的语法具有以下几部分，如表 2-2 所示。

表 2-2 RemoteHost 属性

部分	描述
object	对象表达式，其值是“应用于”列表中的对象
string	远程计算机的名称或地址

✍ 在指定该属性时，应更新 URL 属性来显示新值。如果更新 URL 的主机部分，则也要更新该属性来反映新值。在调用 OpenURL 或 Execute 方法时也可改变 RemoteHost 属性。在运行时直到下一个连接之前，改变该值都不会造成什么影响。

7. RemoteHostIP 属性

返回远程计算机的 IP 地址。

对于客户应用程序来说，已经用 Connect 方法建立连接后，属性就包含了远程计算机的 IP 字符串。

对于服务器应用程序来说，在请求连接（ConnectionRequest 事件）之后，属性包含远程计算机的 IP 字符串，该字符串启动了连接。

当使用 UDP 协议时，在 DataArrival 事件出现之后，属性包含了发送 UDP 数据的计算机的 IP 地址。

使用语法如下：

```
Winsock1.RemoteHostIP
```

数据类型：String

8. RemotePort 属性（ActiveX 控件）

返回或设置要连接的远程端口号

使用语法如下：

```
Winsock1.RemotePort=port
```

要连接的端口。该属性的缺省值是 80。

数据类型：Long

✍ 在设置 Protocol 属性时，将对每个协议自动把 RemotePort 属性设置成适当的缺省端口。表 2-3 列出缺省端口号。

表 2-3　缺省端口号

端口	描述
80	HTTP，通常用于 World WideWeb 连接
21	FTP

9. SocketHandle 属性

返回一个与套接字句柄对应的值，控件用套接字句柄同 Winsock 层通信。在设计时是只读的，而且是不可用的。

使用语法如下：

```
Winsock1.SocketHandle
```

数据类型：Long

✍ 这个属性是为了传递到 WinsockAPIs 而设计的。

10. State 属性

返回控件的状态，用枚举类型来表示。在设计时是只读的，而且是不可用的。

使用语法如下：

```
Winsock1.State
```

object 所在处代表一个对象表达式，其值是“应用于”列表中的对象。

数据类型：Integer

State 属性的设置值如表 2-4 所示。

表 2-4　State 属性值

常数	值	描述
sckClosed	0	缺省值，关闭
sckOpen	1	打开
sckListening	2	侦听
sckConnectionPending	3	连接挂起
sckResolvingHost	4	识别主机
sckHostResolved	5	已识别主机
sckConnecting	6	正在连接
sckConnected	7	已连接
sckClosing	8	同级人员正在关闭连接
sckError	9	错误

2.2.4 Winsock 控件方法

1. Accept 方法

仅适用于 TCP 服务器应用程序。在处理 ConnectionRequest 事件时用这个方法接受新连接。
使用语法如下：

```
Winsock1.Accept requestID
```

数据类型：Long
返回值：Void

✍ 在 ConnectionRequest 事件中使用 Accept 方法。ConnectionRequest 事件有一个对应的参数，即 RequestID 参数，该参数应该传给 Accept 方法。请看下例：

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
   '测试 State 属性，如果当前连接是打开的话，
   '则关闭连接。
   If Winsock1.State <> sckClosed Then Winsock1.Close
   '将 requestID 参数值传递给 Accept 方法。
   Winsock1.Accept requestID
End Sub
```

✍ ConnectionRequest 事件在介绍 Winsock 控件事件中有详细介绍。

2. Bind 方法

指定用于 TCP 连接的 LocalPort 和 LocalIP。如果有多协议适配卡，就用这个方法。
使用语法如下：

```
Winsock1.Bind LocalPort, LocalIP
```

Bind 方法的参数如表 2-5 所示。

表 2-5 Bind 方法的参数说明

部分	描述
LocalPort	用来建立连接的端口
LocalIP	用来建立连接的本地 Internet 地址

✍ 在调用 Listen 方法之前必须调用 Bind 方法。

3. Close 方法

对客户机和服务器应用程序关闭 TCP 连接或侦听套接字。
使用语法如下：

```
Winsock1.Close
```

参数：None
返回值：Void

4. GetData 方法

获取当前的数据块并将其存储在变体类型的变量中。

返回值：Void

使用语法如下：

```
object.GetData data, [type,] [maxLen]
```

GetData **方法的参数如表** 2-6 **所示。**

表 2-6　　GetData **方法的参数说明**

部分	描述
data	在方法成功返回之后存储获取数据的地方。如果对请求的类型没有足够可用的数据，则将 data 设置成 Empty
type	可选的。获取的数据类型，如下例“设置值”所示
maxLen	可选的。在接收到字节数组或字符串时所需大小。如果对字节数组或字符串、参数丢失、则将获取所有可用数据。如果提供的数据类型不是字节数组或字符串，则忽略这个参数

其中，表 2-6 **中的参数** type **的设置值如表** 2-7 **所示。**

表 2-7　　**参数** type **设置值**

描述	常数
Byte	vbByte
Integer	vbInteger
Long	vbLong
Single	vbSingle
Double	vbDouble
Currency	vbCurrency
Date	vbDate
Boolean	vbBoolean
SCODE	vbError
String	vbString
Byte Array	vbArray + vbByte

✍ **通常总是将** GetData **方法与** DataArrival **事件并用，而** DataArrival **事件包含** totalBytes **参数。如果指定一个比** totalBytes **参数小的** maxlen**，则将得到警告** 10040**，以此指出剩余的字节将丢失。**

以下为 GetData **方法（**Winsock **控件）、**DataArrival **事件示例。**

该示例使用 Winsock **控件的** DataArrival **事件中的** GetData **方法。当事件出现时，代码调用** GetData **方法获取数据，并将数据存储在字符串变量中。于是就将数据写到一个** TextBox **控件中。**

```
Private Sub Winsock1_DataArrival (ByVal bytesTotal As Long)
   Dim strData As String
   Winsock1.GetData strData, vbString
   Text1.Text=Text1.Text & strData
End Sub
```

5. Listen 方法

创建套接字并将其设置为侦听模式。该方法仅适用于 TCP 连接。

使用语法如下：

```
object.Listen
```

参数：None

返回值：Void

✍ 当有新连接时就会出现 ConnectionRequest 事件。处理 ConnectionRequest 事件时，应用程序应该（在一个新的控件示例上）用 Accept 方法接受连接。

6. PeekData 方法

PeekData 不从输入队列删除数据，除了这一点之外，方法与 GetData 相似。该方法仅适用于 TCP 连接。

使用语法如下：

```
object.PeekData data, [type,] [maxLen]
```

返回值：Void

PeekData 方法的语法如表 2-8 所示。

表 2-8 PeekData 方法的语法说明

部分	描述
object	对象表达式，其值是“应用于”列表中的对象
data	在方法成功地返回之后存储获取的数据。如果对于没有足够的适于所请求的类型来说没有足够可用的数据，那么 data 将被设置为 Empty
type	可选的。所获取的数据类型，如同“设置值”中所述。缺省值为：vbArray + vbByte
maxLen	可选的。在收到字节数组或字符串时，长度指定了所需要的大小。如果对字节数组或字符串的参数丢失，则将获取所有可用的数据。如果提供的数据类型不是字节数组和字符串的话，则忽略该参数

参数 type 的设置值如表 2-9 所示。

表 2-9 type 的设置值

描述	常数
Byte	vbByte
Integer	vbInteger
Long	vbLong
Single	vbSingle
Double	vbDouble
Currency	vbCurrency
Date	vbDate
Boolean	vbBoolean
SCODE	vbError
String	vbString
Byte Array	vbArray + vbByte

✍ 如果所指定的类型为 vbString，则在返回到用户之前，字符串数据将转化成 Unicode。

7. SendData 方法

将数据发送给远程计算机。

返回值：Void

语法

```
object.SendData data
```

其中参数 data 表示要发送的数据。对于二进制数据应使用字节数组。

✍ 当传进 Unicode 字符串并在网络上发送出去之前，将转化成 ANSI 字符串。

2.2.5 Winsock 控件事件

1. Close 事件

该事件发生在当远程计算机关闭连接时出现。应用程序应正确使用 Close 方法关闭 TCP 连接。

使用语法如下：

```
object_Close()
```

参数

```
None
```

2. Connect 事件

该事件发生在该事件是在到服务器的连接建立之后发生。

```
Private Sub object.Connect()
Private Sub object.Connect(ErrorOccurred As Boolean)
```

参数 ErrorOccurred 是一个 Boolean 表达式，确定连接是否已成功，如设置值中所述。可将 ErrorOccurred 参数设置成表 2-10 中一个值。

表 2-10　　ErrorOccurred 参数设置值

值	描述
True	集合失败
False	集合成功

✍ 可以捕捉 Connect 事件并做在一个新的连接上所要求的任何初始查询，例如对照客户的版本来检验数据库的版本，或在此连接串中设置一个没有被建立起来的缺省数据库。也可以在打开此连接的过程中或者是简单地清除信息消息的 rdoErrors 集合时检查所返回的错误或信息。

3. ConnectionRequest 事件

该事件发生在当远程计算机请求连接时出现。

仅适用于 TCP 服务器应用程序。在请求一个新连接时激活该事件。激活事件之后，RemoteHostIP 和 RemotePort 属性存储有关客户的信息。

使用语法如下：

```
object_ConnectionRequest (requestID As Long)
```

参数 requestID 是新连接请求标识。应把此参数传递给第二个控件示例上的 Accept 方法。

ConnectionRequest 的语法包含下面部分：

✍ 服务器可决定是否接受连接。如果不接受新连接，则同级人员（客户）将得到 Close 事件。（在一个新控件示例上）用 Accept 方法接受新连接。

4. DataArrival 事件

该事件发生在当新数据到达时出现。

使用语法如下：

```
object_DataArrival (bytesTotal As Long)
```

参数 bytesTotal 返回可获取的数据总数量。

✍ 如果没有获取一个 GetData 调用中的全部数据，则事件不会出现。只有存在新数据时才激活事件。可随时用 BytesReceived 属性检查可用的数据量。

5. Error 事件

无论何时，只要后台处理中出现错误（例如，连接失败，或者在后台收发数据失败）事件就会出现。

使用语法如下：

```
object_Error(number As Integer, Description As String, Scode As Long, Source
As String, HelpFile as String, HelpContext As Long, CancelDisplay As Boolean)
```

Error 事件的语法包含以下参数，如表 2-11 所示。

表 2-11 Error 事件的语法参数说明

部分	描述
object	对象表达式，其值是“应用于”列表中的对象
number	定义错误代码的整数。请参阅下述有关常数的“设置值”
description	包含错误信息的字符串
Scode	长 SCODE
Source	描述错误来源的字符串
HelpFile	包含帮助文件名的字符串
HelpContext	Help 文件上下文
CancelDisplay	指示是否取消显示。缺省值为 False，以此显示缺省的错误信息框。如果不想使用缺省的信息框，则将 CancelDisplay 设置成 True
object	对象表达式，其值是“应用于”列表中的对象

参数 number 的设置值如表 2-12 所示。

表 2-12 参数 number 的设置值

常数	值	描述
sckOutOfMemory	7	内存不足
sckInvalidPropertyValue	380	属性值无效
sckGetNotSupported	394	属性不可读
sckSetNotSupported	383	属性是只读的
sckBadState	40006	所请求的事务或请求本身的错误协议或者错误连接状态

续表

常数	值	描述
sckInvalidArg	40014	传递给函数的参数格式不确定，或者不在指定范围内
sckSuccess	40017	成功
sckUnsupported	40018	不受支持的变量类型
sckInvalidOp	40020	在当前状态下的无效操作
sckOutOfRange	40021	参数越界
sckWrongProtocol	40026	所请求的事务或请求本身的错误协议
sckOpCanceled	1004	取消操作。
sckInvalidArgument	10014	所请求的地址是广播地址，但未设置标记
sckWouldBlock	10035	套接字不成块，而指定操作将使之成块
sckInProgress	10036	制造块的 Winsock 操作在进行之中
sckAlreadyComplete	10037	完成操作。未进行制造块的操作
sckNotSocket	10038	描述符不是套接字
sckMsgTooBig	10040	数据报太大，不适于缓冲区的要求，因而被截断
sckPortNotSupported	10043	不支持指定的端口
sckAddressInUse	10048	地址在使用中
sckAddressNotAvailable	10049	来自本地计算机的不可用地址
sckNetworkSubsystemFailed	10050	网络子系统失败
sckNetworkUnreachable	10051	此时不能从主机到达网络
sckNetReset	10052	在设置 SO_KEEPALIVE 时连接超时
sckConnectAborted	11053	由于超时或者其他失败而中止连接
sckConnectionReset	10054	通过远端重新设置连接
sckNoBufferSpace	10055	没有可用的缓冲空间
sckAlreadyConnected	10056	已连接套接字
sckNotConnected	10057	未连接套接字
sckSocketShutdown	10058	已关闭套接字
sckTimedout	10060	已关闭套接字
sckConnectionRefused	10061	强行拒绝连接
sckNotInitialized	10093	应首先调用 WinsockInit
sckHostNotFound	11001	授权应答：未找到主机
sckHostNotFoundTryAgain	11002	非授权应答：未找到主机
sckNonRecoverableError	11003	不可恢复的错误
sckNoData	11004	无效名，对所请求的类型无数据记录。

6. SendComplete 事件

该事件在完成一个发送操作时出现。

使用语法如下：

```
Winsock1_SendComplete
```

参数：None

7. SendProgress 事件

该事件在发送数据期间出现。

使用语法如下：

```
Winsock1_SendProgress (bytesSent As Long, bytesRemaining As Long)
```

参数 bytesSent **表示从上次激活事件以来已发送的字节数；参数** bytesRemaining **表示在发**

送缓冲区等待发送时的字节数。

2.2.6 Winsock 控件的使用

利用 Winsock 控件可以与远程计算机建立连接，并通过用户数据文报协议（UDP）或者传输控制协议（TCP）进行数据交换。这两种协议都可以用来创建客户与服务器应用程序。与 Timer 控件类似，WinSock 控件在运行时是不可见的。Winsock 可能的用途有：

- 创建收集用户信息的客户端应用程序，并将收集的信息发送到某中央服务器。
- 创建一个服务器应用程序，作为多个用户的数据的汇入点。
- 创建“聊天”应用程序。
- 选择通信协议。

在使用 WinSock 控件时，首先需要考虑使用什么协议。可以使用的协议包括 TCP 和 UDP。两种协议之间的重要区别在于它们的连接状态：

TCP 协议控件是基于连接的协议，可以将它同电话系统相比。在开始数据传输之前，用户必须先建立连接。

UDP 协议是一种无连接协议，两台计算机之间的传输类似于传递邮件：消息从一台计算机发送到另一台计算机，但是两者之间没有明确的连接。另外，单次传输的最大数据量取决于具体的网络。到底选择哪一种协议通常是由需要创建的应用程序决定的。

下面的几个问题将有助于选择适宜的协议。

在收发数据的时候，应用程序是否需要得到客户端或者服务器的确认信息？如果需要，使用 TCP 协议，在收发数据之前先建立明确的连接。

数据量是否特别大（例如图像与声音文件）？在连接建立之后，TCP 协议将维护连接并确保数据的完整性。不过，这种连接需要更多的计算资源，因而是比较“昂贵”的。

数据发送是间歇的，还是在一个会话内？例如，如果应用程序在某个任务完成的时候需要通知某个计算机，UDP 协议是更适宜的。UDP 协议适合发送少量的数据。

1. 协议的设置

在设计时，可以按如下方式设置应用程序使用的协议：在“属性”窗口中单击“协议”，然后选择 sckTCPProtocol 或者 sckUDPProtocol。也可以使用程序代码来设置 Protocol 属性，如下所示：

```
Winsock1.Protocol=sckTCPProtocol
```

2. 确定计算机的名称

在与远程计算机相连接的时候，需要知道它的 IP 地址或者它的“好听的名字”。IP 地址是一串数字，每三个数字为一组，中间用点隔开（形如 xxx.xxx.xxx.xxx）。通常，最易记住的是计算机的“好听的名字”。

要确定计算机的名字，请按照以下步骤执行：

- 进入“控制面板”窗口。
- 双击“网络”图标。
- 单击“标识”选项卡。

◆ 在“计算机名称”框中可以找到计算机的名称。

找到的计算机名称可以作为 RemoteHost 属性的值。

3. TCP 连接初步

如果应用程序要使用 TCP 协议，那么首先必须决定应用程序是服务器还是客户端。如果要创建一个服务器端，那么应用程序需要“监听”指定的端口。当客户端提出连接请求时，服务器端能够接受请求并建立连接。在连接建立之后，客户端与服务器端可以自由地互相通信。

下列步骤创建一个非常简单的服务器：

◆ 创建新的 Standard EXE 工程。

◆ 将缺省窗体的名称改为 frmServer。

◆ 将窗体的标题改为“TCP 服务器”。

◆ 在窗体中放入一个 Winsock 控件，并将它的名字改为 tcpServer。

◆ 在窗体上添加两个 TextBox 控件。将第一个命名为 txtSendData，第二个为 txtOutput。

◆ 为窗体添加如下的代码。

```
Private Sub Form_Load()
'将 LocalPort 属性设置为一个整数。
'然后调用 Listen 方法。
tcpServer.LocalPort=1001
tcpServer.Listen
frmClient.Show '显示客户端的窗体。
End Sub

Private Sub tcpServer_ConnectionRequest (ByVal requestID As Long)
'检查控件的 State 属性是否为关闭的。
'如果不是，
'在接受新的连接之前先关闭此连接。
If tcpServer.State <> sckClosed Then _
tcpServer.Close
'接受具有 requestID 参数的
'连接。
tcpServer.Accept requestID
End Sub

Private Sub txtSendData_Change()
'名为 txtSendData 的 TextBox 控件中
'包含了要发送的数据。当用户往文本框中
'键入数据时，使用 SendData 方法
'发送输入的字符串。
```

```
    tcpServer.SendData txtSendData.Text
    End Sub

    Private Sub tcpServer_DataArrival (ByVal bytesTotal As Long)
    '为进入的数据声明一个变量。
    '调用 GetData 方法，并将数据赋予名为 txtOutput
    '的 TextBox 的 Text 属性。
    Dim strData As String
    tcpServer.GetData strData
    txtOutput.Text=strData
    End Sub
```

上面的步骤创建了一个简单的服务器应用程序。为了使它能够工作，还必须为它创建一个客户端的应用程序。

要创建 TCP 客户端，请按照以下步骤执行：

- 在工程中添加一个新的窗体，将其命名为 frmClient。
- 将窗体的标题改为“TCP Client”。
- 在窗体中添加一个 Winsock 控件，并将其命名为 tcpClient。
- 在 frmClient 中添加两个 TextBox 控件。将第一个命名为 txtSend，第二个为 txtOutput。
- 在窗体上放一个 CommandButton 控件，并将其命名为 cmdConnect。
- 将 CommandButton 控件的标题改为 Connect。
- 在窗体中添加如下的代码，必须将 RemoteHost 属性值修改为您的计算机的名字。

```
Private Sub Form_Load()
   'Winsock 控件的名字为 tcpClient。
   '注意：要指定远程主机，可以使用
   'IP 地址（例如："121.111.1.1"），也可以使用
   '计算机的"好听的名字"如下所示。
   tcpClient.RemoteHost="RemoteComputerName"
   tcpClient.RemotePort=1001
End Sub

Private Sub cmdConnect_Click()
   '调用 Connect 方法，初始化连接。
   tcpClient.Connect
End Sub

Private Sub txtSendData_Change()
   tcpClient.SendData txtSend.Text
End Sub
```

```
Private Sub tcpClient_DataArrival _
(ByVal bytesTotal As Long)
   Dim strData As String
   tcpClient.GetData strData
   txtOutput.Text=strData
End Sub
```

上面的代码创建了一个简单的客户/服务器模式的应用程序。我们可以将两者都运行起来：运行工程，然后单击“连接”按钮。在两个窗体之一的 txtSendData 文本框中键入文本，可以看到同样的文字将出现在另一个窗体的 txtOutput 文本框中。

4. 接受多个连接请求

上面设计的基本服务器只能接受一个连接请求。通过创建控件数组，使用一个控件也可以同时接受多个连接请求。利用这种方法，不需要关闭连接，而只需创建新的控件实例（通过设置其索引属性），然后在新的实例上调用 Accept 方法。

下面的代码假定名为 sckServer 的窗体上有一个 Winsock 控件，它的 Index 属性被设置为 0；因此控件是控件数组的一部分。在声明部分，声明了一个模块级的变量 intMax。在窗体的 Load 事件中，intMax 被设置为 0，数组中第一个控件的 LocalPort 属性被设置为 1001。然后调用控件的 Listen 方法，使之成为“监听”控件。在连接请求到达时，代码将检测 Index 是否为 0（“监听”控件的值）。如果为 0，监听控件将增加 intMax 的值，并使用该号码来创建新的控件实例。然后，使用新的控件实例接受连接请求。

```
Private intMax As Long

Private Sub Form_Load()
   intMax=0
   sckServer(0).LocalPort=1001
   sckServer(0).Listen
End Sub

Private Sub sckServer_ConnectionRequest _
(Index As Integer, ByVal requestID As Long)
   If Index=0 Then
      intMax=intMax + 1
      Load sckServer(intMax)
      sckServer(intMax).LocalPort=0
      sckServer(intMax).Accept requestID
      Load txtData(intMax)
   End If
End Sub
```

5. UDP 初步

创建 UDP 应用程序比创建 TCP 应用程序还要简单，因为 UDP 协议不需要显式的连接。在上面的 TCP 应用程序中，一个 Winsock 控件必须显式地进行“监听”，另一个必须使用 Connect 方法初始化连接。

UDP 协议不需要显式的连接。要在两个控件中间发送数据，需要完成以下 3 个步骤（在连接的双方）：

- 将 RemoteHost 属性设置为另一台计算机的名称。
- 将 RemotePort 属性设置为第二个控件的 LocalPort 属性。
- 调用 Bind 方法，指定使用的 LocalPort。（下面将详细地讨论该方法。）

因为两台计算机的地位可以看成“平等的”，这种应用程序也被称为点到点的。为了具体说明这个问题，下面将创建一个“聊天”应用程序，两个人可以通过它进行实时的交谈。

要创建一个 UDP 伙伴，请按照以下步骤执行：

创建一个新的 Standard EXE 工程。

- 将缺省的窗体的名称修改为 frmPeerA。
- 将窗体的标题修改为“Peer A”。
- 在窗体中放入一个 Winsock 控件，并将其命名为 udpPeerA。
- 在“属性”页上，单击“协议”按钮并将协议修改为 UDPProtocol。
- 在窗体中添加两个 TextBox 控件。将第一个命名为 txtSend，第二个命名为 txtOutput。
- 为窗体增加如下的代码。

```
Private Sub Form_Load()
'控件的名字为 udpPeerA
With udpPeerA
'重点:必须将 RemoteHost 的值
'修改为计算机的名字。
.RemoteHost= "PeerB"
.RemotePort=1001   '连接的端口号。
.Bind 1002              '绑定到本地的端口。
End With
frmPeerB.Show              '显示第二个窗体。
End Sub

Private Sub txtSend_Change()
'在键入文本时，立即将其发送出去。
udpPeerA.SendData txtSend.Text
End Sub

Private Sub udpPeerA_DataArrival(ByVal bytesTotal As Long)
Dim strData As String
udpPeerA.GetData strData
```

```
txtOutput.Text=strData
End Sub
```

要创建第二个 UDP 伙伴，请按照以下步骤执行：

- 在工程中添加一个标准窗体。
- 将窗体的名字修改为 frmPeerB。
- 将窗体的标题修改为“Peer B”。
- 在窗体中放入一个 Winsock 控件，并将其命名为 udpPeerB。
- 在“属性”页上，单击“协议”按钮并将协议修改为“UDPProtocol”。
- 在窗体上添加两个 TextBox 控件。将第一个命名为 txtSend，第二个命名为 txtOutput。
- 在窗体中添加如下的代码。

```
Private Sub Form_Load()
'控件的名字为 udpPeerB。
With udpPeerB
'重点:必须将 RemoteHost 的值改为
'计算机的名字。
.RemoteHost= "PeerA"
.RemotePort=1002    '要连接的端口。
.Bind 1001              '绑定到本地的端口上。
End With
End Sub

Private Sub txtSend_Change()
'在键入后立即发送文本。
udpPeerB.SendData txtSend.Text
End Sub

Private Sub udpPeerB_DataArrival _
(ByVal bytesTotal As Long)
Dim strData As String
udpPeerB.GetData strData
txtOutput.Text=strData
End Sub
```

如果要试用上面的例子，按 F5 键运行工程，然后在两个窗体的 txtSend TextBox 中分别键入一些文本。键入的文字将出现在另一个窗体的 txtOutput TextBox 中。

在上面的代码中，在创建 UDP 应用程序时调用了 Bind 方法，这是必须的。Bind 方法的作用是为控件“保留”一个本地端口。例如，如果将控件绑定到 1001 号端口，那么其他应用程序将不能使用该端口进行“监听”。该方法阻止其他应用程序使用同样的端口。

Bind 方法的第二个参数是任选的。如果计算机上存在多个网络适配器，可以用 LocalIP 参数来指定使用哪一个适配器。如果忽略该参数，控件使用的将是计算机上“控制面板”设

置中“网络”控制面板对话框中列出的第一个适配器。

在使用 UDP 协议的时候，可以任意地改变 RemoteHost 和 RemotePort 属性，同时始终保持绑定在同一个 LocalPort 上。TCP 协议与此不同，在改变 RemoteHost 和 RemotePort 属性之前，必须先关闭连接。

2.3 Internet Transfer 控件

Internet Transfer 控件实现了两种广泛使用的 Internet 协议：超文本传送协议（HyperText Transfer Protocol，HTTP）和文件传送协议（File Transfer Protocol，FTP）。使用 Internet Transfer 控件可以通过 OpenURL 或 Execute 方法连接到任何使用这两个协议的站点并检索文件。

可能有以下的用途：

- 在应用程序中添加 FTP 浏览器。
- 创建自动从公共 FTP 站点下载文件的应用程序。
- 分析 World WideWeb 站点中的图形引用，并只下载图形。
- 提供以自定义格式显示从 Web 页获得的动态数据。

Internet Transfer 控件的功能依赖于将要使用的协议。由于所支持的两种协议工作起来不尽相同，所能够进行的操作就依赖于正在使用的协议。例如，GetHeader 方法只能用于 HTTP（HTML 文档）协议。

然而，有些过程对两个协议是通用的。最基本的，如果要使用任何一个协议，则必须符合以下条件。

- 将 AccessType 属性设置为合法的代理服务器。
- 用合法的 URL 调用 OpenURL 方法。
- 用合法的 URL 和协议支持的命令调用 Execute 方法。
- 用 GetChunk 方法从缓冲区获取数据。

2.3.1 Internet Transfer 控件属性

1. AccessType 属性

设置或返回一个值，决定该控件用来与 Internet 网进行通信的访问类型（通过代理访问或直接访问）。正在处理异步请求时，该值可以改变，但直到创建了下一个连接时，改变才会生效。

使用语法如下：

```
Winsock1.AccessType=type
```

其中参数 type 是整数（枚举型）。数值表达式，决定所使用的访问类型，参见“设置值”中的描述。

type 设置值如表 2-13 所示。

表 2-13　　type 的设置值

常数	值	描述
icUseDefault	0	使用缺省值。控件使用在注册表中找到的缺省设置值来访问 Internet
icDirect	1	直接连到 Internet 网。控件直接连到 Internet
icNamedProxy	2	命名代理。指示控件使用 Proxy 属性中指定的代理服务器

2. Document 属性

返回或设置与 Execute 方法一起使用的文件或文档。如果未指定该属性，将返回服务器中的缺省文档；如果不指定文档写操作会发生错误。

使用语法如下：

```
Winsock1.Document=string
```

参数 string 表示与 Execute 方法一起使用的文件或文档的名称。

3. hInternet 属性

从下一级的 Wininet.dll API 返回 Internet 句柄。可直接使用该句柄来调用该 API。从 Visual Basic 访问控件时，不能使用该属性。

使用语法如下：

```
Winsock1.hInternet
```

返回数据类型：Long

4. Password 属性

设置或返回一个密码，该密码将和请求一起被发送，用以在远程计算机上登录。如果该属性为空，控件将发送一个缺省的密码。

使用语法如下：

```
Winsock1.Password=string
```

参数 string 表示当登录到远程计算机时，要发送的密码。

如表 2-14 所示，控件发送的缺省密码将取决于确切的方案。

表 2-14　　用户名和密码的设置值

UserName 属性	Password 属性	发送到 FTP 服务器的 UserName	发送到 FTP 服务器的 Password
Null 或 ""	Null 或 ""	"anonymous"	用户的电子邮件名
非空字符串	Null 或 ""	UserName 属性	""
Null	非空字符串	错误	错误
非空字符串	非空字符串	UserName 属性	Password 属性

5. Protocol 属性

设置或返回一个值，指定和 Execute 方法一起使用的协议。

使用语法如下：

```
Winsock1.Protocol=integer
```

参数 integer 是整数。数值表达式，决定所用的协议。

Protocol 的有效设置值如表 2-15 所示。

表 2-15　　Protocol 的有效设置值

常数	值	描述
icUnknown	0	未知的
icDefault	1	缺省协议
icFTP	2	FTP。文件传输协议
icReserved	3	为将来预留
icHTTP	4	HTTP，超文本传输协议
icHTTPS	5	安全 HTTP

✍ 指定该属性后，URL 属性被更新以显示新值。另外，如果此 URL 的协议部分被更新，Protocol 属性也将被更新以体现新值。OpenURL 和 Execute 方法都可能会修改该属性值。直到调用下一个 Execute 或 OpenURL 方法时，该属性值的改变才会有效。

6. Proxy 属性

设置或返回用以和 Internet 网进行通信的代理服务器的名称。只有当 AccessType 属性设置为 icNamedProxy(3)时，才使用该属性。

使用语法如下：

```
Winsock1.Proxy=proxy
```

参数 proxy 表示所用的代理服务器的名称。

数据类型：String

✍ 直到调用下一个 Execute 或 OpenURL 方法时，该属性值的改变才会有效。

可以为每一个协议设置单独代理。例如，如果网络有一个用于 ftp 协议的名为“ CorpFTP ”的代理，且使用了端口 123，则可以将期设置为：

```
Inet1.Proxy="ftp=CorpFTP:123"
```

可以指定多个网关，用空格将其相互分开。例如，如果 HTTP 代理名称为“ CorpHTTP ”，并且使用了端口 131，则为两个协议做如下设置：

```
Inet1.Proxy="ftp=CorpFTP:123 HTTP=CorpHTTP:131"
```

如果没有指定要设置哪一个协议，则为所有的协议使用相同的代理名称。例如，即使协议是 http，也使用以下的代理名称：

```
Inet1.Proxy="CorpFTP:123"
```

7. RemoteHost 属性

返回或设置远程计算机，控件向它发送数据或从它那里接收数据。既可提供主机名，比如 “ FTP://ftp.microsoft.com ”，也可提供点格式下的 IP 地址字符串，比如“ 100.0.1.1 ”。

使用语法如下：

```
Winsock1.RemoteHost=string
```

参数 string 表示远程计算机的名称或地址。

✍ 在指定该属性时，应更新 URL 属性来显示新值。如果更新 URL 的主机部分，则也要更新该属性来反映新值。在调用 OpenURL 或 Execute 方法时也可改变 RemoteHost 属性。在运行时直到下一个连接之前，改变该值都不会造成什么影响。

8. RemotePort 属性

返回或设置要连接的远程端口号

使用语法如下：

```
Winsock1.RemotePort=port
```

参数 port 表示要连接的端口。该属性的缺省值是 80。

数据类型：Long

✎ 在设置 Protocol 属性时，将对每个协议自动把 RemotePort 属性设置成适当的缺省端口。表 2-16 表列出缺省端口号。

表 2-16　　缺省端口号

端口	描述
80	HTTP，通常用于 World WideWeb 连接
21	FTP

9. RequestTimeout 属性

设置或返回在超时截止之前，按秒计算的等待时间长度。如果请求在指定的时间内还没有响应，并且该请求使用 OpenURL 方法（同步地），就会产生错误；如果请求使用 Execute 方法，将引发带错误码的 StateChanged 事件。把该属性设置为 0，则意味着不限定等待时间。

使用语法如下：

```
object.RequestTimeout=time
```

参数 time 表示在错误出现之前，按秒计算的等待时间长度。

数据类型：Long

10. ResponseCode 属性

StateChanged 事件中出现 icError (11)状态时，从连接返回错误码。要获得此错误的描述，检查 ResponseInfo 属性。

使用语法如下：

```
object.ResponseCode= code
```

参数 code 表示远程服务器返回的数字。

数据类型：Long

如下所示，可使用 StateChanged 事件来接收关于错误的通知：

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
  Dim strMess As String '消息变量。
  Select Case State
  '...Other cases not shown.
  Case icError  '11
    '得到错误文本。
    strMess="ErrorCode: " & Inet1.ResponseCode & _
    " : " & Inet1.ResponseInfo
  End Select
```

```
    Debug.Print strMess
End Sub
```

11. ResponseInfo 属性

返回最后发生的错误的文本。想得到该错误码，检查 ResponseCode 属性。

使用语法如下：

```
Winsock1.ResponseInfo
```

参数 info 表示连接返回的响应。

返回类型：String

如下所示，使用 StateChanged 事件来接收错误的通知。

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim strMess As String '消息变量。
    Select Case State
    '...没有列举其他情况。
    Case icError  ' 11
        '得到错误文本。
        strMess="ErrorCode: " & Inet1.ResponseCode & _
        " : " & Inet1.ResponseInfo
    End Select

    Debug.Print strMess
End Sub
```

12. StillExecuting 属性

返回一个值，指明此 Internet Transfer 控件是否处于忙状态。如果该控件正在做诸如从 Internet 网上检索文件之类的操作，将返回 True。当该控件处于忙的时候，不响应其他的请求。

使用语法如下：

```
Winsock1.StillExecuting=boolean
```

参数 boolean 是布尔表达式，指明该控件是否处于忙状态。

数据类型：Boolean

boolean 的设置值如表 2-17 所示。

表 2-17 boolean 的设置值

常数	值	描述
True	-1	该控件忙
False	0	该控件空闲

13. URL 属性

设置或返回 Execute 或 OpenURL 方法使用的 URL。

使用语法如下：

```
Winsock1.URL [= url]
```

参数 url 是字符串，指定 Execute 方法中使用的 URL。

```
数据类型：String
```

✍ 调用 OpenURL 或 Execute 方法会改变该属性的值。直到下一次调用 OpenURL 或 Execute 方法时，对该属性所做的改变才有效。URL 属性至少必须包含一个协议和一个远程主机名。

URL 属性可以是目录或文件。例如，下面这两个 URL 都是有效的。

```
'设置该 URL，仅返回文件目录：
Inet1.URL="HTTP://www.microsoft.com"
'然而，该 URL 将返回文件的文本：
Inet1.URL="HTTP://www.microsoft.com/disclaimer.txt
```

14. UserName 属性

设置或返回与请求一起发送到远程计算机的名称。如果该属性为空，当提出请求时，该控件将把“ anonymous ”作为用户名来发送。

使用语法如下：

```
object.UserName[= name]
```

参数 name 是字符串，指定 Execute 方法使用的 UserName。

UserName 属性的语法包含下面部分：

```
数据类型：String
```

✍ 调用 OpenURL 或 Execute 方法会改变该属性的值。直到下一次调用 OpenURL 或 Execute 方法时，对该属性所做的改变才有效。

15. Cancel 方法

取消当前请求，并关闭当前创建的所有连接。

使用语法如下：

```
Winsock1.Cancel
返回值：无返回值
```

2.3.2 Internet Transfer 控件方法

1. Execute 方法

执行对远程服务器的请求。只能发送对特定的协议有效的请求。

使用语法如下：

```
Winsock1.Execute url, operation, data, requestHeaders
```

Execute 属性的语法如表 2-18 所示。

表 2-18　　Execute 属性的语法

部分	描述
url	可选的。字符串，指定控件将要连接的 URL。如果这里未指定 URL，将使用 URL 属性中指定的 URL
operation	可选的。字符串，指定将要执行的操作类型。所支持的操作的列表，参见下面的“设置值”
data	可选的。字符串，指定用于操作的数据（参见下面的“设置值”。）
requestHeaders	可选的。字符串，指定由远程服务器传来的附加的标头。它们的格式为：header name: header value vbCrLf

operation 的有效设置值如表 2-19 所示（支持 http 的命令）：

表 2-19　　operation 的有效的设置值

运算	描述
GET	检索由 URL 属性指定的 URL 中的数据
HEAD	发送请求的标头
POST	传递数据给服务器。该数据在 data 参数中。这是 GET 的替代方法，附加的指令在 data 参数中指定
PUT	Put 操作。被替代的页面名在 data 参数中

下面介绍支持的 FTP 命令。

✍ FTP 协议使用单个字符串，该字符串包含操作名以及操作所需的其他参数。换句话说，即不使用 data 和 requestHeaders 参数；所有的操作及操作的参数是在 operation 参数中作为单个字符串来传递的。各参数间由空格分隔。在下面的描述中，不要把“ file1 ”、“ file2 ”与 data , requestHeaders 参数搞混。

FTP 操作的语法为：

```
operationName file1 file2.
```

例如，为获得一个文件，下面的代码调用 Execute 方法，该方法包含着操作名("GET") 以及此操作所需的两个文件名：

```
Inet1.Execute "FTP://ftp.microsoft.com", "GET Disclaimer.txt
C:\Temp\Disclaimer.txt"
```

✍ 不支持嵌入空格的文件名。

operation 有效的 FTP 设置值，如表 2-20 所示。

表 2-20　　operation 有效的 FTP 设置值

操作	描述	示例
CD file1	改变目录。改变到由 file1 指定的目录中	Execute , "CD docs\mydocs"
CDUP	改变到父目录。功能与“CD ..”相同	Execute , "CDUP"
DELETE file1	删除由 file1 指定的文件	Execute , "DELETE discard.txt"
DIR [file1]	在由 file1 指定的目录中查找。如果没有指定 file1 目录，则查找当前工作目录。使用 GetChunk 方法返回数据	Execute , "DIR /mydocs"
GET file1 file2	获取由 file1 指定的远程文件，并创建由 file2 指定的新的本地文件	Execute , _ "GET getme.txt C:\gotme.txt"

续表

操作	描述	示例
MKDIR file1	创建由 file1 指定的目录。是否能够成功地执行，取决于用户在远程主机上的权限	Execute , "MKDIR /myDir"
PUT file1 file2	将由 file1 指定的本地文件，复制到由 file2 指定的远程主机文件中	Execute , _ "PUT C:\putme.txt /putme.txt"
PWD	打印工作目录。返回当前目录的名称。用 GetChunk 方法返回数据。	Execute , "PWD"
QUIT	结束当前连接	Execute , "QUIT"
RECV file1 file2	与 GET 相同	Execute , _ "RECV getme.txt C:\gotme.txt"
RENAME file1 file2	文件重命名。是否能够成功地执行，取决于用户在远程主机上的权限	Execute , "RENAME old.txt new.txt"
RMDIR file1	删除目录。是否能够成功地执行，取决于用户在远程主机上的权限	Execute , "RMDIR oldDir"
SEND file1	将文件复制到 FTP 站点（与 PUT 相同）	Execute , _ "SEND C:\putme.txt /putme.txt"
SIZE file1	返回由 file1 指定文件的大小	Execute "SIZE /largefile.txt"

返回类型：None

✍ **上面列出的许多命令都只有当用户在主机服务器上具有相应的权限时，才能够执行。例如，匿名的 FTP 站点不允许任何人删除文件或目录。**

2. GetChunk **方法**

从 StateChanged **事件中检索数据。把** Execute **方法当作** GET **操作来调用之后使用该方法。使用语法如下：**

```
Winsock.GetChunk( size [,datatype] )
```

参数 Size **必需，长整型数值表达式，决定被检索的块的大小。**
参数 datatype**，可选的。整数，决定被检索块的数据类型，如下面的“设置值”所示。**
datatype **的设置值如表** 2-21 **所示。**

表 2-21　　datatype **的设置值**

常数	值		描述
icString	0		缺省值。把数据作为字符串来检索
icByteArray	1		把数据作为字节数组来检索

返回类型：Variant

✍ **在** StateChanged **事件中使用** GetChunk **方法。当** State **属性为** icResponseCompleted(12) **时，使用** GetChunk **方法检索缓冲区的内容。**

3. GetHeader **方法**

GetHeader **方法用于检索** HTTP **文件的标头文本。**
使用语法如下：

```
Winsock1.GetHeader (hdrName)
```

参数 hdrName 可选的。字符串，指定将被检索的标头

返回类型：String

✍ 如果没有已命名的标头，将返回所有的标头。

表格 2-22 列举了一些典型的可用标头：

表 2-22 典型的可用标头

标头	描述
Date	返回文档传输的日期和时间。返回的数据格式为：Wednesday, 27-April-96 19:34:15 GMT
MIME-version	返回 MIME 协议的版本号，目前为 1.00
Server	返回服务器的名称
Content-length	返回数据的字节长度
Content-type	返回数据的 MIM 的当前类型
Last-modified	返回最后一次修改文档的日期和时间。返回的数据格式为：Wednesday, 27-April-96 19:34:15 GMT

4. OpenURL 方法

打开并返回指定 URL 的文档。文档以变体型返回。该方法完成时，URL 的各种属性（以及该 URL 的一些部分，如协议）将被更新，以符合当前的 URL。

使用语法如下：

```
Winsock1.OpenUrlURL[,datatype]
```

参数 url 必需的。表示被检索文档的 URL。

参数 datatype 可选的。是整数，如“设置值”所示，指定数据类型。

datatype 的设置值如表 2-23 所示。

表 2-23 datatype 的设置值

常数	值	描述
icString	0	缺省值。把数据作为字符串来检索
icByteArray	1	把数据作为字节数组来检索

返回类型：Variant

✍ OpenURL 方法的返回值取决于 URL 的目标。例如，如果 URL 的目标是某个 FTP 服务器的目录，将返回该目录。另一方面，如果目标是一个文件，则检索该文件。

OpenURL 方法等效于：调用带 GET 操作的 Execute 方法，然后在 StateChanged 事件中调用 GetChunk 方法。但是，OpenURL 方法会导致从站点返回同步数据流。

如下所示，如果正在检索一个二进制文件，在把它写到磁盘上之前，请务必使用一个字节数组作为临时变量。

```
Dim b() As Byte
Dim strURL As String
'设置 strURL 为一个有效的地址。
strURL="FTP://ftp.GreatSite.com/China.exe"
b()=Inet1.OpenURL(strURL, icByteArray)
```

```
Open "C:\Temp\China.exe" For Binary Access _
Write As #1
Put #1, , b()
Close #1
```

✍ 当使用 OpenURL 方法时，在设置 Password 和 UserName 属性之前，设置 URL 属性。如果最后设置 URL 属性，UserName 和 Password 属性将被置为""。

2.3.3　Internet Transfer 控件事件

StateChanged 事件

连接中状态发生改变，就会引发该事件。

使用语法如下：

```
Winsock1_StateChanged(ByVal State As Integer)
```

参数 State，整数。如下面的“设置值”所示，指定状态。

State 的设置值如表 2-24 所示。

表 2-24　　State 参数的设置值

常数	值		描述
icNone	0		无状态可报告
icHostResolvingHost	1		该控件正在查询所指定的主机的 IP 地址
icHostResolved	2		该控件已成功地找到所指定的主机的 IP 地址
icConnecting	3		该控件正在与主机连接
icConnected	4		该控件已与主机连接成功
icRequesting	5		该控件正在向主机发送请求
icRequestSent	6		该控件发送请求已成功
icReceivingResponse	7		该控件正在接收主机的响应
icResponseReceived	8		该控件已成功地接收到主机的响应
icDisconnecting	9		该控件正在解除与主机的连接
icDisconnected	10		该控件已成功地与主机解除了连接
icError	11		与主机通信时出现了错误
icResponseCompleted	12		该请求已经完成，并且所有数据均已接收到

✍ 一般来说，使用 StateChanged 事件决定何时使用 GetChunk 方法来检索数据。要这样做，须使用 Select Case 语句，并测试 icResponseReceived(8)或 icResponseCompleted(12)。

当该控件已完成一个操作时，且此操作在缓冲区中没有产生任何数据，此时 icResponseReceived 状态也可能出现。例如，当与某个 FTP 站点进行连接时，该控件将与此 FTP 站点“握手”，但没有在缓冲区中产生任何数据，此时会出现 icResponseReceived 状态。

另一方面，一个操作完全完成后，会出现 icResponseCompleted 状态。例如，如果正在使用 Execute 方法和 GET 操作来检索某个文件，在此文件被完全检索之后，将出现 icResponseCompleted 事件，且仅出现一次。

实际上，使用 icResponseReceived 状态可以对数据做语法分析，直到检索到所需信息为止（例如，检索 HTML 文件时，只对标头进行检索）。获得该信息后，就可以取消这次检索。

另一方面，如果想检索整个文件，icResponseCompleted 状态还会通知传输已经完成，可以继续。

2.3.4 Internet Transfer 控件的使用

1. 设置 AccessType 属性

为了与 Internet 建立任何形式的连接，必须确定计算机如何连接到 Internet 上。如果在 intranet 上，可能需要提供代理服务器才能连接到 Internet 上。

简单地说，代理服务器是计算机和 Internet 之间的媒介。Intranet 上所有需要连接到 Internet 上的计算机，都必须通过代理服务器。代理行使 Intranet 和 Internet 之间的防火墙功能，能够阻止非法的最终用户和外部请求，也就保护了 Intranet 不受破坏。

要查找计算机中的代理设置值，请按照以下步骤执行：

✍ 下面的步骤只能用于 Windows 9x 和 Windows NT(R) 4.0 系统。

- ◆ 进入“控制面板”窗口，“双击“Internet”图标。
- ◆ 在 Internet 属性对话框中，单击“连接”选项卡。
- ◆ 在“代理服务器”项目中，确认选中了“通过代理服务器连接”复选框。

如果选中了，则单击“设置”按钮。在该对话框中可以找到能够用于多种协议的代理服务器的名称。如果没有定义代理服务器，请与系统管理员联系，以获得可用的代理服务器。

如果希望使用对话框中未列出的代理服务器，可将 AccessType 属性设置为 icNamedProxy (2)。然后将 Proxy 属性设置为代理服务器的名称，如下面的代码所示：

```
Inet1.Proxy="myProxyName"
Inet1.AccessType=icNamedProxy
```

另一方面，如果希望使用缺省代理服务器（由计算机的注册表决定），则可以忽略 Proxy 属性，而只需将 AccessType 设置为 icUseDefault (0)。

AccessType 设置值在表 2-25 中列出：

表 2-25　　AccessType 的设置值

常数	值	描述
icUseDefault	0	（缺省）用作缺省。控件使用注册表中的缺省设置访问 Internet
icDirect	1	直接连接 Internet。该控件可直接连接到 Internet
icNamedProxy	2	命名的代理服务器。指示该控件使用 Proxy 属性确定的代理服务器

2. 使用 OpenURL 方法

设置完 AccessType 属性后，最基本的操作就是用合法地 URL 调用 OpenURL 方法。使用 OpenURL 方法时，操作所得到的结果将依赖于目标 URL。例如下面的 URL 将返回在 www.microsoft.com 中找到的 HTML 文档：

```
'名为"Text1"的 TextBox 控件保存了
'该方法的结果。Internet 传输
'控件的名称是"Inet1"。
Text1.Text=Inet1.OpenURL("http://www.microsoft.com")
```

在这种情况下，缺省操作返回的是 URL 定位的 HTML 文档。然而，如果 URL 被改为指向文本文件，则将获得实际的文件。例如，下面的代码：

```
Text1.Text=Inet1.OpenURL("ftp://ftp.microsoft.com/disclaimer.txt")
```

✍ 在使用 OpenURL 或 Execute 方法时，不需要设置 Protocol 属性。Internet Transfer 控件会自动按 URL 的协议部分确定的协议来设置。

最后，可以用包含附加数据的 URL 调用 OpenURL 方法。例如，很多 Web 站点提供了搜索数据库的能力。要搜索数据库，则需要在发送的 URL 中包含搜索条件。例如下面的代码用条件“ find=Maui ”调用名为“ search.exe ”的搜索引擎。

```
Dim strURL As String
strURL=_
"http://www.howzit.com/cgi-bin/search.e xe?find=maui
Text1.Text=Inet1.OpenURL(strURL)
```

如果搜索引擎找到了符合条件的内容，将合成一个 HTML 文档并携带适当的信息返回。

3. 数据存盘

如果需要将通过 OpenURL 方法获取的数据保存到文件，可以使用 Open、Put 和 Close 语句，如下面的代码所示。该示例先将获得的二进制文件传入 Byte 数组，然后将该数据保存到磁盘中：

```
Dim strURL As String
Dim bData() As Byte      '数据变量
Dim intFile As Integer   '可用文件变量
strURL="ftp://ftp.microsoft.com/Softlib/Softlib.e xe"
intFile=FreeFile()      '将 intFile 设置为未使用的文件
'OpenURL 方法的结果首先传入 Byte 数组，
'然后将 Byte 数组保存到磁盘。
bData()=Inet1.OpenURL(strURL, icByteArray)
Open "C:\Temp\Softlib.e xe" For Binary Access Write _
As #intFile
Put #intFile, , bData()
Close #intFile
```

可用类似的过程将文本文件写入磁盘中，不同的只是不再需要 Byte 数组了，数据可以直接保存到文件中：

```
Dim strURL As String     'URL 字符串
Dim intFile As Integer   '可用文件变量
IntFile=FreeFile()
strURL="http://www.microsoft.com"
Open "c:\temp\MSsource.txt" For Output _
As #IntFile
Write #IntFile, Inet1.OpenURL(strURL)
```

```
Close #IntFile
```

4. 同步异步传输

OpenURL 方法以同步方式传输数据。在这里，同步指的是传输操作未完成之前，不能执行其他过程。这样数据传输就必须在执行其他代码之前完成。

而 Execute 方法以异步方式传输数据。在调用 Execute 方法时，传输操作与其他过程无关。这样，在调用 Execute 方法后，在后台接收数据的同时，即可同时执行其他代码。

对 InternetTransfer 控件的使用者来说这意味着什么？简单地说，用 OpenURL 方法能够直接得到可保存到磁盘的数据流（如上所述），或者直接在 TextBox 控件中阅览（如果数据是文本格式的）。从另一方面说，如果用 Execute 方法获取数据，则必须用 StateChanged 事件监视该控件的连接状态。当达到适当的状态时，调用 GetChunk 方法从控件的缓冲区获取数据。下面更详细地讨论这一操作。

（1）在 FPT 协议中使用 Execute 方法

Execute 方法具有 4 个参数：URL、operation、data 和 requestHeaders。FTP 操作只用到了 operation 参数和 URL 参数，其中后者是可选的。例如，要从远程计算机中得到一个文件，可用下面的代码：

Inet1.Execute “FTP://ftp.microsoft.com”, “GET disclaimer.txt C:\Temp\Disclaimer.txt”

如果正在用 FTP 从匿名 FTP 服务器中获取文件，就应熟悉在服务器目录树中漫游的特定命令，以及将其中文件获取到本地的硬盘中的命令。例如，要用 FPT 协议改变目录，应使用带有希望改变到的目录路径的“CD”命令。

对绝大多数通用操作，如将文件传入服务器，以及从服务器获取文件，Internet 传输控件在 Execute 方法中使用了（与 FTP）相同或相近的命令。例如，下面的代码将“CD”命令作为 Execute 方法的参数以改变路径：

```
'txtURL 文本框包含了要打开的路径。
'txtRemotePath 文本框包含了要改变到的路径。
Inet1.Execute txtURL.Text, "CD " & txtRemotePath.Text
```

✍ 在 Execute 方法中使用 FTP 命令时，没有用到 data 和 requestHeaders 参数。所有的操作和它们的参数都在 operation 参数中作为字符串进行传递；参数之间用空格进行分隔。在下面的描述中，不要把“file1”和“file2”项与 data 和 requestHeaders 参数搞混。

FTP 操作的语法是：

operationName file1 file2

例如，要获取文件，在下面的代码中包含了操作的名称（“获取”），以及该操作所需的两个文件名：

```
'得到名为"Disclaimer.txt"的文件，并将其复制到
' C:\Temp\Disclaimer.txt。
Inet1.Execute, _
"GET Disclaimer.txt C:\Temp\Disclaimer.txt"
```

✍ 如果代理服务器是 CERN 代理服务器，就不允许使用直接的 FTP 连接（使用 Execute

方法)。在这种情况下，要获得文件，则需使用带 Open、Put 和 Close 语句的 OpenURL 方法，就象前面“用 OpenURL 方法保存到文件”提到的那样。还可以用 OpenURL 方法得到目录列表，即将目标目录作为 URL，并调用该方法。

（2）在 HTTP 协议上使用 Execute 方法

HTTP 协议允许客户机用 GET、HEAD、POST 和 PUT 命令向服务器请求数据。表 2-26 中列出了这些操作：

表 2-26　　HTTP 协议的各种命令

值	描述	描述
GET	获取 URL 中命名的文件	Execute “http://www.microsoft.com” & _ “/default.htm”, “GET”
HEAD	只获取 URL 属性中命名的文件的文件标头	Execute , “HEAD”
POST	提供附加数据，以支持向远程主机的请求	Execute , “POST”, strFormData
PUT	替换指定的 URL 中的数据	Execute , “PUT”, “replace.htm”

（3）通用网关接口和 Execute 方法

很多 World Wide Web 站点提供了搜索数据库的能力。它是通过 HTTP 协议用通用网关接口 （CGI） 发送查询的能力完成的。

在这里不再讨论 CGI 了。然而，如果对 CGI 比较了解，就可用 Execute 方法构造一个应用程序模拟 World WideWeb 站点的行为。例如，下面的代码给出了典型的 CGI 查询字符串：

```
http://www.yippee.com/cgi-bin/find.e xe?find=Hangzhou
```

用 Execute 方法也可以发送同样的查询，介绍如下。

```
Dim strURL As String, strFormData As String
strURL="//www.yippee.com/cgi-bin/find.e xe"
strFormData="find=Hangzhou"
Inet1.E xecute strURL, "POST", strFormData
```

如果希望得到从服务器发回的结果（如上面的示例所示），就必须使用 GetChunk 方法以获取作为结果的 HTML 文档。

（4）在 State 事件中使用 GetChunk 方法

在从远程计算机下载数据时，将建立异步连接。例如，在 Execute 方法中使用“获取”操作，将使服务器获取请求的文件。当获取了整个文件之后，State 参数将返回 icResponseCompleted(12)。在这时候，就可以用 GetChunk 方法从缓冲区中获取数据了。下面的示例中给出了这种情况：

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
   Dim vtData As Variant '数据变量。
   Select Case State
   '…没有给出其他情况。
```

```
    Case icResponseCompleted '12
      '打开文件用于写入。
      Open txtOperation For Binary Access _
      Write As #intFile

      '得到第一个大块。注意：指定 Byte 数组
      ' (icByteArray) 以获取二进制文件。
      VtData=Inet1.GetChunk(1024, icString)

      Do While LenB(vtData) > 0
        Put #intFile, , vtData
        '得到下一大块。
        VtData=Inet1.GetChunk(1024, icString)
      Loop
      Put #intFile, , vtData
      Close #intFile
    End Select
  End Sub
```

登录到 FTP 服务器

FTP 服务器有两种：公用的和私用的。公用服务器，顾名思义，是对所有人开放的。而私用服务器，除了该服务器的真正用户，其他人不能登录。在这两种情况下，FTP 协议都要求提供用户名和密码。这两个参数用来验明用户，并允许（或禁止）进一步的操作。

要登录到公用服务器，通常的做法是以“匿名”登录 (UserName=“anonymous”)，然后发送用户的电子邮件名称作为密码。然而使用 Internet 传输控件这一过程还能够进一步简化。按照缺省规定，如果没有提供 UserName 和 Password 属性值，该控件以“匿名”作为 UserName，用户的电子邮件名称作为 Password。

如果要登录到私用服务器，只需适当地设置 UserName、Password 和 URL 属性，并调用 Execute 方法，如下面的示例所示：

```
  With Inet1
    .URL="ftp://ftp.someFTPSite.com"
    .UserName="John Smith"
    .Password="mAuI&9$6"
    .Execute ,"DIR"   '返回该目录。
    .Execute ,"CLOSE" '关闭连接。
  End With
```

在调用 Execute 方法之后，FTP 连接仍旧打开着。这时可以继续使用 Execute 方法完成其他 FTP 操作，比如 CD 和 GET。如果会话已经完成，使用 Execute 方法执行 CLOSE 操作以关闭连接。也可以通过改变 URL 属性，并调用 OpenURL 或 Execute 方法，自动关闭该

连接，这样的操作会关闭当前 FTP 连接，并打开新的 URL。

2.4　MSComm 控件

MSComm 控件通过串行端口传输和接收数据，为应用程序提供串行通信功能。MSComm 控件提供下列两种处理通信的方式：

事件驱动通信是处理串行端口交互作用的一种非常有效的方法。在许多情况下，在事件发生时需要得到通知，例如，在 Carrier Detect (CD)或 Request To Send (RTS)线上一个字符到达或一个变化发生时。在这些情况下，可以利用 MSComm 控件的 OnComm 事件捕获并处理这些通信事件。OnComm 事件还可以检查和处理通信错误。所有通信事件和通信错误的列表，参阅 CommEvent 属性。

在程序的每个关键功能之后，可以通过检查 CommEvent 属性的值来查询事件和错误。如果应用程序较小，并且是自保持的，这种方法可能是更可取的。例如，如果写一个简单的电话拨号程序，则没有必要对每接收一个字符都产生事件，因为唯一等待接收的字符是调制解调器的“确定”响应。

每个使用的 MSComm 控件对应着一个串行端口。如果应用程序需要访问多个串行端口，必须使用多个 MSComm 控件。可以在 Windows“控制面板”中改变端口地址和中断地址。

2.4.1　MSComm 控件的属性

1. CommPort 属性

设置或返回连接的串口编号。必须指定该属性，Windows 将会利用该串口和外界通信。CommPort 属性值用 1，2，…表示串口 COM1，COM2…，其取值范围为 l~16，缺省为 1。MSComm 控件支持的最大串口号是 16 个，CommPort 属性值超过 16，系统会通知出错，并返回前一次设定的值。

2. Settings 属性

设置或返回通信参数，值为字符串型表示，主要设定数据传输率、奇偶检验、数据位数，停止位等 4 个参数。组成格式为：

```
"BBBB，P，D，S"
```

其中 BBBB 为数据传输率，P 为奇偶校验，D 为数据位数，S 为停止位数。默认的为“19200，N，8，1”表示传输速率 19200bit/s，无奇偶校验位，8 位数据位，l 位停止位。

数据传输率的合法值在 110，300，600，1200，2400，4800，9600（默认），14400，19200，28800，38400，56000，57600，115200，128000，256000

奇偶校验的值可以设置为如表 2-27 所示的值之一。

表 2-27　　奇偶校验设定值

设定值	描述
E	偶校验 (EVEN)
M	标号校验(MARK)
N	无检验(NONE)
0	奇校验(ODD)
S	空格检验(Space)

数据位的值可以是：4、5、6、7、8（默认值）。
停止位的值有：1（默认值）、1.5、2。
Settings 的值只有通信的双方都一样，通信连接才有效。

3. Handshaking 属性

设置或返回硬件握手协议，即 PC 与通信设备（如 Modem）之间为了控制流速而约定的内部协议，取值如表 2-28 所示。

表 2-28　　奇偶校验设定值

设定值	值	描述
ComNone	0	默认值，无握手协议
ComXOnXOff	1	Xon/Xoff 握手协议
ComRTS	2	RTS/CTS 握手协议
comRTSXOnXOff	3	Xon/Xoff 握手协议和 RTS/CTS 握手协议

4. PortOpen 属性

打开或关闭喘口。值为 Boolean 型。设为 True 可以打开端口；设为 False 可以关闭端口。一般在程序开始时打开端口，在程序结束时关闭端口。

5. OutBuffersize 属性

设置或返回发送缓冲区大小，值为 Integer 型，表示传输缓冲区的字节数，缺省为 512 字节。

6. OutBufferCounter 属性

返回发送缓冲区内等待发送的字符数，可用来清空缓冲区。

7. OutPut 属性

向发送缓冲区写数据流。值为 Variant 型变量。

✍ 注意：传输文本数据时，应将 String 型数据放入 Variant 变量，传输二进制数据（即按字节）时，应将 Byte 型数组数据放入 Variant 变量。

8. Sthreshold 属性

该属性为一阀值，它确定当输出缓冲区内字节个数小于该值后就产生 OnComm 事件，并

且 CommEvent 属性会被设定为 ComEvSend。如果该值为 0（默认值），则数据传输事件不会产生 OnComm 事件

9. InBuffersize 属性

设置或返回输入缓冲区的大小，缺省为 1024 字节。

10. InBufferCount 属性

返回输入缓冲区内的等待读取的字节个数，可将该属性设置为 0 来清除接收缓冲区。

11. InputLen 属性

设置或返回接收缓冲区内用 Input 读人的个数。若取 0，则 INPUT 读取整个缓冲区的内容。

12. Input 属性

该属性表示从接收缓冲区移走一串字符，将缓冲区中收到的数据读入变量。值为 Variant 变量。

✍ 注意：当 InPutMode 属性值为 0（文本模式）时，变量中含 String 型数据。当 InputMode 属性值为 1（二进制模式）时，变量中含 Byte 型数组数据。

13. Rthreshold 属性

该属性为一阀值，它确定当接收缓冲区内字节个数达到或超过该值后就产生 OnComm 事件，并且 CommEvent 属性会被设定为 ComEvReceive。如果该值为 0（默认值），则数据输入缓冲区无论有多少数据都不会产生 OnComm 事件

14. InputMode 属性

设置或返回接收数据的类型。取值为 0，则用 Input 属性接收文本型数据。取值为 1，则用 Input 属性接收二进制数据。

15. CommEvent 属性

如果在通信过程中发生错误或事件，就会引发 OnComm 事件并且改变属性值，由 CommEvent 属性代码反映错误或事件类型，在通信程序的设计中可根据该属性值来执行不同的操作。

通信错误的设定值如表 2-29 所示。

表 2-29　　通信错误设定值

常数	值	描述
comEventBreak	1001	接受到一个中断信号
comEventFrame	1004	帧出错。硬件检测到一个帧出错。当双方设置的格式不一致时，就会引发此错误
comEventOverrun	1006	端口超速错误。一个字符没有在下一个字符到达之前被硬件读取，该字符就丢失了

续 表

常数	值	描述
comEventRxOver	1008	输入缓冲区溢出。输入缓冲区没有空间
comEventRxParity	1009	奇偶校验出错。硬件检验到奇偶校验出错
comEventTxFull	1010	输出缓冲区溢出。输出缓冲区已满，不能将字符排入输入缓冲区
comEventDCB	1011	从端口获得 DCB 数据时，发生未知的错误

通信事件常数的设定如表 2-30 所示。

表 2-30 通信事件常数设定值

常数	值	描述
ComEvSend	1	发送缓冲区的内容少于 SThreshold 指定的值
ComEvReceive	2	接收缓冲区内字符数达到 RThreshold 值，该事件在缓冲区中数据被移走前将持续产生，利用此事件可编写接收数据的过程
comEvCTS	3	CTS 线的状态发生变化
comEvDSR	4	DSR 线的状态发生变化。该事件只在 DSR 从 1 变化到 0 时才发生
comEvCD	5	CD(Carrier Detect)线发生变化
comEvRing	6	探测到振铃信号。一些 UART 可能不支持此事件
ComEvEOF	7	接收数据中出现文件结束（ASCII 码为 26）字符

16. EOFEnable 属性

若置 TRUE，则当输入中出现 EOF，就停止输入并产生 OnComm 事件。

17. DTREnable 属性

确定在通信时是否使 Data Terminal Ready（DTR）线有效。Data Terminal Ready 是计算机发送到调制解调器的信号，指示计算机在等待接受传输。

18. RTSEnable 属性

确定是否使 Request To Send（RTS）线有效。一般情况下，由计算机发送 Request To Send 信号到联接的调制解调器，以请示允许发送数据。

19. DSRHolding 属性

确定 Data Set Ready（DSR）线的状态。Data Set Ready 信号由调制解调器发送到相连计算机，指示作好操作准备。该属性在设计时无效，在运行时为只读。

20. CTSHolding 属性

确定是否可通过查询 Clear To Send（CTS）线的状态发送数据。Clear To Send 是调制解调器发送到相联计算机的信号，指示传输可以进行。该属性在设计时无效，在运行时为只读。

21. CDHolding 属性

通过查询 Carrier Detect（CD）线的状态确定当前是否有传输。 Carrier Detect 是从调制解调器发送到相联计算机的一个信号，指示调制解调器正在联机。该属性在设计时无效，在

运行时为只读。

2.4.2　MSComm 控件的事件

MSComm 控件只有一个事件，即 OnComm 事件。

利用 MSComm 控件编写的应用程序在通信过程中只发生错误或事件，就会引发 OnComm 事件并且改变属性值，并由 CommEvent 属性代码反映错误类型。在通信程序的设计中可根据 CommEvent 属性值来执行不同的操作。

2.4.3　利用 MSComm 控件通信步骤

通常以下面的步骤来使用 VB 的 MSComm 控件作通信控制：

- 加入 MSComm 对象。
- 设定通信端口号码，即 CommPort 属性。
- 设定通信协议，即 HandShaking 属性。
- 设定传输速度等参数，即 Settings 属性。
- 设定其他参数，若必要时再加上其他的属性设定。
- 开启通信端口，即将 PortOpen 属性设为 TRUE。
- 送出字符串或读入字符串，使用 Input 及 Output 属性。
- 使用完 MSComm 通信对象后，将通信端口关闭。

2.5　Winsock API

在 VB 中基本上所有的应用都可以用 Winsock 控件来实现，但是对于稍微低层的一些应用，仅仅靠控件是很难实现的，而且缺乏灵活性。在本书中，大部分的应用都是通过控件来实现，但是也有一部分是通过 API 来实现的。通过 API 来实现，主要有两个目的，第一就是让读者了解到除了控件以外，还有功能更为强大的 API 可以来实现各种功能；其次就是有些功能控件不能够实现，只能够通过 API 来实现。

在 API 的编程中，由于 VB 不能像 VC 一样能够直接在程序中调用 API，而必须首先在程序中声明，所以，为了避免在各个程序中声明 API，这里介绍一个 API 模块，读者如果想通过 API 来编程，只需要在程序中直接加入这些模块就可以了。

具体模块名称就是 Winsock.bas，在开发的时候，直接引入就可以。

2.5.1　Winsock API 的函数声明

下面列出该文件的具体代码，具体程序详见本书附录光盘。

```
    ------------------------------------------Winsock.bas----------------
-------------------------------------
    Option Explicit
```

1. 常量声明

```
    '选项标志
    '
    Global Const SO_DEBUG=&H1                '调试信息记录
    Global Const SO_ACCEPTCONN=&H2           'socket 已经在侦听
    Global Const SO_REUSEADDR=&H4            '允许本地地址能够再次使用
    Global Const SO_KEEPALIVE=&H8            '保持连接
    Global Const SO_DONTROUTE=&H10           '使用接口地址
    Global Const SO_BROADCAST=&H20           '允许发送广播信息
    Global Const SO_USELOOPBACK=&H40         '
    Global Const SO_LINGER=&H80              '
    Global Const SO_OOBINLINE=&H100          '让收到的 OOB 数据在线

    Global Const SO_DONTLINGER=SO_LINGER Xor &HFFFFFFFF
    Global Const SO_EXCLUSIVEADDRUSE=SO_REUSEADDR Xor &HFFFFFFFF '不允许本地地址
被再次使用
    '
    '其他选项
    Global Const SO_SNDBUF=&H1001            '发送缓冲区大小
    Global Const SO_RCVBUF=&H1002            '结喉缓冲区大小
    Global Const SO_SNDLOWAT=&H1003          '发送标志
    Global Const SO_RCVLOWAT=&H1004          '接受标志
    Global Const SO_SNDTIMEO=&H1005          '发送超时
    Global Const SO_RCVTIMEO=&H1006          '接收超时
    Global Const SO_ERROR=&H1007             '获得错误状态并清除
    Global Const SO_TYPE=&H1008              '获得 socket 类型

    'WinSock 2 扩展新特性
    Global Const SO_GROUP_ID=&H2001          'socket 组的 ID
    Global Const SO_GROUP_PRIORITY=&H2002    '相关权重
    Global Const SO_MAX_MSG_SIZE=&H2003      '最大信息长度
    Global Const SO_PROTOCOL_INFOA=&H2004    'WSAPROTOCOL_INFOA 结构体
    Global Const SO_PROTOCOL_INFOW=&H2005    'WSAPROTOCOL_INFOW 结构体
    Global Const PVD_CONFIG=&H3001           '服务提供者的配置信息

    'TCP 选项
    Global Const TCP_NODELAY=&H1
    Global Const IP_OPTIONS=1        '设置或者获得 IP 选项
    Global Const IP_HDRINCL=2        '数据中包括 IP 头
```

```
Global Const IP_TOS=3           '服务的 IP 类型
Global Const IP_TTL=4           'IP time to live
Global Const IP_MULTICAST_IF=9      '设置或者获得 IP 广播
Global Const IP_MULTICAST_TTL=10        '设置或者获得 IP 广播 TTL
Global Const IP_MULTICAST_LOOP=11       '设置或者获得广播 LOOPBACK
Global Const IP_ADD_MEMBERSHIP=12       '增加 IP 组
Global Const IP_DROP_MEMBERSHIP=13      '删除 IP 组
Global Const IP_DONTFRAGMENT=14    '不要分隔 IP 数据

Type ip_mreq
   imr_multiaddr As Long   'IP 多播组
   imr_interface As Long   '本地 IP 地址接口
End Type

Global Const AF_UNSPEC=0                '不确定

Global Const AF_UNIX=1                  '管道协议
Global Const AF_INET=2                  'TCP、UDP 协议
Global Const AF_IMPLINK=3               'imp 地址
Global Const AF_PUP=4                   'pup 协议
Global Const AF_CHAOS=5                 'mit CHAOS 协议
Global Const AF_NS=6                    'XEROX NS 协议
Global Const AF_IPX=AF_NS               'IPX 协议如 IPX, SPX
Global Const AF_ISO=7                   'ISO 协议
Global Const AF_OSI=AF_ISO              '
Global Const AF_ECMA=8                  '欧洲计算机厂商
Global Const AF_DATAKIT=9               'datakit 协议
Global Const AF_CCITT=10                'CCITT 协议如 X.25
Global Const AF_SNA=11                  'IBM SNA
Global Const AF_DECnet=12               'DECnet
Global Const AF_DLI=13                  'Direct data link interface
Global Const AF_LAT=14                  'LAT
Global Const AF_HYLINK=15               'NSC Hyperchannel
Global Const AF_APPLETALK=16            'AppleTalk
Global Const AF_NETBIOS=17              'NetBios-style 地址
Global Const AF_VOICEVIEW=18            '
Global Const AF_FIREFOX=19              'Firefox 协议
Global Const AF_UNKNOWN1=20             '
Global Const AF_BAN=21                  'Banyan
```

```
Global Const AF_ATM=22                    'ATM 服务
Global Const AF_INET6=23                  '网络 6
Global Const AF_CLUSTER=24                '
Global Const AF_12844=25                  'IEEE 1284.4 WG AF
Global Const AF_IRDA=26                   'IrDA

Global Const AF_MAX=27

Global Const NSPROTO_IPX=1000
Global Const NSPROTO_SPX=1256
Global Const NSPROTO_SPXII=1257

Global Const ATPROTO_BASE=(1000 * AF_APPLETALK)
Global Const SOL_APPLETALK=(ATPROTO_BASE)

Global Const DDPPROTO_RTMP=(ATPROTO_BASE + 1)
Global Const DDPPROTO_NBP=(ATPROTO_BASE + 2)
Global Const DDPPROTO_ATP=(ATPROTO_BASE + 3)
Global Const DDPPROTO_AEP=(ATPROTO_BASE + 4)
Global Const DDPPROTO_RTMPRQ=(ATPROTO_BASE + 5)
Global Const DDPPROTO_ZIP=(ATPROTO_BASE + 6)
Global Const DDPPROTO_ADSP=(ATPROTO_BASE + 7)

Global Const DDPPROTO_MAX=(ATPROTO_BASE + 255)

'定义 appletalk 协议的上层协议
Global Const ATPROTO_ADSP=(DDPPROTO_MAX + 1)
Global Const ATPROTO_ATP=(DDPPROTO_MAX + 2)
Global Const ATPROTO_ASP=(DDPPROTO_MAX + 3)
Global Const ATPROTO_PAP=(DDPPROTO_MAX + 4)

Global Const ATMPROTO_AALUSER=0           '用户自定义 AAL
Global Const ATMPROTO_AAL1=1              'AAL 1
Global Const ATMPROTO_AAL2=2              'AAL 2
Global Const ATMPROTO_AAL34=3             'AAL 3/4
Global Const ATMPROTO_AAL5=5              'AAL 5

'协议
Global Const IPPROTO_IP=0                         'IP 协议
```

```
Global Const IPPROTO_ICMP=1                    '控制信息协议
Global Const IPPROTO_IGMP=2                    'Internet 组管理协议
Global Const IPPROTO_GGP=3                     '网关
Global Const IPPROTO_TCP=6                     'tcp
Global Const IPPROTO_PUP=12                    'pup
Global Const IPPROTO_UDP=17                    '用户数据报协议
Global Const IPPROTO_IDP=22                    'xns idp
Global Const IPPROTO_ND=77                     '非正是网络盘协议

Global Const IPPROTO_RAW=255                   '新地 IP 包
Global Const IPPROTO_MAX=256

'端口数值：网络标准函数
Global Const IPPORT_ECHO=7
Global Const IPPORT_DISCARD=9
Global Const IPPORT_SYSTAT=11
Global Const IPPORT_DAYTIME=13
Global Const IPPORT_NETSTAT=15
Global Const IPPORT_FTP=21
Global Const IPPORT_TELNET=23
Global Const IPPORT_SMTP=25
Global Const IPPORT_TIMESERVER=37
Global Const IPPORT_NAMESERVER=42
Global Const IPPORT_WHOIS=43
Global Const IPPORT_MTP=57

'端口数量：主机确定函数
Global Const IPPORT_TFTP=69
Global Const IPPORT_RJE=77
Global Const IPPORT_FINGER=79
Global Const IPPORT_TTYLINK=87
Global Const IPPORT_SUPDUP=95

'UNIXTCPsockets
Global Const IPPORT_EXECSERVER=512
Global Const IPPORT_LOGINSERVER=513
Global Const IPPORT_CMDSERVER=514
Global Const IPPORT_EFSSERVER=520
```

```
' UNIXUDPsockets
Global Const IPPORT_BIFFUDP=512
Global Const IPPORT_WHOSERVER=513
Global Const IPPORT_ROUTESERVER=520
                                '     注意端口 520+1 也被使用
' 保留端口号 1024
Global Const IPPORT_RESERVED=1024

'标志位定义
Global Const PFL_MULTIPLE_PROTO_ENTRIES=&H1
Global Const PFL_RECOMMENDED_PROTO_ENTRY=&H2
Global Const PFL_HIDDEN=&H4
Global Const PFL_MATCHES_PROTOCOL_ZERO=&H8

Global Const WSABASEERR=10000

'Winsock 错误常量
Global Const WSAEINTR=(WSABASEERR + 4)
Global Const WSAEBADF=(WSABASEERR + 9)
Global Const WSAEACCES=(WSABASEERR + 13)
Global Const WSAEFAULT=(WSABASEERR + 14)
Global Const WSAEINVAL=(WSABASEERR + 22)
Global Const WSAEMFILE=(WSABASEERR + 24)

' Winsock 标准伯克力错误常量
Global Const WSAEWOULDBLOCK=(WSABASEERR + 35)
Global Const WSAEINPROGRESS=(WSABASEERR + 36)
Global Const WSAEALREADY=(WSABASEERR + 37)
Global Const WSAENOTSOCK=(WSABASEERR + 38)
Global Const WSAEDESTADDRREQ=(WSABASEERR + 39)
Global Const WSAEMSGSIZE=(WSABASEERR + 40)
Global Const WSAEPROTOTYPE=(WSABASEERR + 41)
Global Const WSAENOPROTOOPT=(WSABASEERR + 42)
Global Const WSAEPROTONOSUPPORT=(WSABASEERR + 43)
Global Const WSAESOCKTNOSUPPORT=(WSABASEERR + 44)
Global Const WSAEOPNOTSUPP=(WSABASEERR + 45)
Global Const WSAEPFNOSUPPORT=(WSABASEERR + 46)
Global Const WSAEAFNOSUPPORT=(WSABASEERR + 47)
Global Const WSAEADDRINUSE=(WSABASEERR + 48)
```

```
Global Const WSAEADDRNOTAVAIL=(WSABASEERR + 49)
Global Const WSAENETDOWN=(WSABASEERR + 50)
Global Const WSAENETUNREACH=(WSABASEERR + 51)
Global Const WSAENETRESET=(WSABASEERR + 52)
Global Const WSAECONNABORTED=(WSABASEERR + 53)
Global Const WSAECONNRESET=(WSABASEERR + 54)
Global Const WSAENOBUFS=(WSABASEERR + 55)
Global Const WSAEISCONN=(WSABASEERR + 56)
Global Const WSAENOTCONN=(WSABASEERR + 57)
Global Const WSAESHUTDOWN=(WSABASEERR + 58)
Global Const WSAETOOMANYREFS=(WSABASEERR + 59)
Global Const WSAETIMEDOUT=(WSABASEERR + 60)
Global Const WSAECONNREFUSED=(WSABASEERR + 61)
Global Const WSAELOOP=(WSABASEERR + 62)
Global Const WSAENAMETOOLONG=(WSABASEERR + 63)
Global Const WSAEHOSTDOWN=(WSABASEERR + 64)
Global Const WSAEHOSTUNREACH=(WSABASEERR + 65)
Global Const WSAENOTEMPTY=(WSABASEERR + 66)
Global Const WSAEPROCLIM=(WSABASEERR + 67)
Global Const WSAEUSERS=(WSABASEERR + 68)
Global Const WSAEDQUOT=(WSABASEERR + 69)
Global Const WSAESTALE=(WSABASEERR + 70)
Global Const WSAEREMOTE=(WSABASEERR + 71)

'扩展 Winsock 错误常量定义
Global Const WSASYSNOTREADY=(WSABASEERR + 91)
Global Const WSAVERNOTSUPPORTED=(WSABASEERR + 92)
Global Const WSANOTINITIALISED=(WSABASEERR + 93)
Global Const WSAEDISCON=(WSABASEERR + 101)
Global Const WSAENOMORE=(WSABASEERR + 102)
Global Const WSAECANCELLED=(WSABASEERR + 103)
Global Const WSAEINVALIDPROCTABLE=(WSABASEERR + 104)
Global Const WSAEINVALIDPROVIDER=(WSABASEERR + 105)
Global Const WSAEPROVIDERFAILEDINIT=(WSABASEERR + 106)
Global Const WSASYSCALLFAILURE=(WSABASEERR + 107)
Global Const WSASERVICE_NOT_FOUND=(WSABASEERR + 108)
Global Const WSATYPE_NOT_FOUND=(WSABASEERR + 109)
Global Const WSA_E_NO_MORE=(WSABASEERR + 110)
Global Const WSA_E_CANCELLED=(WSABASEERR + 111)
```

```
Global Const WSAEREFUSED=(WSABASEERR + 112)

'主机没有找到
Global Const WSAHOST_NOT_FOUND=(WSABASEERR + 1001)

'主机没有找
Global Const WSATRY_AGAIN=(WSABASEERR + 1002)

'FORMERR, REFUSED, NOTIMP
Global Const WSANO_RECOVERY=(WSABASEERR + 1003)

'合法名字
Global Const WSANO_DATA=(WSABASEERR + 1004)

' 定义 QOS 相关返回码
Global Const WSA_QOS_RECEIVERS=(WSABASEERR + 1005)
        '至少一个 reserve 到达
Global Const WSA_QOS_SENDERS=(WSABASEERR + 1006)
        '至少一个 path 到达
Global Const WSA_QOS_NO_SENDERS=(WSABASEERR + 1007)
        '没有发送者
Global Const WSA_QOS_NO_RECEIVERS=(WSABASEERR + 1008)
        '没有接收者
Global Const WSA_QOS_REQUEST_CONFIRMED=(WSABASEERR + 1009)
        'Reserve 被确认
Global Const WSA_QOS_ADMISSION_FAILURE=(WSABASEERR + 1010)
        '缺少资源
Global Const WSA_QOS_POLICY_FAILURE=(WSABASEERR + 1011)
        '被拒绝
Global Const WSA_QOS_BAD_STYLE=(WSABASEERR + 1012)
        '不确定或者冲突的错误
Global Const WSA_QOS_BAD_OBJECT=(WSABASEERR + 1013)
        'filterspec 和 providerspecific 的部分错误
Global Const WSA_QOS_TRAFFIC_CTRL_ERROR=(WSABASEERR + 1014)
        'flowspec 错误
Global Const WSA_QOS_GENERIC_ERROR=(WSABASEERR + 1015)
        '一般错误

'连接类型
```

```
Global Const SOCK_STREAM=1              '流式 socket
Global Const SOCK_DGRAM=2               '数据报式 socket
Global Const SOCK_RAW=3               'raw-protocol 界面
Global Const SOCK_RDM=4               '可靠发送消息
Global Const SOCK_SEQPACKET=5               'sequenced packet stream

Global Const XP1_CONNECTIONLESS=&H1
Global Const XP1_GUARANTEED_DELIVERY=&H2
Global Const XP1_GUARANTEED_ORDER=&H4
Global Const XP1_MESSAGE_ORIENTED=&H8
Global Const XP1_PSEUDO_STREAM=&H10
Global Const XP1_GRACEFUL_CLOSE=&H20
Global Const XP1_EXPEDITED_DATA=&H40
Global Const XP1_CONNECT_DATA=&H80
Global Const XP1_DISCONNECT_DATA=&H100
Global Const XP1_SUPPORT_BROADCAST=&H200
Global Const XP1_SUPPORT_MULTIPOINT=&H400
Global Const XP1_MULTIPOINT_CONTROL_PLANE=&H800
Global Const XP1_MULTIPOINT_DATA_PLANE=&H1000
Global Const XP1_QOS_SUPPORTED=&H2000
Global Const XP1_INTERRUPT=&H4000
Global Const XP1_UNI_SEND=&H8000
Global Const XP1_UNI_RECV=&H10000
Global Const XP1_IFS_HANDLES=&H20000
Global Const XP1_PARTIAL_MESSAGE=&H40000

Global Const BIGENDIAN=&H0
Global Const LITTLEENDIAN=&H1

Global Const SECURITY_PROTOCOL_NONE=&H0

'WinSock 2 扩展 --验证函数 WSAJoinLeaf()常量
Global Const JL_SENDER_ONLY=&H1
Global Const JL_RECEIVER_ONLY=&H2
Global Const JL_BOTH=&H4

'Winsock 2 扩展 -- 验证 WSASocket()常量
Global Const WSA_FLAG_OVERLAPPED=&H1
Global Const WSA_FLAG_MULTIPOINT_C_ROOT=&H2
```

```
Global Const WSA_FLAG_MULTIPOINT_C_LEAF=&H4
Global Const WSA_FLAG_MULTIPOINT_D_ROOT=&H8
Global Const WSA_FLAG_MULTIPOINT_D_LEAF=&H10

'Winsock2 扩展 -- 验证WSAIoctl()常量
Global Const IOC_UNIX=&H0
Global Const IOC_WS2=&H8000000
Global Const IOC_PROTOCOL=&H10000000
Global Const IOC_VENDOR=&H18000000

Global Const IOCPARM_MASK=&H7F            '参数必须少于128字节
Global Const IOC_VOID=&H20000000          '没有参数
Global Const IOC_OUT=&H40000000           '复制外部参数
Global Const IOC_IN=&H80000000            '复制内部参数
Global Const IOC_INOUT=IOC_IN Or IOC_OUT

Global Const SIO_ASSOCIATE_HANDLE=IOC_IN Or IOC_WS2 Or 1
Global Const SIO_ENABLE_CIRCULAR_QUEUEING=IOC_VOID Or IOC_WS2 Or 2
Global Const SIO_FIND_ROUTE=IOC_OUT Or IOC_WS2 Or 3
Global Const SIO_FLUSH=IOC_VOID Or IOC_WS2 Or 4
Global Const SIO_GET_BROADCAST_ADDRESS=IOC_OUT Or IOC_WS2 Or 5
Global Const SIO_GET_EXTENSION_FUNCTION_POINTER=IOC_INOUT Or IOC_WS2 Or 6
Global Const SIO_GET_QOS=IOC_INOUT Or IOC_WS2 Or 7
Global Const SIO_GET_GROUP_QOS=IOC_INOUT Or IOC_WS2 Or 8
Global Const SIO_MULTIPOINT_LOOPBACK=IOC_IN Or IOC_WS2 Or 9
Global Const SIO_MULTICAST_SCOPE=IOC_IN Or IOC_WS2 Or 10
Global Const SIO_SET_QOS=IOC_IN Or IOC_WS2 Or 11
Global Const SIO_SET_GROUP_QOS=IOC_IN Or IOC_WS2 Or 12
Global Const SIO_TRANSLATE_HANDLE=IOC_INOUT Or IOC_WS2 Or 13
Global Const SIO_ROUTING_INTERFACE_QUERY=IOC_INOUT Or IOC_WS2 Or 20
Global Const SIO_ROUTING_INTERFACE_CHANGE=IOC_IN Or IOC_WS2 Or 21
Global Const SIO_ADDRESS_LIST_QUERY=IOC_OUT Or IOC_WS2 Or 22
Global Const SIO_ADDRESS_LIST_CHANGE=IOC_VOID Or IOC_WS2 Or 23
Global Const SIO_QUERY_TARGET_PNP_HANDLE=IOC_OUT Or IOC_WS2 Or 24

Type tcp_keepalive
    onoff As Long
    keepalivetime As Long
    keepaliveinterval As Long
```

```
End Type

Global Const SIO_RCVALL=IOC_IN Or IOC_VENDOR Or 1
Global Const SIO_RCVALL_MCAST=IOC_IN Or IOC_VENDOR Or 2
Global Const SIO_RCVALL_IGMPMCAST=IOC_IN Or IOC_VENDOR Or 3
Global Const SIO_KEEPALIVE_VALS=IOC_IN Or IOC_VENDOR Or 4
Global Const SIO_ABSORB_RTRALERT=IOC_IN Or IOC_VENDOR Or 5
Global Const SIO_UCAST_IF=IOC_IN Or IOC_VENDOR Or 6
Global Const SIO_LIMIT_BROADCASTS=IOC_IN Or IOC_VENDOR Or 7
Global Const SIO_INDEX_BIND=IOC_IN Or IOC_VENDOR Or 8
Global Const SIO_INDEX_MCASTIF=IOC_IN Or IOC_VENDOR Or 9
Global Const SIO_INDEX_ADD_MCAST=IOC_IN Or IOC_VENDOR Or 10
Global Const SIO_INDEX_DEL_MCAST=IOC_IN Or IOC_VENDOR Or 11
```

2. **结构体变量声明**

```
'Socket 地址信息
Type SOCKET_ADDRESS
   lpSockaddr As Long
   iSockaddrLength As Long
End Type
'
'CSAddr 地址信息
Type CSADDR_INFO
   LocalAddr As Long
   RemoteAddr As Long
   iSocketType As Long
   iProtocol As Long
End Type
'
'通过 SIO_ADDRESS_LIST_QUERY 返回的地址列表
Type SOCKET_ADDRESS_LIST
   iAddressCount As Long
   Address(20) As SOCKET_ADDRESS 'here we hack a little bit
End Type
'
' 地址族
Type AFPROTOCOLS
   iAddressFamily As Long
```

```
    iProtocol As Long
End Type

'地址常量
Global Const INADDR_NONE=&HFFFF
Global Const INADDR_ANY=&H0

Global Const SOL_SOCKET=&HFFFF&
Global Const INVALID_SOCKET=-1
Global Const SOCKET_ERROR=-1

Global Const MAXGETHOSTSTRUCT=1024
Type GUID
    Data1 As Long
    Data2 As Integer
    Data3 As Integer
    Data4(7) As Byte
End Type

Global Const MAX_PROTOCOL_CHAIN=7

Global Const BASE_PROTOCOL=1
Global Const LAYERED_PROTOCOL=0

Type WSAPROTOCOLCHAIN
     ChainLen As Long                              'chain 的长度
                                                  '长度=0 表示层协议
                                                  '长度=1 表示基协议
                                                  '长度> 1 表示协议链
    ChainEntries(MAX_PROTOCOL_CHAIN - 1) As Long   'dwCatalogEntryIds 列表
End Type

Global Const WSAPROTOCOL_LEN=255

Type WSAPROTOCOL_INFO
    dwServiceFlags1 As Long
    dwServiceFlags2 As Long
    dwServiceFlags3 As Long
    dwServiceFlags4 As Long
```

```
    dwProviderFlags As Long
    ProviderId As GUID
    dwCatalogEntryId As Long
    ProtocolChain As WSAPROTOCOLCHAIN
    iVersion As Long
    iAddressFamily As Long
    iMaxSockAddr As Long
    iMinSockAddr As Long
    iSocketType As Long
    iProtocol As Long
    iProtocolMaxOffset As Long
    iNetworkByteOrder As Long
    iSecurityScheme As Long
    dwMessageSize As Long
    dwProviderReserved As Long
    szProtocol(WSAPROTOCOL_LEN) As Byte
End Type

Global Const NS_ALL=0

Global Const NS_SAP=1
Global Const NS_NDS=2
Global Const NS_PEER_BROWSE=3

Global Const NS_TCPIP_LOCAL=10
Global Const NS_TCPIP_HOSTS=11
Global Const NS_DNS=12
Global Const NS_NETBT=13
Global Const NS_WINS=14

Global Const NS_NBP=20

Global Const NS_MS=30
Global Const NS_STDA=31
Global Const NS_NTDS=32

Global Const NS_X500=40
Global Const NS_NIS=41
Global Const NS_NISPLUS=42
```

```
Global Const NS_WRQ=50

Type WSANAMESPACE_INFO
    NSProviderId As GUID
    dwNameSpace As Long
    fActive As Long
    dwVersion As Long
    lpszIdentifier As Long
End Type

Type sockaddr
    sin_family As Integer
    sin_port As Integer
    sin_addr As Long
    sin_zero As String * 8
End Type
Global Const sockaddr_size=16

Type HostEnt
    h_name As Long
    h_aliases As Long
    h_addrtype As Integer
    h_length As Integer
    h_addr_list As Long
End Type
Global Const hostent_size=16

Global Const WSA_DESCRIPTIONLEN=256
Global Const WSA_DescriptionSize=WSA_DESCRIPTIONLEN + 1
Global Const WSA_SYS_STATUS_LEN=128
Global Const WSA_SysStatusSize=WSA_SYS_STATUS_LEN + 1

Type WSADataType
    wVersion As Integer
    wHighVersion As Integer
    szDescription As String * WSA_DescriptionSize
    szSystemStatus As String * WSA_SysStatusSize
    iMaxSockets As Integer
```

```
    iMaxUdpDg As Integer
    lpVendorInfo As Long
End Type

Type LingerType
    l_onoff As Integer
    l_linger As Integer
End Type


Global Const FD_READ_BIT=0
Global Const FD_READ=1        '(1 << FD_READ_BIT)

Global Const FD_WRITE_BIT=1
Global Const FD_WRITE=&H2       '(1 << FD_WRITE_BIT)

Global Const FD_OOB_BIT=2
Global Const FD_OOB=&H4         '(1 << FD_OOB_BIT)

Global Const FD_ACCEPT_BIT=3
Global Const FD_ACCEPT=&H8      '(1 << FD_ACCEPT_BIT)

Global Const FD_CONNECT_BIT=4
Global Const FD_CONNECT=&H10     '(1 << FD_CONNECT_BIT)

Global Const FD_CLOSE_BIT=5
Global Const FD_CLOSE=&H20      '(1 << FD_CLOSE_BIT)

Global Const FD_QOS_BIT=6
Global Const FD_QOS=&H40         '(1 << FD_QOS_BIT)

Global Const FD_GROUP_QOS_BIT=7
Global Const FD_GROUP_QOS=&H80   '(1 << FD_GROUP_QOS_BIT)

Global Const FD_ROUTING_INTERFACE_CHANGE_BIT=8
Global   Const   FD_ROUTING_INTERFACE_CHANGE=&H100              '(1   <<
FD_ROUTING_INTERFACE_CHANGE_BIT)

Global Const FD_ADDRESS_LIST_CHANGE_BIT=9
```

```
    Global     Const     FD_ADDRESS_LIST_CHANGE=&H200              '(1     <<
FD_ADDRESS_LIST_CHANGE_BIT)

    Global Const FD_MAX_EVENTS=10
    Global Const FD_ALL_EVENTS=&H3FF     '(1 << FD_MAX_EVENTS) - 1)

    Type WSANETWORKEVENTS
       lNetWorkEvents As Long
       iErrorCode(FD_MAX_EVENTS - 1) As Long
    End Type

    Global Const WSA_WAIT_FAILED=&HFFFFFFFF
    Global Const WSA_WAIT_EVENT_0=0
    Global Const WSA_WAIT_TIMEOUT=&H102

    Public Const GMEM_FIXED=&H0
```

3. API 函数声明

```
    Public Declare Function GlobalAlloc Lib "Kernel32" (ByVal wFlags As Long, ByVal
dwBytes As Long) As Long
    Public Declare Function GlobalFree Lib "Kernel32" (ByVal hMem As Long) As Long
    Public Declare Sub CopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (dest As
Any, src As Any, ByVal cb As Long)
    Public Declare Sub CopyMemory2 Lib "Kernel32" Alias "RtlMoveMemory" (dest As
Any, ByVal src As Long, ByVal cb As Long)
    Public Declare Sub CopyMemory3 Lib "Kernel32" Alias "RtlMoveMemory" (dest As
Long, ByVal src As Long, ByVal cb As Long)
    Public Declare Sub ZeroMemory Lib "Kernel32" Alias "RtlZeroMemory" (dest As
Any, ByVal numBytes As Long)
    Public Declare Function StringFromGUID2 Lib "ole32.dll" (pGUID As GUID, ByVal
_
    PointerToString As String, ByVal MaxLength As Long) As Long
    Public Declare Function lstrcpy Lib "Kernel32" Alias "lstrcpyA" (ByVal lpString1
As String, lpString2 As Byte) As Long
    Public Declare Function lstrlen Lib "Kernel32" Alias "lstrlenA" (ByVal lpString
As String) As Long
    Public Declare Function lstrcpy1 Lib "Kernel32" Alias "lstrcpyA" (ByVal
lpString1 As String, ByVal lpString2 As Long) As Long
    Public Declare Function GetCurrentProcessId Lib "Kernel32" () As Long
```

```
    Public Declare Function GetTickCount Lib "Kernel32" () As Long
    Public Declare Function Sleep Lib "Kernel32" (ByVal dwMilliseconds As Long)
As Long

    Declare Function getsockname Lib "ws2_32.DLL" (ByVal s As Long, sname As sockaddr,
namelen As Long) As Long
    Declare Function WSAStartup Lib "ws2_32.DLL" (ByVal wVR As Long, lpWSAD As
WSADataType) As Long
    Declare Function WSACleanup Lib "ws2_32.DLL" () As Long
    Declare Function bind Lib "ws2_32.DLL" (ByVal s As Long, addr As sockaddr,
ByVal namelen As Long) As Long
    Declare FunctionAcceptLib "ws2_32.DLL" (ByVal s As Long, addr As sockaddr,
namelen As Long) As Long
    Declare Function socket Lib "ws2_32.DLL" (ByVal af As Long, ByVal s_type As
Long, ByVal protocol As Long) As Long
    Declare Function WSASocket Lib "ws2_32.DLL" Alias "WSASocketA" (ByVal af As
Long, ByVal s_type As Long, ByVal protocol As Long, lpProtocolInfo As Any, ByVal
g As Long, ByVal dwFlags As Long) As Long
    Declare Function closesocket Lib "ws2_32.DLL" (ByVal s As Long) As Long
    Declare Function connect Lib "ws2_32.DLL" (ByVal s As Long, addr As sockaddr,
ByVal namelen As Long) As Long
    Declare Function gethostbyname Lib "ws2_32.DLL" (ByVal host_name As String)
As Long
    Declare Function gethostbyaddr Lib "ws2_32.DLL" (addr As Any, ByVal nlen As
Long, ByVal ntype As Long) As Long
    Declare Function gethostname Lib "ws2_32.DLL" (ByVal host_name As String, ByVal
namelen As Long) As Long
    Declare Function recv Lib "ws2_32.DLL" (ByVal s As Long, Buf As Any, ByVal
buflen As Long, ByVal flags As Long) As Long
    Declare Function recvfrom Lib "ws2_32.DLL" (ByVal s As Long, Buf As Any, ByVal
buflen As Long, ByVal flags As Long, from As sockaddr, fromlen As Long) As Long
    Declare Function send Lib "ws2_32.DLL" (ByVal s As Long, Buf As Any, ByVal
buflen As Long, ByVal flags As Long) As Long
    Declare Function sendto Lib "ws2_32.DLL" (ByVal s As Long, Buf As Any, ByVal
buflen As Long, ByVal flags As Long, to_addr As Any, ByVal tolen As Long) As Long
    Declare Function htonl Lib "ws2_32.DLL" (ByVal hostlong As Long) As Long
    Declare Function htons Lib "ws2_32.DLL" (ByVal hostshort As Long) As Integer
    Declare Function ntohs Lib "ws2_32.DLL" (ByVal netshort As Long) As Integer
    Declare Function ntohl Lib "ws2_32.DLL" (ByVal netlong As Long) As Long
```

```
    Declare Function inet_addr Lib "ws2_32.DLL" (ByVal cp As String) As Long
    Declare Function inet_ntoa Lib "ws2_32.DLL" (ByVal in_n As Long) As Long
    Declare Function setsockopt Lib "ws2_32.DLL" (ByVal s As Long, ByVal level
As Long, ByVal optname As Long, optval As Long, ByVal optlen As Long) As Long
    Declare Function setsockopt2 Lib "ws2_32.DLL" Alias "setsockopt" (ByVal s As
Long, ByVal level As Long, ByVal optname As Long, optval As Any, ByVal optlen As
Long) As Long
    Declare Function getsockopt Lib "ws2_32.DLL" (ByVal s As Long, ByVal level
As Long, ByVal optname As Long, optval As Long, optlen As Long) As Long
    Declare Function getsockopt2 Lib "ws2_32.DLL" Alias "getsockopt" (ByVal s As
Long, ByVal level As Long, ByVal optname As Long, optval As Any, optlen As Long)
As Long
    Declare Function listen Lib "ws2_32.DLL" (ByVal s As Long, ByVal backlog As
Long) As Long
    Declare Function WSAIoctl Lib "ws2_32.DLL" (ByVal s As Long, ByVal
dwIoControlCode As Long, lpvInBuffer As Any, ByVal cbInBuffer As Long, lpvOutBuffer
As Any, ByVal cbOutBuffer As Long, lpcbBytesReturned As Long, lpOverlapped As Any,
lpCompletionRoutine As Any) As Long
    Declare Function WSAEnumProtocols Lib "ws2_32.DLL" Alias "WSAEnumProtocolsA"
(ByVal lpiProtocols As Long, ByVal lpProtocolBuffer As Long, lpdwBufferLength As
Long) As Long
    Declare Function WSACreateEvent Lib "ws2_32.DLL" () As Long
    Declare Function WSACloseEvent Lib "ws2_32.DLL" (ByVal hEvent As Long) As Boolean
    Declare Function WSAEventSelect Lib "ws2_32.DLL" (ByVal s As Long, ByVal
hEventOjbect As Long, ByVal lNetWorkEvents As Long) As Long
    Declare Function WSAEnumNetworkEvents Lib "ws2_32.DLL" (ByVal s As Long, ByVal
hEventOjbect As Long, lpNetWorkEvents As WSANETWORKEVENTS) As Long
    Declare Function WSAWaitForMultipleEvents Lib "ws2_32.DLL" (ByVal cEvents As
Long,     lphEvents As Long, ByVal fWaitAll As Boolean,     ByVal dwTimeOUT As Long,
ByVal fAlertable As Boolean) As Long
    Declare Function WSAResetEvent Lib "ws2_32.DLL" (ByVal hEvent As Long) As Boolean

    Declare Function WSAEnumNameSpaceProviders Lib "ws2_32.DLL" Alias
"WSAEnumNameSpaceProvidersA" (lpdwBufferLength As Long, ByVal lpnspBuffer As Long)
As Boolean

    Public Const ICMP_ECHO=8
    Public Const ICMP_ECHOREPLY=0
    Public Const ICMP_MIN=8  '最小 8 字节的 ICMP 包
```

```
Public Const IP_RECORD_ROUTE=7

' IP 头
Type IpHeader
   h_len As Byte
   tos As Byte
   total_len As Integer
   ident As Integer
   frag_and_flags As Integer
   ttl As Byte
   proto As Byte
   CheckSum As Integer
   sourceIP As Long
   destIP As Long
End Type

' ICMP header
Type IcmpHeader
  i_type As Byte
  i_code As Byte
  i_cksum As Integer
  i_id As Integer
  i_seq As Integer
  timestamp As Long   '不是标准头部，但是一般保留空间
End Type

' IP 选项头
Type IpOptionHeader
    code As Byte       '选项类型
    len As Byte        '长度
    ptr As Byte        '间距
    addr(8) As Long    'IP 地址列表
End Type

Function GetHostByNameAlias(ByVal hostname As String) As Long
    On Error Resume Next
    '返回 IP 地址

    Dim phe As Long    ' 主机信息入口指针
```

```
    Dim heDestHost As HostEnt ’hostent 结构
    Dim addrList As Long
    Dim retIP As Long
    ’首先检查是否是一个合法的 IP
    retIP=inet_addr(hostname)
    If retIP=INADDR_NONE Then
        ’不是 IP，进行 DNS 转换
        phe=gethostbyname(hostname)
        If phe <> 0 Then
            ’指针非空，复制 hostent 结构
            CopyMemory heDestHost, ByVal phe, hostent_size
            ’获得地址列表中的第一个指针
            CopyMemory addrList, ByVal heDestHost.h_addr_list, 4
            CopyMemory retIP, ByVal addrList, heDestHost.h_length
        Else
            ’不是合法地址
            retIP=INADDR_NONE
        End If
    End If
    GetHostByNameAlias=retIP
    If Err Then GetHostByNameAlias=INADDR_NONE
End Function

Public Function TCPIPStartup() As Boolean
  Dim rc As Integer   ’返回码
  Dim wVersionRequested As Long   ’winsock 版本
  Dim WSAData As WSADataType         ’

  wVersionRequested=&H202
  TCPIPStartup=True
  rc=WSAStartup(wVersionRequested, WSAData)
  If rc <> 0 Then
    MsgBox("RC:"&rc&"Unable to start winsocks"&",Error"&Err.LastDllError)
    Call TCPIPShutDown
    TCPIPStartup=False
    Exit Function
  End If

End Function
```

```
    Public Function TCPIPShutDown() As Boolean
       WSACleanup
    End Function

    ------------------------------------------------    声    明    结    束
------------------------------------------------
```

2.5.2　WinsockAP 的函数使用

不论是流套接字还是数据报套接字编程，一般都采用客户机/ 服务器方式，它们的运作过程基本类似，下面着重介绍流套接字的编程模型。

1. 流套接字编程模型（TCP 模型）

流套接字的服务进程和客户进程在通信前必须创建各自的套接字并建立连接，然后才能对相应的套接字进行“读”、“写”操作，实现数据的传输。具体编程步骤如下：

① 服务器进程创建套接字。

服务进程总是先于客户进程启动，服务进程首先调用 socket 函数创建一个流套接字。Socket 函数的原型如下：

socket(ByVal af As Long, ByVal s_type As Long, ByVal protocol As Long) As Long;

其中，参数 af 用于指定网络地址类型，一般取 AF_INET，表示该套接字在 Internet 域中进行通信。参数 type 用于指定套接字类型，若取 SOCK_STREAM 表示要创建的套接字是流套接字，而取 SOCK_DGRAM 创建数据报套接字，这里取 SOCK_STREAM。参数 protocol 用于指定网络协议，一般取 0，表示默认为 TCP/IP 协议。若套接字创建成功则该函数返回所创建的套接字句柄 SOCKET，否则产生 INVALID_SOCKET 错误。

② 将本地地址绑定到所创建的套接字上以在网络上标识该套接字。这个过程是通过调用 bind 函数来完成的，该函数原型如下：

```
    bind((ByVal s As Long, addr As sockaddr, ByVal namelen As Long) As Long
```

其中，第一个参数 s 标识一未捆绑套接字的句柄，它用来等待客户机的连接。第二个参数 name 是赋予套接字的地址，它由 struct sockaddr 结构表示 ，该结构的格式如下：

```
    Type sockaddr
    sin_family As Integer
    sin_port As Integer
    sin_addr As Long
    sin_zero As String * 8
    End Type
```

其中，sin_family 字段必须设为 AF_INET，表示该 socket 处于 Internet 域。Sin_port 字段用于指定服务端口，在选择端口时必须特别小心，因为有些可用的端口号已为固定的服务保留（比如说文件传输协议和超文本传输协议，即 FTP 和 HTTP），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，可从 RFC 1700 中查阅。此外还应把端口号由主机字节顺序转换为网络字节顺序，如果端口号置为 0，则 Winsock 将为应用程序分配一个值

在 1024～5000 之间的唯一的端口。Sin_addr 字段用于把一个 IP 地址保存为一个 4 字节的数，它是无符号长整数类型；根据这个字段的不同用法，它还可表示一个本地或远程 IP 地址。最后一个字段 sin_ zero ，只充当填充项的职责，以使 sockaddr_in 结构和 sockaddr 结构的长度一样。一个有用的、名为 inet_addr 的函数，可把一个点式 IP 地址转换成一个 32 位的无符号长整数；这里取 IP 地址为 INADDR_ANY，以允许服务器应用监听主机计算机上面每个网络接口上的客户机活动。第三个字段 sin_zero 表示要传递的、由协议决定的地址的长度。

一旦出错，bind 函数就会返回 SOCKET_ERROR。对 bind 来说，最常见的错误是 WSAEADDRINUSE。如果使用的是 TCP/IP，那么该错误表示另一个进程已经同本地 IP 接口和端口号绑定到了一起，或者那个 IP 接口和端口号处于 TIME_WAIT 状态。假如对一个已经绑定的套接字调用 bind，便会返回 WSAEFFAULT 错误。

③ 将套接字置入监听模式并准备接受连接请求。Bind 函数的作用只是将一个套接字和一个指定的地址关联在一起，让一个套接字等候进入连接的 API 函数则是 listen，其原型为：

```
listen (ByVal s As Long, ByVal backlog As Long) As Long
```

其中，参数 s 标识一个已捆绑未连接套接字的描述字，当没有可用的描述字时，listen()函数仍试图正常地工作，它仍接受请求直至队列变空。当有可用描述字时，后续的一次 listen()或 accept()调用会将队列按照当前或最近的“后备日志”重新填充，如有可能的话，将恢复监听申请进入的连接请求。Backlog 参数用于指定正在等待连接的最大队列长度，这个参数非常重要，因为完全可能同时出现几个服务器连接请求。例如，假定 backlog 参数为 2，如果三个客户机同时发出请求，那么头两个会被放在一个等待处理队列中，以便应用程序依次为它们提供服务，而第三个连接（既队列以满）会造成一个 WSAECONNREFUSED 错误；一旦服务器接受了一个连接，那个连接请求就会从队列中删去，以便别人可继续发出请求，backlog 参数本身是由基层的协议提供者决定的，如果出现非法值，那么会用与之最接近的一个合法值来取代。

如无错误发生，listen 函数返回 0，若失败则返回 SOCKET_ERROR 错误，最常见的错误是 WSAEINVAL，该错误通常表示套接字在 listen 之前没有调用 bind。

进入监听状态之后，通过调用 accept 函数使套接字作好接受客户连接的准备。Accept 函数的原型为：

```
accept (ByVal s As Long, addr As sockaddr, namelen As Long) As Long
```

其中，参数 s 是处于监听模式的套接字描述字。第二个参数应该是一个有效的 SOCKADDR_IN 结构的地址，而 addrlen 是 SOCKADDR_IN 结构的长度。这样，服务器便可为等待连接队列中的第一个连接请求提供服务了。Accept 函数返回后，addr 参数变量中会包含发出连接请求的那个客户机的 IP 地址信息，而 addrlen 参数则指出该结构的长度，并返回一个新的套接字描述字，它对应于已经接受的那个客户机连接。对于该客户机后续的所有操作，都应使用这个新套接字，至于原来那个监听套接字，它仍然用于接受其他客户机连接，而且仍处于监听模式。如果无连接请求，服务进程被阻塞。

④ 客户进程调用 socket 函数创建客户端套接字。

⑤ 客户向服务进程发出连接请求。通过调用 connect 函数可以建立一个端的连接。Connet 函数原型为：

```
connect (ByVal s As Long, addr As sockaddr, ByVal namelen As Long) As Long
```

s 标识一个未连接的数据报或流类套接字的描述字。Name 是针对 TCP 的套接字地址结构，它标识服务进程 IP 地址信息，若 name 结构中的地址域为全零的话，则 connect 函数将返回 WSAEADDRNOTAVAIL 错误。Namelen 则用于标识 name 参数的长度。

如果欲连接的计算机没有侦听指定端口的这一进程，connect 调用就会失败，并发生错误 WSAECONNREFUSED。另一个常见的错误是 WSAETIMEDOUT，表示连接超时。

⑥　当连接请求到来后，被阻塞服务进程的 accpet()函数如③中所述生成一个新的套接字与客户套接字建立连接，并向客户返回接受信号。

⑦　一旦客户机的套接字收到来自服务器的接受信号，则表示客户机与服务器已实现连接，则可以进行数据传输了。Send、recv 函数是进行数据的收发的函数。

Send 函数的原型为：

```
    send (ByVal s As Long, Buf As Any, ByVal buflen As Long, ByVal flags As Long)
As Long
```

其中，SOCKET 参数是已建立连接的套接字描述字，表示发送数据操作将在这个套接字上进行。第二个参数 buf 是字符缓冲区，包含即将发送的数据。第三个参数 len 用于指定即将发送的缓冲区内的字符数。最后一个参数 flags 可取的值有 0、MSG_DONTROUTE 或 MSG_OOB 或这些标志的按位“或”运算，MSG_DONTROUTE 标志要求传送层不要将它发出的包路由出去，MSG_OOB 表志数据应该被带外发送。

Send 函数返回发送数据的字节数，若发生错误，就返回 SOCKET_ERROR。常见的错误是 WSAECONNABORTED，这一错误一般发生在虚拟回路由于超时或协议有错而中断的时候。发生这种情况时，应该关闭这个套接字。如果远程主机上的套接字被强行或意外关闭，则会发生 WSAECONNRESET 错误。最后一个常见错误是 WSAETIMEOUT，它发生在连接由于网络故障或远程连接系统异常死机而引起的连接中断时。但成功地完成 send 调用并不意味着数据传送到达，如果传送系统的缓冲区空间不够保存需传送的数据，除非套接字处于非阻塞 I/O 方式，否则 send 函数将阻塞。对于非阻塞 SOCK_STREAM 类型的套接字，实际写的数据数目可能在 1 到所需大小之间，其值取决于本地和远端主机的缓冲区大小。

Recv 函数的原型为：

```
    recv (ByVal s As Long, Buf As Any, ByVal buflen As Long, ByVal flags As Long)
As Long
```

其中，参数 s 是准备接收数据的套接字。第二个参数 buf 是即将收到数据的字符缓冲区，而 len 则是准备接收的字节数或 buf 缓冲的长度。Flags 参数可以是 0、MSG_PEEK 或 MSG_OOB 或这些标志的按位“或”运算，0 表示无特殊行为；MSG_PEEK 使有用的数据复制到所提供的接收端缓冲内，但是没有从系统缓冲区中将它删除。Recv 函数返回发送字节数。

⑧　关闭套接字。一旦任务完成，就必须关掉连接以释放套接字占用的所有资源。通常调用 closesocket 函数即可达到目的。

Closesocket()函数的原型为：

```
    closesocket (ByVal s As Long) As Long
```

其中，参数 s 是要关闭的套接字描述字，再利用套接字执行调用就会失败，并出现 WSAE-OTSOCK 错误。

为帮助读者进一步理解流套接字的编程模型，图 2-2 列出了流套接字编程的时序图。

2. **数据报套接字编程模型**

数据报套接字是无连接的，它的编程过程比流套接字要简单一些。

对于接收端（一般为服务端），先用 socket **函数建立套接字，再通过** bind **函数把这个套接字和准备接收数据的** IP **地址信息绑定在一起，这和前面流套接字一样，但不同的是它不必调用** listen **和** accept**，只需等待接收数据。并且由于它是无连接的，因此它可以接收网络上任何一台计算机所发的数据报。常用的接收数据函数是** recvfrom**，它的原型为：**

```
    recvfrom(ByVal s As Long, Buf As Any, ByVal buflen As Long, ByVal flags As
Long, from As sockaddr, fromlen As Long) As Long
```

其中，前面的 4 **个参数和** recv **函数一样，而参数** from **是一个** SOCKADDR **结构指针，**fromlen **参数是带有指向地址结构的长度的指针。当它返回数据时，**SOCKADDR **结构内便填入发送数据端的地址。**

对于发送端的数据报套接字来说，可以有两种方法发送数据。第一种，也是最简单的一种，便是建立一个套接字，然后调用 sendto **函数。**sendto **函数的原型为：**

```
    sendto(ByVal s As Long, Buf As Any, ByVal buflen As Long, ByVal flags As Long,
to_addr As Any, ByVal tolen As Long) As Long
```

除了 buf **是发送数据的缓冲，**len **指明发送的字节数外，其余参数和** recvfrom **的参数一样，**to **参数是一个指向** SOCKADDR **结构指针，带有接收数据的套接字目标地址信息。**

因为数据报套接字没有连接，所以也不会有正式的关闭和从容关闭之分，在接收端或发送端结束收发数据时，只是在套接字句柄上调用 closesocket()**函数来释放套接字资源。数据报套接字编程时序如图** 2-3 **所示。**

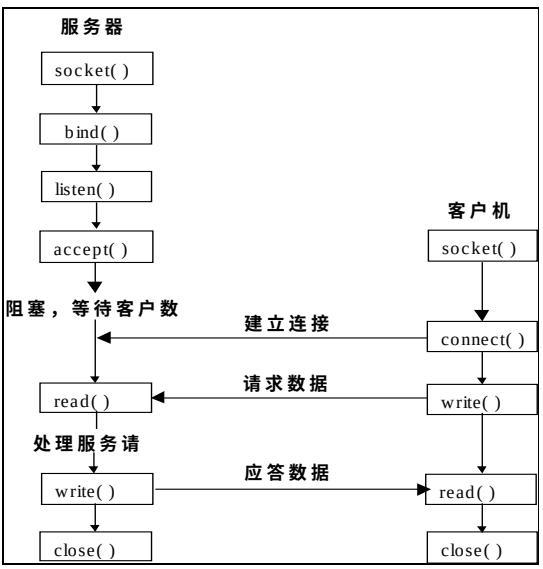


图 2-2　**流套接字编程时序图**

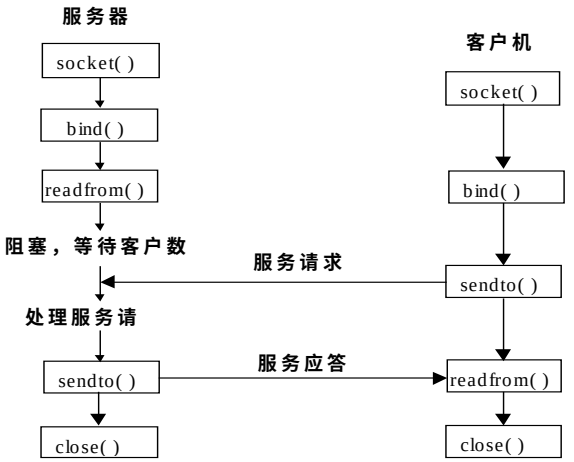


图 2-3　无连接套接字应用程序时序图

2.6　串口通信 API

以 Windows 9x/NT/2000 环境中的通信 API 函数为例，来介绍串口通信 API。Windows 9x/NT/2000 中的 API 一般支持 32 位的操作，因而又叫 Win32 API。

Win32 中的通信 API 不是很适合于局域网（LAN）通信。在线路上发送数据之前，LAN 通常将数据位串行化，就像串口或调制解调器所做的一样，但 LAN 使用的线路通常比串口要少，而且还使用与串口协议很少有类似之处的访问、路由、安全性和纠错协议。LAN 通信所需要的协议层，使 Win32 通信 API 对于这些应用来说很不理想。相比之下，TCP／IP 比 Win32 通信 API 更适合网络的通信和连接。

Windows 9x/NT/2000 是一个可抢占式的操作系统，因而 Windows 应用程序经常会有被别的程序抢占时间片的可能，所以 Win32 通信 API 达不到实时通信的要求。实时通信的质量与时间密切相关。例如，多媒体数据是实时数据，因为多媒体的质量依赖于采集和播放它的速率，在录制多媒体时，它就以某个速度被数字化了，该速度就是人们所熟知的采样速率。多媒体必须以相同的采样率进行重放，否则听起来就会太慢或者太快。实际中的视频点播，也不是实时数据，我们所看到的实际上是存放在缓冲中的那部分数据。

Windows 通信 API 用到了某些 RS-232 信号，如 RTS、RLSD、DTR 等，同时由于在 VB 中调用 API 的特殊性，在调用 Windows 通信 API 之前一定要进行声明 API 函数，常数和类型。

2.6.1　打开和关闭串口

1.　打开串口

在 Win32 中，串口和其他通信设备是作为文件处理的。串口的打开、关闭、读取和写入

所用的函数与操作文件的函数相同。

通信会话以调用 CreateFile 开始。CreateFile 为读访问权限、写访问权限或读写访问权限“打开”串口的。按照 Windows 的用法，CreateFile 返回一个句柄，随后在打开的端口的操作中使用。如下所示，CreateFile 函数很复杂，原因之一它是通用的内核函数。可以使用 CreateFile 打开已存在的文件，创建新文件或打开根本就不是文件的设备，例如串口、并口和调制解调器。

CreateFile 函数在 VB 中的声明如下：

```
Public Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" (ByVal
lpFileName As String, ByVal dwDesiredAccess As Long, ByVal dwShareMode As Long,
lpSecurityAttributes As SECURITY_ATTRIBUTES, ByVal dwCreationDisposition As Long,
ByVal dwFlagsAndAttributes As Long, ByVal hTemplateFile As Long) As Long
```

该函数表明函数位于系统内核动态库，其中函数的参数解释如下：

```
lpFileName：
```

该参数指定要打开的串口逻辑名，用字符串来表示，例如“COM1”或 “COM2”来分别表示串口一和串口二。

```
dwDesiredAccess：
```

用来指定端口访问的类型。与文件一样，串口也可以打开以供读取、写入或两者兼而有之。 标志 GENERIC_READ 为读取访问打开串口，标志 GENERIC_WRITE 为写访问打开串口。GENERIC_READ 和 GENERIC_WRITE 常数定义如下：

```
Public Const GENERIC_READ=&H80000000;
Public Const GENERIC_WRITE=&H40000000
```

可以用 OR 操作符将两个标志连接起来，就可以为读／写访问权限打开串口。大部分串口通信是双向的，因此通常在设置中两个标志均使用。

例如：

```
dwDesiredAccess=GENERIC_READ Or GENERIC_WRITE
dwShareMode：
```

该参数指定该端口的共享属性。该参数是为那些有许多应用程序共享的文件提供的。对于不能共享的串口，它必须置为 0。这就是文件与通信设备之间的主要差异之一。如果在当前的应用程序调用 CreateFile 时，另一个应用程序已经打开了串口，该函数就会返回错误代码，原因是两个应用程序不能共享一个端口。然而，同一个应用程序的多个线程可以共享由 CreateFile 返回的端口句柄，并且根据安全性属性设置，该句柄可以被打开端口的应用程序的子程序所继承。

第 4 个参数 lpSecurityAttributes 引用安全性属性 SECURITY_ATTRIBUTES 结构，该结构定义了一些属性，例如通信句柄如何被打开端口的应用程序的子程序所继承。将该参数设置为 NULL 将为该端口分配缺省的安全性属性。子应用程序所继承的缺省属性是该端口不能被继承。SECURITY_ATTRIBUTES 的声明如下：

```
Public Type SECURITY_ATTRIBUTES
       nLength As Long
       lpSecurityDescriptor As Long
```

```
        bInheritHandle As Long
End Type
```

SECURITY_ATTRIBUTES 结构成员 nLength 指明该结构的字节长度，lpSecurityDescriptor 是指向一个安全描述字符，bInheritHandle 是指句柄是否能被继承。

第 5 个参数 dwCreationDisposition 指定如果 CreateFile 正在被已有的文件调用时应做些什么。既然串口是实物，总是存在的，dwCreationDisposition 就必须设置成 OPEN_EXISTING。该标志告诉 Windows 不要企图创建新端口，而是打开已经存在的端口。OPEN_EXISTING 常数定义如下：

```
Public Const OPEN_EXISTING=3
```

第 6 个参数 dwFlagsAndAttributes 描述了该端口的各种属性。对于文件来说，可能具有很多属性，但对于串口硬件，唯一有意义的设置是 FILE_FLAG_OVERLAPPED。当创建时指定该设置时，端口 I/O 可以在后台进行（后台 I/O 也叫异步 I/O）。FILE_FLAG_OVERLAPPED 常数定义如下：

```
Public Const FILE_FLAG_OVERLAPPED=&H40000000
```

第 7 个参数 hTemplateFile 是指向模板文件的句柄。串口没有模板文件，因而当打开串口时，该参数必须置成 0。

调用 CreateFile 打开串口进行读写操作的例子如下：

```
handle=CreateFile(devname, GENERIC_READ Or GENERIC_WRITE, 0, 0, OPEN_EXISTING,
FILE_FLAG_OVERLAPPED, 0)
```

一旦端口处于打开状态，就可以分配一个发送缓冲区和一个接收缓冲区，可以通过调用 SetupComm 函数实现其他初始化工作，也可以不调用 SetupComm，Windows 也会分配缺省的发送和接收缓冲区，并且初始化端口。但为了保证缓冲区的大小与实际需要的一致，最好还是调用该函数。

SetupComm 的声明如下：

```
Public Declare Function SetupComm Lib "kernel32" Alias "SetupComm" (ByVal hFile
As Long, ByVal dwInQueue As Long, ByVal dwOutQueue As Long) As Long
```

其中参数：

hFile：

是由 CreateFile 函数返回的指向已打开串口的句柄。

dwInQueue 和 dwOutQueue：

分别定义了接收缓冲区的大小和发送缓冲区的大小。这两个定义并非是实际上的缓冲区的大小，这只是“推荐的”大小，Windows 设备驱动程序获得这两个数据，并不直接分配大小，而是用来优化性能和避免缓冲区超限，系统的其他部分已经提供了这项功能。

2. 关闭串口

关闭串口要比打开串口简单多了，只需要用 CloseHandle 关闭由 CreateFile 返回的句柄。

CloseHandle 的函数声明如下：

```
Public Declare Function CloseHandle Lib "kernel32" Alias "CloseHandle" (ByVal
hObject As Long) As Long
```

在使用串口后通常要关闭它，如果忘记关闭串口，它就会始终处于打开状态，其他的应用程序就不能够打开或使用它，只有注销当前用户或者重启才能再次使用该串口。

调用 CloseHandle 关闭串口进行读写操作的例子如下：

```
Call CloseHandle(handle)
```

2.6.2 串口配置和串口属性

在 Windows9X/NT/2000 中配置串口要比在 Windows 的早期版本中更为复杂，但复杂的程度越大，其功能也就越强大。

在 CreateFile 函数中打开串口时，系统将根据上次打开串口时设置的值来初始化串口。可以继承从上次打开操作后的数值，包括设备控制块（DCB）和超时控制结构（COMMTIMEOUTS）。如果是第一次打开串口，Windows 系统将会使用缺省的配置值。

1. 串口配置

Windows9X/NT/2000 使用 GetCommState 函数获取串口的当前配置，使用 SetCommState 函数重新分配串口资源的各个参数。

GetCommState 的函数声明如下：

```
Public Declare Function GetCommState Lib "kernel32" Alias "GetCommState" (ByVal nCid As Long, lpDCB As DCB) As Long
```

GetCommState 函数的第一个参数 nCid 是由 CreateFile 函数返回指向已打开串口的句柄。第二个参数是一个非常重要的结构-设备控制块 DCB。

DCB 结构的声明如下：

```
Public Type DCB
        DCBlength As Long
        BaudRate As Long
        fBitFields As Long ’See Comments in Win32API.Txt
        wReserved As Integer
        XonLim As Integer
        XoffLim As Integer
        ByteSize As Byte
        Parity As Byte
        StopBits As Byte
        XonChar As Byte
        XoffChar As Byte
        ErrorChar As Byte
        EofChar As Byte
        EvtChar As Byte
        wReserved1 As Integer ’Reserved; Do Not Use
End Type
```

其中成员：

DCBlength：1Byte 为单位指定的 DCB 的大小。

BaudRate：指定串口设备通信的数据传输率。它可以是实际的数据传输率数值，也可以是下列数据之一 ：

- CBR_110
- CBR_19200
- CBR_300
- CBR_38400
- CBR_600
- CBR_56000
- CBR_1200
- CBR_57600
- CBR_2400
- CBR_115200
- CBR_4800
- CBR_128000
- CBR_9600
- CBR_256000
- CBR_14400

用到以上数值时，也要在 VB 中用 API 的常数定义。

fbitFields：这 4 个字节指定了 14 个属性。每个属性可以通过对 fbitFields 的逐位逻辑与或逐位逻辑或得到，如表 2-31 所示。

表 2-31　　fbitFields 的成员列表

属性名称	所在位号	描述
Fbinary	1	二进制模式
Fparity	2	进行奇偶校验
fOutxCtsFlow	3	CTS 输出流量控制
fOutxDsrFlow	4	DSR 输出流量控制
fDtrControl	5	DTR 流量控制(2 位)
fDsrSensitivity	7	DSR 敏感度
fTXContinueOnXoff	8	XOFF 后发送是否停止
FoutX	9	XON/XOFF 输出控制有效
FinX	10	XON/XOFF 输入控制有效
fErrorChar	11	允许奇偶错误替换
Fnull	12	允许删除 NULL
fRtsControl	13	RTS 流量控制
fAbortOnError	15	出错时是否终止读写操作
fDummy2	16	系统保留

以下详细说明属性功能：

- fBinary：指定是否允许二进制模式。由于 Win32 API 不支持非二进制模式传递，因此该位必须设为 1。如果设为 0，则通信设备不工作。
- fParity：指明是否进行奇偶校验。如果该位设为 1，则进行奇偶校验，并报告出错信息。

- fOutxCtsFlow：指定 CTS 是否用于检测发送流控制。当该位为 1，而 CTS 为 OFF 时，发送将被挂起，直到 CTS 置 ON。
- fOutxDsrFlow：指定 DSR 是否用于检测发送流控制。当该位为 1 而 DSR 为 OFF 时，发送将被挂起，直到 DSR 置 ON。
- FdtrControl：指定 DTR 流量控制，它可以是表 2-32 中的各值之一。

表 2-32　　DTR 流量控制

值	描述
DTR_CONTROL_DISABLE	禁止 DTR 线，并保持禁止状态
DTR_CONTROL_ENABLE	允许 DTR 线，并保持允许状态
DTR_CONTROL_HANDSHAKE	允许 DTR 握手。如果允许握手，则不允许应用程序使用 EscapeCommFunction 函数调整线路

- fDsrSensitivity：指定通信驱动程序对 DTR 信号是否敏感。如果该位设为 1 时，DSR 信号为 OFF 时，接收的任何字节将被忽略。
- fTXContinueOnXoff：指定当接收缓冲区已满，并且驱动程序已经发送出 XoffChar 字符时发送是否停止。当该位置为 1 时，在接收缓冲区接收到了缓冲区已满的字节 XoffLim，并且驱动程序已经发送出 XoffChar 字符中止接收字节之后，发送继续进行。该成员置为 0 时，接收缓冲区接收到代表缓冲区已空的字节 XonChar，并且驱动程序已经发送出恢复发送的 XonChar 字符后，发送可以继续进行。
- fOutX：指明在数据发送中是否使用 XON/XOFF 信息流控制。该位为 1 时，接收到 XoffChar 之后便停止发送，接收到 XonChar 之后发送将重新开始。
- fInX：指明在数据接受中是否使用 XON/XOFF 信息流控制。当位为 1 时，接收缓冲区接收到代表缓冲区满的 XoffLim 之后，XoffChar 发送出去，接收缓冲区接收到代表缓冲区已空的字符 XonLim 之后，XonChar 发送出去。
- fErrorChar：指定是否使用 ErrorChar 成员指定的字符来代替奇偶校验错误的字符。当该位为 1，并且 fParity 为 1 时，就会用 ErrorChar 成员指定的字符来代替奇偶校验错误的字符。
- fNull：指明是否丢弃接收到的 NULL（ASCII 0）字符。如果置为 1，则丢弃接收到 NULL（ASCII 0）字符；反之则不丢弃。
- fRtsControl：指定 RTS 的流量控制，它的各值如表 2-33 所示。

表 2-33　　RTS 流量控制

值	描述
RTS_CONTROL_DISABLE	打开设备时禁止 RTS 线，并保持禁止状态
RTS_CONTROL_ENABLE	打开设备时允许 RTS 线，并保持允许状态
RTS_CONTROL_HANDSHAKE	允许握手。在接收缓冲区小于半满时将 RTS 置为 ON，在接收缓冲区超过四分之三满的时候将 RTS 置为 OFF。如果允许握手，则不允许应用程序使用 EscapeCommFunction 函数调整线路
RTS_CONTROL_TOGGLE	当发送的字节有效，则将 RTS 置为 ON。发送完缓冲区中的所有字节后，RTS 置为 OFF

如果 RTS 为 0 值，缺省值为 RTS_CONTROL_HANDSHAKE。

- fAbortOnError：如果发送错误，指定是否终止读、写操作。若该位为 1，当发生错

误时，则驱动程序以出错态终止所有的读写操作。只有当应用程序调用 ClearCommError 函数处理后，串口才能接受随后的通信操作。

- fDummy2：保留的，没有使用。

DCB 中的其他成员：

- wReserved：没有使用，必须为零。
- XonLim：指定在 XOFF 字符发送之前接收缓冲区中可允许的最小字节数。
- XoffLim:指定在 XOFF 字符发送之前接收缓冲区中可允许的最小可用字节数。
- ByteSize：指定端口当前使用的数据位数。
- Parity：指定端口当前使用的奇偶校验方法。它的可能值如表 2-34 所示。

表 2-34　　奇偶校验方法

值	方法
EVENPARITY	偶校验
MARKPARITY	标号校验
NOPARITY	无校验
ODDPARITY	奇校验
SPACEPARITY	空格校验

StopBits:指定串口当前使用的停止位数，可能值如表 2-35 所示。

表 2-35　　停止位描述

值	描述
ONESTOPBIT	1 位停止位
ONE5STOPBITS	1.5 位停止位
TWOSTOPBITS	2 位停止位

- XonChar：指明发送和接受的 XON 字符值，它表明允许继续传输。
- XoffChar：指明发送和接受的 XOFF 字符值，它表示暂停数据传输。
- ErrorChar: 本字符用来代替接收到的奇偶校验发生错误的字符。
- EofChar：用来指示数据的结束。
- EvtChar：事件字符。当接收到此字符时的时候，会产生一个事件。
- wReservedl：保留的，没有使用。

由上可知，DCB 结构成员指定了数据传输率、数据位数等丰富的信息。

如果 GetCommState 函数调用成功，则返回值不为 0；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

GetLastError 函数也是 Win32 API 函数，它的声明如下：

```
    Public Declare Function GetLastError Lib "kernel32" Alias "GetLastError" ()
As Long
```

使用 GetCommState 的例子如下：

```
    Private DCBStr As DCB
    res=GetCommState(hCommDev, DCBStr)  ' hCommDev 为打开的句柄。
    If res <> 0
      MsgBox "获得串口信息正确"
```

当应用程序只要修改一部分配置时，可以通过 GetCommState 函数获得当前的 DCB 结构；然后更改参数，调用 GetCommState 函数设置修改过的 DCB 来配置串口。

SetCommState 的函数声明如下：

```
Public Declare Function SetCommState Lib "kernel32" Alias "SetCommState" (ByVal
hCommDev As Long, lpDCB As DCB) As Long
```

如果函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

SetCommState 函数的第一参数 hCommDev 是由 CreateFile 函数返回的指向已打开串口的句柄。第二个参数也是指向 DCB 结构。

如果 SetCommState 函数调用的 DCB 结构中的 XonChar 成员等于 XoffChar 成员，SetCommState 函数会调用失败。

DCB 最常改变的设置是数据传输率、奇偶校验方法以及数据位和停止位数。Windows 为改变改变这些设置的提供了一个很方便的函数是 BuildCommDCB。

BuildCommDCB 函数的声明如下：

```
Public Declare Function BuildCommDCB Lib "kernel32" Alias "BuildCommDCBA"
(ByVal lpDef As String, lpDCB As DCB) As Long
```

BuildCommDCB 参数包含新设置的字符串和一个 DCB 结构的参数，该设置将提供给 DCB 结构。新设置的字符串与 MS-DOS 或 Windows NT/2000 中 Mode 命令格式相同，如下所示：

```
baud=1200 parity=N data=8 stop=1
```

该字符串将数据传输率改变为 1200 bit/s，关闭奇偶校验，数据位数设为 8，停止位数设为 1。与在 MS-DOS 和 Windows NT 中一样，该字符串不包括串口的名称，该函数实际上并不改变端口的设置，因此没有必要标识该串口。新的设置只是简单地拷贝到已提供好的 DCB 结构中。要使新设置处于有效，需使用 SetCommState 函数。

BuildCommDCB 支持老的和新的各种版本的 mode 命令。缺省的情况下，BuildCommDCB 函数禁止 XON/XOFF 和硬件流的控制。如果要使用硬件流控制，则必须设置 DCB 结构的各个成员的数值。

如果函数调用成功，则返回值为 1；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

Win32 还提供 BuildCommDCBAndTimeouts 函数来提供设置 DCB 的功能，该函数还可以设置超时结构。

BuildCommDCBAndTimeouts 函数的声明如下：

```
Public Declare Function BuildCommDCBAndTimeouts Lib "kernel32" Alias
BuildCommDCBAndTimeoutsA" (ByVal lpDef As String, lpDCB As DCB, lpCommTimeouts
As COMMTIMEOUTS) As Long
```

BuildCommDCBAndTimeouts 函数的设置 DCB 跟在 BuildCommDCB 中基本相同。该函数是根据在 lpDef 字符串中的“TO=xxx”子串的出现、空缺等设置来修改它的超时结构：

- 如果包含“TO=ON”，则函数根据超时结构 lpCommTimeouts 为串口设置超时值。

- 如果包含“ TO=OFF ”，则函数不为串口设置超时值。
- 如果不包含以上提到的“ TO=xxx ”的字串的任何一个，则函数忽略 lpCommTimeouts 的超时结构，该超时结构不能被获取。

如果 BuildCommDCBAndTimeouts 函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

2. 串口属性

在配置串口的属性时，应该先了解串口设备的属性，串口的性能可由 GetCommProperties 函数获取。

GetCommProperties 的声明如下：

```
Public Declare Function GetCommProperties Lib "kernel32" Alias "GetCommProperties" (ByVal hFile As Long, lpCommProp As COMMPROP) As Long
```

其中参数：

- hFile ：是 CreateFile 函数返回的句柄。
- lpCommProp：指向一个 COMMPROP 的结构，串口的性能就从 COMMPROP 中返回。

COMMPROP 结构不仅仅用于串口，它们也可以用来返回并口的有关信息。COMMPROP 的内容是静态的，不允许任何应用程序改变该结构中返回的信息。因此，没有任何相应的函数可将 COMMPROP 结构中的设置写向串口。

COMMPROP 的类型定义如下：

```
Public Type COMMPROP
        wPacketLength As Integer
        wPacketVersion As Integer
        dwServiceMask As Long
        dwReserved1 As Long
        dwMaxTxQueue As Long
        dwMaxRxQueue As Long
        dwMaxBaud As Long
        dwProvSubType As Long
        dwProvCapabilities As Long
        dwSettableParams As Long
        dwSettableBaud As Long
        wSettableData As Integer
        wSettableStopParity As Integer
        dwCurrentTxQueue As Long
        dwCurrentRxQueue As Long
        dwProvSpec1 As Long
        dwProvSpec2 As Long
        wcProvChar(1) As Integer
```

```
End Type
```

其中：

- wPacketLength：**不管申请的数据大小，指明整个数据包的大小。**
- wPacketVersion：**指定数据包结构的版本号。**
- dwServiceMask：**指定一位掩码。它指出由设备实现的服务，通常为通信设备设置** SP_SERIALCOMM **位，**MODEM **服务也可以指定为次位。**
- dwReserved1：**保留的，未用。**
- dwMaxTxQueue：**以字节为单位指定驱动程序内部发送缓冲区的最大长度。如果这个数为** 0**，则表示串口设备不能使用这个值。**
- dwMaxRxQueue ：**以字节为单位指定驱动程序内部接收缓冲区的最大长度。如果这个数为** 0**，则表示串口设备不能使用这个值。**
- dwMaxBaud：**以** bit/s **为单位指定可用的最大数据传输率。它可以是如表** 2-36 **所示的值之一。**

表 2-36 **数据传输率**

数据传输率	描述	数据传输率	描述
BAUD_075	75bit/s	BAUD_110	110bit/s
BAUD_134_5	134.5bit/s	BAUD_150	150bit/s
BAUD_300	300bit/s	BAUD_600	600bit/s
BAUD_1200	1200bit/s	BAUD_1800	1800bit/s
BAUD_2400	2400bit/s	BAUD_4800	4800bit/s
BAUD_7200	7200bit/s	BAUD_9600	9600bit/s
BAUD_14400	14400bit/s	BAUD_19200	19200bit/s
BAUD_38400	38400bit/s	BAUD_56k	56kbit/s
BAUD_57600	57600bit/s	BAUD_115200	115200bit/s
BAUD_128k	128kbit/s	BAUD_USER	**可用于编程的数据传输率**

- dwProvSubType：**指定特定通信设备的类型，如表** 2-37 **所示。**

表 2-37 **通信设备类型**

值	描述	数据传输率	描述
PST_FAX	**传真机设备**	PST_LAT	LAT **协议**
PST_MODEM	MODEM **设备**	PST_NETWORK_BRIDGE	**未指明的网桥**
PST_PARALLELPORT	**并行口**	PST_RS232	RS-232 **串口**
PST_RS422	RS-422 **口**	PST_RS423	RS-423 **口**
PST_RS449	RS-449 **口**	PST_SCANNER	**扫描仪设备**
PST_TCPIP_TELNET	TCP/IP Telnet **协议**	PST_UNSPECIFIED	**未指明的设备**
PST_TCPIP_TELNETPST_X25	X25 **标准**		

对于串口设备，该成员可以设为 PST_RS232 **或** PST_MODEM**，用来指明是** RS-232 **口或者是** Modem **设备。**

- dwProvCapabilities：**指定一个位掩码，表示设备能够实现的功能，可以是如表** 2-38 **所示的值。**

表 2-38　　设备功能

值	描述
PCF_16BITMODE	支持特殊的 16 位方式
PCF_DTRDSR	支持 DTR/DSR
PCF_INTTIMEOUTS	支持间隔超时
PCF_PARITY_CHECK	支持奇偶校验
PCF_RLSD	支持 RLSD
PCF_RTSCTS	支持 RTS/CTS
PCF_SETXCHAR	支持可设置的 XON/XOFF 流量控制
PCF_SPECIALCHARS	支持提供的特殊字符
PCF_TOTALTIMEOUTS	支持总超时
PCF_XONXOFF	支持 XON/XOFF 流量控制

- dwSettableParams：指定一个位掩码，它表示可修改的通信参数，如表 2-39 所示。

表 2-39　　通信参数

值	描述
SP_BAUD	1 位停止位
SP_DATABITS	1.5 位停止位
SP_HANDSHAKING	握手（流量控制）
SP_PARITY	奇偶位
SP_PARITY_CHECK	奇偶校验
SP_RLSD	载波检测
SP_STOPBITS	停止位

- dwSettableBaud：指定一个位掩码，表示可以设置的数据传输率。
- wSettableData：指定一个位掩码，表示可以设置的数据位数，如表 2-40 所示的之一。

表 2-40　　数据位数

值	描述
DATABITS_5	5 位数据位
DATABITS_6	6 位数据位
DATABITS_7	7 位数据位
DATABITS_8	8 位数据位
DATABITS_16	16 位数据位
DATABITS_16X	通过串口硬件线路的特殊宽度通道

- wSettableStopParity：指定一个位掩码，表示可以设置的停止位数和奇偶检验位，如表 2-41 表示。

表 2-41　　停止位数和奇偶检验位

值	描述
STOPBITS_10	1 个停止位
STOPBITS_15	1．5 个停止位
STOPBITS_20	2 个停止位
PARITY_NONE	无校验
PARITY_ODD	奇校验
PARITY_EVEN	偶校验
PARITY_MARK	标号校验
PARITY_SPACE	空格校验

- dwCurrentTxQueue：以字节为单位指定当前发送缓冲区大小。

- dwCurrentRxQueue :以字节为单位指定当前接受缓冲区大小。
- dwProvSpec1：指定设备的特定数据。在调用 GetCommProperties 函数之前必须设置该成员为 COMMPROP_INITIALIZED，表明 wPacketLength 成员有效。
- dwProvSpec2 和 wcProvChar(1) ：指定设备的特定数据。若没有关于设备要求的数据格式信息，则应用程序忽略该成员。

如果 dwProvSubType 成员为 PST_MODEM, 则 dwProvSpec1 、 dwProvSpec2 和 wcProvChar(1)成员分别为：

- dwProvSpec1：未用。
- dwProvSpec2：未用。
- wcProvChar(1)：包含一个 MODEMDEVCAPS 结构。

3. 通用通信设备配置

Windows 9x/NT/2000 提供了一个函数 CommConfigDialog 来提供通用通信设备配置，该函数显示了一个对话框，可以从中改变数据传输率、数据位、奇偶校验方法、停止位和流控制方法，如图 2-4 所示。

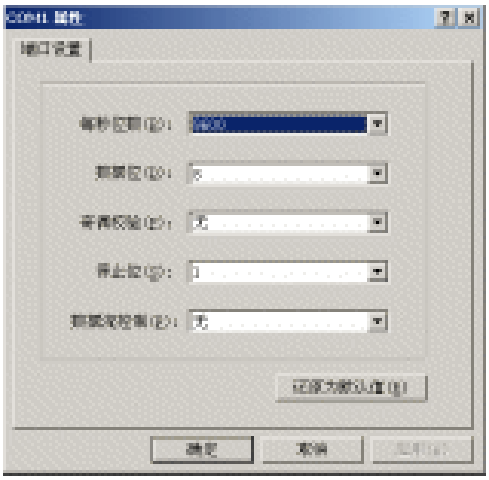


图 2-4 通用通信设备配置对话框

该函数的声明如下：

```
Public Declare Function CommConfigDialog Lib "kernel32" Alias "CommConfigDialogA"(ByVal lpszName As String, ByVal hWnd As Long, lpCC As COMMCONFIG) As Long
```

其中参数：

- lpszName：指明要配置的端口名。
- hWnd：是将拥有此对话框的窗口句柄。
- lpCC：指向一个 COMMCONFIG 结构。

COMMCONFIG 结构的声明如下：

```
Public Type COMMCONFIG
```

```
    dwSize As Long
    wVersion As Integer
    wReserved As Integer
    dcbx As DCB
    dwProviderSubType As Long
    dwProviderOffset As Long
    dwProviderSize As Long
    wcProviderData As Byte
End Type
```

其中成员：

- dwSize：指明整个结构长度，必须指明为一个实际的长度，不能用 Len(COMMCONFIG)代替。
- wVersion：指明该结构的当前的版本号。
- wReserved：保留未用。
- dcbx：是一个 DCB 结构的成员。
- dwProviderSubType，dwProviderOffset，dwProviderSize，wcProviderData：提供有关供应商的信息。

当 CommConfigDialog 函数返回时，选定的设置在 COMMCONFIG 的 DCB 成员中返回。对已打开端口的句柄，CommConfigDialog 没有作为参数传递，实际上并不改变端口的设置，这时的更改要通过 SetCommState 函数来改变串口的设置。

下面给出实际上如何调用 CommConfigDialog 部分代码，实际的项目参见本书所附光盘 chap03 目录下的 showcom 项目。

```
Private Sub Command1_Click()
Dim comm As COMMCONFIG
comm.dwSize=10000
Debug.Print comm.dwSize
CommConfigDialog "COM1", Me.hWnd, comm
Debug.Print comm.dwSize
End Sub
```

读者可以运行该项目，以检验效果。

4. 缓冲区控制

Win32 通信 API 除了提供 SetupComm 函数实现初始化的缓冲区操作外，还提供了 PurgeComm 函数和 FlushFileBuffers 函数来进行缓冲区操作。

PurgeComm 函数的声明如下：

```
Public Declare Function PurgeComm Lib "kernel32" Alias "PurgeComm" (ByVal hFile As Long, ByVal dwFlags As Long) As Long
```

参数 hFile 指向由 CreateFile 函数返回的句柄；参数 dwFlags 表示执行的动作，它可以是

如表 2-42 各值的组合。

表 2-42　　停止位数和奇偶检验位

值	描述
PURGE_TXABORT	即使发送操作没有完成，也终止所有的重叠发送操作，立即返回
PURGE_RXABORT	即使接受操作没有完成，也终止所有的重叠接受操作，立即返回
PURGE_TXCLEAR	清除发送缓冲区
PURGE_RXCLEAR	清除接受缓冲区

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

可见，PurgeComm 函数可以在读写操作的同时，清空缓冲区。当应用程序在读写操作时调用 PurgeComm 函数，不能保证缓冲区的所有字符都被发送。如果要保证缓冲区的所有字符都被发送，应该调用 FlushFileBuffers()函数。该函数只受流量控制的支配，不受超时控制的支配，它在所有的写操作都完成后才返回。

FlushFileBuffers 函数的声明如下：

```
    Public Declare Function FlushFileBuffers Lib "kernel32" Alias
"FlushFileBuffers" (ByVal hFile As Long) As Long
```

参数 hFile 指向由 CreateFile 函数打开的句柄。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

2.6.3　读写串口

Win32 API 不仅支持对串口的同步操作，还支持异步（重叠）操作。函数在进行同步操作时，直到操作完成后才返回。这意味在每次操作时只能等待冗长的读写操作完成。而在重叠（异步）操作中，函数可以立即返回，即使读写操作还没完成，这使费时的读写操作在后台执行的同时，程序还可以做其他的工作。

1. 读串口操作

程序可以使用 Win32 API ReadFile 函数或者 ReadFileEx 函数从串口中读取数据。ReadFile 对同步或异步操作都支持；而 ReadFileEx 只支持异步操作。这两个函数都受到函数是否异步操作、超时操作等有关参数的影响和限定。

ReadFile 的函数声明如下：

```
    Public Declare Function ReadFile Lib "kernel32" Alias "ReadFile" (ByVal hFile
As Long, lpBuffer As Any, ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead
As Long, lpOverlapped As OVERLAPPED) As Long
```

其中参数：

- hFile：指向标识的句柄。对串口来说，就是由 CreateFile 函数返回的句柄。该句柄必须拥有 GENERIC_READ 的权限。
- lpBuffer：指向一个缓冲区。该缓冲区主要用来存放从串口设备中读取的数据。

- NNumberOfBytesToRead：指定要从串口设备中读取的字节数。
- lpNumberOfBytesRead：指向调用该函数读出的字节数。ReadFile 函数在读操作前，首先将其置为 0。Windows NT/2000 中当 lpOverlapped 没有设置时，lpNumberOfBytesRead 必须设置；当 lpOverlapped 设置时，lpNumberOfBytesRead 可以不设置，这时可以调用 GetOverlappedResult 函数获取实际的读取数值。Winows 9x 中这个参数一定要设置。
- LpOverlapped：是一个 OVERLAPPED 的结构，该结构将在后面介绍。如果 hFile 以 FILE_FLAG_OVERLAPPED 方式创建，则需要此结构；否则，不需要此结构。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

ReadFileEx 函数和 ReadFile 函数不同之处在于，它只能异步执行。该函数是对 ReadFile 函数的扩充，它允许应用程序继续其他的操作，异步地报告读操作的完成状态。

ReadFileEx 的函数声明如下：

```
    Public Declare Function ReadFileEx Lib "kernel32" Alias "ReadFileEx" (ByVal
hFile As Long, lpBuffer As Any, ByVal nNumberOfBytesToRead As Long, lpOverlapped
As OVERLAPPED, ByVal lpCompletionRoutine As Long) As Long
```

其中参数：

- hFile：指向标识的句柄。对串口来说，就是由 CreateFile 函数返回的句柄。该句柄必须拥有 GENERIC_READ 的权限。
- lpBuffer：指向一个缓冲区。该缓冲区主要用来存放从串口设备中读取的数据。
- nNumberOfBytesToRead：指定要从串口设备中读取的字节数
- lpOverlapped：是一个 OVERLAPPED 的结构，该结构将在后面介绍。此参数是必需的。
- lpCompletionRoutine：指明一个 I/O 完成后的常用例子的地址。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

当使用 ReadFileEx 函数时，返回不为零时，也应该调用 GetLastError 函数来获取有关这次调用的更多信息。

2. 写串口操作

程序可以使用 Win32 API WriteFile 函数或者 WriteFileEx 函数向串口中写入数据。WriteFile 对同步或异步操作都支持；而 WriteFileEx 只支持异步操作。这两个函数也都受到函数是否异步操作、超时操作等有关参数的影响和限定。

WriteFile 的函数声明如下：

```
    Public Declare Function WriteFile Lib "kernel32" Alias "WriteFile" (ByVal hFile
As Long, lpBuffer As Any, ByVal nNumberOfBytesToWrite As Long,
lpNumberOfBytesWritten As Long, lpOverlapped As OVERLAPPED) As Long
```

其中参数：

- hFile：指向标识的句柄。对串口来说，就是由 CreateFile 函数返回的句柄。该句柄

必须拥有 GENERIC_WRITE 的权限。

- lpBuffer：指向一个缓冲区。该缓冲区主要用来存放待写入串口设备的数据。
- nNumberOfBytesToWrite：指定要从串口设备中读取的字节数
- lpNumberOfBytesWritten：指向调用该函数已写入的字节数。WriteFile 函数在读操作前，首先将其置为 0。Windows NT/2000 中当 lpOverlapped 没有设置时，lpNumberOfBytesWritten 必须设置 当 lpOverlapped 设置时，lpNumberOfBytesWritten 可以不设置，这时可以调用 GetOverlappedResult 函数获取实际的读取数值。在 Winows 9x 中这个参数一定要设置。
- LpOverlapped：是一个 OVERLAPPED 的结构，该结构将在后面介绍。如果 hFile 以 FILE_FLAG_OVERLAPPED 方式创建，则需要此结构；否则，不需要此结构。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

与 ReadFileEx 函数和 ReadFile 函数之间的差别一样，WriteFileEx 函数和 WriteFile 函数不同之处在于，WriteFileEx 只能异步执行。该函数是对 WriteFile 函数的扩充，它允许应用程序继续其他的操作，异步的报告写操作的完成状态。

WriteFileEx 函数声明如下：

```
Public Declare Function WriteFileEx Lib "kernel32" Alias "WriteFileEx" (ByVal hFile As Long, lpBuffer As Any, ByVal nNumberOfBytesToWrite As Long, lpOverlapped As OVERLAPPED, ByVal lpCompletionRoutine As Long) As Long
```

其中参数：

- hFile：指向标识的句柄。对串口来说，就是由 CreateFile 函数返回的句柄。该句柄必须拥有 GENERIC_READ 的权限。
- lpBuffer：指向一个缓冲区。该缓冲区主要用来存放从串口设备中读取的数据。
- nNumberOfBytesToWrite：指定调用该函数要写入的字节数
- lpOverlapped：是一个 OVERLAPPED 的结构，该结构将在后面介绍。此参数是必需的。
- lpCompletionRoutine：指明一个 I/O 完成后的常用例子的地址。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

当使用 WriteFileEx 函数时，返回不为零时，也应该调用 GetLastError 函数来获取有关这次调用的更多信息。

Windows 还提供 TransmitCommChar 函数在发送时指定一字符的传递权限级别最高，该字符可在缓冲区的其他字符发送之前发送。

TransmitCommChar 函数的声明如下：

```
Public Declare Function TransmitCommChar Lib "kernel32" Alias "TransmitCommChar" (ByVal nCid As Long, ByVal cChar As Byte) As Long
```

其中参数 nCid 表示通信的端口，是由 CreateFile 函数返回的句柄。CChar 是指定要发送的字符。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一

步的错误信息，可以调用 GetLastError 函数来获取。

该函数是同步函数，也受流量控制和写超时支配。因而，当串口没有传输时，TransmitCommChar 函数不能重复调用。当 TransmitCommChar 把一个字符放到输出缓冲区中，在下次这个函数调用前，这个字符必须被传出；否则 TransmitCommChar 函数就会出错。

3. 重叠 I/O 操作

重叠（异步）I/O 操作是指应用程序可以在后台读或者写数据，而在前台做其他事情。例如，应用程序可以开始时对 10000 个直接进行读或写操作，然后返回执行其他的响应；在读写完成以后，Windows 就会产生一个信号，应用程序得到此信号，可以进行其他的读写操作。

要使用 OVERLAPPED，CreateFile 的 dwFlagsAndAttributes 参数必须设为 FILE_FLAG_OVERLAPPED 标志，读写串口函数必须指定 OVERLAPPED 结构。异步 I/O 在 Windows 里用的很多。

OVERLAPPED 结构类型声明如下：

```
Public Type OVERLAPPED
       Internal As Long
       InternalHigh As Long
       offset As Long
       OffsetHigh As Long
       hEvent As Long
End Type
```

其中成员：

- InternalHigh：为操作系统保留，指出一个和系统相关的状态。当 GetOverlappedResult 函数返回时，如果为将扩展错误信息设置为 ERROR_IO_PENDING.，该成员有效。
- InternalHigh：为操作系统保留，指出发送或接受的数据长度。当 GetOverlappedResult 函数返回非 0 时，该成员有效。
- offset 和 OffsetHigh 成员指明文件传送的开始位置和字节偏移量的高位字。当进行端口操作时，这两个成员被忽略。
- hEvent：指定一个 I/O 完成时触发的事件（信号）。在调用读写函数进行 I/O 操作之前，必须设置该成员。

在设置了重叠 I/O 操作后，I/O 操作和函数返回有以下两种情况：

- 函数返回时 I/O 操作已经完成：此时结果好象是同步执行的，但实际上这是异步执行的结果。
- 函数返回时 I/O 操作还没完成：此时一方面，函数返回为 0，并且 GetLastError 函数返回 ERROR_IO_PENDING.；另一方面，系统把 OVERLAPPED 中的信号事件设为无信号状态。当 I/O 操作完成时，系统把它设置为有信号状态。

重叠 I/O 操作可以由特定的函数 GetOverlappedResult 来获取结果，也可以用 Windows 信号函数来处理。

GetOverlappedResult 函数的声明如下：

```
    Public Declare Function GetOverlappedResult Lib "kernel32" Alias
"GetOverlappedResult" (ByVal hFile As Long, lpOverlapped As OVERLAPPED,
lpNumberOfBytesTransferred As Long, ByVal bWait As Long) As Long
```

其中参数：

- hFile：表示通信句柄。它应该与开始调用重叠结构的 ReadFile, WriteFile, WaitCommEvent 函数的参数一致。
- LpOverlapped：是指在启动重叠操作时指定的 OVERLAPPED 结构。
- LpNumberOfBytesTransferred：指向一个长整型变量。该变量接收由一个读或写操作实际传递的字节数。
- BWait：指定函数是否等待挂起的重叠操作完成。如果该参数设为 1，则该函数直到 I/O 操作完成后才返回。如果该参数被设为 0，并且操作处于挂起状态，则该函数返回 0，并且 GetLastError 函数返回 ERROR_IO_INCOMPLETE.

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

Windows 也使用等待函数来检查事件对象的当前状态或等待 Windows 状态信号,在 WaitForSingleObject, WaitForSingleObjectEx 函数，以及 WaitForMultipleObjects, WaitForMultipleObjectsEx 函数中指定 OVERLAPPED 结构中的 hEvent，即可获取函数返回事件。

以 WaitForSingleObject 函数为例，介绍以下 Win32 等待事件函数。

WaitForSingleObject 的函数声明如下：

```
    Public Declare Function WaitForSingleObject Lib "kernel32" Alias
"WaitForSingleObject" (ByVal hHandle As Long, ByVal dwMilliseconds As Long) As
Long
```

其中参数：

- hHandle：指定一个同步事件的句柄。在这里是指 OVERLAPPED 结构中的 hEvent 句柄。
- dwMilliseconds：以 ms 为单位指定超时时间。

该函数如果调用成功，则返回值表示导致该函数返回的事件，否则返回值为 WAIT_FAILED。成功的返回值可以是以下的三值之一：

- WAIT_ABANDONED：等待的对象是互斥对象，并且拥有该对象的线程在终止时没有释放该对象
- WAIT_OBJECT_0：指定的对象处于有信号状态
- WAIT_TIMEOUT：超时事件发生后，指定的对象仍处于无信号状态。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

✍ 注意，事件对象在应用的时候必须先行创建。创建对象的函数用 CreeatEvent 函数，重置函数采用 ResetEvent 函数。

4. **超时设置**

Winows 9x/NT/2000 中读写串口引入了超时的结构。超时参数直接影响读和写的操作行为。当领先设定的超时间隔消逝时，ReadFile、ReadFileEx、WriteFile 或 WriteFileEx 操作仍未结束，那么超时将无条件结束读写操作，而不管是否已读出或写入指定数量的字符。

在读或写操作期间发生的超时将不按错误处理，即读或写操作返回指示成功的值。对于同步读／写操作，实际传输的字节数由 ReadFile 或 WriteFile 报告。对于重叠 I/O 操作，则由 OVERLAPPED 结构来获取。

超时结构 COMMTIMEOUTS 的声明如下：

```
Public Type COMMTIMEOUTS
       ReadIntervalTimeout As Long
       ReadTotalTimeoutMultiplier As Long
       ReadTotalTimeoutConstant As Long
       WriteTotalTimeoutMultiplier As Long
       WriteTotalTimeoutConstant As Long
End Type
```

其中成员：

- ReadlntervalTimeout：以 ms 为单位指定通信线路上两个字符到达之间的最大时间。在 ReadFile 操作期间，从接收到第一个字符时开始计时。若任意两个字符到达之间的时间间隔超过这个最大值，则 ReadFile 操作完成，并返回缓冲数据。如果被置为 0，则表示不使用间隔超时。
- ReadTotalTimeoutMultiplier：以 ms 为单位指定一个系数，该系数用来计算读操作的总超时时间。
- ReadTotalTimeoutConstant：以 ms 为单位指定一个常数，该常数也用来计算读操作的总超时时间。
- WriteTotalTimeoutMultiplier：以 ms 为单位指定一个系数，该系数用来计算写操作的总超时时间。

WriteTotalTimeoutConstant：以 ms 为单位指定一个常数，该常数也用来计算写操作的总超时时间。

由此可知，总超时时间计算方法如下：

```
   Tlmeout= (MULTIPLIER * number_of_bytes) +CONSTANT
```

其中 MULTIPLIER 为总超时系数，CONSTANT 为总超时常数。使用系数和常数使总超时时间能够根据所请求的数据变化。应用程序通过设置系数为 0 而只使用常数，或通过设置常数为 0 而只使用系数。如果系数和常数都为 0，则没有使用总超时。

因此每个读操作的总超时时间等于 ReadTotalTimeoutMultiplier 成员值乘以读操作所需字节数再加上 ReadTotalTimeoutConstant 成员值的和。如果将 ReadTotalTimeoutMultiplier 和 ReadTotalTimeoutConstant 成员值都置为零则表示读操作不使用总超时时间。

如果读间隔超时参数 ReadlntervalTimeout 被置为 MAXDWORD，而且两个读总超时参数都被置为 0，那么表示只要一读完接收缓冲区而不管得到什么字符就完成读操作，即使它是空的。

当接收中有间隔时，间隔超时将迫使读操作返回。因此使用间隔超时的进程可以设置一个非常短的间隔超时参数，这样它就可以实现对一个或一些字符的小的、孤立的数据作出反应。

每个写操作的总超时时间等于 WriteTotalTimeoutMultiplier 成员值乘上要写的字节数，再加上 WriteTotalTimeoutConstant 成员值的和。如果将 WriteTotalTimeoutMultiplier 和 WriteTotalTimeoutConstant 都置为 0 则表示写操作不使用超时时间。

在传输被某种流量控制阻塞时或调用 SetCommBreak 函数把字符挂起时，写操作的超时可能有用。

如果所有的读超时参数都为 0，那么没有使用读超时，而且读操作直到读完要求的字节数或发生错误时为止。同样，如果所有的写超时参数都为 0，那么写操作直到写完要求的字节数或发生错误时为止。

当打开通信资源时，超时参数将根据上次设备打开时所设置的值设置。如果资源从未打开过或如果调用 SetupComm 函数，那么所有的超时参数都设置为 0。

如果要获取当前超时参数，应用程序可以调用 GetCommTimeouts 函数。

该函数的声明如下：

```
    Public Declare Function GetCommTimeouts Lib "kernel32" Alias "GetCommTimeouts"
(ByVal hFile As Long, lpCommTimeouts As COMMTIMEOUTS) As Long
```

其中参数：

- hFile：标识通信设备，CreateFile 函数返回该句柄。
- lpCommTimeouts：指向一个 COMMTIMEOUTS 结构，返回超时信息。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

如果要设置或改变原来的超时参数，应用程序可以调用 SetCommTimeouts 函数。

该函数的声明如下：

```
    Public Declare Function SetCommTimeouts Lib "kernel32" Alias "SetCommTimeouts"
(ByVal hFile As Long, lpCommTimeouts As COMMTIMEOUTS) As Long
```

其中参数：

- hFile：标识通信设备，CreateFile 函数返回该句柄。
- lpCommTimeouts：指向一个 COMMTIMEOUTS 结构，是设定的超时信息。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

5. 通信状态和通信错误

如果在串口通信中发生错误，如发生中断、奇偶错误等等，I/O 操作将会终止。如果程序想进一步 I/O 操作，必须调用 ClearCommError 函数。

ClearCommError 具有双重目的，它的第一个目的是清除错误条件。第二个目的是确定端口通信状态。

ClearCommError 函数的声明如下：

```
    Public Declare Function ClearCommError Lib "kernel32" Alias "ClearCommError"
```

```
(ByVal hFile As Long, lpErrors As Long, lpStat As COMSTAT) As Long
```

其中参数：

- hFile：**标识通信设备，是由** CreateFile **函数返回的句柄。**
- lpErrors：**指向将用一个指明错误类型的掩码填充的** 32 **位变量。**

该参数是如表 2-43 **各值的组合。**

表 2-43　　**通信错误列表**

值	说明
CE_BREAK	硬件检测到一个中断条件
CE_FRAME	硬件检测到一个帧出错
CE_IOE	发生 I/O 错误
CE_MODE	模式出错，或者是句柄无效
CE_OVERRUN	超速错误
CE_RXOVER	接收缓冲区超限，或是输入缓冲区中没有空间，或是在文件结束符（EOF）接收后接收到一个字符
CE_RXPARITY	奇偶校验错误
CE_TXFULL	发送缓冲区满

在上表中有关并口和网络错误类型 CE_DNS、CE_OOP、CE_PTO **在串口通信用不到，本书没有列出。**

- lpStat：**指向一个** COMSTAT **结构，该结构接收设备的状态信息。如果** lpStat **不设置，则没有设备状态信息被返回。**

例如，以下代码不返回接收设备的状态信息。

```
Call  ClearCommError(handle, errors, 0)
```

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0**。如果想获得进一步的错误信息，可以调用** GetLastError **函数来获取。**

在同步操作时，可以调用 ClearCommError **函数来确定串口的接收缓冲区中处于等待状态的字节数，而后可用** ReadFile **或者** WriteFile **一次性写完。**

COMSTAT **结构存放有关通信设备的当前信息。该结构内容由**ClearCommError **函数填写。其定义如下：**

```
Public Type COMSTAT
       fBitFields As Long ’See Comment in Win32API.Txt
       cbInQue As Long
       cbOutQue As Long
End Type
```

其中成员：

- fBitFields：**这** 4 **个字节指定了** 8 **个属性。每个属性可以通过对** fbitFields **的逐位的逻辑与和逻辑或得到,如表** 2-44 **所示。**

表 2-44　　fBitFields **列表**

属性	所在位数	说明
fCtsHold	1	指明是否等待 CTS 信号。若为 1，则发送等待
fDsrHold	2	指明是否等待 DSR 信号。若为 1，则发送等待
fRlsdHold	3	指明是否等待 RLSD 信号。若为 1，则发送等待

续表

属性	所在位数	说明
fXoffHold	4	指明收到 XOFF 字符后发送是否等待。若为 1，则发送等待
fXoffSent	5	指明发送完 XOFF 字符后发送是否等待。若为 1，则发送等待。当把 XOFF 字符发送给一系统时，该系统就把下一个字符当作 XON，而不管实际是什么字符，此时发送将停止
fEof	6	EOF 字符送出
fTxim	7	指明字符是否正等待被发送。若为 1，则字符正等待被发送
fReserved	8	系统保留（25）

- cbInQue：指明串行设备接收到的字节数。它并不是指 ReadFile 操作要求读的字节数。
- cboutQue：指明发送缓冲区中尚未发送的字节数。当进行不重叠写操作时，该值为 0。

2.6.4 通信事件

Windows 进程监视发生在通信资源中的一组事件，这样应用程序可以不检查端口状态就知道某些条件何时发生。例如，应用程序可以利用事件监视确定 CTS 和 DSR 信号何时改变状态。

1. 通信事件

Windows 可以利用 GetCommMask 和 SetCommMask 来控制如表 2-45 所列出的通信事件。

表 2-45　　Windows 通信事件列表

通信事件	说明
EV_BREAK	检测到输入的中止
EV_CTS	CTS（清除发送）信号改变状态
EV_DSR	DSR（数据设置就绪）信号改变状态
EV_ERR	发生了线路状态错误。线路状态错误是 CE_FRAME（帧错误）、CE_OVERRUN（接收缓冲区超限）和 CE_RXPARITY（奇偶校验错误）
EV_RING	检测到振铃
EV_RLSD	RLSD（接收线路信号检测）信号改变状态
EV_RXCHAR	接收到一个字符，并放入输入缓冲区
EV_RXFLAG	接收到事件字符（DCB 结构的 EvtChar 成员），并放入输入缓冲区
EV_TXEMPTY	输出缓冲区中最后一个字符发送出去

2. 操作通信事件

应用程序可以通过使用 SetCommMask 函数建立事件掩模来监视指定通信资源上的事件。SetCommMask 函数的声明如下：

```
    Public Declare Function SetCommMask Lib "kernel32" Alias "SetCommMask" (ByVal
hFile As Long, ByVal dwEvtMask As Long) As Long
```

其中参数：

- hFile：标识通信设备，CreateFile 函数返回这个句柄。
- dwEvtMask：事件掩模，表示将被监视的通信事件。如果该参数被置为 0，则表示禁止所有事件。它可以是表 2-45 所列各事件的组合。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为零。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

调用 SetCommMask 函数的例子如下：

```
Private CurrentEventMask&
CurrentEventMask=EV_ERR
Call SetCommMask(handle, CurrentEventMask)
```

如果想要获取特定通信资源的当前事件掩模，应用程序可以使用 GetCommMask 函数。GetCommMask 函数的声明如下：

```
Public Declare Function GetCommMask Lib "kernel32" Alias "GetCommMask" (ByVal
hFile As Long, lpEvtMask As Long) As Long
```

其中参数：

- hFile：标识通信设备，CreateFile 函数返回这个句柄。
- lpEvtMask：指向一个被监视的通信事件的掩模填充的 32 位变量。它可以是表 3-15 所列各事件的组合。

如果该函数调用成功，则返回值为非零；若函数失败，则返回值为 0。如果想获得进一步的错误信息，可以调用 GetLastError 函数来获取。

3. 监视通信事件

在指定被监视的事件掩模之后，进程使用 WaitCommEvent 函数等待其中一个事件发生。WaitCommEvent 既可以同步使用，也可以用作重叠操作。

WaitCommEvent 函数的声明如下：

```
Public Declare Function WaitCommEvent Lib "kernel32" Alias "WaitCommEvent"
(ByVal hFile As Long, lpEvtMask As Long, lpOverlapped As OVERLAPPED) As Long
```

其中参数：

- hFile：标识通信设备，CreateFile 函数返回的句柄。
- lpEvtMask：指向一个 32 位变量，该变量接收一个指示所发生的事件类型的掩码。如果出错，该参数被置为 0；否则它是表 3-15 所列各事件的之一。
- lpOverlapped：指向一个 OVERLAPPED 结构。如果打开 hFile 标识的通信设备时未指定 FILE_FLAG_OVERLAPPED 标志，则该参数被忽略。如果不需要重叠操作，则该参数不用设置。

如果函数调用成功，则返回值为非零；否则返回值为 0。为获取进一步错误信息，可以调用 GetLastError 函数。

如果 lpOverlapped 参数不设置或打开 hFile 标识的通信设备时未指定 FILE_FLAG_OVERLAPPED 标志，则直到发生了指定事件之一或出错时，WaitCommEvent 函数才返回。如果 lpOverlapped 参数指向一个 OVERLAPPED 结构，并且打开 hFile 标识的通信设备时指定了 FILE_FLAG_OVERLAPPED 标志，则 WaitCommEvent 函数以重叠操作实现。

这种情况下，OVERLAPPED 结构中必须含有一个人工复位事件对象的句柄。和所有重叠函数一样，如果重叠操作不能立即完成，则该函数返回 0，并且 GetLastError 函数返回 ERROR_IO_PENDING，以指示该操作正在后台执行。此时在 WaitCommEvent 函数返回之前，系统将 OVERLAPPED 结构的 hEvent 成员置为无信号状态；当发生了指定事件之一或出错时，系统将其置为有信号状态。调用程序可使用等待函数确定事件对象的状态，然后使用 GetoverlappedResult 函数确定 WaitCommEvent 函数的操作结果。GetoverlappedResult 函数报告该操作成功或失败，并且 lpEvtMask 参数所指向的变量被设置以指示所发生的事件。

如果一个进程在 WaitCommEvent 操作进行期间使用 SetCommMask 函数试图改变通信设备的事件掩码，则 WaitCommEvent 函数立即返回，并且由 EvtMask 参数指向的变量被设置为 0。

另外请读者注意，WaitCommEvent 只检测发生在等待开始后的事件。例如，如果指定 EV_RXCHAR 事件，则只有当收到任何字符并放进接收缓冲区后才满足等待条件。WaitCommEvent 调用时已在接收缓冲区中的字符不符合等待条件。

监视 CTS、DSR 等信号状态改变的事件时，WaitCommEvent 只报告此变动，但不报告当前状态。如果要查询这些信号状态，进程可以使用 GetCommModemstatus 函数。

2.6.5 设备控制命令

1. 控制握手信号

Win32 也提供了 EscapeCommFunction 函数。该函数指示一个给定通信设备来执行一个扩展功能，例如改变硬件握手信号状态和软件握手信号状态。

EscapeCommFunction 函数的声明如下：

```
Public Declare Function EscapeCommFunction Lib "kernel32" Alias "EscapeCommFunction" (ByVal nCid As Long, ByVal nFunc As Long) As Long
```

其中：

- nCid：标识通信设备，它是由 CreateFile 函数返回这个句柄。
- nFunc：指定要执行的扩展功能的代码，它可以是如表 2-46 所列的值之一。

表 2-46　　扩展功能代码

值	说明
CLRDTR	清除 DTR 信号
CLRRTS	清除 RTS 信号
SETDTR	设置 DTR 信号
SETRTS	设置 RTS 信号
SETXOFF	假设 XOFF 字符收到，停止传输
SETXON	假设 XON 字符收到，停止传输
SETBREAK	终止字符传输，使传输处于中断状态，直到调用 ClearCommBreak 函数（或者用 EscapeCommFunction 函数调用 CLRBREAK 扩展功能）为止。该扩展功能与 SetCommBreak 函数相同。注意该功能不刷新未传的数据
CLRBREAK	恢复字符传输，使传输处于不中断状态。该扩展功能与 ClearCommBreak 函数相同

因此可以通过设置 CLRDTR 和 SETDTR 码可以实现手工 DTR 流量控制。

如果函数调用成功，则返回值为非 0；否则返回值为 0。为获取进一步错误信息，可以调用 GetLastError 函数。

如果进程在通信设备已配置使 DTR 握手信号有效时，再使用 EscapeCommFunction 操作 DTR 信号线，或者配置 RTS 握手信号有效时，操作 RTS 信号线会发生冲突。

2. 设备操作

在通信过程中也是使用 SetCommBreak 和 GetCommBreak 函数实现通信设备的挂起和通信设备的恢复。

SetCommBreak 函数实现挂起字符传输以及将通信设备置为间断状态，直到调用了 ClearCommBreak 函数为止。该函数不清除尚未传输的数据。

SetCommBreak 函数的声明如下：

```
    Public Declare Function SetCommBreak Lib "kernel32" Alias "SetCommBreak" (ByVal
nCid As Long) As Long
```

如果函数调用成功，则返回值为非零；否则返回值为 0。为获取进一步错误信息，可以调用 GetLastError 函数。

如果通信设备是通过 SetCommBreak 或 EscapeCommFunction 函数被置成间断状态的。那么将暂停字符传输，直到调用 ClearCommBreak 函数来清除该间断状态为止。

ClearCommBreak 函数的声明如下：

```
    Public Declare Function ClearCommBreak Lib "kernel32" Alias "ClearCommBreak"
(ByVal nCid As Long) As Long
```

如果函数调用成功，则返回值为非 0；否则返回值为 0。为获取进一步错误信息，可以调用 GetLastError 函数。

2.7 小结

本章并没有介绍具体的编程，主要是为了初中级读者而准备的，后面的章节中主要是使用了这些控件来进行开发的，因此对于不熟悉这些控件的读者来说，必须先学习这些控件的使用。在后面的章节中，是对这些控件的高级应用，因此必须首先打好基础。同时也介绍了相关的 API 函数。

第 3 章　实现网络基本应用

本章重点介绍一些在网络中常用的小应用程序，这些小程序一般只需要用某些控件的简单属性或者简单的 API 函数就能够实现。本章实现了如下基本应用：

- 端口扫描程序
- Ping 程序的实现
- 根据域名获得计算机的 IP 地址
- 获取网卡地址
- 超级链接和发送 E-mail

3.1　端口扫描程序

端口扫描程序是网络中最常见的应用之一，通过端口扫描，可以知道本地计算机或者是远程计算机的某一个端口是否被占用了。因为在开发 TCP/IP 程序的时候，基本上都是基于客户机/服务器模式的，而这种模式的网络开发服务器端通常需要在一个特定的端口进行监听，而在监听之前，首先需要知道计算机上什么端口被占用，什么端口没有被占用。

实现端口扫描的方法很多，首先介绍直接使用 Winsock 控件来进行判断本地计算机某一端口是否被占用的方法。

程序运行界面如图 3-1 所示，读者可以输入端口的扫描范围，然后单击“开始扫描”按钮，进行端口扫描。

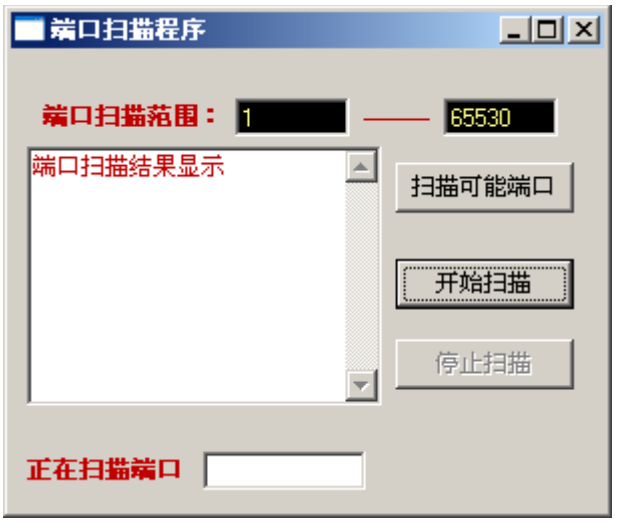


图 3-1　端口扫描程序界面

1. **实例原理**

本程序的关键是通过使用 Winsock **控件的一个属性来实现的，即** LocalPort **属性，当给该控件设定一个端口以后，然后调用侦听的方法，即** Winsock1.Listen**，如果该端口已经被占用，则程序会出错，为了能够判断是否出错，我们在程序中做了错误处理，并对错误进行分析，如果错误的编号为** 10048**，则表明是端口被占用而导致的错误，则输出被占用的端口号，然后回到出错误的地方，继续进行下一个端口的扫描，知道所有被定义的端口范围扫描完毕。该程序中关键的部分是函数** scanningports**，读者可以仔细看看。**

2. **建立工程项目**

本实例的实现非常简单，就是在程序中增加了一个 Winsock **控件，然后通过使用该控件的一些属性以及** VB **的编程技巧来实现的。其中设计界面如图** 3-2 **所示。**

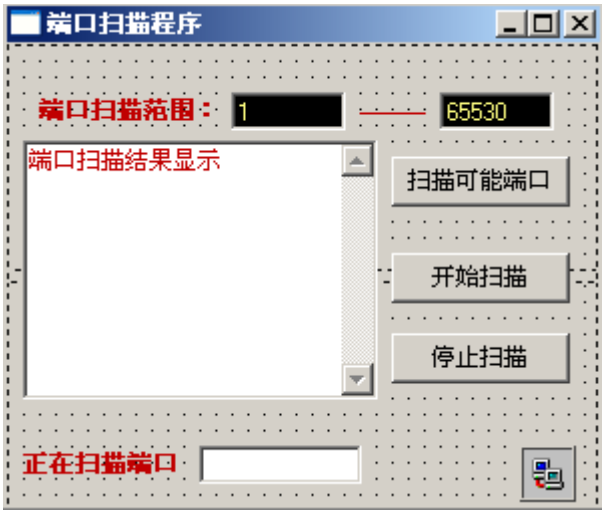


图 3-2　**端口扫描程序设计界面**

在图 3-2 **中的右下角的控件为** Winsock **控件。**

3. **代码分析**

下面的代码实现了端口扫描的功能，具体程序请参见附赠光盘的“ chap03\PortScanner **”目录。**

```
    ------------------------------------------------    程    序    代    码
------------------------------------------------
    Dim portnum As Long
    Dim start As String
    Dim TimerOnOff As String

    '“开始扫描”按钮单击事件
    Private Sub Command1_Click()
    Command2.Enabled = True
```

```
    If Text1.Text = "" Then
    MsgBox "输入开始扫描端口"
    Exit Sub
    End If
    If Text2.Text = "" Then
    MsgBox "输入最大扫描端口"
    Exit Sub
    End If

    Text1.Locked = True
    Text2.Locked = True
    Command1.Enabled = False
    Winsock1.Close
    start = True
    logport.Text = "开始扫描"

    Call scanningports
    logport.Text = logport.Text & vbCrLf & "端口" & Text1.Text & "- " & Text3.Text
& "扫描完毕."
    End Sub

    '端口扫描程序
    Sub scanningports()
    Dim porttwo As Long
    portnum = Text1.Text
    porttwo = Text2.Text
    Command2.Enabled = True
    On Error GoTo viriio
    Do
       portnum = portnum + 1
       DoEvents
       If start = True Then
          Winsock1.Close '
          DoEvents '增加程序的稳定性
          Winsock1.LocalPort = portnum
          DoEvents
          Text3.Text = portnum
          Winsock1.Listen '开始侦听
          DoEvents
```

```
        Else '如果单击停止按钮
            portnum = 0
            Command1.Enabled = True
            Text1.Locked = False
            Text2.Locked = False
            Exit Sub
        End If
        Winsock1.Close
        DoEvents
    Loop Until portnum >= porttwo
        portnum = 0
        Command1.Enabled = True
        logport.Text = logport.Text & vbCrLf & "完成端口扫描!" & vbCrLf
        Text1.Locked = False
        Text2.Locked = False
    viriio:

        If Err.Number = 10048 Then
            logport.Text = logport.Text & vbCrLf & "端口" & Winsock1.LocalPort &
" 已经被使用!"
            Resume Next ' 回到出错的地方，继续下一个端口扫描
        End If

    End Sub

    '“停止”按钮单击事件
    Private Sub Command2_Click()
    Command2.Enabled = False
    start = False
    End Sub

    '“扫描所有端口”按钮单击事件
    Private Sub Command3_Click()
    Text1.Text = "1"
    Text2.Text = "65530"
    Text1.Locked = True
    Text2.Locked = True
    Command1.Enabled = False
    Winsock1.Close
```

```
    start = True

    Call scanningports
    End Sub

    ------------------------------------------      代      码      完      毕
------------------------------------------------------------
```

3.2 Ping 程序的实现

Ping 是最常用的网络状态判断指令，Ping 程序编写的目的是为了测试另一台主机是否可达。该程序发送一份 ICMP 回显请求报文给主机，并等待返回 ICMP 回显应答。

Windows 操作系统中都自带了这样一个工具，用户能够很方便地进行 Ping 命令的测试，只要在开始菜单中单击“运行”菜单，然后在弹出的对话框中输入 Ping 命令，在 Ping 命令后面输入要验证的网站域名或者 IP 地址，如图 3-3 所示。

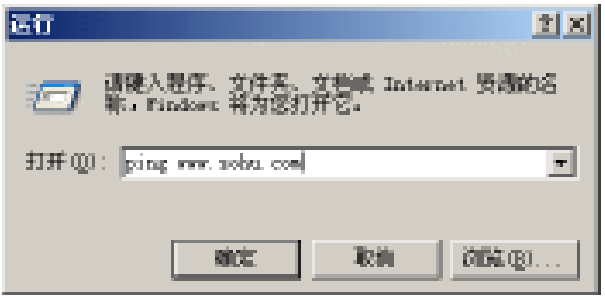


图 3-3　系统 Ping 命令测试

Ping 命令的结果如图 3-4 所示。

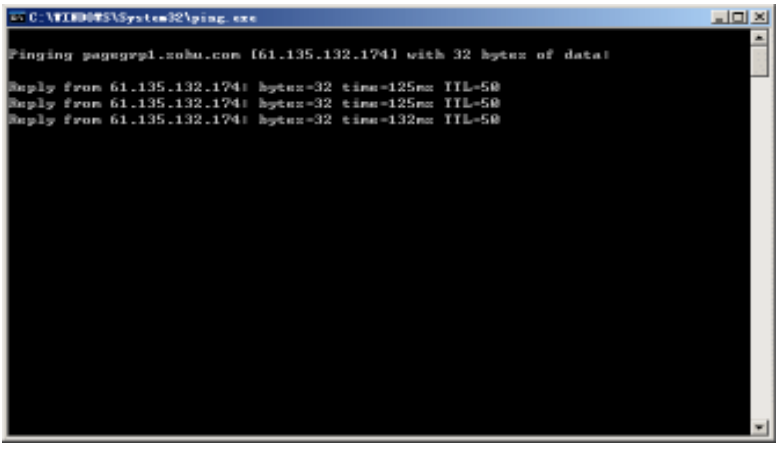


图 3-4　系统 Ping 命令结果图

一般来说，如果用 Ping 命令不能检测到某台主机，那么也就不能使用 Telnet 或者 FTP 命令。但是，如果不能 Telnet 到某台主机，那么通常可以用 Ping 程序来确定问题出在哪里。

Ping 程序还能测出到这台主机的往返时间，以表明该主机离我们有“ 多远 ”。大多数的 TCP/IP 实现都在内核中直接支持 Ping 服务器，它不是一个用户进程。

Ping 程序的实现方法是：主机向远程计算机发出 ICMP 回显请求以后，远程计算机会拦截这个请求，然后生成一条回显应答消息，再通过网络传回给主机。假如出于某些方面的原因，不能抵达目标主机，就会生成对应的 ICMP 错误消息（比如“ 目标主机访问不可到达 ”），由原先打算建立通信的那个路径上某处的一个路由器返回。假定与远程主机的物理性连接并不存在问题，但远程主机已经关机或没有设置对网络事件作出响应，便需由用户自己的程序来执行超时检定，来侦测出这样的情况。

本实例是通过 Winsock API 函数来实现可视化界面的 Ping 函数，可以集成到许多大的应用当中。

本实例的运行界面如图 3-5 所示。

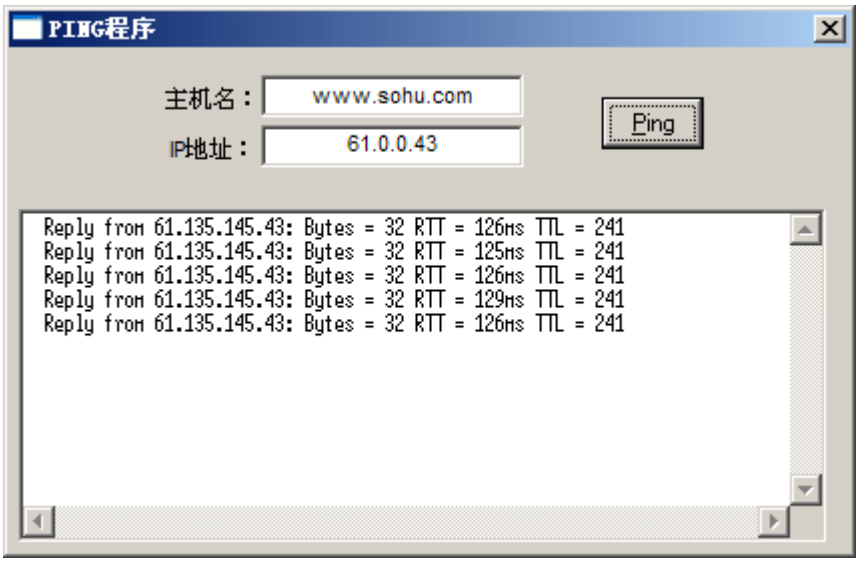


图 3-5　Ping 程序运行界面

1. 实例原理

Ping 的应用也非常简单，只需要通过几个 API 函数就能够实现。不过在实际运用中，需要对函数的返回码有一定的了解，因为函数返回的并不是普通的直接的含义，需要翻译过来。当用户单击“ Ping ”按钮时，先后调用了以下几个函数：vbWSAStartup 、vbGetHostByName、vbIcmpCreateFile、vbIcmpSendEcho、vbIcmpCloseHandle、vbWSACleanup。vbWSAStartup 函数主要是进行 Winsock API 的初始化；其中 vbGetHostByName 用于通过给定的域名获得相应的 IP 地址；而 vbIcmpCreateFile 函数则是调用 API 函数 IcmpCreateFile 创建一个 ICMP 句柄，只有先创建该句柄，才能够在后面进行 ICMP 的请求；接下来是调用了函数 vbIcmpSendEcho，该函数是 Ping 功能的核心函数，它是调用 API 函数 IcmpSendEcho，该函数调用成功以后，将把返回的结果保存在结构体 pIPe 中，它的具体结构声明参见 IP_ECHO_REPLY，在 ICMP 模块中；通过分析 pIPe 的各个字段的值就可以很容易得到 Ping 的结果了，当前所处的状态可以通过 pIPe.Status 来获得。最后调用 vbIcmpCloseHandle、

vbWSACleanup 来关闭 ICMP 句柄和 Winsock 句柄。

2. 建立工程项目

该程序地界面设计非常简单，不需要重要的控件，而只需要使用 Winsock API 函数就可以了，如图 3-6 所示。

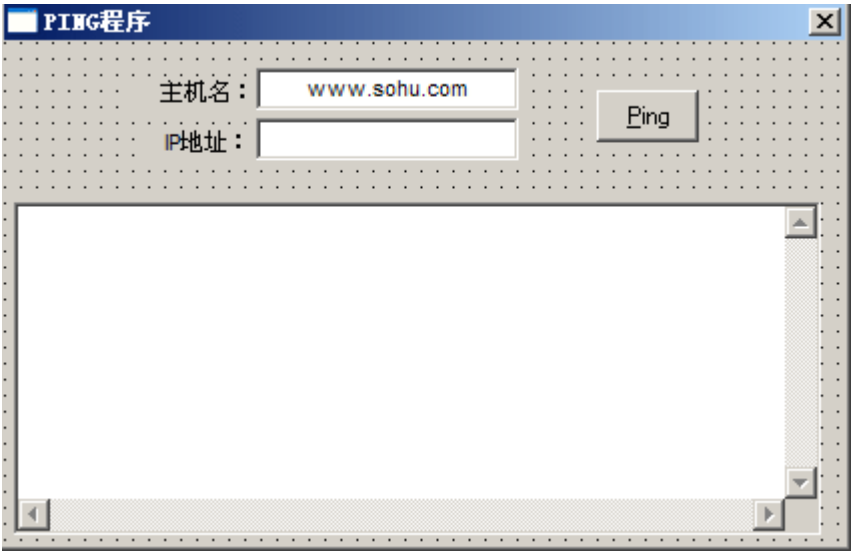


图 3-6 Ping 程序设计界面

3. 代码分析

详细代码请参见附送光盘，本实例共有两个文件，分别是 Form1.frm 和 ICMP.bas。Form1.frm 文件代码如下：

```
    ----------------------------------------              代                码
----------------------------------------------------------------------
    ' WSock32 变量
    Dim iReturn As Long, sLowByte As String, sHighByte As String
    Dim sMsg As String, HostLen As Long, Host As String
    Dim Hostent As Hostent, PointerToPointer As Long, ListAddress As Long
    Dim WSAdata As WSAdata, DotA As Long, DotAddr As String, ListAddr As Long
    Dim MaxUDP As Long, MaxSockets As Long, i As Integer
    Dim Description As String, Status As String

    ' ICMP 变量
    Dim bReturn As Boolean, hIP As Long
    Dim szBuffer As String
    Dim Addr As Long
    Dim RCode As String
```

```
Dim RespondingHost As String

' TRACERT 变量
Dim TraceRT As Boolean
Dim TTL As Integer

' WSock32 常数
Const WS_VERSION_MAJOR = &H101 \ &H100 And &HFF&
Const WS_VERSION_MINOR = &H101 And &HFF&
Const MIN_SOCKETS_REQD = 0
'将状态码翻译成相关的说明
Sub GetRCode()
    If pIPe.Status = 0 Then RCode = "Success"
    If pIPe.Status = 11001 Then RCode = "Buffer too Small"
    If pIPe.Status = 11002 Then RCode = "Dest Network Not Reachable"
    If pIPe.Status = 11003 Then RCode = "Dest Host Not Reachable"
    If pIPe.Status = 11004 Then RCode = "Dest Protocol Not Reachable"
    If pIPe.Status = 11005 Then RCode = "Dest Port Not Reachable"
    If pIPe.Status = 11006 Then RCode = "No Resources Available"
    If pIPe.Status = 11007 Then RCode = "Bad Option"
    If pIPe.Status = 11008 Then RCode = "Hardware Error"
    If pIPe.Status = 11009 Then RCode = "Packet too Big"
    If pIPe.Status = 11010 Then RCode = "Rqst Timed Out"
    If pIPe.Status = 11011 Then RCode = "Bad Request"
    If pIPe.Status = 11012 Then RCode = "Bad Route"
    If pIPe.Status = 11013 Then RCode = "TTL Exprd in Transit"
    If pIPe.Status = 11014 Then RCode = "TTL Exprd Reassemb"
    If pIPe.Status = 11015 Then RCode = "Parameter Problem"
    If pIPe.Status = 11016 Then RCode = "Source Quench"
    If pIPe.Status = 11017 Then RCode = "Option too Big"
    If pIPe.Status = 11018 Then RCode = " Bad Destination"
    If pIPe.Status = 11019 Then RCode = "Address Deleted"
    If pIPe.Status = 11020 Then RCode = "Spec MTU Change"
    If pIPe.Status = 11021 Then RCode = "MTU Change"
    If pIPe.Status = 11022 Then RCode = "Unload"
    If pIPe.Status = 11050 Then RCode = "General Failure"
    RCode = RCode + " (" + CStr(pIPe.Status) + ")"
    DoEvents
    If TraceRT = False Then
```

```
            If pIPe.Status = 0 Then
                Text3.Text = Text3.Text + "  Reply from " + RespondingHost + ": Bytes
= " + Trim$(CStr(pIPe.DataSize)) + " RTT = " + Trim$(CStr(pIPe.RoundTripTime))
+ "ms TTL = " + Trim$(CStr(pIPe.Options.TTL)) + Chr$(13) + Chr$(10)
            Else
                Text3.Text = Text3.Text + "  Reply from " + RespondingHost + ": "
+ RCode + Chr$(13) + Chr$(10)
            End If
        Else
            If TTL - 1 < 10 Then Text3.Text = Text3.Text + "  Hop # 0" + CStr(TTL
- 1) Else Text3.Text = Text3.Text + "  Hop # " + CStr(TTL - 1)
            Text3.Text = Text3.Text + "  " + RespondingHost + Chr$(13) + Chr$(10)
        End If
    End Sub

    Sub vbGetHostByName()
        Dim szString As String
        Host = Trim$(Text1.Text)              ' 获得主机地址
        szString = String(64, &H0)
        Host = Host + Right$(szString, 64 - Len(Host))
        If gethostbyname(Host) = SOCKET_ERROR Then            ' WSock32 错误
            sMsg = "Winsock Error" & Str$(WSAGetLastError())
            MsgBox sMsg, vbOKOnly, "VB4032-ICMPEcho"
        Else
            PointerToPointer = gethostbyname(Host)            ' 获得 winsock 结构地
址指针 Get the pointer to the address of the winsock hostent structure
            CopyMemory Hostent.h_name, ByVal _
            PointerToPointer, Len(Hostent)                   ' 将 Winsock 结构复制成
VB 结构
            ListAddress = Hostent.h_addr_list                ' 获得地址列表
            CopyMemory ListAddr, ByVal ListAddress, 4         ' 复制内存
            CopyMemory IPLong, ByVal ListAddr, 4             ' 获得 Address List
的第一个入口
            CopyMemory Addr, ByVal ListAddr, 4
            Label3.Caption    =    Trim$(CStr(Asc(IPLong.Byte4))    +    "."    +
CStr(Asc(IPLong.Byte3))              + "." + CStr(Asc(IPLong.Byte2)) + "." +
CStr(Asc(IPLong.Byte1)))
        End If
    End Sub
```

```
    Sub vbGetHostName()
        Host = String(64, &H0)          ' 获得一个空字符串
        If gethostname(Host, HostLen) = SOCKET_ERROR Then    '
            sMsg = "WSock32 Error" & Str$(WSAGetLastError())  ' 如果有 WSOCK32 错误
            MsgBox sMsg, vbOKOnly, "VB4032-ICMPEcho"
        Else
            Host = Left$(Trim$(Host), Len(Trim$(Host)) - 1)   ' 去掉结果种的空格
            Text1.Text = Host                                 ' 显示主机名
        End If
    End Sub
    Sub vbIcmpSendEcho()
        Dim NbrOfPkts As Integer
        szBuffer                                                                    =
"abcdefghijklmnopqrstuvwabcdefghijklmnopqrstuvwabcdefghijklmnopqrstuvwabcdefg
hijklmnopqrstuvwabcdefghijklmnopqrstuvwabcdefghijklm"
        '数据包大小
        szBuffer = Left$(szBuffer, 32)
        '包发送次数
        For NbrOfPkts = 1 To 5
            DoEvents
            bReturn = IcmpSendEcho(hIP, Addr, szBuffer, Len(szBuffer), pIPo, pIPe,
Len(pIPe) + 8, 2700)
            If bReturn Then
                RespondingHost = CStr(pIPe.Address(0)) + "." + CStr(pIPe.Address(1))
+ "." + CStr(pIPe.Address(2)) + "." + CStr(pIPe.Address(3))
                GetRCode
            Else       ' ICMP 超时
                      '
                If TraceRT Then
                    TTL = TTL - 1
                Else
                    Text3.Text = Text3.Text + "ICMP Request Timeout" + Chr$(13) +
Chr$(10)
                End If
            End If
        Next NbrOfPkts
    End Sub
```

```
    Sub vbWSAStartup()
        ' 初始化 Winsock
        iReturn = WSAStartup(&H101, WSAdata)
        If iReturn <> 0 Then
            MsgBox "WSock32.dll is not responding!", vbOKOnly, "VB4032-ICMPEcho"
        End If
        If LoByte(WSAdata.wVersion)<WS_VERSION_MAJOR Or (LoByte(WSAdata.wVersion)
= WS_VERSION_MAJOR And HiByte(WSAdata.wVersion) < WS_VERSION_MINOR) Then
            sHighByte = Trim$(Str$(HiByte(WSAdata.wVersion)))
            sLowByte = Trim$(Str$(LoByte(WSAdata.wVersion)))

            sMsg = "WinSock Version " & sLowByte & "." & sHighByte
            sMsg = sMsg & " is not supported "
            MsgBox sMsg, vbOKOnly, "VB4032-ICMPEcho"
            End
        End If
        If WSAdata.iMaxSockets < MIN_SOCKETS_REQD Then
            sMsg = "This application requires a minimum of "
            sMsg = sMsg & Trim$(Str$(MIN_SOCKETS_REQD)) & " supported sockets."
            MsgBox sMsg, vbOKOnly, "VB4032-ICMPEcho"
            End
        End If

        MaxSockets = WSAdata.iMaxSockets
        'WSAdata.iMaxSockets 是一个无符号 Short 类型，必须转化成为无符号 Long 型
        If MaxSockets < 0 Then
            MaxSockets = 65536 + MaxSockets
        End If
        MaxUDP = WSAdata.iMaxUdpDg
        If MaxUDP < 0 Then
            MaxUDP = 65536 + MaxUDP
        End If
        '  处理 Winsock 描述信息
        Description = ""
        For i = 0 To WSADESCRIPTION_LEN
            If WSAdata.szDescription(i) = 0 Then Exit For
            Description = Description + Chr$(WSAdata.szDescription(i))
        Next i
        '  处理 winsock 状态信息
```

```
    Status = ""
    For i = 0 To WSASYS_STATUS_LEN
        If WSAdata.szSystemStatus(i) = 0 Then Exit For
        Status = Status + Chr$(WSAdata.szSystemStatus(i))
    Next i
End Sub

Function HiByte(ByVal wParam As Integer)
    HiByte = wParam \ &H100 And &HFF&
End Function
Function LoByte(ByVal wParam As Integer)
    LoByte = wParam And &HFF&
End Function

Sub vbWSACleanup()
    ' 终止 winsock
    iReturn = WSACleanup()
    If iReturn <> 0 Then
        sMsg = "WSock32 Error - " & Trim$(Str$(iReturn)) & " occurred in Cleanup"
        MsgBox sMsg, vbOKOnly, "VB4032-ICMPEcho"
        End
    End If
End Sub

Sub vbIcmpCloseHandle()
    bReturn = IcmpCloseHandle(hIP)
    If bReturn = False Then
        MsgBox "ICMP Closed with Error", vbOKOnly, "VB4032-ICMPEcho"
    End If
End Sub

Sub vbIcmpCreateFile()
    hIP = IcmpCreateFile()
    If hIP = 0 Then
        MsgBox "Unable to Create File Handle", vbOKOnly, "VBPing32"
    End If
End Sub

Private Sub Command1_Click()
```

```
    vbWSAStartup            ' 初始化 winsock
    If Len(Text1.Text) = 0 Then
        vbGetHostName
    End If

    If Text1.Text = "" Then
        MsgBox "No Hostname Specified!", vbOKOnly, "VB4032-ICMPEcho"
        vbWSACleanup
        Exit Sub
    End If
    vbGetHostByName          ' 获得 IP 地址
    vbIcmpCreateFile         ' 获得 ICMP 句柄
    '设置 TTL 为 255
    pIPo.TTL = 255
    vbIcmpSendEcho           ' 发送 ICMP 回显请求
    vbIcmpCloseHandle        ' 关闭 ICMP 句柄
    vbWSACleanup             ' 关闭 winsock
End Sub
-------------------代码完毕----------------------
```

ICMP.bas 文件代码如下：

```
---------------------------代码-------------------------
' WSock32 UDTs

Type Inet_address
    Byte4 As String * 1
    Byte3 As String * 1
    Byte2 As String * 1
    Byte1 As String * 1
End Type

Public IPLong As Inet_address

Type WSAdata
    wVersion As Integer
    wHighVersion As Integer
    szDescription(0 To 255) As Byte
    szSystemStatus(0 To 128) As Byte
```

```
    iMaxSockets As Integer
    iMaxUdpDg As Integer
    lpVendorInfo As Long
End Type

Type Hostent
    h_name As Long
    h_aliases As Long
    h_addrtype As Integer
    h_length As Integer
    h_addr_list As Long
End Type

Type IP_OPTION_INFORMATION
    TTL As Byte                ' TTL
    Tos As Byte                ' 服务类型，通常为 0
    Flags As Byte              ' IP 标头，通常为 0
    OptionsSize As Long        ' 选择数据的大小，通常为 0，最大为 40
    OptionsData As String * 128   ' Options 数据缓冲区
End Type

Public pIPo As IP_OPTION_INFORMATION

Type IP_ECHO_REPLY
    Address(0 To 3) As Byte        ' 回复地址
    Status As Long                 ' 回复状态
    RoundTripTime As Long          ' 周期，单位是微秒
    DataSize As Integer            ' 回复数据大小
    Reserved As Integer            ' 系统保留字段
    data As Long                   ' 回显数据的指针
    Options As IP_OPTION_INFORMATION  ' 回复选项
End Type

Public pIPe As IP_ECHO_REPLY

' WSock32 API 函数

Declare Function gethostname Lib "wsock32.dll" (ByVal hostname$, HostLen&)
As Long
```

```
    Declare Function gethostbyname& Lib "wsock32.dll" (ByVal hostname$)
    Declare Function WSAGetLastError Lib "wsock32.dll" () As Long
    Declare Function WSAStartup Lib "wsock32.dll" (ByVal wVersionRequired&,
lpWSAData As WSAdata) As Long
    Declare Function WSACleanup Lib "wsock32.dll" () As Long

    ' Kernel32 API 函数

    Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (hpvDest As Any,
hpvSource As Any, ByVal cbCopy As Long)

    ' ICMP API 函数
       '
    '创建一个 ICMP 请求句柄
    Declare Function IcmpCreateFile Lib "icmp.dll" () As Long
    '关闭 ICMP 句柄
    Declare Function IcmpCloseHandle Lib "icmp.dll" (ByVal HANDLE As Long) As Boolean
       ' IcmpHandle 从函数 IcmpCreateFile 返回得到
       ' DestAddress 是指向地址列表的第一项的指针
       ' RequestData 是一个没有结束符的 64 字节的字符串，里面包含字符的 ASCII 为 170
       ' RequestSize 大小为 64 字节    ' RequestOptions is a NULL at this time
       ' ReplyBuffer
       ' ReplySize
       ' Timeout 为超时时间
    Declare Function IcmpSendEcho Lib "ICMP" (ByVal IcmpHandle As Long, ByVal
DestAddress As Long,    ByVal RequestData As String, ByVal RequestSize As Integer,
RequestOptns As IP_OPTION_INFORMATION,     ReplyBuffer As IP_ECHO_REPLY, ByVal
ReplySize As Long, ByVal TimeOut As Long) As Boolean
    ------------------------------代码完毕----------------------------
```

3.3 根据域名或者计算机名获取 IP 地址

获取 IP 地址在很多时候都是必须的，首先是作为网络的一个重要属性，必须告诉用户，同时在很多时候需要统计用户的 IP 地址等。还有一种情况就是在局域网或者广域网中，需要根据域名或者计算机名称来确定对方的 IP 地址。在本例题中就介绍如何使用 Winsock 控件和 Winsock API 来实现这些功能。

3.3.1　获取本机 IP 地址

获取本机的 IP 地址很简单，只需要通过 Winsock 控件就可以很容易地实现这个功能。

1.　实例原理

该实例很简单，因此不需要多作分析，读者只需要在窗体上面加上 1 个 Winsock 控件，然后直接利用 Winsock 控件的 LocalIP 属性就能够实现了。

程序的工作界面如图 3-7 所示。

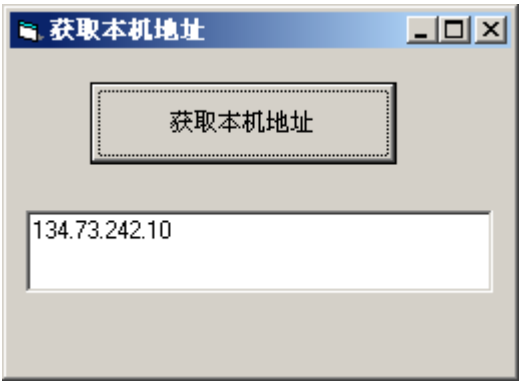


图 3-7　获取本机 IP 地址运行界面

2.　建立工程项目

该实例只有一个窗体，窗体上有 3 个控件、1 个按钮、1 个文本框和 1 个 Winsock 控件。设计界面如图 3-8 所示。

图 3-8　获取本机 IP 地址设计界面

3.　代码分析

本实例只有一个文件 getlocalip.frm，具体代码也非常简单，如下：

```
Private Sub Command1_Click()
Text1.Text = Winsock1.LocalIP
End Sub
```

3.3.2 根据域名或者计算机名获取 IP 地址

下面的实例读者会在开发过程中需要经常用到的，在局域网中经常需要根据计算机名来确定局域网内的 IP 地址，或者在广域网中需要根据域名来确定远程计算机的 IP 地址。

1. 实例原理

当用户单击按钮“获取”后，调用函数 GetIPAddress(TxtCmpName.Text)来获得 IP 地址，其中 TxtCmpName.Text 就是要获取 IP 地址的域名或者是计算机名。其中函数 GetIPAddress 调用了一系列的函数来实现这个功能，首先调用 lpHost = gethostbyname(sHostName)来获得指向一个结构体 lpHost 的指针，其中 sHostName 就是输入的域名；然后调用 CopyMemory 将 HOST.hAddrList 的 IP 地址复制到一个变量中，在通过一定的方法获取 IP 地址。

2. 建立工程项目

该程序都是通过 API 来实现的，所以设计起来很简单，只需要按钮和文本框就可以了。该程序的运行界面如图 3-9 所示。

图 3-9 根据域名获取 IP 地址的运行界面

3. 代码分析

本例程也只有一个窗体，文件名为 form1.frm，详见附赠光盘，具体代码如下：

```
Private Const SOCKET_ERROR As Long = -1
Private Const MAX_WSADescription = 256
Private Const MAX_WSASYSStatus = 128
Private Const ERROR_SUCCESS      As Long = 0
Private Const WS_VERSION_REQD     As Long = &H101
Private Const MIN_SOCKETS_REQD    As Long = 1
Private Const WS_VERSION_MAJOR    As Long = WS_VERSION_REQD \ &H100 And &HFF&
Private Const WS_VERSION_MINOR    As Long = WS_VERSION_REQD And &HFF&
```

```
    Private Type HOSTENT
        hName     As Long
        hAliases   As Long
        hAddrType  As Integer
        hLen      As Integer
        hAddrList  As Long
    End Type

    Private Type WSADATA
        wVersion     As Integer
        wHighVersion  As Integer
        szDescription(0 To MAX_WSADescription)   As Byte
        szSystemStatus(0 To MAX_WSASYSStatus)    As Byte
        wMaxSockets   As Integer
        wMaxUDPDG     As Integer
        dwVendorInfo  As Long
    End Type

    Private Declare Function gethostname Lib "WSOCK32.DLL" (ByVal szHost As String,
ByVal dwHostLen As Long) As Long
    Private Declare Function gethostbyname Lib "WSOCK32.DLL" (ByVal szHost As String)
As Long
    Private Declare Function WSAStartup Lib "WSOCK32.DLL" (ByVal wVersionRequired
As Long, lpWSADATA As WSADATA) As Long
    Private Declare Function WSAGetLastError Lib "WSOCK32.DLL" () As Long
    Private Declare Function WSACleanup Lib "WSOCK32.DLL" () As Long
    Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (hpvDest
As Any, ByVal hpvSource As Long, ByVal cbCopy As Long)

    Private Function GetIPAddress(Optional sHost As String) As String
    '返回给定计算机名的 Ip 地址，计算机名为空时返回本机 Ip 地址
        Dim sHostName   As String * 256
        Dim lpHost     As Long
        Dim HOST       As HOSTENT
        Dim dwIPAddr    As Long
        Dim tmpIPAddr() As Byte
        Dim i          As Integer
        Dim sIPAddr     As String
        Dim werr       As Long
```

```
If Not SocketsInitialize() Then
    GetIPAddress = ""
    Exit Function
End If

If sHost = "" Then
    If gethostname(sHostName, 256) = SOCKET_ERROR Then
        werr = WSAGetLastError()
        GetIPAddress = ""
        SocketsCleanup
        Exit Function
    End If

    sHostName = Trim$(sHostName)
Else
    sHostName = Trim$(sHost) & Chr$(0)
End If

lpHost = gethostbyname(sHostName)

If lpHost = 0 Then
    werr = WSAGetLastError()
    GetIPAddress = ""
    SocketsCleanup
    Exit Function
End If

CopyMemory HOST, lpHost, Len(HOST)
CopyMemory dwIPAddr, HOST.hAddrList, 4

ReDim tmpIPAddr(1 To HOST.hLen)
CopyMemory tmpIPAddr(1), dwIPAddr, HOST.hLen

For i = 1 To HOST.hLen
    sIPAddr = sIPAddr & tmpIPAddr(i) & "."
Next

GetIPAddress = Mid$(sIPAddr, 1, Len(sIPAddr) - 1)
```

```
    SocketsCleanup
End Function

Private Function SocketsInitialize(Optional sErr As String) As Boolean
    Dim WSAD As WSADATA, sLoByte As String, sHiByte As String
    If WSAStartup(WS_VERSION_REQD, WSAD) <> ERROR_SUCCESS Then
        sErr = "The 32-bit Windows Socket is not responding."
        SocketsInitialize = False
        Exit Function
    End If

    If WSAD.wMaxSockets < MIN_SOCKETS_REQD Then
        sErr = "This application requires a minimum of " & _
               CStr(MIN_SOCKETS_REQD) & " supported sockets."

        SocketsInitialize = False
        Exit Function
    End If



    If LoByte(WSAD.wVersion) < WS_VERSION_MAJOR Or _
           (LoByte(WSAD.wVersion) = WS_VERSION_MAJOR And _
           HiByte(WSAD.wVersion) < WS_VERSION_MINOR) Then

        sHiByte = CStr(HiByte(WSAD.wVersion))
        sLoByte = CStr(LoByte(WSAD.wVersion))

        sErr = "Sockets version " & sLoByte & "." & sHiByte & _
               " is not supported by 32-bit Windows Sockets."

        SocketsInitialize = False
        Exit Function
    End If
    SocketsInitialize = True
End Function

Private Sub SocketsCleanup()
    If WSACleanup() <> ERROR_SUCCESS Then
        App.LogEvent "Socket error occurred in Cleanup.", vbLogEventTypeError
```

```
    End If
End Sub

Private Function HiByte(ByVal wParam As Integer)
    HiByte = wParam \ &H1 And &HFF&
End Function

Private Function LoByte(ByVal wParam As Integer)
    LoByte = wParam And &HFF&
End Function

Private Sub Command1_Click()
    On Error Resume Next
    Screen.MousePointer = vbHourglass
    TxtIp.Text = GetIPAddress(TxtCmpName.Text)
    Screen.MousePointer = vbDefault
End Sub
-------------------------代码完毕-----------------------------
```

3.4 获取网卡地址

很多时候，程序中除了需要 IP 地址以外，还需要网卡的物理地址，实际上 IP 地址是计算机在网络中的地址，而要进行的数据传输，还必须要找到计算机的网卡物理地址才可以。网卡的物理地址在网卡出产的时候就会有一个全球唯一的号码，长度为 48 位。下面这个实例就是给出网卡的实际物理地址。

1. 实例原理

该实例是通过 API 函数 HeapAlloc 来实现的，通过该函数返回的值 pASTAT 传递给网络控制块结构体的 ncb_buffer 分量，即 myNcb.ncb_buffer = pASTAT，然后通过 Netbios 处理 myNcb 结构体即 Netbios(myNcb)，最后通过 CopyMemory 函数将 myNcb.ncb_buffer 复制给 myASTAT，myASTAT 是一个 ASTAT 的结构变量，通过分析以下几个部分：

myASTAT.adapt.adapter_address(0))
myASTAT.adapt.adapter_address(1))
myASTAT.adapt.adapter_address(2))
myASTAT.adapt.adapter_address(3))
myASTAT.adapt.adapter_address(4))
myASTAT.adapt.adapter_address(5))

可以获得网卡的地址。

2. 建立工程项目

程序设计非常简单，建一个窗体，然后在窗体上增加一个命令按钮和一个文本框就可以了。程序运行界面如图 3-10 所示。

图 3-10　Ping 程序设计界面

3. 代码分析

本实例只有一个文件 form1.frm，具体代码如下：

```
----------------------代码-----------------------------

Option Explicit
Private Const NCBASTAT = &H33
Private Const NCBNAMSZ = 16
Private Const HEAP_ZERO_MEMORY = &H8
Private Const HEAP_GENERATE_EXCEPTIONS = &H4
Private Const NCBRESET = &H32

Private Type NCB
  ncb_command As Byte ’Integer
  ncb_retcode As Byte ’Integer
  ncb_lsn As Byte ’Integer
  ncb_num As Byte ’ Integer
  ncb_buffer As Long ’String
  ncb_length As Integer
  ncb_callname As String * NCBNAMSZ
  ncb_name As String * NCBNAMSZ
  ncb_rto As Byte ’Integer
  ncb_sto As Byte ’ Integer
```

```
    ncb_post As Long
    ncb_lana_num As Byte ’Integer
    ncb_cmd_cplt As Byte ’Integer
    ncb_reserve(9) As Byte ’ Reserved, must be 0
    ncb_event As Long
  End Type

  Private Type ADAPTER_STATUS
    adapter_address(5) As Byte ’As String * 6
    rev_major As Byte ’Integer
    reserved0 As Byte ’Integer
    adapter_type As Byte ’Integer
    rev_minor As Byte ’Integer
    duration As Integer
    frmr_recv As Integer
    frmr_xmit As Integer
    iframe_recv_err As Integer
    xmit_aborts As Integer
    xmit_success As Long
    recv_success As Long
    iframe_xmit_err As Integer
    recv_buff_unavail As Integer
    t1_timeouts As Integer
    ti_timeouts As Integer
    Reserved1 As Long
    free_ncbs As Integer
    max_cfg_ncbs As Integer
    max_ncbs As Integer
    xmit_buf_unavail As Integer
    max_dgram_size As Integer
    pending_sess As Integer
    max_cfg_sess As Integer
    max_sess As Integer
    max_sess_pkt_size As Integer
    name_count As Integer
  End Type
  Private Type NAME_BUFFER
    name As String * NCBNAMSZ
    name_num As Integer
```

```
      name_flags As Integer
    End Type
    Private Type ASTAT
      adapt As ADAPTER_STATUS
      NameBuff(30) As NAME_BUFFER
    End Type

    Private Declare Function Netbios Lib "netapi32.dll" (pncb As NCB) As Byte
    Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (hpvDest
As Any, ByVal hpvSource As Long, ByVal cbCopy As Long)
    Private Declare Function GetProcessHeap Lib "kernel32" () As Long
    Private Declare Function HeapAlloc Lib "kernel32" (ByVal hHeap As Long, ByVal
dwFlags As Long, ByVal dwBytes As Long) As Long
    Private Declare Function HeapFree Lib "kernel32" (ByVal hHeap As Long, ByVal
dwFlags As Long, lpMem As Any) As Long

    Private Sub Command1_Click()
      Dim myNcb As NCB
      Dim bRet As Byte
      myNcb.ncb_command = NCBRESET
      bRet = Netbios(myNcb)
      myNcb.ncb_command = NCBASTAT
      myNcb.ncb_lana_num = 0
      myNcb.ncb_callname = "*      "
      Dim myASTAT As ASTAT, tempASTAT As ASTAT
      Dim pASTAT As Long
      myNcb.ncb_length = Len(myASTAT)
      Debug.Print Err.LastDllError
      pASTAT   =   HeapAlloc(GetProcessHeap(),   HEAP_GENERATE_EXCEPTIONS   Or
HEAP_ZERO_MEMORY, myNcb.ncb_length)
      If pASTAT = 0 Then
        Debug.Print "memory allcoation failed!"
        Exit Sub
      End If
      myNcb.ncb_buffer = pASTAT
      bRet = Netbios(myNcb)
      Debug.Print Err.LastDllError
      CopyMemory myASTAT, myNcb.ncb_buffer, Len(myASTAT)
      Text1.Text    =    Hex(myASTAT.adapt.adapter_address(0))    &    "    "    &
```

```
Hex(myASTAT.adapt.adapter_address(1)) _
       & " " & Hex(myASTAT.adapt.adapter_address(2)) & " " _
       & Hex(myASTAT.adapt.adapter_address(3)) _
       & " " & Hex(myASTAT.adapt.adapter_address(4)) & " " _
       & Hex(myASTAT.adapt.adapter_address(5))
     HeapFree GetProcessHeap(), 0, pASTAT
   End Sub
   --------------------------------代码完毕-------------------------------
```

3.5 增加超级链接和发送 E-mail

当鼠标指针指到窗体上某一个链接以后，单击就会自动链接到相关的程序，而且相关的参数已经传递进去，比如发送电子邮件给某人，会自动弹出默认的邮件客户端程序，而不需要输入对方的 E-mail 地址了，在 VB 中很容易就可以实现这些功能。

1. 实例原理

这个实例也很简单，主要是利用一个 API 函数来实现的，即 ShellExecute，通过设定不同的参数，可以把参数传给相应的应用程序，然后启动该程序进行编辑，除了能够实现这些功能，还可以通过设置参数实现把文件名传送给 Office 等编辑软件，由于这些和网络编程关系不大，所以不作详细介绍。

2. 建立工程项目

该程序只需要在窗体中增加两个标签控件就可以了，一个标签链接到浏览器，一个链接到邮件程序。注意，这里的链接和 E-mail 都是启动系统默认的程序来实现的。如默认的邮件客户端程序是 Outlook，则会启动 Outlook；程序运行界面如图 3-11 所示。

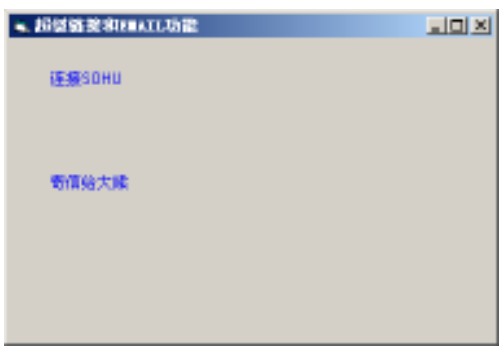


图 3-11 程序运行界面

图 3-12 为用鼠标单击“寄信给大熊”标签后，出现的界面，可以看出程序是自动连接到了 Outlook 的邮件发送程序。

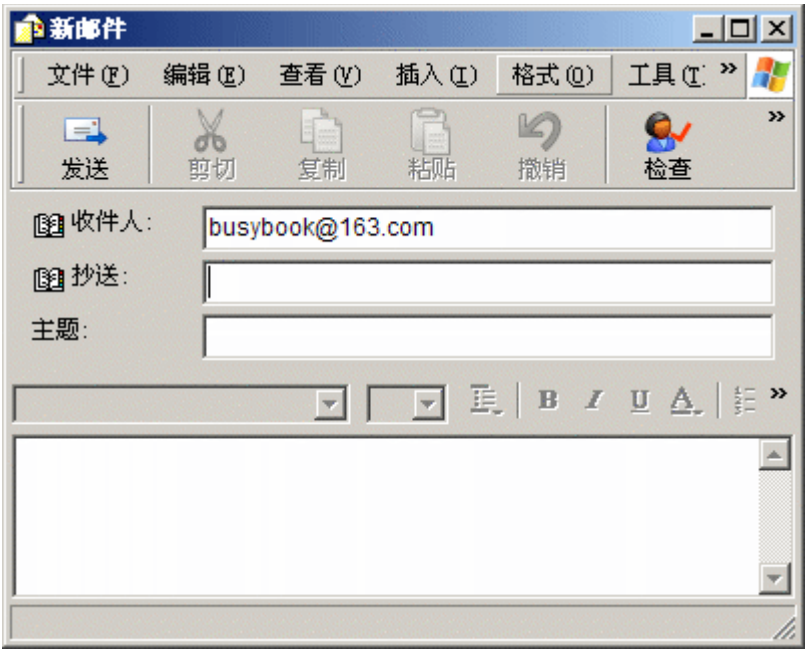


图 3-12　程序运行界面

3. 代码分析

该实例共有两个文件，分别是 form1.frm **和** winsock.bas**，其中** winsock.bas **在前面一章中有详细介绍，因此这里就只介绍** form1.frm **的代码，具体** form1.frm **代码如下：**

```
    ----------------------------代码--------------------------------
    Option Explicit
    Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA"
_
        (ByVal hWnd As Long, ByVal lpOperation As String, ByVal lpFile As String,
_
        ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd
As Long) As Long
    Private Const SW_SHOW = 5


    Private Sub Form_Load()
        Label1.Caption = "连接 SOHU"
        Label1.ForeColor = &HFF0000
        Label2.Caption = "寄信给大熊"
        Label2.ForeColor = &HFF0000
    End Sub
```

```
    Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Label1.Font.Underline = False
    Label2.Font.Underline = False

    End Sub

    Private Sub Label1_Click()
      Call ShellExecute(Me.hWnd, "open", "http://www.sohu.com", "", "", SW_SHOW)
    End Sub

    Private Sub Label1_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Label1.Font.Underline = True

    End Sub

    Private Sub Label2_Click()
      Call  ShellExecute(Me.hWnd,  "open",  "mailto:busywxp@163.com",  "",  "",
SW_SHOW)
    End Sub

    Private Sub Label2_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Label2.Font.Underline = True

    End Sub

    --------------------------------代码完毕---------------------------------
```

3.6 小结

本章主要介绍如何通过 VB 实现一些小的网络应用程序，这些应用程序都非常简单，因此代码中的注释并不多，读者也只需要稍微花点时间就能够看懂了。

第 4 章　TCP/UDP 应用开发

第 2 章重点介绍了 VB 网络通信编程的一些基本方法和途经，本章主要是想让初学者对 TCP/UDP 编程有一定的了解，通过一些简单的例题让读者了解一些 TCP/UDP 编程的区别，以及编程思路。以便为后续章节内容的学习作一个准备。

本章例题主要是围绕 TCP/UDP 编程来写的，主要有以下几个实例：

- Winsock API 实现 TCP 聊天程序
- Winsock API 实现 UDP 聊天程序
- Winsock 控件实现 TCP 聊天程序
- Winsock 控件实现 UDP 聊天程序

4.1　Winsock API 实现 TCP 聊天

在 VB 中实现聊天程序很简单，但是也最能够突出 TCP/IP 程序的核心，再复杂的例题也都是这些基本的应用演变过来的。目前大部分的网络应用都是基于 TCP 连接的，比如 E-mail、FTP、HTTP 等。

4.1.1　建立工程项目

本程序是通过 API 来实现的，因此不需要在窗体上增加 Winsock 控件。但是由于基于 TCP/IP 的程序都是基于客户机/服务器模式的，因此聊天网络程序总是要包含服务器程序和客户端程序的。

本程序中包含了服务器和客户端两个程序，其功能严格说来不是一个完整的聊天程序，服务器端只是把客户端发送出去的信息回显给客户端。

其中该程序一共包含 4 个文件：

- myclient.frm（窗体）
- myserver.frm（窗体）
- startup.bas（模块）
- Winsock.bas（模块）

从上面可以看出，有两个窗体，一个是服务器端的，一个是客户端的；还有两个模块，其中 startup.bas 模块主要是在启动的时候进行一些处理，而 Winsock.bas 模块，在第 2 章中已经介绍。

其中实现网络通信的 API 函数主要在 Winsock.bas。程序的运行结果如下，其中图 4-1 是程序运行的时候弹出来的对话框。

图 4-1　程序运行界面

在图 4-1 中，可以选择运行服务器端程序和客户端程序，当然首先需要选择运行服务器端程序，因此单击“是”按钮，弹出服务器端运行界面，如图 4-2 所示。

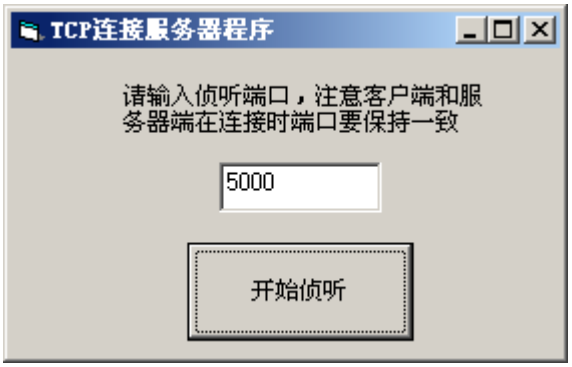


图 4-2　服务器端运行界面

用户要想使应用程序工作，首先必须启动服务器端程序，服务器程序运行后，需要单击“开始侦听”按钮来运行服务器端程序，否则客户端程序不能连接到服务器。

运行完服务器端程序以后，必须要运行客户端程序，在图 4-1 中，单击“否”按钮，就可以运行客户端程序，如图 4-3 所示。

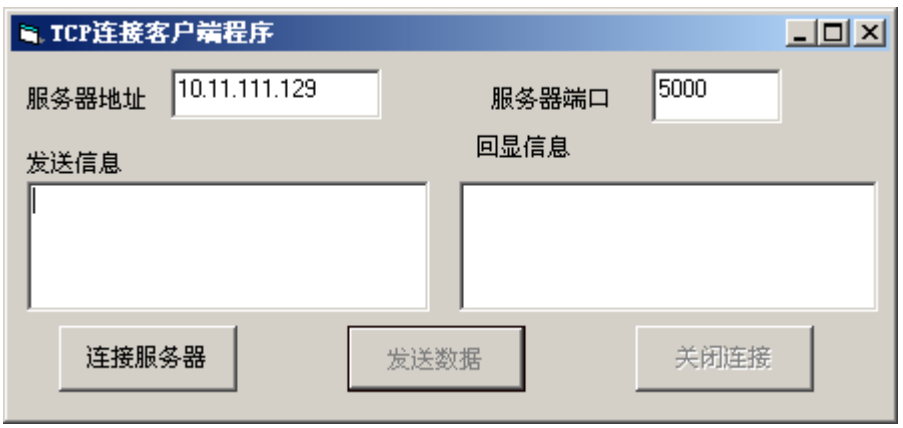


图 4-3　客户端程序运行界面

在图 4-3 中，要正确输入服务器地址以及服务器的侦听端口，这里服务器地址是“10.11.111.129”，服务器侦听端口是“5000”，同图 4-2 的设置是一样的，当一切准备好了以后，就可以单击“连接服务器”按钮，如果连接成功，则“发送数据”和“关闭连接”按钮则会变成可以使用状态，如图 4-4 所示。

图 4-4　客户端发送信息界面

在图 4-4 中，表示连接已经成功，然后在发送信息文本框中输入要发送的信息“ hello world ”，则服务器端也会把相同的信息也发送到客户端，如果是多用户聊天，其实很简单，只要把其他客户端发出的信息，通过服务器端转发给目的客户端就可以了。

4.1.2　代码分析

1. **窗体** myclient

```
Option Explicit

Dim soc As Long, dwRc As Long
Dim RemoteAddr As sockaddr
Dim RetMsg As String

'
' cmdSendRecv_Click 事件
'
'    该事件是处理按钮“发送”单击事件
'    当用户单击该按钮时，发送文本况中的信息将被
'    发送到服务器，并读取从服务器回显的数据。
Private Sub cmdSendRecv_Click()
    If soc = INVALID_SOCKET Then
        MsgBox "Create Socket First"
        Exit Sub
    End If

    RetMsg = String(7000, 0)
```

```
        '
        ' send data
        dwRc = send(soc, ByVal txtSend.Text, Len(txtSend.Text), 0)
        If dwRc = SOCKET_ERROR Then
            MsgBox "Couldn't send data to remote Socket. Error: " & Err.LastDllError
        Else
            ' receive data
            dwRc = recv(soc, ByVal RetMsg, 7000, 0)
            If dwRc = SOCKET_ERROR Then
                MsgBox "Couldn't recieve data from remote Socket. Error: " &
Err.LastDllError
            Else
                txtRecv.Text = Left(RetMsg, InStr(RetMsg, Chr(0)))
            End If
        End If
    End Sub

    '
    ' cmdConnect_Click 事件
    '    该事件是用户单击“连接”按钮的时候发生
    '    当该事件发生时，客户端尝试连接服务器端
    '    在连接服务器之前，首先要指定服务器地址和服务器端口号
    '
    Private Sub cmdConnect_Click()

        soc = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
        If soc = INVALID_SOCKET Then
            MsgBox "Couldn't Create Socket. Error: " & Err.LastDllError
        Else
            RemoteAddr.sin_family = AF_INET     ' address family, internet: 2
            RemoteAddr.sin_port = 5000 'htons(CLng(txtPort.Text))     'port 7, must
convert
            RemoteAddr.sin_addr = GetHostByNameAlias(txtServer.Text)

            'connect
            dwRc = connect(soc, RemoteAddr, sockaddr_size)

            If dwRc = SOCKET_ERROR Then
                MsgBox "Couldn't connect to remote Socket. Error: " & Err.LastDllError
```

```
        Else
            cmdConnect.Enabled = False
            cmdSendRecv.Enabled = True
            cmdDisconnect.Enabled = True
        End If

    End If
End Sub

' cmdDisconnect_Click 事件
'    该事件当单击“关闭连接”的时候发生
'    当用户单击该按钮时，连接被关闭
Private Sub cmdDisconnect_Click()
    txtSend.Text = ""
    txtRecv.Text = ""
    If soc <> INVALID_SOCKET Then closesocket (soc)
    cmdConnect.Enabled = True
    cmdSendRecv.Enabled = False
    cmdDisconnect.Enabled = False
    soc = INVALID_SOCKET
End Sub

' Form_Load 事件
'    载入窗体事件，该事件确保 Winsock DLL
'    被正确地载入系统
Private Sub Form_Load()
    If TCPIPStartup Then
        cmdSendRecv.Enabled = True
    Else
        MsgBox "Windows Sockets not initialized. Error: " & Err.LastDllError
    End If
    soc = INVALID_SOCKET
    cmdConnect.Enabled = True
    cmdSendRecv.Enabled = False
    cmdDisconnect.Enabled = False
End Sub

' Form_Unload 事件
'    卸载窗体事件，该事件中卸载 Winsock DLL
```

```
Private Sub Form_Unload(Cancel As Integer)
    TCPIPShutDown
End Sub
```

2. **窗体** myserver

程序说明：

该程序包含了一个 TCP **服务器程序和客户端程序。服务器端创建了一个** TCP **侦听** Socket **并等待客户端连接。一旦连接建立成功，服务器端接受客户端的程序，并把接受到的数据返回到客户端。**

服务器端使用了阻塞式 Sockets **来接受和发送数据。这意味着服务器端程序一旦建立，用户界面不能被刷新。**

客户端程序非常简单，用户确定服务器名字和端口号，然后就可以连接服务器了。用户只要在发送数据文本框中输入文字，并单击“发送”按钮，就可以在显示文本框中读取回显数据。

```
Option Explicit
'窗体载入事件
'该事件中，要确定 Winsock DLL 已经被正确
'的载入
'
Sub Form_Load()
    If TCPIPStartup Then
        cmdStart.Enabled = True
    Else
        MsgBox "Windows Socketsb 不能被初始化：" & Err.LastDllError
    End If
End Sub

'Form_Unload 窗体卸载事件
'卸载掉 Winsock DLL
Private Sub Form_Unload(Cancel As Integer)
    TCPIPShutDown
End Sub

' CmdStart_Click 事件
'该事件中启动服务器程序，程序首先创建了一个 socket，该 socket 被绑定到
'一个指定的端口，然后等待客户端的连接。连接建立完毕，便等待
'客户端的数据发送，然后把接受到的数据回显给客户端。
```

```
    '注意采用的式阻塞式 socket。
    '
    Private Sub CmdStart_Click()
       Dim addr As sockaddr, client_addr As sockaddr, addrlen As Long, client_addrlen
As Long, pSockAddr As Long
       Dim msg_sock As Long, accept_sock As Long, msgstr As String
       Dim i As Long

       cmdStart.Enabled = False
       '获取 socket 句柄
       accept_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
       If accept_sock = INVALID_SOCKET Then
          MsgBox "Couldn't create socket(). Error:" & Err.LastDllError, "Accept"
          Exit Sub
       End If
       '
       '设定 socket 端口
       addr.sin_family = AF_INET
       addr.sin_port = 5000 'htons(CLng(txtPort.Text))
       addr.sin_addr = INADDR_ANY

       '绑定到 socket 上
       If bind(accept_sock, addr, Len(addr)) = SOCKET_ERROR Then
          MsgBox "Couldn't bind() to socket. Error: " & Err.LastDllError
          closesocket (accept_sock)
          Exit Sub
       End If

       '开始侦听
       If listen(accept_sock, 1) = SOCKET_ERROR Then
          MsgBox "Couldn't listen() to socket. Error: " & Err.LastDllError
          closesocket (accept_sock)
          Exit Sub
       End If

       client_addrlen = Len(client_addr)
       '
       ' 等待客户端连接
       msg_sock = accept(accept_sock, client_addr, client_addrlen)
```

```
      If msg_sock = INVALID_SOCKET Then
         MsgBox "Couldn't accept an socket connection. Error: " & Err.LastDllError
         closesocket (accept_sock)
         Exit Sub
      End If

      Dim RetMsg As String '循环接收和发送知道其他程序关闭 socket
      Dim dwRc As Long
      'DoEvents

      Do
       RetMsg = String(7000, 0)
       dwRc = recv(msg_sock, ByVal RetMsg, 7000, 0)
       If dwRc = SOCKET_ERROR Then
           MsgBox "Couldn't recieve data from remote Socket. Error: " &
Err.LastDllError
           closesocket (accept_sock)
           closesocket (msg_sock)
           cmdStart.Enabled = True
           Exit Sub
        End If
        '
        ' 读取数据，并返回数据到客户端
        dwRc = send(msg_sock, ByVal RetMsg, Len(RetMsg), 0)
        If dwRc = SOCKET_ERROR Then
           MsgBox "Couldn't send data to remote Socket. Error: " & Err.LastDllError
           closesocket (accept_sock)
           closesocket (msg_sock)
           cmdStart.Enabled = True
           Exit Sub
        End If
       Loop
       closesocket (msg_sock)
       closesocket (accept_sock)
       cmdStart.Enabled = True
   End Sub
```

3. **模块** startup

```
Option Explicit
'启动程序 Main
'该程序在主窗体载入前执行，主要是让用户选择是启动服务器程序还是客户端程序
Sub Main()
    Dim dwRet As Long
    dwRet = MsgBox("单击""是""启动服务器进程，单击""否""启动客户端程序。", vbYesNo)
    If dwRet = vbYes Then
        Load myserver
        myserver.Show
    Else
        Load myclient
        myclient.Show
    End If
End Sub

'子程序 VBntoaVers
'该函数用来获取 Winsock 版本
Function VBntoaVers(ByVal vers As Long) As String
    Dim szVers As String
    szVers = String(5, 0)
    szVers = (vers And &HFF) & "." & ((vers And &HFF00) / 256)
    VBntoaVers = szVers
End Function

'子程序 hibyte
'该函数返回一个整数地高位字节
Function hibyte(ByVal wParam As Integer)
   hibyte = wParam \ &H100 And &HFF&
End Function

'字函数 lobyte
'该函数返回一个整数地低位字节
Function lobyte(ByVal wParam As Integer)
   lobyte = wParam And &HFF&
End Function
```

4. **模块** Winsock.bas

该模块在第 2 章中有详细介绍，主要是各种网络 API 的声明。

4.2 Winsock API 实现 UDP 聊天

在第 2 章中已经解释了 TCP 和 UDP 两种不同方式的网络传输方式，相对于 TCP 连接方式而言，UDP 则显得容易了很多。原则上并没有服务器端和客户端的区别，只有发送端和接受端。

UDP 是无连接协议，与 TCP 操作不同，计算机间并不需要建立连接，也就是说无须传递握手信号，也不用知道对方主机是否在网上，只要知道对方主机的 IP 地址就可以传递数据。

UDP 协议的另一个特点就是它的每一个应用程序可同时作为客户方和服务器方。由于 UDP 协议并不需要建立一个明确的连接，因此建立 UDP 应用要比建立 TCP 应用简单得多。在 TCP 应用中，作为服务方的 Socket 必须明确地设置成监听（Listen）模式，而其它 Socket 则必须使用 Connect 方法来初始化一个连接。在使用 UDP 协议时就可以省略这些过程（但是在建立的时候需要 Bind 一个端口）。

UDP 协议的母体和 TCP 协议一样，也是 IP 协议，它的数据流结构依次包括 IP 头、UDP 头、数据三部分。在 IP 头里包括发送方和接收方的 IP 地址、端口和使用的协议类型。UDP 头里包括发送方和接收方的端口。

4.2.1 建立工程项目

该程序只有两个文件，一个是 myudp.frm，另外一个是模块文件 Winsock.bas。

由于是通过 API 函数来实现的，因此，窗体上不需要增加 Winsock 控件，只需要增加一些普通的控件，然后增加一个定时器控件就可以了。

该程序同前面的一个程序也是一样的，发送端和客户端都集成在一个程序中，这样可以对比学习，同时也较少了一些无关的代码。

程序运行界面如图 4-5 所示。

图 4-5 UDP 聊天程序运行界面

要测试或者运用本程序，只需要单击“绑定到本地端口”按钮，然后如果绑定成功，“发送信息”和“关闭连接”按钮就可以变成使用状态。注意远程服务器地址和远程端口要设置

正确，因为对于 UDP 而言，本地端口并不重要，但是接受数据的远程计算机端口和地址必须设置清楚，否则不能正确地使用。由于本程序是接受端和发送端都集成一起的，所以在一个界面上就可以测试。

在图 4-5 上单击“绑定到本地端口”按钮以后，由于绑定成功，因此在发送数据文本框中输入“hello world”，则在接受数据文本框中，也会显示同样的字符串，如图 4-6 所示。

图 4-6　UDP 聊天界面

如果其他计算机上运行该程序，只要输入正确的服务器地址和远程端口好，也可以发送数据到远程计算机。图 4-7 是在另外一台机器上运行的该程序，其中远程端口要设置为“5000”，地址要设置为“134.73.242.10”。

图 4-7　其他机器的运行的 UDP 程序发送消息

则运行在本机的 UDP 程序就会接受到其他机器发送来的信息，如图 4-8 所示。

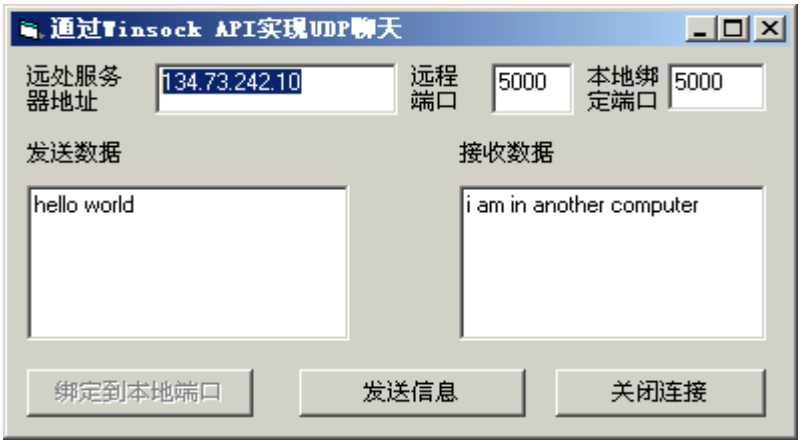


图 4-8　接受到其他机器发送的信息

4.2.2　代码分析

窗体 myudp

```
Option Explicit

Dim msg_sock As Long, msgstr As String
Dim addr As sockaddr, remote_addr As sockaddr, from_addr As sockaddr
Dim fromlen As Long
Dim hEvent As Long
Dim dwRet As Long, dwyes As Long, dwRc As Long
'
' "关闭连接"按钮单击事件
'程序说明：
'主要是清空文本框内容，同时设置相关按钮的 enable 属性
'并关闭定时器
Private Sub cmdClose_Click()
    If msg_sock <> INVALID_SOCKET Then
        closesocket msg_sock
    End If
    txtRecv.Text = ""
    txtSend.Text = ""
    cmdBind.Enabled = True
    cmdSend.Enabled = False
    cmdClose.Enabled = False
```

```
        Timer1.Enabled = False
    End Sub

    '"发送信息"按钮单击事件
    '该事件主要发送 UDP 数据到远程主机和端口
    Private Sub cmdSend_Click()
        '建立远程主机和端口
        remote_addr.sin_family = AF_INET    '
        remote_addr.sin_port = htons(CLng(txtRemotePort.Text))
        remote_addr.sin_addr = GetHostByNameAlias(txtRemotePeer.Text)

        '发送数据
        dwRet = sendto(msg_sock, ByVal txtSend.Text, Len(txtSend.Text), 0,
remote_addr, LenB(remote_addr))
        If dwRet = SOCKET_ERROR Then
            MsgBox "sendto failed. Error: " & Err.LastDllError
        End If
    End Sub

    '"绑定到本地端口"按钮单击事件
    '该事件就是把 Winsock 绑定到指定的本地端口上，这样是为了在该端口能够接受数据

    Private Sub CmdBind_Click()
        '获得 Winsock 句柄
        msg_sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)
        If msg_sock = INVALID_SOCKET Then
            MsgBox "Couldn't create socket(). Error: " & Err.LastDllError
        Else
            ' 建立本地地址
            addr.sin_family = AF_INET
            addr.sin_port = htons(CLng(txtLocalPort.Text))
            addr.sin_addr = INADDR_ANY

            dwyes = 1
            '设定是否能多个进程绑定到同一个端口
            dwRet = setsockopt(msg_sock, SOL_SOCKET, SO_REUSEADDR, dwyes,
LenB(dwyes))
            If dwRet = SOCKET_ERROR Then
                MsgBox "SO_REUSERADDR failed. Error: " & Err.LastDllError
```

```
        End If
        dwyes = 1
        '设定是否进行广播
        dwRet = setsockopt(msg_sock, SOL_SOCKET, SO_BROADCAST, dwyes,
LenB(dwyes))
        If dwRet = SOCKET_ERROR Then
            MsgBox "SO_BROADCAST failed. Error: " & Err.LastDllError
        End If

        '绑定到 socket
        If bind(msg_sock, addr, Len(addr)) = SOCKET_ERROR Then
           MsgBox "Couldn't bind() to socket locally. Error: " & Err.LastDllError
           closesocket (msg_sock)
           msg_sock = INVALID_SOCKET
        Else
           cmdSend.Enabled = True
           cmdBind.Enabled = False
           cmdClose.Enabled = True
           Timer1.Enabled = True
        End If
     End If
  End Sub

  'Form_Load 事件
  '窗体载入时初始化 Winsock
  Private Sub Form_Load()
     If TCPIPStartup Then
        cmdBind.Enabled = True
        cmdSend.Enabled = False
        cmdClose.Enabled = False
        Timer1.Enabled = False
     Else
        MsgBox "Windows Sockets not initialized. Error: " & Err.LastDllError
     End If
     '
     '创建一个事件用来获得数据
     msg_sock = INVALID_SOCKET
     hEvent = WSACreateEvent
     If hEvent = 0 Then
```

```
        MsgBox "Failed to create event. Error: " & Err.LastDllError
    End If

End Sub

'窗体卸载事件
'窗体卸载事件卸载 Winsock 相关资源
Private Sub Form_Unload(Cancel As Integer)
    If msg_sock <> INVALID_SOCKET Then
        closesocket msg_sock
    End If
    TCPIPShutDown
    If hEvent <> 0 Then WSACloseEvent hEvent
End Sub

'
'定时器事件

'这始一个回调函数，当启动定时器时，将触发事件 FD_READ
'然后调用 WSAWaitForMultipleEvents 决定是否有数据到达
'如果没有，则退出程序，有的话则调用 recvfrom 函数来读取数据报
Private Sub Timer1_Timer()
    If msg_sock = INVALID_SOCKET Then
        Debug.Print "Create Socket First"
        Exit Sub
    End If

    Dim RetMsg As String

    RetMsg = String(7000, 0)
    dwRc = WSAEventSelect(msg_sock, hEvent, FD_READ)
    If (dwRc = SOCKET_ERROR) Then
        MsgBox "Failed to select event. Error: " & Err.LastDllError
        Exit Sub
    End If
    '
    '查看是否有数据可以获取，不要使用阻塞模式
    '如果等待参数为 0 则立即返回
    dwRc = WSAWaitForMultipleEvents(1, hEvent, False, 0, False)
```

```
        Select Case dwRc
        Case WSA_WAIT_TIMEOUT
            Debug.Print "Recv timed out"

        Case WSA_WAIT_EVENT_0
            Dim NetworkEvents As WSANETWORKEVENTS
            Dim dwRet As Long
            NetworkEvents.lNetWorkEvents = 0
            dwRet = WSAEnumNetworkEvents(msg_sock, hEvent, NetworkEvents)
            If (dwRet = SOCKET_ERROR) Then
                MsgBox "WSAEnumNetworkEvents failed to select event. Error: " &
Err.LastDllError
            Else
                If (FD_READ And NetworkEvents.lNetWorkEvents) Then
                    fromlen = LenB(from_addr)
                    dwRc = recvfrom(msg_sock, ByVal RetMsg, 7000, 0, from_addr,
fromlen)
                    If dwRc = SOCKET_ERROR Then
                        MsgBox "Couldn't recieve data from remote Socket. Error: "
& Err.LastDllError
                    Else
                        txtRecv.Text = Left(RetMsg, InStr(RetMsg, Chr(0)))
                    End If
                End If
            End If
        Case WSA_WAIT_FAILED
            MsgBox "WSAWaitForMultipleEvents failed. Error: " & Err.LastDllError
        Case Else
            MsgBox "Unexpected WSAWaitForMultipleEvents return. Error: " &
Err.LastDllError
        End Select

        dwRc = WSAEventSelect(msg_sock, hEvent, 0)
        If (dwRc = SOCKET_ERROR) Then
            MsgBox "Failed to select event. Error: " & Err.LastDllError
            Exit Sub
        End If
    End Sub
```

4.3　Winsock 控件实现 TCP 聊天

前面都是通过 API 函数来实现“聊天”功能的，相对而言，通过 API 函数来实现难度较大，但是灵活性高很多，对于一些要求不高的 TCP/IP 程序，可以直接通过 Winsock 控件来实现,本节通过具体实例来让读者更深入地了解利用 Winsock 控件进行应用开发的工作流程。

4.3.1　建立工程项目

该程序的设计界面如图 4-9 所示。

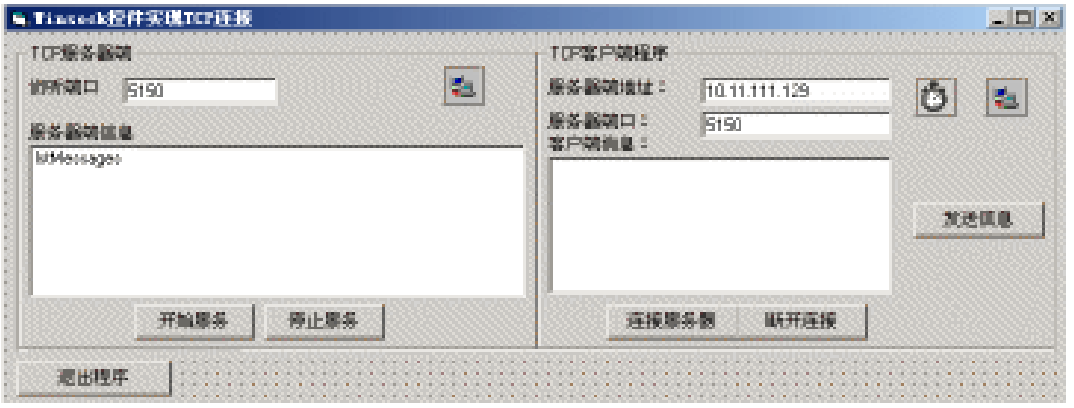


图 4-9　TCP　程序的设计界面

从图 4-9 中可以看出，设计界面上有两个 Winsock 控件。一个用于服务器端的侦听，一个用于客户端的连接。

要运行该程序，首先要启动服务器端的服务，单击“开始服务”按钮，服务器便会在端口“5150”侦听，端口可以自己设定；客户端程序首先要连接服务器端，所以当服务器端开始服务以后，单击“连接服务器”按钮，注意端口的设置要一致，如图 4-10 所示。

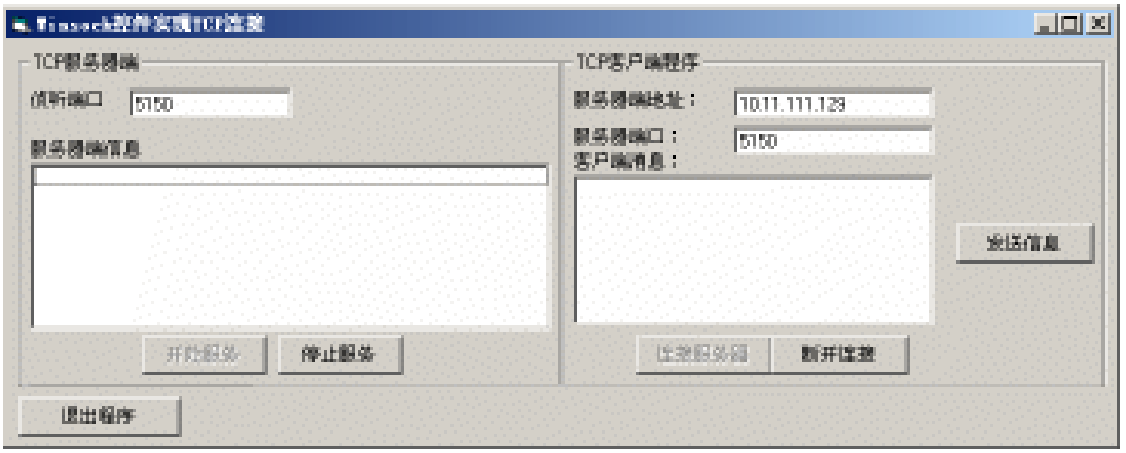


图 4-10　发送信息界面

从图 4-10 可以看出，连接通以后，客户端发出“hello”的信息以后，服务器端会显示出

相应的消息。

4.3.2 代码分析

本实例也是集成客户端和服务器端与一体的，由于使用了 Winsock 控件，所以不需要额外的 API 函数声明，直接使用一个窗体就可以了。窗体名称为 mytcp。

窗体 mytcp

```
Option Explicit

' 该变量表示第几个 socket 被载入
Private ServerIndex As Long

Private Sub cmdCloseListen_Click()
    Dim itemx As Object
    '停止服务器端侦听
    sockServer(0).Close
    cmdListen.Enabled = True
    cmdCloseListen.Enabled = False

    Set itemx = lstStates.ListItems.Item(2)
    itemx.SubItems(2) = "-1"
End Sub

Private Sub cmdConnect_Click()
    ' 客户端程序连接服务器
    '
    sockClient.LocalPort = 0
    sockClient.RemoteHost = txtServerName.Text
    sockClient.RemotePort = CInt(txtPort.Text)
    sockClient.Connect

    cmdConnect.Enabled = False
End Sub

Private Sub cmdDisconnect_Click()
    Dim itemx As Object
    '关闭客户端连接
    '
    sockClient.Close
```

```
    cmdConnect.Enabled = True
    cmdSendData.Enabled = False
    cmdDisconnect.Enabled = False
    ' 设定端口号为-1 表示没有连接
    '
    Set itemx = lstStates.ListItems.Item(1)
    itemx.SubItems(2) = "-1"
End Sub

Private Sub cmdExit_Click()
    Unload Me
End Sub

Private Sub cmdListen_Click()
    Dim itemx As Object
    ' 将服务器端控件设置为侦听模式
    '
    sockServer(0).LocalPort = CInt(txtServerPort.Text)
    sockServer(0).Listen

    Set itemx = lstStates.ListItems.Item(2)
    itemx.SubItems(2) = sockServer(0).LocalPort

    cmdCloseListen.Enabled = True
    cmdListen.Enabled = False
End Sub

Private Sub cmdSendData_Click()
    ' 如果连接成功，则发送数据到服务器
    '
    If (sockClient.State = sckConnected) Then
        sockClient.SendData txtSendData.Text
    Else
        MsgBox "以外错误，连接关闭!"
        Call cmdDisconnect_Click
    End If
End Sub
```

```
Private Sub Form_Load()
    Dim itemx As Object

    lblLocalHostname.Caption = sockServer(0).LocalHostName
    lblLocalHostIP.Caption = sockServer(0).LocalIP

    ' 初始化协议，设定为 TCP 协议
    '
    ServerIndex = 0
    sockServer(0).Protocol = sckTCPProtocol
    sockClient.Protocol = sckTCPProtocol
    ' 设置按钮相关属性
    '
    cmdDisconnect.Enabled = False
    cmdSendData.Enabled = False
    cmdCloseListen.Enabled = False
    ' 初始化 ListView 控件，让她包含目前所有 Winsock 控件的状态
    '
    Set itemx = lstStates.ListItems.Add(1, , "Local Client")
    itemx.SubItems(1) = "sckClosed"
    itemx.SubItems(2) = "-1"
    Set itemx = lstStates.ListItems.Add(2, , "Local Server")
    itemx.SubItems(1) = "sckClosed"
    itemx.SubItems(2) = "-1"
    ' 初始化定时器，它控制状态的刷新
    '
    Timer1.Interval = 500
    Timer1.Enabled = True
End Sub


Private Sub sockClient_Close()
    sockClient.Close
End Sub

Private Sub sockClient_Connect()
    Dim itemx As Object

    ' 如果连接成功，则使得按钮发送数据可以使用
```

```
    cmdSendData.Enabled = True
    cmdDisconnect.Enabled = True

    Set itemx = lstStates.ListItems.Item(1)
    itemx.SubItems(2) = sockClient.LocalPort
End Sub

Private Sub sockClient_Error(ByVal Number As Integer, _
        Description As String, ByVal Scode As Long, _
        ByVal Source As String, ByVal HelpFile As String, _
        ByVal HelpContext As Long, CancelDisplay As Boolean)
    ' 客户端连接出现错误的时候，弹出错误信息
    ' 并关闭连接

    MsgBox Description
    sockClient.Close
    cmdConnect.Enabled = True
End Sub

Private Sub sockServer_Close(index As Integer)
    Dim itemx As Object
    ' 关闭控件
    '
    sockServer(index).Close
    Set itemx = lstStates.ListItems.Item(index + 2)
    lstStates.ListItems.Item(index + 2).Text = "---.---.---.---"
    itemx.SubItems(2) = "-1"

End Sub
Private Sub sockServer_ConnectionRequest(index As Integer, _
        ByVal requestID As Long)
    Dim i As Long, place As Long, freeSock As Long, itemx As Object
    ' 寻找是否有空闲的关闭 Winsock 控件可以使用
    freeSock = 0
    For i = 1 To ServerIndex
        If sockServer(i).State = sckClosed Then
            freeSock = i
            Exit For
        End If
```

```
    Next i
    ' 如果没有空闲的控件，则重新创建一个新的 Winsock
    ' so load a new one
    '
    If freeSock = 0 Then
        ServerIndex = ServerIndex + 1
        Load sockServer(ServerIndex)

        sockServer(ServerIndex).Accept requestID
        place = ServerIndex
    Else
        sockServer(freeSock).Accept requestID
        place = freeSock
    End If
    ' 在 ListView 中增加新 Winsock 控件的状态
     If freeSock = 0 Then
        Set itemx = lstStates.ListItems.Add(, , _
            sockServer(ServerIndex).RemoteHostIP)
    Else
        Set itemx = lstStates.ListItems.Item(freeSock + 2)
        lstStates.ListItems.Item(freeSock + 2).Text = _
            sockServer(freeSock).RemoteHostIP
    End If
    itemx.SubItems(2) = sockServer(place).RemotePort
    End Sub

Private Sub sockServer_DataArrival(index As Integer, _
        ByVal bytesTotal As Long)
    Dim data As String, entry As String
     ' 创建一个大的缓冲区以存放数据
    data = String(bytesTotal + 2, Chr$(0))
    sockServer(index).GetData data, vbString, bytesTotal
    '
    entry = sockServer(index).RemoteHostIP & ": " & data
    lstMessages.AddItem entry
End Sub

Private Sub sockServer_Error(index As Integer, _
        ByVal Number As Integer, Description As String, _
```

```
        ByVal Scode As Long, ByVal Source As String, _
        ByVal HelpFile As String, ByVal HelpContext As Long, _
        CancelDisplay As Boolean)
    ' 错误处理
    MsgBox Description
    sockServer(index).Close
End Sub

Private Sub Timer1_Timer()
    Dim i As Long, index As Long, itemx As Object

    ' 定时刷新各个 Winsock 控件的状态
    '
    Set itemx = lstStates.ListItems.Item(1)
    Select Case sockClient.State
        Case sckClosed
            itemx.SubItems(1) = "sckClosed"
        Case sckOpen
            itemx.SubItems(1) = "sckOpen"
        Case sckListening
            itemx.SubItems(1) = "sckListening"
        Case sckConnectionPending
            itemx.SubItems(1) = "sckConnectionPending"
        Case sckResolvingHost
            itemx.SubItems(1) = "sckResolvingHost"
        Case sckHostResolved
            itemx.SubItems(1) = "sckHostResolved"
        Case sckConnecting
            itemx.SubItems(1) = "sckConnecting"
        Case sckConnected
            itemx.SubItems(1) = "sckConnected"
        Case sckClosing
            itemx.SubItems(1) = "sckClosing"
        Case sckError
            itemx.SubItems(1) = "sckError"
        Case Else
            itemx.SubItems(1) = "unknown: " & sockClient.State
    End Select
    ' 当有客户连接时，设置服务器端状态
```

```
    '   index = 0
    For i = 2 To ServerIndex + 2
       Set itemx = lstStates.ListItems.Item(i)
        Select Case sockServer(index).State
          Case sckClosed
             itemx.SubItems(1) = "sckClosed"
          Case sckOpen
             itemx.SubItems(1) = "sckOpen"
          Case sckListening
             itemx.SubItems(1) = "sckListening"
          Case sckConnectionPending
             itemx.SubItems(1) = "sckConnectionPending"
          Case sckResolvingHost
             itemx.SubItems(1) = "sckResolvingHost"
          Case sckHostResolved
             itemx.SubItems(1) = "sckHostResolved"
          Case sckConnecting
             itemx.SubItems(1) = "sckConnecting"
          Case sckConnected
             itemx.SubItems(1) = "sckConnected"
          Case sckClosing
             itemx.SubItems(1) = "sckClosing"
          Case sckError
             itemx.SubItems(1) = "sckError"
          Case Else
             itemx.SubItems(1) = "unknown"
       End Select
       index = index + 1
    Next i
End Sub
```

4.4 Winsock 控件实现 UDP 聊天

同上一个应用一样，该例题是通过 Winsock 控件来实现 UDP 聊天程序的。因为几乎所有的应用程序都是基于 TCP/UDP 来实现的，所以本章主要从各个方面来阐述这些知识和用法。后面的例题都是综合应用，而且本书的重心不在于如何使用这些简单的控件，而是如何结合协议以及相关的理论来开发高级程序。仍然继续前几个例题的风格，发送端和接收端程

序都集成在一个应用程序里面。

4.4.1　建立工程项目

本实例也只有一个窗体，同时集成了接收端和发送端，其设计界面如图 4-11 所示。

图 4-11　实例设计界面

从图 4-11 可以了解到，一共用了两个 Winsock 控件，一个用于发送，一个用于接收。当程序运行起来以后，首先要设定接收端和发送端的一些参数。对于发送端而言，必须知道接收端的地址以及接收端的侦听端口，本地绑定端口可以随便选取一个不常用的即可。

而对于接收端，则需要设定侦听端口。一切设定好以后，只需要在发送端发送信息“ hello,world ”，则接收端会显示出相同的信，如图 4-12 所示。

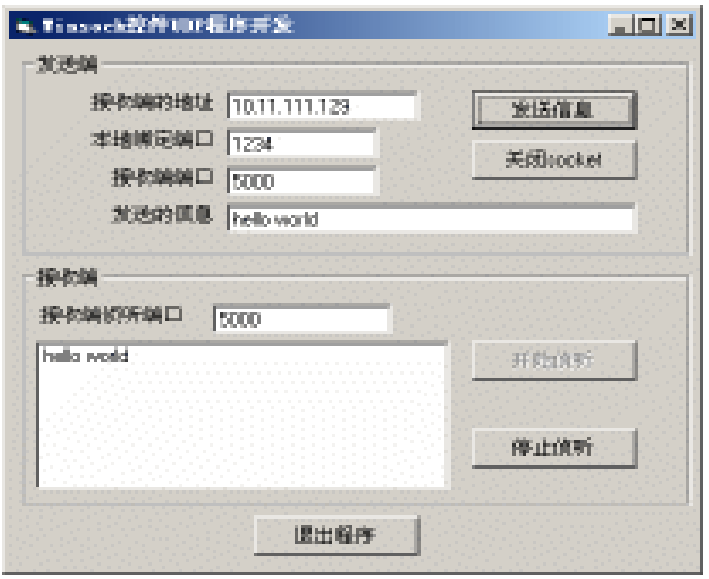


图 4-12　UDP 发送消息

4.4.2 代码分析

窗体 udp

```
Option Explicit

Private Sub cmdExit_Click()
    Unload Me
End Sub

Private Sub cmdSendDgram_Click()
    ' 绑定本地端口，然后发送数据
    If (sockSend.State = sckClosed) Then
        sockSend.RemoteHost = txtRecipientIP.Text
        sockSend.RemotePort = CInt(txtSendRemotePort.Text)
        sockSend.Bind CInt(txtSendLocalPort.Text)

        cmdCloseSend.Enabled = True
    End If
    '
    ' 现在发送数据
    '
    sockSend.SendData txtSendData.Text
End Sub
'"开始侦听"按钮单击事件
Private Sub cmdListen_Click()
    ' 绑定到本地端口
    sockRecv.Bind CInt(txtRecvLocalPort.Text)
    ' 改变控件的 enable 属性，因为单击侦听按钮两次会出现错误
    cmdListen.Enabled = False
    cmdCloseListen.Enabled = True
End Sub

Private Sub cmdCloseSend_Click()
    ' 关闭发送 socket，然后 disable 发送信息按钮
    sockSend.Close
    cmdCloseSend.Enabled = False
End Sub

Private Sub cmdCloseListen_Click()
```

```
    ' 关闭服务 socket
    sockRecv.Close
    ' Enable 侦听按钮
    cmdListen.Enabled = True
    cmdCloseListen.Enabled = False
    lstRecvData.Clear
End Sub

Private Sub Form_Load()
    ' 初始化 socket
    sockSend.Protocol = sckUDPProtocol
    sockRecv.Protocol = sckUDPProtocol

    lblHostName.Caption = sockSend.LocalHostName
    lblLocalIP.Caption = sockSend.LocalIP

    cmdCloseListen.Enabled = False
    cmdCloseSend.Enabled = False

End Sub

Private Sub sockSend_Error(ByVal Number As Integer, _
       Description As String, ByVal Scode As Long, _
       ByVal Source As String, ByVal HelpFile As String, _
       ByVal HelpContext As Long, CancelDisplay As Boolean)
    MsgBox Description
End Sub

Private Sub sockRecv_DataArrival(ByVal bytesTotal As Long)
    Dim data As String

    ' 创建一个足够大的缓冲区来接收数据
    data = String(bytesTotal + 2, Chr$(0))
    sockRecv.GetData data, , bytesTotal
    lstRecvData.AddItem data
    ' 修改远程服务器属性
    lblRemoteIP.Caption = sockRecv.RemoteHostIP
    lblRemotePort.Caption = sockRecv.RemotePort
End Sub
```

```
Private Sub sockRecv_Error(ByVal Number As Integer, _
       Description As String, ByVal Scode As Long, _
       ByVal Source As String, ByVal HelpFile As String, _
       ByVal HelpContext As Long, CancelDisplay As Boolean)
   MsgBox Description
End Sub
```

4.5 小结

本章从各个角度阐述了 TCP/UDP **开发的技术，从某种意义上说，它代表后面章节的** E-mail、FTP、Telnet **等协议的高级开发中的网络通信技术，读者要好好揣摩。由于本章中的程序例程并不复杂，而重要的控件和** API **函数在第** 2 **章中都有详细的说明，因此本章代码除了在代码中间进行注释以外，并没有进行专门的代码说明。**

第 5 章　E-mail 协议及高级编程

本章将从各个角度来阐述 E-mail 客户端软件的开发技术，而对于 E-mail 软件的服务器端，因为适用性不大，所以本章并没有介绍，但是根据网络应用的理论，以及第 4 章介绍的 TCP/UDP 网络编程原理，可以写出简单的邮件服务器程序。

本章大致包括以下程序内容：

- 标准 E-mail 客户端软件 Foxmail 的使用
- 发送无附件的 E-mail 程序
- 接收 E-mail 程序
- MIME 编码解码与发送附件
- E-mail 客户端高级编程
- E-mail 乱码处理
- MAPI 高级编程

5.1　Foxmail 发送接收 E-mail

通过 Foxmail 等这些成熟产品的使用，能够更加深刻地理解 E-mail 程序开发的本质和原理，同时这些产品在功能界面设计上也有很好的参考价值。

启动 Foxmail，界面如图 5-1 所示（这里使用的是目前 Foxmail 的最新版本 4.2 版）。

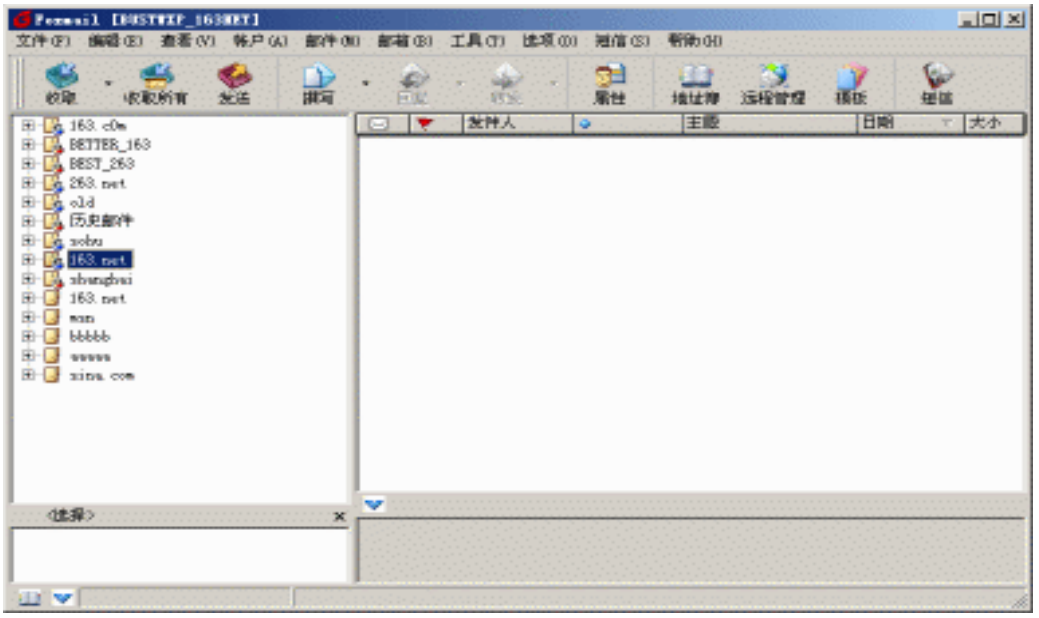


图 5-1　Foxmail 程序运行界面

由于目前流行的 E-mail 客户端软件都能够支持多用户管理，因此在程序运行以后，首先要建立一个新的用户，单击“账户”菜单，选择“新建”命令，弹出新建用户向导界面，单击“下一步”按钮，进入新用户建立界面，如图 5-2 所示。

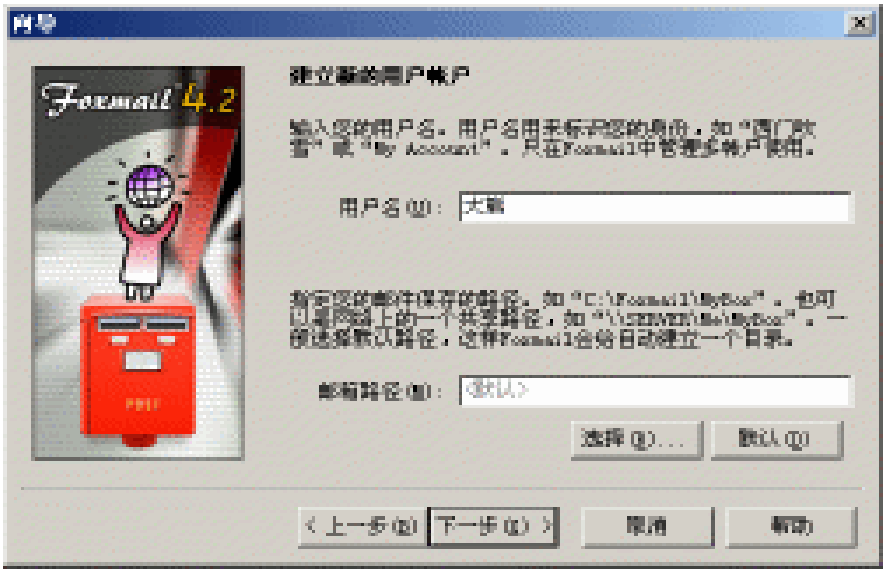


图 5-2　输入用户名和油箱路径

在图 5-2 中输入用户名，该用户名不是 E-mail 的用户名，而是一个标志名，就是在 Foxmail 中显示的用户名。在输入用户名以后，请选择一个邮件保存地址。

为了保证邮件的安全，不被误删，请自己选择目录，而不要使用默认的保存路径，因为这样，当以后重新安装系统或者其他原因删除该软件的时候，很可能一不小心而被自己或者别人删除。

输入完用户名以及邮件保存地址以后，单击“下一步”按钮，如图 5-3 所示。

图 5-3　输入发送者姓名以及邮件地址

在图 5-3 中，需要输入“发送者姓名”以及“邮件地址”。输入完毕以后，单击“下一步”按钮，进入邮件服务器设置阶段，这个很重要，也是要介绍这个软件的主要目的，就是要搞清楚邮件服务器的几种服务，如图 5-4 所示。

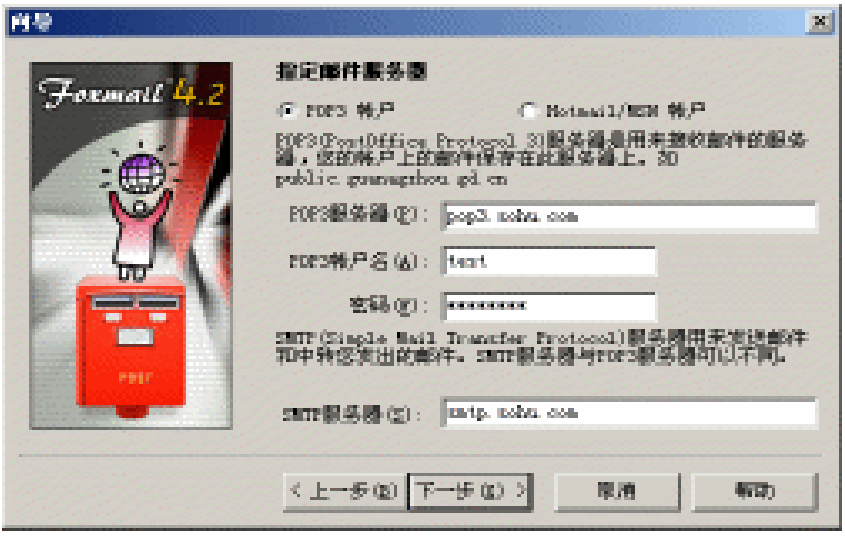


图 5-4　邮件服务器设定

其中首先要设置 POP3 服务器地址，这也是通过 Foxmail 收取信件的服务器地址，正是有了服务器提供 POP3 服务器，客户才可以不登录到它们的网站，而只需要通过客户端的软件就可以下载邮件了。SMTP 服务器就是用于发信的服务器地址，因为发送邮件都是需要通过一个邮件服务器转发到目的邮件服务器的，因此必须设置这个服务器地址。这个地址可以设置成任何一个可用的地址，这里输入的是“smtp.sohu.com”，当然即使使用“smtp.163.com”也是可以的，要注意一点的就是现在很多提供商也都需要进行用户名和密码验证，才能够发送信件了。

单击“下一步”按钮，进入密码设置和其他设置，如图 5-5 所示。

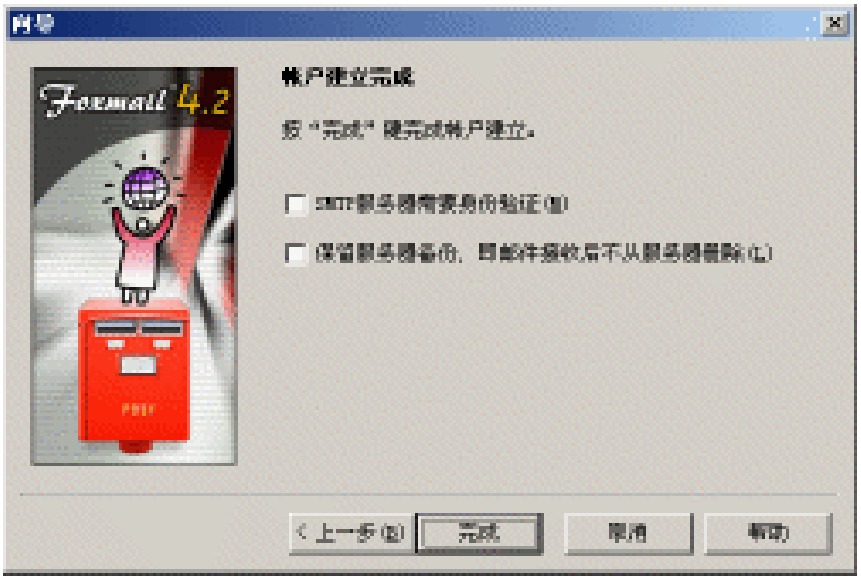


图 5-5　账户设置完毕

如果 SMTP 服务器需要身份验证，则在检查框上打勾，然后单击“完成”按钮，这样在发送邮件的时候，会弹出提示框，提示输入用户名和密码，目前提供商需要的账户和密码都是用户自己邮件的用户名和密码；另外一个选项就是是否在服务器上保留备份，如果不保留，用户下载到本地计算机的时候，服务器的信件全部被删除，这样当用户在其他计算机上登录服务器，则看不到原来的信件了。

单击“完成”按钮以后，就可以进行邮件的接收和发送了，至于邮件的接收和发送，读者可以从菜单中或者工具栏中找到这些命令。另外如果想修改刚才设定的这些属性，可以通过账户菜单中的“属性”来修改，如图 5-6 所示。

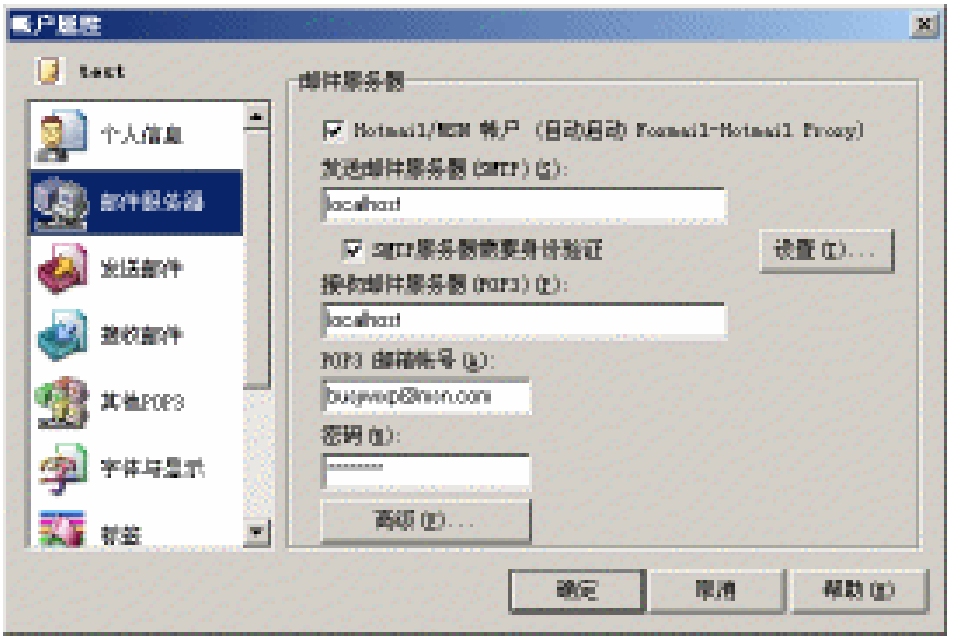


图 5-6　账户属性设置修改

如果 SMTP 服务器需要身份验证，则检查框会打上勾，如果不想每次都输入，可以预先输入，只要单击“设置”按钮，输入用户名和密码就可以了，如图 5-7 所示。

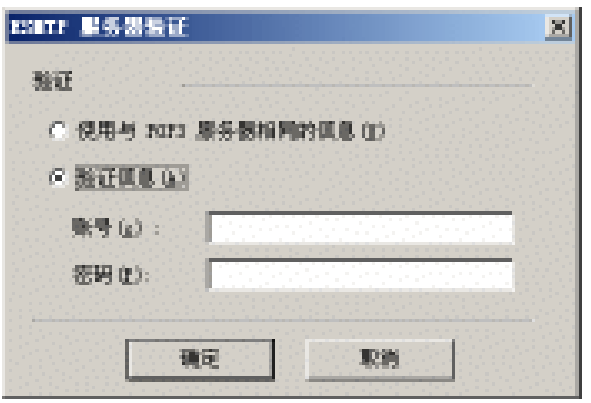


图 5-7　SMTP 服务器验证

这样，我们就可以很方便地接收和发送邮件了。

5.2　SMTP、POP3 与 E-mail

网络上的 E-mail 传送有点像我们日常生活中发送的普通信件一样，我们首先把信件交给邮局，再由邮局按照信封上的地址投递，最后投递到收信人的信箱中，收信人只要凭着他的信箱钥匙就可以拿到信件。同样的一封 E-mail 从用户的手中发出，首先要有一个程序，把这封 E-mail 发到 SMTP 服务器（发送邮件服务器）上，再由发送邮件服务器负责传递邮件到达目的信箱。然后再需要一个程序接收信箱的邮件，当然也不是只要随便有一个程序就可以收的，还需要账号（就像收信人的姓名）和口令（信箱钥匙）。以此看来很明显发送邮件至少需要 4 个处理不同任务，有不同功能的程序：发送邮件到服务器的程序，我们把它叫做发送邮件客户端程序；负责在网络上传送邮件到信箱的程序，我们称之为 SMTP 服务器程序；接收并储存邮件并给用户提取的服务器程序，我们称之为 POP3 服务器程序；从 POP3 服务器上收取邮件的程序，我们称之为接收邮件客户端程序（当然有些功能可以合并在同一个程序里，如发送邮件和接收邮件）。

通常我们进行 E-mail 的程序编程工作集中在收发电子邮件的客户端程序上。收发电子邮件是依靠一套标准的会话协议的，其中与发送有关的协议叫 SMTP（Simple Mail Transfer Protocal 简单邮件发送协议），这个协议规定了与 SMTP 服务器进行对话的一系列命令及过程标准。与接收有关的协议叫 POP3 协议（Post Office Protocal 邮局协议）该协议规定了与 POP3 服务器进行对话的一系列命令及过程标准。因此有必要从这两个协议开始了解收发 E-mail 的原理及过程。

5.3　SMTP 及发送电子邮件

5.3.1　SMTP 的模型描述

SMTP 模型采用的也是现在流行的 C/S（客户机/服务器）模式，用户直接使用的是用于编写和发送的客户端软件，而通常的 SMTP 服务器运行在远程站点上。客户机/服务器之间的通信是通过 TCP/IP 协议进行的。

前面介绍过，使用 SMTP 服务器并不是邮件的目的地，它只是邮件的中间传递机构。发送邮件的客户端软件不用了解如何把邮件发送到目的信箱的服务器上，只要告诉具有传递机制的 SMTP 服务器一些必要的信息，接下来怎么投递邮件就是 SMTP 服务器的事情了。

5.3.2　SMTP 的会话过程

在介绍 SMTP 的命令和标准之前，先来看一段比较简单的 SMTP 对话。

```
 (connect to the SMTP server……)
220  BigFox ESMTP service ready (MDaemon v2.1 rB b1 32-T)
HELO zhongjun
```

```
250 ZHONG Hello worldcomputers.com, pleased to meet you
MAIL FROM:zhong@10.10.10.10
250 <from: zhong@10.10.10.10>, Sender accepted
RCPT TO: zhjuna@263.net
250 < zhjuna@263.net>, Recipient ok
(发送邮件内容)
354 Enter mail, end with <CRLF>.<CRLF>
(邮件发送完毕，发送 crlf & . & crlf)
250 Ok, message saved
QUIT
221 See ya in cyberspace
```

在上面的对话过程中，黑体字部分是发送邮件的客户端软件发送的内容，其他部分是SMTP 服务器的应答内容。从以上的这段简单的会话内容中我们可以明显地看出以下的一些特点：

- 会话的过程全部是发送文本来完成的，过程为交互式的请求应答模式
- 命令是文本形式的命令
- 每次会话服务器总是返回一定的响应码，表示客户端的请求是否被正确地回答
- 会话过程有一定的顺序

到这里可能读者急于想知道如何连接 SMTP 服务器、如何与服务器对话、对话的命令和含义是什么、如何从服务器的响应码中知道服务器的意思？

1. 连接服务器

E-mail 的通信过程是基于 TCP/IP 协议的，所以在 Windows 中采取 Winsock 就可以达到和服务器进行通信的目的。在 Visual Basic 中可以使用 Winsock 控件来实现。在这里有必要对 Winsock 的通信过程进行简单的介绍。

利用 Winsock 控件进行 TCP 通信（对话）通常包含以下的几个步骤：

① 设置服务器在指定 IP 端口监听客户端的连接请求。
② 告诉客户端连接到特定 IP 地址和端口的服务器。
③ 服务器通过接受连接响应连接请求。
④ 客户端和服务器向对方发送数据。
⑤ 客户端或服务器读取缓冲区中的数据。
⑥ 必要时重复④和⑤。
⑦ 关闭客户端和服务器之间的连接。

对于这个要与 SMTP 服务器进行通信的客户端程序，所要做的事情就是设置 Winsock 连接的 IP 地址或域名，指定端口号。SMTP 的端口号一般为 25。在 VB 中形式如下：

```
Winsock.Protocol=sckTCPProtocol
Winsock.RemoteHost="10.10.10.10"
Winsock.RemotePort=25
```

```
Winsock.Connect
```

这样如果指定的地址运行着 SMTP 服务器程序，那么就可以与服务器进行通信了。

读者可以在 VB 中试试看，指定一个知道的发送邮件服务器地址，然后连接上去，看看接收到的从服务器返回的响应文字，如果是类似：

```
220 BigFox ESMTP service ready (MDaemon v2.1 rB b1 32-T)
```

的文字，那就对了，说明你已经成功地实现了第一步，接着往下看。

2. 发送命令及接收响应信息

命令是利用 Winsock 控件在连接上服务器之后发送的，如：

```
Winsock.SendData "HELO ZHONG" & vbCrLf
```

其中 HELO 是编写的程序（客户端程序）与 SMTP 服务器对话的命令，ZHONG 是 HELO 命令参数，在每个命令的结尾加上回车换行符（VB 中使用常数变量 vbCrLf 表示）。

如果服务器接收客户机的命令，会给客户机返回一个响应码和消息行，告诉请求的命令是否被准确执行。在 VB 编写的程序中，利用 Winsock 控件与服务器进行通信，客户机接收到服务器的数据会触发该控件的 Wsock_DataArrival 事件。可以在该事件中取得服务器返回的响应码。

```
Private Sub Wsock_DataArrival(ByVal bytesTotal As Long)
Wsock.GetData Information
End Sub
```

3. SMTP 标准命令

在成功连接到 SMTP 服务器之后，接着要进行的是与它的会话，SMTP 有其认识的一整套命令，当然这些命令是事先约定好的，SMTP 协议很重要的一部分就是会话的标准命令。表 5-1 列出了比较常用的 SMTP 命令。

表 5-1　　常用 SMTP 命令

命令	描述
QUIT	终止会话
HELP	请求 SMTP 命令的帮助
NOOP	空操作
VRFY	验证地址（不要求一定启用）
EXPN	扩展一个别名
HELO	客户机问候服务器
MAIL	指定邮件的发送者
RCPT	指定邮件的接收者
DATA	发送邮件的数据状态
RSET	复位会话状态
SEND	指定要发送到用户终端的邮件的发送者
SOML	Send 或 Mail
SAML	Send 和 Mail
TURN	交换客户机/服务器角色

命令详解：

（1）退出：QUIT

该命令用来终止与 SMTP 的会话。

```
客户机：QUIT
服务器：221 smtp.bigfox.com ESMTP server closing connection
```

当客户机发送 QUIT 命令时，服务器使用 221 响应代码应答确认终止，然后关闭连接。

（2）帮助：HELP

该命令用于请求服务器发送回来各种类型的帮助信息。

```
客户机：HELP
服务器：
214-This SMTP server is a part of the Netscape Mail Server system.  For
214-information   about   the   Netscape   Mail   Server,   please   see
http://www.netscape.com
214-
214-     Supported commands:
214-
214-          EHLO    HELO    MAIL    RCPT    DATA
214-          VRFY    RSET    NOOP    QUIT
214-
214-     SMTP Extensions supported through EHLO:
214-
214-          EXPN    HELP    SIZE
214-
214-For more information about a listed topic, use "HELP <topic>"
214 Please report mail-related problems to Postmaster at this site.
```

在这个例子中服务器应答可用命令的摘要信息，以及关于服务器软件的一般信息。

HELP 命令还可以带参数请求特定的 SMTP 命令的帮助，如向服务器请求 DATA 命令的帮助信息，得到下面的返回结果：

```
客户机：HELP DATA
服务器：
214-Usage:  DATA
214-
214-The DATA command is used to transfer the message body.  Lines
214 are read until a line consisting of a single "." is encountered.
```

（3）空操作：NOOP

这个命令什么都不做，不改变与服务器的会话状态。服务器通常以 250 作为响应码应答，确认成功。

```
客户机：NOOP
```

```
服务器：250 OK
```

用该命令可以测试与服务器连接。

（4）验证：VRFY

该命令用于验证指定的字符串是否是一个合法的信箱。

```
客户机：VRFY zhong
服务器：550 Unknown address: <zhong>
```

在上面的例子中，VRFY 的主要失败响应码是 550。

在下例中服务器认为该信箱合法，还无法验证是否存在，但愿意尝试发送。

```
客户机：VRFY zhong@263.net
服务器：252 Couldn’t verify <zhong@263.net> but will attempt delivery anyway
```

（5）扩展：EXPN

该命令把指定的参数扩展为指向一组组成成员的邮件列表。假如 SMTP 服务器上有一个含有 4 个成员的邮件列表：

```
Vb-club：busyants@263.net
        Bigfox@163.net
        BigShark@hotmail.com
        Busyzhong@netease.com
```

EXPN 命令可以用于扩展该列表，显示它的成员。如：

```
客户机：EXPN Vb-club
服务器：
250-< busyants@263.net>
250-<Bigfox@163.net>
250-<BigShark@hotmail.com>
250-<Busyzhong@netease.com>
```

遗憾的是，该命令由于被认为会显示太多的信息而通常在 SMTP 服务器上是禁止的。所以请求该命令时，通常返回以下的应答：

```
252 Cannot EXPN mailbox,but will accept address
```

（6）问候：HELO

该命令用于客户机向服务器标识自己。通常在连接到服务器之后，向服务器发送该命令。如果 HELO 命令完成，客户机和服务器都准备好继续 SMTP 对话的其余部分。

如果服务器接受该命令，返回 250 的响应码。例如在上例中：

```
客户机：HELO ZHONG
服务器：250 ZHONG Hello worldcomputers.com, pleased to meet you
```

由于安全性或其他的原因，服务器想拒绝到达的客户机请求，可以发出 550 错误响应码。

```
550 Access Denied.
```

（7）邮件来自：MAIL

该命令用于指定邮件的发送者。实际上这个命令所带的参数并不是十分重要，当投递失败时，服务器将邮件返回给发送者，该命令指定返回的信箱地址。

客户机：`mail from:<bigfox@163.net>`

服务器接受该命令，发出 250 响应码，指示会话准备好继续到下一个步骤。

```
250 OK
```

或 `250 <from: bigfox@163.net >, Sender accepted`

这个命令如果失败，服务器返回失败的响应码。如：

服务器不想授予访问权，发出 550 错误。

```
550 access denied
```

对于语法错误，服务器应答 553 错误：

```
553 mailbox syntax incorrect
```

值得注意的是，该命令是邮件事务中的第一个命令。服务器成功响应该命令之后，将会复位任何状态表和数据缓冲区，使新事务可以开始。

（8）收信人：RCPT

该命令用于指定收信人的地址。

客户机：`RCPT TO: zhjuna@263.net`

服务器成功接受该命令，通常产生以下响应：

```
250 < zhjuna@263.net>, Recipient ok
```

命令失败，服务器返回相应的响应码，如：

信箱地址不合法，产生 553 响应码：

```
553 Invalid address syntax
```

（9）数据（信体内容）：DATA

该命令用于向 SMTP 服务器请求发送邮件的内容。

指定发送邮件地址的 mail 或 send 命令和指定接收人地址的 rcpt 命令必须在 data 命令前执行，否则服务器将应答 503，表示命令顺序不对。

这个命令与一般的 SMTP 命令有点不同，当客户端发出这个命令请求后，服务器通常以 354 作为响应码，表示服务器已经准备好让客户端发送数据。接着客户端发送符合 RFC822 规范的信体。如：

```
data
354 Enter mail, end with <CRLF>.<CRLF>
to:busyant@hotmail.com
from:busyboy@hotmail.com
date:Sat,14 Oct 2000 13:02:22 -0700
X-Mailer: EBT Reporter v 2.x
Subject:Hello,how are you?
```

```
Hello,how are you?I am busyboy.
.
250 Ok, message saved
```

发送信体完成，客户端发送句点（.）作为结束，当服务器接到句点后，应答一个代码指示发送是否成功。

如果发送的信体中的一行是以句点（.）作为开始的，由于句点可以用于终止信体的发送，服务器可能会误认为客户端发送的信体数据已经完成。这就要采取有效措施来确保服务器能够知道发送的句点是信体数据还是结束邮件的发送。解决这个问题，客户端应该在发送邮件时检查每一行，如果发现某一行是以句点开始的，则在该行前面多加一个句点，当服务器处理来自 data 命令后的数据时，如果检查到一行以句点开始，则从该行的开始删除一个句点，把数据还原到原来的形式。

例如：发送以下的两行文字

```
I will enjoy it tonight.
…but I have sth to do.
```

第二句中以句点开头。经过处理后变成：

```
I will enjoy it tonight.
.…but I have sth to do
```

当服务器接收完信体时，给客户端返回一个响应码，成功的响应码为 250，如：

```
250 Ok, message saved
```

如果由于其他原因使 data 命令失败，服务器返回表示错误的响应码，响应码的类型根据错误情况有所不同。下面列举几个错误的响应码：

```
552 Requested mail action aborted:exceeded storage allocation
554 Local configuration error
451 Requested action aborted :local error in processing
452 Requested action not taken:insufficient system storage
```

响应码的意思请参照表 5-2。

（10）复位：RSET

该命令用于复位连接状态。

```
客户端：RSET
服务器：250 OK
```

服务器接到这个命令后，将清除以前所接收的命令请求及内容，复位会话状态到发出 HELO 命令时的状态。

4. SMTP 服务器响应码

在每次发送一条 SMTP 命令后，服务器给客户机返回一条响应。响应都是以一个响应码开头，后面接着响应的描述信息。如果 SMTP 服务器不一样，响应的描述信息可能不一样，如对于 mail 命令：

```
MAIL FROM:zhong@163.net
```

服务器响应可能为

```
250 <from: zhong@163.net>, Sender accepted
```

也可能只是简单的响应

```
250 OK
```

表示服务器已经接受该命令，客户机可以接着往下进行。尽管响应描述不太一样，但是对于同一条命令，相同的响应结果响应码是一致的。从上面的例子也可以看出这一点。

响应码由 3 位数字数字组成，后接一个空格和文本描述信息。在表 5-2 中列出了响应码的含义。

表 5-2　　常用 SMTP 响应码

响应码	含义
211	系统状态或系统帮助应答
214	帮助信息
220	服务就绪
221	服务关闭传输通道
250	请求得邮件操作完成
251	用户不是本地的，将转发到<forwaed-path>
354	启动邮件输入，用<CRLF>.<CRLF>结束
421	服务不可用，关闭传输通道
450	没有采取请求的邮箱操作：信箱不可用
451	请求的操作终止：处理中有错误
452	请求的操作没有发生：系统存储空间不足
500	语法错误，命令不可识别
501	参数或变元中的语法错误
502	命令不能实现
503	错误的命令序列
504	命令参数不能实现
550	请求的操作不能发生：信箱不可用
551	用户不在本地，请尝试发送到<forward-path>
552	请求的邮件操作终止：超出存储分配
553	未采用请求的操作：信箱名不允许（例如信箱语法错误）
554	事务失败

响应码的每一位都有特定的含义。其中第一位在 1~5 之间。每个数字的含义如下：

- 1——没有用在 SMTP 中。
- 2——表示一条关于传输线路的消息。
- 3——表示一条肯定的中间响应，它意味着服务器正在等待更多的信息。
- 4——表示命令没有被接受且要求的操作还没有发生，但这个状态是暂时的。
- 5——表示绝对的失败。

响应码的第二位表示类别如：

- 0——表示一个语法错误。
- 1——表示消息内容。
- 2——表示一条关于传输路线的消息。
- 3 和 4 没有使用。
- 5——表示一条关于邮件系统状态的消息。

响应码的第三位仅仅指定了某个特定类别中的消息的间隔等级。

表 5-3 以分类的形式给出响应码的含义。

表 5-3　　分类 SMTP 响应码的含义

响应码	含义
2xx	肯定应答
3xx	中间肯定应答
4xx	暂时否定完成应答
5xx	永久否定完成应答
x0x	语法错
x1x	信息
x2x	连接
x3x	还未使用
x4x	还未使用
x5x	邮件系统

5. 发送步骤

利用 Winsock 连接上 SMTP 服务器后：

（1）打开邮件发送的对话

这个命令是使邮件服务器知道发送客户端程序想与它通信。

语法：`HELO <domain><crlf>`

假如我们发送这样的命令：

```
Winscok.SendData "HELO" & " " & "zhongjun" & vbCrLf
```

如果对话成功，服务器的返回信息是这样的。

```
250 ZHONG Hello zhongjun, pleased to meet you
```

当然不同的 SMTP 服务器可能响应的信息不一样。但是返回的前三个字符即响应码，肯定是相同的。我们可以根据响应码判断请求是否被接收。

（2）指定发送者的信箱

这个过程是必需要有的，就像我们写信一样，写信的人的地址是必需的。有些发送邮件的服务器如果无法发送邮件到目的信箱，会根据发送者的信箱地址返回一个邮件无法发送的消息。

语法：`MAIL FROM: <sendermailbox> <crlf>`

假如此命令被接受，服务器返回的响应码为 250。

（3）指定接收者的信箱

接下来指定接收者的信箱。

语法：`RCPT TO: <receivemailbox> <crlf>`

假如我们要将邮件发送给“busyants@263.net”，只需写入以下代码：

```
Winscok.SendData "RCPT TO:" & " " & "busyants@263.net" & vbcrlf
```

假如此命令被接受，则服务器的响应码为 250。如果有多个接收者，则重复这一步即可。

（4）发送邮件的内容

语法：DATA <crlf>

假如该命令被接受，则服务器的响应码为 354，并认为随后接到的数据是邮件的内容。

邮件内容有一定的格式，以下是一封电子邮件内容的基本格式。（包括发送者姓名、接受者姓名、发送时间、发送邮件软件名称、主题和邮件内容）

```
From: BigAnt
Date: Fri, 07 Apr 2000 15:24:59 -0600
X-Mailer: Zhongjun-Mailer
To: Busywxp
Subject: 你接到信了吗？
接到信之后请给我回信
```

.在发送邮件内容之后，客户端程序发送字符串“vbCrlf & . & vbCrlf”。服务器接收到这个字符串后认为邮件内容已经发送完成。服务器返回为“250 OK”表示接受了邮件数据。

（5）结束邮件发送对话

发送邮件完毕，结束对话。

语法：QUIT <CRLF>

服务器返回的响应码为 221。

5.4 发送无附件 E-mail 程序

在了解发送电子邮件的 SMTP 协议以及相应的命令之后，下面以一个简单的例子来实现发送电子邮件。

5.4.1 建立工程项目

在本实例中，主要利用了 Winsock 控件 TCP 连接实现 SMTP 会话过程，工程项目非常简单，由两个窗体模块构成，即 frmSetup 窗体和 SMTP 窗体。程序运行时界面如图 5-8 所示。

图 5-8 发送 E-mail 程序运行时界面

5.4.2　代码分析

1. **窗体** SMTP

```
Option Explicit
Public ServerIp As String       'SMTP 服务器地址
Public ServerPort As Long       'SMTP 服务器端口

Dim strSendName As String   '发送人姓名
Dim strReceiveName As String    '接收人姓名
Dim strFromMail As String       '发送人地址
Dim strToMail As String         '接收人地址
Dim m_Date As String        '发送日期
Dim strSubject As String        '主题
Dim strContent As String        '正文
Dim Information As String       '从服务器接收响应消息
```

以上代码声明该窗体所用的变量，变量的含义见其后面的注释。

```
Private Sub cmdSend_Click()
'设置 Winsock
Wsock.Close
Wsock.RemoteHost=ServerIp
Wsock.RemotePort=ServerPort
strSendName=txtSName.Text
strReceiveName=txtRName.Text
strFromMail=txtFrom.Text
strToMail=txtTo.Text
m_Date=Format(Date, "Ddd") & ", " & Format(Date, "dd Mmm YYYY") & " " &
Format(Time, "hh:mm:ss") & "" & " -0600"
strSubject=txtSubject.Text
strContent=txtContent.Text
Dim mData As String
'构造邮件标题字段
mData="From:" & Chr(32) & strSendName & vbCrLf & _
       "Date:" & Chr(32) & m_Date & vbCrLf & _
       "X-Mailer: BigAnt Smtp Mailer V1.0" & vbCrLf & _
       "To:" & Chr(32) & strReceiveName & vbCrLf & _
       "Subject:" & Chr(32) & strSubject & vbCrLf
Wsock.Close
'连接 SMTP 服务器
Wsock.Connect
```

```
    If Not WaitForResponse("220", 10) Then
       txtMsg.Text="邮件服务器连接不上......"
       Exit Sub
    End If
    '打开对话
    Wsock.SendData "HELO" & " " & Wsock.LocalHostName & vbCrLf
    If Not WaitForResponse("250", 10) Then
       txtMsg.Text=txtMsg.Text & "无法打开邮件发送对话" & vbCrLf
       Exit Sub
    End If
    '发送发送方地址
    Wsock.SendData "MAIL FROM:" & " " & strFromMail & vbCrLf
    If Not WaitForResponse("250", 10) Then
       txtMsg.Text=txtMsg.Text & "无法发送发送方地址" & vbCrLf
       Exit Sub
    End If
    '发送接收方地址
    Wsock.SendData "RCPT TO:" & " " & strToMail & vbCrLf
    If Not WaitForResponse("250", 10) Then
       txtMsg.Text=txtMsg.Text & "无法发送接收方地址" & vbCrLf
       Exit Sub
    End If
    '发送消息体
    Wsock.SendData "DATA" & vbCrLf
    If Not WaitForResponse("354", 10) Then
       txtMsg.Text=txtMsg.Text & "无法发送消息体" & vbCrLf
       Exit Sub

    End If
    Wsock.SendData mData & vbCrLf
    Wsock.SendData strContent & vbCrLf
    Wsock.SendData "." & vbCrLf
    If Not WaitForResponse("250", 20) Then
       txtMsg.Text=txtMsg.Text & "消息体发送不成功" & vbCrLf
       Exit Sub
    End If
    '结束邮件发送对话
    Wsock.SendData "QUIT" & vbCrLf
    If Not WaitForResponse("221", 10) Then
```

```
    Exit Sub
End If
Wsock.Close
txtMsg.Text=txtMsg.Text & "邮件发送成功"
txtMsg.Text=txtMsg.Text & mData & vbCrLf & strContent & vbCrLf
End Sub
```

以上代码用于“发送”按钮的单击事件，首先从窗体上填写内容，设置相应的变量，构造邮件的标题字段。然后连接 SMTP 服务器，成功后按照 SMTP 会话步骤发送命令及参数，在每一步等待服务器正确的响应码，如果服务器返回的响应码与期望的一致，进行下一个步骤，如果服务器返回表示错误的响应码，则退出该过程停止发送，发送失败。此外，如果等待时间过长，也会认为接收不到正确的响应码。邮件发送成功，关闭 Winsock 连接。

```
'该按钮事件过程用于设置 SMTP 服务器
Private Sub cmdSetUp_Click()
frmSetup.Show
End Sub
```

以上的过程用于响应设置 SMTP 服务器的按钮事件，弹出设置窗体。

```
'程序加载时读出上次的设置
Private Sub Form_Load()
ServerIp=GetSetting("email", "smtpserver", "serverip", "")
ServerPort=GetSetting("email", "smtpserver", "serverport", 25)
Wsock.Protocol=sckTCPProtocol
End Sub

'程序退出时保存设置
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
SaveSetting "email", "smtpserver", "serverip", ServerIp
SaveSetting "email", "smtpserver", "serverport", ServerPort
End Sub
```

以上的过程用于响应窗体加载和卸载过程，从注册表中读出或保存 SMTP 服务器的信息，方便使用。

```
'接收服务器的响应消息
Private Sub Wsock_DataArrival(ByVal bytesTotal As Long)
Wsock.GetData Information
txtMsg.Text=txtMsg.Text & Information & vbCrLf
End Sub
```

以上代码用于响应 Winsock 的数据到达事件，从缓冲区中读取数据，赋给变量 Information。

```
'该函数用于等待服务器响应码
Private Function WaitForResponse(strResponse As String, WaitTime As Integer)
```

```
As Boolean
    Dim WaitSt As Date
    WaitSt=Now()
    While InStr(1, Information, strResponse, vbTextCompare) < 1
       DoEvents
       If DateDiff("s", WaitSt, Now) > WaitTime Then
          Information=""
          WaitForResponse=False
          Exit Function
       End If
    Wend
    Information=""
    WaitForResponse=True
    End Function
```

以上代码用于判断接收服务器返回的数据（在 Information **变量中）是否有指定的字符串。可以借此来判断服务器是否返回了响应码。如果在指定的时间内没有接收到指定的响应码字符串，该函数返回** False**，认定没有接收到服务器的响应码。**

2. **窗体** frmSetUp

该窗体用于设置 SMTP **服务器的地址和端口，程序代码很简单，请读者自己分析。其界面如图** 5-9 **所示。**

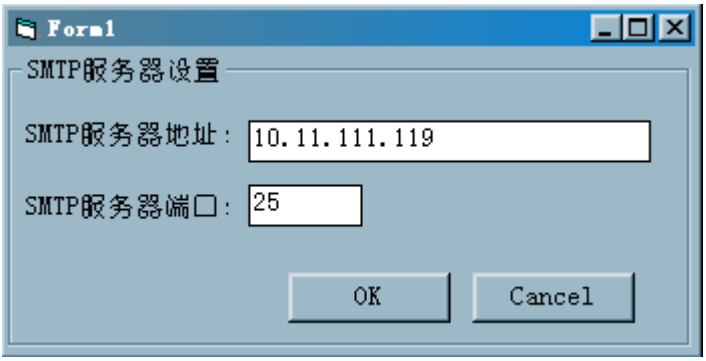


图 5-9　**设置** SMTP **服务器**

```
    Private Sub cmdQuit_Click(Index As Integer)
    If Index=0 Then
       smtp.ServerIp=txtIp.Text
       smtp.ServerPort=CLng(txtPort.Text)
    End If
    Unload Me
    End Sub

    Private Sub Form_Load()
```

```
    txtIp.Text=smtp.ServerIp
    txtPort.Text=smtp.ServerPort
    End Sub
```

5.5　POP3 与接收电子邮件

5.5.1　POP3 的模型描述

与 SMTP 一样，POP 也有一套协议。第一个 POP RFC 是 1984 年发表的（RFC918，邮局协议），描述了一个基本的实验性的 POP 实现。随后在 1985 年发表 RFC937（邮局协议版本 2），增加了几个新的功能，并且使用更加广泛。第三个版本 RFC1081（邮局协议版本 3）是在 1988 年发表的，这个版本修改过几次，产生了以下的几个修订版的 RFC：

RFC1225，RFC1460，RFC1725，RFC1939

其中 RFC1939 是当前 POP 标准。

以下我们介绍的 POP 协议都是指邮局协议版本 3，即 POP3 协议。虽然尽管有些 POP2 的实现还在使用，但由于大量使用的是 POP3，我们只介绍 POP3。

5.5.2　POP3 的会话过程

与介绍 SMTP 的模型一样，先看一段比较简单的 POP3 对话。

```
(connect to the POP3 server……)
+OK  BigFox POP3 server ready (MDaemon v2.1 rB b1 32-T)
user busyzhong
+OK
pass 123321
+OK
stat
+OK 2 4065
retr 1
+OK
(POP3 server 发送邮件 1，客户端软件接收邮件内容)
retr 2
+OK
(POP3 server 发送邮件 2，客户端软件接收邮件内容)
dele 1
+OK
dele 2
+OK
QUIT
```

```
+OK
```

在上面的对话过程中，黑体字部分是接收邮件的客户端软件发送的内容，其他部分是 POP3 服务器的应答内容。从以上的这段简单的会话内容我们可以明显地看出以下的一些特点：

- 与 SMTP 会话的过程一样是发送文本来完成的，过程为交互式的请求应答模式
- 命令也是文本形式的命令
- 每次会话服务器总是返回一定的响应码，表示客户端的请求是否被正确回答
- 会话过程有一定的顺序

1. 连接服务器

POP3 通信与 SMTP 一样，在 Windows 里可以通过 Winsock 来实现，读者可以参看前面讲过的连接 SMTP 服务器。

与 POP3 服务器进行通信的客户端程序，设置 Winsock 连接的 IP 地址或域名，指定端口号。POP3 的端口号为 110。在 VB 中如下：

```
Winsock.Protocol=sckTCPProtocol
Winsock.RemoteHost="10.10.10.10"
Winsock.RemotePort=110
Winsock.Connect
```

这样如果指定的地址运行着 POP3 服务器程序，那么就可以与服务器进行通信了。

通常会返回类似以下的信息：

```
+OK  BigFox POP3 server ready (MDaemon v2.1 rB b1 32-T)
```

2. POP3 标准命令

POP3 的标准命令如表 5-4 所示。

表 5-4　　常用 POP3 命令

命令	描述
USER	标识用户进行验证
PASS	发送密码进行验证
APOP	转换验证机制
QUIT	终止会话
NOOP	空操作
TAT	提供信箱大小信息
LIST	提供邮件大小信息
RETR	从服务器取出邮件
TOP	取出信头和邮件的前 N 行
DELE	标记邮件被删除
RSET	复位 POP 会话
UIDL	取出邮件的唯一标识符

命令详解：

（1）用户：USER

该命令用来验证用户信箱。

```
客户机：USER 用户信箱
服务器：+OK
```

在成功连接 POP3 服务器后，客户端要发送“ user + 用户名 ”的命令进行验证，如果成功则返回+OK 的应答，如果失败，则返回-ERR。如：

```
user
-ERR missing user name argument
```

这是由于 User 命令之后没有提供用户信箱名的原因。提示信息根据服务器不同而异。值得注意的是，User 命令不是必需的，但通常 POP 服务器至少必须实现一个验证机制。大多数的 POP 服务器 User 命令还是必须的。

User 命令只能在会话的验证状态时进行，验证完成后进入事务状态。如果 User 命令验证没通过，一些服务器会在提示错误信息后，客户端可以再次发出这个命令以请求再次验证。

（2）密码：PASS

该命令用来为 User 命令指定的信箱提供密码。

```
客户机：PASS 密码
服务器：+OK
```

成功返回的应答为+OK，如果密码与用户信箱名不匹配，返回-ERR。

如果验证成功，会话进入事务状态；如果失败，则服务器保持在验证状态中。

除了因为密码和用户信箱不匹配之外，如果服务器不能获得对信箱的独占访问锁，服务器也应答一个错误，并保持在验证状态中。

```
Pass 123321
-ERR maildrop is already locked
```

通常在已经有其他用户登录该信箱进行操作时会产生这样的情况。POP 服务器是不允许同时有两个进程访问同一个信箱的。

如果 Pass 命令失败，不允许客户端简单地发送另一个 Pass 命令进行验证，而是在发送 Pass 之前要重新发送 User 命令。

（3）退出：QUIT

该命令终止会话，断开与服务器的连接。

```
客户机：QUIT
服务器：+OK
```

成功响应为+OK，失败响应为-ERR。

如果会话在验证状态，Quit 引起服务器关闭连接，如果会话在事务状态，POP 服务器进入更新状态，在关闭连接前删除标记为删除的任何邮件。如果用户不是通过 Quit 命令关闭连接而是在客户端进行强行关闭，则在事务状态标记删除的邮件并没有被删除。

如果在更新状态时删除邮件碰到错误，服务器应答一个错误信息。但不管 Quit 命令成功与否，信箱锁都被释放，连接关闭。

（4）空操作：NOOP

该命令只是检测服务器的连接。

客户机：QUIT
服务器：+OK
成功返回+OK，失败返回-ERR。

该命令只能在事务状态中使用，除了检测与服务器的连接是否正常，还可以防止一个自动注销定时器定时关闭连接。

（5）状态：STAT

该命令请求服务器返回信箱大小的信息，但是不包括标记为删除的邮件。

客户机：STAT
服务器：+OK 2 4065

服务器的应答包含有信箱邮件的数量以及所有邮件的大小。Stat 命令仅在事务状态时是可用的。

（6）列表：LIST

该命令请求服务器返回一个信箱中特定邮件的大小信息或没有删除标记的所有邮件的大小信息。

客户机：LIST 2
服务器：+OK 2 1046

该命令有两种使用情形：带参数和不带参数。List 后面如果指定邮件，则返回邮件的大小信息 List 指定的邮件如果被标志为删除或不存在，则出错：

客户机：LIST 3
服务器：-ERR no such message, only messages 1 thru 1 are present in your inbox

如果 List 不带参数，如：

客户机：LIST

服务器将返回多行应答。

```
+OK
1 1045
2 2204
.
```

应答成功先是响应+OK，接着每一行含有一个邮件号和邮件的大小（字节数）。最后是以句点“.”作为结束行。

如果没有邮件，服务器将响应一个成功的多行应答，在初始行和结束行之间没有任何应答语句：

客户机：LIST
服务器：
+OK
.

（7）取出邮件：RETR

该命令用于取出指定邮件号的邮件。

```
客户机：RETR 1
服务器：
+OK 13100 octets
Received: from company.mail [127.0.0.1] by company.mail [127.0.0.1] with RAW
(MDaemon.v2.7.SP3.T) for <wang@company.mail>; Tue, 04 Dec 2001 16:19:02 +0800
Date: Tue, 04 Dec 2001 16:19:02 +0800
From: MDaemon@company.mail
X-MDSend-Notifications-To: [trash]
Subject: Welcome to the email system for domain company.mail
To: wang@company.mail
Reply-To: MDaemon@company.mail
Message-ID: <MDAEMON30003200112041619.AA1902015@company.mail>
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
X-Actual-To: wang@company.mail
X-Actual-From: MDaemon@company.mail
Welcome wang!
--
.
```

如果请求成功，服务器返回的是多行应答，先是“+OK**”表示响应成功，接着返回邮件的所有内容，包括信头、信体，如果有附件，附件的内容也以文本的形式返回。最后以一个句点“.”表示结束。为了防止单个句点引起客户机提前结束邮件的读取，对** Retr **命令使用与前面所介绍的** SMTP **标准命令** Data **相同的点填补。**

指定的邮件不存在：

```
客户机：RETR 3
服务器：-ERR no such message
```

如果邮件标志删除后，同样会出现错误：

```
客户机：DELE 1
服务器：+OK message 1 deleted
客户机：RETR 1
服务器：-ERR no such message
```

Retr **后面所带的指定邮件标号是必须的，如果不带这个参数就会出错。**

```
客户机：RETR
服务器：-ERR missing requered message number parameter
```

（8）取出前几行：TOP

```
客户机：TOP 1 1
```

```
    服务器：
    +OK
    Received: from company.mail [127.0.0.1] by company.mail [127.0.0.1] with RAW
(MDaemon.v2.7.SP3.T) for <wang@company.mail>; Tue, 04 Dec 2001 16:19:02 +0800
    Date: Tue, 04 Dec 2001 16:19:02 +0800
    From: MDaemon@company.mail
    X-MDSend-Notifications-To: [trash]
    Subject: Welcome to the email system for domain company.mail
    To: wang@company.mail
    Reply-To: MDaemon@company.mail
    Message-ID: <MDAEMON30003200112041619.AA1902015@company.mail>
    Mime-Version: 1.0
    Content-Type: text/plain; charset=us-ascii
    X-Actual-To: wang@company.mail
    X-Actual-From: MDaemon@company.mail
    Welcome wang!
    .
```

Top 命令格式：Top 邮件序号行数

Top 命令取出指定邮件的信头和指定信体的行数。该命令仅在事务状态中才是可用的。

TOP 1 2 取出的是邮件号为 1 的信件头和前 2 行的内容。

与 Retr 命令一样，使用 Top 命令服务器返回的内容结束标志为句点“.”。

如果指定的行数为 0 或不指定行数，则服务器仅返回信头和空白行。如：

```
    客户机：top 1 0 （或：top 1）
    服务器：
    +OK
    Received: from company.mail [127.0.0.1] by company.mail [127.0.0.1] with RAW
(MDaemon.v2.7.SP3.T) for <wang@company.mail>; Tue, 04 Dec 2001 16:19:02 +0800
    Date: Tue, 04 Dec 2001 16:19:02 +0800
    From: MDaemon@company.mail
    X-MDSend-Notifications-To: [trash]
    Subject: Welcome to the email system for domain company.mail
    To: wang@company.mail
    Reply-To: MDaemon@company.mail
    Message-ID: <MDAEMON30003200112041619.AA1902015@company.mail>
    Mime-Version: 1.0
    Content-Type: text/plain; charset=us-ascii
    X-Actual-To: wang@company.mail
    X-Actual-From: MDaemon@company.mail
    .
```

返回的结束标志也是句点“.”。有关 E-mail 的发送日期，主题，接收人等信息在信头中，

通常使用 Top 命令取回信件头之后进行分析，可以知道 E-mail 除内容之外的信息。

同样对于已经标志为删除的邮件和不存在的邮件，使用 Top 命令服务器返回出错信息。

Top 命令对于服务器是可选的，在有些服务器上并没有实现。只有在可以实现 Top 的服务器才可以使用该命令。

（9）删除：DELE

```
客户机：DELE 1
服务器：+OK message 1 deleted
```

该命令标志指定的邮件为删除邮件。用于事务状态中。

如果指定的邮件不存在或已经标志为删除，返回出错提示，如：

```
客户机：DELE 1
服务器：+OK message 1 deleted
客户机：DELE 1
服务器：-ERR message 1 already marked for deletion
```

只有在 Quit 命令之后进入更新状态，标志为删除的邮件才会被真正地删除。

（10）复位：RSET

```
客户机：RSET
服务器：+OK
```

该命令在事务状态中复位 POP 会话。被标记为删除的邮件取消标记，以后发出的 Quit 命令不删除以前标记为删除的邮件。

（11）唯一 ID 列表：UIDL

```
客户机：UIDL
服务器：
+OK
1 MD00001:MSG:13100:3337811140
.
```

该命令用户请求返回邮件的唯一标识符。

在与 POP 服务器的会话中，只是用简单的序号来标识邮件是不行的，邮件序号一般是按照信箱中邮件时间的先后从 1 开始。假如信箱中一共有三封信。序号按时间先后分别是 1、2、3。假如在一次 POP 会话中删除了邮件 2，邮件 3 的序号在下次会话中变成 2。客户机根据序号标识邮件有很大困难。Uidl 命令可以返回该 POP3 服务器上的邮件的唯一编号，根据这个编号，客户机可以在不同的 POP3 会话中辨认邮件。

命令格式：UIDL [邮件序号]

其中邮件序号是可选的，如果不带参数，则返回的是多行应答。多行应答最后以句点“ . ”作为结束。如果是指定邮件号，则返回指定邮件的唯一 ID，且应答是单行的，如：

```
客户机：UIDL  1
服务器：+OK 1 MD00001:MSG:13100:3337811140
```

3. POP3 服务器的应答

在 POP3 中服务器的应答比 SMTP 应答简单的多。命令操作的应答状态码只有“ +OK ”和“ -ERR ”两个，分别表示成功和失败。

4. POP3 会话的 3 个状态

POP3 会话一共有 3 个状态：验证状态（authorization status）、事务状态（transaction status）和更新状态（update status）。每个状态是会话过程中的特定阶段。

当连接服务器后，首先进入验证状态，在这个阶段里，可以使用的 POP3 命令是：

```
User   Pass   Apop   Quit
```

通过服务器验证后，服务器锁定信箱，这样做是可以防止多个 POP 客户端进行邮件操作，比如删除，取信等。但可以让新的邮件加入。这时会话过程转变为事务状态，接收邮件的 POP 对话大部分时间都是处在事务状态中的。

在事务状态可以使用的 POP3 命令有：

```
Noop   Stat   Quit   List   Retr  Top   Dele   Rset   Uidl
```

会话过程的最后一个状态为更新状态，在事务状态结束后，发出 Quit 命令进入该状态，但是由于异常原因导致的与服务器终止对话并没进入更新状态。在事务状态进行的一些操作，最终在更新状态才得到体现。如删除邮件是在事务状态使用 Dele 命令，但是服务器并没有实际删除，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，如果是异常原因导致没有发出 quit 命令进入更新状态，则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。

更新状态只是会话过程的一个过程，目的是用户在事务状态后用以确认已经进行的操作，该状态没有可使用的命令。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。

5. 接收步骤

利用 Winsock 连接上 POP3 服务器后：

（1）发送用户信箱名

这个命令是让客户端程序给服务器发送用户信箱名的。

语法：`user <username> <crlf>`

假如我们发送这样的命令：

```
Winscok.SendData "user" & " " & "busyzhong" & vbCrLf
```

如果对话成功，服务器的返回信息是这样的。

```
+OK
```

如果对话失败（账号不存在等原因），服务器的返回信息是这样的：

```
-ERR
```

（2）发送信箱密码

这个命令是让客户端程序给服务器发送信箱密码的。

```
语法：pass <password> <crlf>
Winscok.SendData "pass " & " " & "123321" & vbcrlf
```

如果在这个步骤中的密码与信箱不匹配，则不是简单地再发送一个 pass 命令，而必须要重新进行发送信箱名的上一步骤。

（3）对信箱邮件进行操作

该阶段称为事务状态。在这一个阶段，有许多 POP3 的命令可以使用，总结下来可以分为下面几类：

- 取得信箱及邮件状态的命令

  Stat：提供信箱大小信息。

  List：提供邮件大小信息。

  Uidl：取出邮件的唯一标识符
- 取得邮件内容的命令

  Retr：从服务器取出邮件。

  Top：取出信头和邮件的前 N 行。
- 对邮件进行操作的命令

  Dele：标记邮件被删除。

  Rset：复位 POP 会话。

（4）接收邮件完毕，结束 POP3 对话

```
语法：QUIT <CrLf>
Winscok.SendData "quit" & vbcrlf
```

5.6　接收 E-mail 的程序

5.6.1　建立工程项目

读者可以通过下面的实例进一步了解 POP3 的有关原理和内容，该实例只是简单地介绍接收邮件、提取信头标题字段的有关内容，解释符合 RFC822 规范的邮件。由于目前邮件结构非常复杂，读者看到的是收信的原始信息，只是将标题字段和邮件体（正文）简单地分开。

本工程项目一共由一个窗体和两个类模块构成：窗体 frmMain、类模块 Cmessage、类模块 Cmessages。

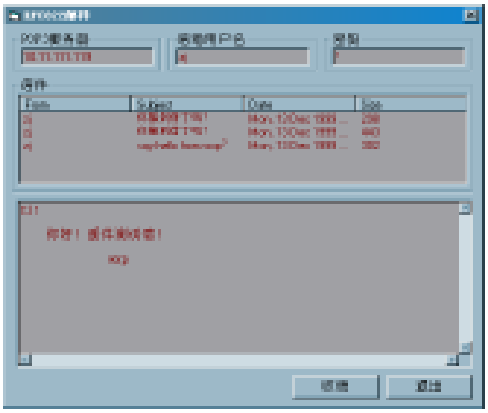


图 5-10　接收邮件程序运行界面

5.6.2　代码分析

1. **窗体** frmMain

窗体 frmMain **是用于管理接收和查看邮件的界面。运行界面如图** 5-10 **所示。**

```
'定义 POP3 会话状态的枚举类型
Private Enum POP3States
    POP3_Connect
    POP3_USER
    POP3_PASS
    POP3_STAT
    POP3_RETR
    POP3_DELE
    POP3_QUIT
End Enum
'表示当前 POP3 会话的枚举变量
Private m_State        As  POP3States
```

首先定义 POP3 **会话过程的枚举类型及变量，在与** POP3 **服务器会话过程中，设置该变量并依此判断处于哪个会话阶段。**

```
Private m_oMessage     As CMessage
Private m_colMessages   As New CMessages
```

上面代码定义、分析和存储当前接收的邮件类变量 m_oMessage，m_colMessages **是用于存储当前** POP3 **会话的所有消息（邮件）的类对象。**

```
Private Sub cmdCheckMail_Click()
'检查除了 txtBody 之外的 TextBox 是否为空
For Each c In Controls
    If TypeOf c Is TextBox And c.Name <> "txtBody" Then
```

```
        If Len(c.Text)=0 Then
            MsgBox c.Name & " can't be empty", vbCritical
            Exit Sub
        End If
    End If
Next
'
'改变表示当前状态的变量
m_State=POP3_Connect

'在打开一个新的会话之前先关闭 Winsock
Winsock1.Close

'指定服务器地址和端口并连接服务器
Winsock1.RemotePort=110
Winsock1.RemoteHost=txtHost.Text
Winsock1.Connect
End Sub
```

上面的代码用于响应“cmdCheckMail**”按钮的单击事件，判断是否填写了必要的信息，再连接** POP3 **服务器，并标识当前的会话状态处于连接（**POP3_Connect**）。**

```
Private Sub cmdDel_Click()
Unload Me
End Sub
```

上面的代码响应“cmdDel**”按钮的单击事件，退出程序。**

```
Private Sub lvMessages_ItemClick(ByVal Item As ComctlLib.ListItem)
    txtBody=m_colMessages(Item.Key).MessageBody


End Sub
```

上面代码用于响应“lvMessages**”中的** ListItem **对象单击事件，在** txtBody **显示相应的信件体内容。**

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
Dim strData As String
Static intMessages         As Integer '记录信箱的邮件数
Static intCurrentMessage    As Integer '当前下载的邮件
Static strBuffer           As String  '接收消息的字符串变量
'
'从 Winsock 接收缓冲区中读取数据
Winsock1.GetData strData
Debug.Print strData
If Left$(strData, 1)="+" Or m_State=POP3_RETR Then
```

```
    '如果接收字符串的第一个字符为“+”，表示服务器接收了来自客户端的请求命令。
    'If the first character of the server's response is "+" then
    '如果接到的第一个字符为“-”，则表示服务器认为来自客户端的请求是错误的。
    '如果会话状态正处于接收邮件，则不需要判断第一个字符。
       Select Case m_State
          Case POP3_Connect
             '
             '将消息（邮件）记数设置为 0 Reset the number of messages
             intMessages=0
             '
             '改变当前会话状态为发送用户名 Change current state of session
             m_State=POP3_USER
             '
             '发送 USER 及其参数。
             Winsock1.SendData "USER " & txtUserName & vbCrLf
             Debug.Print "USER " & txtUserName
          Case POP3_USER
             '
             '如果能够执行到这一步，表示发送给服务器的用户名已经通过服务器的验证
             '改变当前会话状态为发送密码阶段
             m_State=POP3_PASS
             '发送密码
             Winsock1.SendData "PASS " & txtPassword & vbCrLf
             Debug.Print "PASS " & txtPassword
          Case POP3_PASS
             '
             ' Change the state of the session
             '如果能够执行到这一步，表示发送给服务器的密码已经通过服务器的验证
             '改变当前会话状态为取得邮箱信息阶段
             m_State=POP3_STAT
             '
             '发送 STAT 命令，要求服务器返回邮箱信息
             Winsock1.SendData "STAT" & vbCrLf
             Debug.Print "STAT"
          Case POP3_STAT
             '
             '从接收的从服务器返回的邮箱状态字符串中获得邮件数量
             intMessages=CInt(Mid$(strData, 5, _
                         InStr(5, strData, " ") - 5))
             If intMessages > 0 Then
```

```
            '如果邮件的数量>0，设置当前会话状态为取回邮件阶段
            m_State=POP3_RETR
            '设置变量表明当前取回的是哪个邮件
            intCurrentMessage=intCurrentMessage + 1
            '
            '发送 RETR 及参数取回第一封邮件
            Winsock1.SendData "RETR 1" & vbCrLf
            Debug.Print "RETR 1"
        Else
            '如果是处于其他状态，设置当前状态为退出会话阶段
            m_State=POP3_QUIT
            '发送 QUIT 命令退出会话过程
            Winsock1.SendData "QUIT" & vbCrLf
            Debug.Print "QUIT"
            MsgBox "You have not mail.", vbInformation
        End If
    Case POP3_RETR
        '如果执行到这个阶段，说明邮箱中至少存在一封以上的邮件
        '循环执行该过程，以取得邮件的内容
        strBuffer=strBuffer & strData
        '
        If InStr(1, strBuffer, vbLf & "." & vbCrLf) Then
            '如果接收到的邮件内容的字符串中存在 vbLf & "." & vbCrLf 表示
            '使用 RETR 返回的邮件数据已经完毕
            '
            '从服务器接收到邮件的数据中将响应行去掉
            strBuffer=Mid$(strBuffer, InStr(1, strBuffer, vbCrLf) + 2)
            '
            '从邮件的数据中去掉最后的 vbLf & "." & vbCrLf
            strBuffer=Left$(strBuffer, Len(strBuffer) - 3)
            '
            '创建 CMessage 分析取得的邮件数据
            Set m_oMessage=New CMessage
            m_oMessage.CreateFromText strBuffer
            '将分析结果加入到 m_colMessages 对象的集合中
            m_colMessages.Add m_oMessage, m_oMessage.MessageID
            Set m_oMessage=Nothing
            '
            '清除接收邮件数据的字符串
            strBuffer=""
```

```
                '判断是否已经取完邮件
                If intCurrentMessage=intMessages Then
                    m_State=POP3_QUIT
                    '退出 POP3 会话
                    Winsock1.SendData "QUIT" & vbCrLf
                    Debug.Print "QUIT"
                Else
                    intCurrentMessage=intCurrentMessage + 1
                    m_State=POP3_RETR
                    '发送 RETR 命令取回下一封信
                    Winsock1.SendData "RETR " & _
                    CStr(intCurrentMessage) & vbCrLf
                    Debug.Print "RETR " & intCurrentMessage
                End If
            End If
        Case POP3_QUIT
            '关闭 Winsock 连接
            Winsock1.Close
            '调用该函数将邮件内容加入到 ListView 中
            Call ListMessages
    End Select
Else

    '显示关闭 Winsock 连接出错并提示
    Winsock1.Close
    MsgBox "POP3 Error: " & strData, _
    vbExclamation, "POP3 Error"
End If
End Sub
```

以上代码是当 Winsock 接收到连接的服务器的数据被触发，按照一般的 POP3 会话过程，首先从接收到的数据判断响应是否正确，如果正确，设置有关变量并发送命令进入下一个会话状态。如果服务器返回的响应表示出错，则退出会话过程并关闭 Winsock。

```
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String,
ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal
HelpContext As Long, CancelDisplay As Boolean)

MsgBox "Winsock Error: #" & Number & vbCrLf & _
       Description

End Sub
```

以上代码用于 Winsock 出错时显示提示信息。

```
Private Sub ListMessages()
'将邮件的主题，发送日期，大小加入到 ListView 中
Dim oMes As CMessage
Dim lvItem As ListItem
For Each oMes In m_colMessages
    Set lvItem=lvMessages.ListItems.Add
    lvItem.Key=oMes.MessageID
    lvItem.Text=oMes.From
    lvItem.SubItems(1)=oMes.Subject
    lvItem.SubItems(2)=oMes.SendDate
    lvItem.SubItems(3)=oMes.Size
Next

End Sub
```

以上代码用于将 POP3 会话取回的邮件（存放在类对象 m_colMessages 中）加入到 ListView 中。

2. **类模块 CMessage**

这个类模块用于分析从 POP3 服务器接收的邮件原始数据中，提取标题字段或其他邮件信息，并将信头与信体分隔存放到变量中。同时提供了外部程序访问或设置这些信息的接口。分析的过程比较简单，并且只能分析最简单的邮件内容（符合 RFC822，下一节会介绍），所以实用性不是很强，但是可以让读者从中了解这方面的原理。这部分代码不是很复杂，请读者自己分析。

```
'存储标题字段或其他邮件信息的变量
Private m_strReturnPath     As String
Private m_strReceived       As String
Private m_strSendDate       As String
Private m_strMessageID      As String
Private m_strMessageTo      As String
Private m_strFrom          As String
Private m_strSubject        As String
Private m_strReplyTo        As String
Private m_strSender         As String
Private m_strCC            As String
Private m_strBCC           As String
Private m_strInReplyTo      As String
Private m_strReferences     As String
Private m_strKeywords       As String
Private m_strComments       As String
```

```
Private m_strEncrypted      As String
Private m_strMessageText    As String
Private m_strMessageBody    As String
Private m_strHeaders        As String
Private m_lSize             As Long

'从接收到的邮件字符串中提取有关信息
Public Sub CreateFromText(strMessage As String)

Dim intPosA        As Integer
Dim vHeaders       As Variant
Dim vField         As Variant
Dim strHeader      As String
Dim strHeaderName  As String
Dim strHeaderValue As String

intPosA=InStr(1, strMessage, vbCrLf & vbCrLf)
If intPosA Then
   m_strHeaders=Left$(strMessage, intPosA - 1)
   m_strMessageBody=Right$(strMessage, Len(strMessage) - intPosA - 3)
   m_strMessageText=strMessage
Else
   Err.Raise vbObjectError + 512 + 101, "CMessage.CreateFromText", _
             "Invalid message format"
   Exit Sub
End If

vHeaders=Split(m_strHeaders, vbCrLf)
For Each vField In vHeaders
   strHeader=CStr(vField)
   intPosA=InStr(1, strHeader, ":")
   If intPosA Then
      strHeaderName=LCase(Left$(strHeader, intPosA - 1))
   Else
      strHeaderName=""
   End If
   strHeaderValue=Trim$(Right$(strHeader, Len(strHeader) - intPosA))
   '提取有关信头标题字段
   Select Case strHeaderName
      Case "return-path"
```

```
            m_strReturnPath=strHeaderValue
        Case "received"
            m_strReceived=strHeaderValue
        Case "from"
            m_strFrom=strHeaderValue
        Case "sender"
            m_strSender=strHeaderValue
        Case "reply-to"
            m_strReplyTo=strHeaderValue
        Case "to"
            m_strMessageTo=strHeaderValue
        Case "cc"
            m_strCC=strHeaderValue
        Case "bcc"
            m_strBCC=strHeaderValue
        Case "message-id"
            m_strMessageID=strHeaderValue
        Case "in-reply-to"
            m_strInReplyTo=strHeaderValue
        Case "references"
            m_strReferences=strHeaderValue
        Case "keywords"
            m_strKeywords=strHeaderValue
        Case "subject"
            m_strSubject=strHeaderValue
        Case "comments"
            m_strComments=strHeaderValue
        Case "encrypted"
            m_strEncrypted=strHeaderValue
        Case "date"
            m_strSendDate=strHeaderValue
    End Select
Next

End Sub

Public Function CombineMessage() As String

End Function
```

```
Public Property Let Headers(ByVal vData As String)
    m_strHeaders=vData
End Property

Public Property Get Headers() As String
    Headers=m_strHeaders
End Property

Public Property Let MessageBody(ByVal vData As String)
    m_strMessageBody=vData
End Property

Public Property Get MessageBody() As String
    MessageBody=m_strMessageBody
End Property

Public Property Let MessageText(ByVal vData As String)
    m_strMessageText=vData
End Property

Public Property Get MessageText() As String
    MessageText=m_strMessageText
End Property

Public Property Let Encrypted(ByVal vData As String)
    m_strEncrypted=vData
End Property

Public Property Get Encrypted() As String
    Encrypted=m_strEncrypted
End Property

Public Property Let Comments(ByVal vData As String)
    m_strComments=vData
End Property

Public Property Get Comments() As String
    Comments=m_strComments
End Property
```

```
Public Property Let Keywords(ByVal vData As String)
    m_strKeywords=vData
End Property

Public Property Get Keywords() As String
    Keywords=m_strKeywords
End Property

Public Property Let References(ByVal vData As String)
    m_strReferences=vData
End Property

Public Property Get References() As String
    References=m_strReferences
End Property

Public Property Let InReplyTo(ByVal vData As String)
    m_strInReplyTo=vData
End Property

Public Property Get InReplyTo() As String
    InReplyTo=m_strInReplyTo
End Property

Public Property Let BCC(ByVal vData As String)
    m_strBCC=vData
End Property

Public Property Get BCC() As String
    BCC=m_strBCC
End Property

Public Property Let CC(ByVal vData As String)
    m_strCC=vData
End Property

Public Property Get CC() As String
    CC=m_strCC
End Property
```

```
Public Property Let Sender(ByVal vData As String)
    m_strSender=vData
End Property

Public Property Get Sender() As String
    Sender=m_strSender
End Property

Public Property Let ReplyTo(ByVal vData As String)
    m_strReplyTo=vData
End Property

Public Property Get ReplyTo() As String
    ReplyTo=m_strReplyTo
End Property

Public Property Let Subject(ByVal vData As String)
    m_strSubject=vData
End Property

Public Property Get Subject() As String
    Subject=m_strSubject
End Property

Public Property Let From(ByVal vData As String)
    m_strFrom=vData
End Property

Public Property Get From() As String
    From=m_strFrom
End Property

Public Property Let MessageTo(ByVal vData As String)
    m_strMessageTo=vData
End Property

Public Property Get MessageTo() As String
    MessageTo=m_strMessageTo
End Property
```

```
Public Property Let MessageID(ByVal vData As String)
    m_strMessageID=vData
End Property

Public Property Get MessageID() As String
    MessageID=m_strMessageID
End Property

Public Property Let SendDate(ByVal vData As String)
    m_strSendDate=vData
End Property

Public Property Get SendDate() As String
    SendDate=m_strSendDate
End Property

Public Property Let Received(ByVal vData As String)
    m_strReceived=vData
End Property

Public Property Get Received() As String
    Received=m_strReceived
End Property

Public Property Let ReturnPath(ByVal vData As String)
    m_strReturnPath=vData
End Property

Public Property Get ReturnPath() As String
    ReturnPath=m_strReturnPath
End Property

Public Property Get Size() As Long
    Size=Len(m_strMessageText)
End Property
```

3. **类模块** CMessages

该类模块用于存储或读取分析邮件内容。

```
'保存消息类的集合变量
```

```
Private mCol As Collection
Public Sub Add(oMessage As CMessage, Optional sKey As String)
If Len(sKey)=0 Then
    mCol.Add oMessage
Else
    mCol.Add oMessage, sKey
End If
End Sub
Public Property Get Item(vntIndexKey As Variant) As CMessage
    Set Item=mCol(vntIndexKey)
End Property
Public Property Get Count() As Long
    Count=mCol.Count
End Property

Public Sub Remove(vntIndexKey As Variant)
    mCol.Remove vntIndexKey
End Sub

Public Property Get NewEnum() As IUnknown
    Set NewEnum=mCol.[_NewEnum]
End Property

Private Sub Class_Initialize()
    Set mCol=New Collection
End Sub

Private Sub Class_Terminate()
    Set mCol=Nothing
End Sub
```

5.7 信件结构详述

5.7.1 RFC822 信件的格式和内容

电子邮件的主要部分是信件，信件的内容是 ASCII **码构成的一系列字符，就是通常所说的文本文件。文本的构成都有一定的格式和规范。无论使用** SMTP **或者是** POP3**，了解其信件的基本格式是非常重要的。对于** POP3 **而言，从** POP3 **接收的信件是没有经过整理的，有**

关信件的一些信息，如发送人、主题、时间、内容都包含在取回的文本中，而一个接收邮件的程序，不仅仅能够把信件从服务器上取回来，而且必须能够对取回来的内容进行分类整理，最终让用户所看到的是经过处理过的有条理的信件内容。这一点对于 SMTP 而言，同样很重要，发送的信件要使得对方接收的客户端软件收到信件时，能够识别信件所包含的信息。

如果发送和接收 E-mail 的软件只有一种，并且都是同一个开发小组编写的，不考虑升级的因素，那么信件的格式可以自己定义。但这在网络世界里是不可能和不现实的。许多程序要求对电子邮件进行编制特定的其他功能，以满足需要。和其他客户机/服务器的应用程序一样，指定规范的协议是必须的，根据协议编写的无论是发送还是接收 E-mail 的软件，彼此在进行邮件交流时不会存在障碍。以上所说的 SMTP 和 POP3 协议只不过 Internet E-mail 协议的一部分，它给出了与 SMTP 服务器或 POP3 服务器进行邮件对话的规范。同样信件的格式也有协议。

最早规定电子邮件核心结构的是在 1982 发表的 RFC822，它是 Internet E-mail 信件的当前标准。RFC822 定义了信件从主机传送到主机时需要的格式化方式。它的主要用途是为信件提供规范化的格式。这样使得不同类型的网络可以相互传送电子邮件。

1989 年发表的 RFC1123 含有对许多 Internet 标准的重要更新。在本节中主要介绍 RFC822 制定的格式标准。我们先以一封从 POP3 服务器上接收的信件来分析其基本格式和内容。

```
Date: Tue, 04 Dec 2001 16:19:02 +0800
From: MDaemon@company.mail
Subject: Welcome to the email system for domain company.mail
To: wang@company.mail
Hello wang!
Say Hello From busyzhong
```

从直观上看，信件非常简单，就是一系列的文本行，每一行以回车换行（vbCrlf）结束。由 ASCII 字符组成。每一行的长度和信件的长度在 RFC822 中没有规定，但是值得注意的是，与信件有关的大多数长度限制在 SMTP 的 RFC 中进行了相当规定。这些限制决定了 E-mail 程序在产生信件时的最大长度。

在 SMTP 中规定了每一行最多能有 1000 个字符，包括行终止符。为了增加可读性，每一行应该少于 80 个字符。

对于信件的行数没有特别的限制，但一些软件包和一些传送机制（如 SMTP）有它们的限制。

上面的信件可以说是结构最简单的信件了，主要有 3 部分组成，信头：

```
Date: Tue, 04 Dec 2001 16:19:02 +0800
From: MDaemon@company.mail
Subject: Welcome to the email system for domain company.mail
To: wang@company.mail
```

空白行：

```

```

信件体：

```
Hello wang!
Say Hello From busyzhong
```

信件结构可以说分为两大部分：信头和信件体。空白行通常分离各个元素，以便于进行分析，在信件头和信件体之间有一个空白行，用于分隔信头和信件的其余部分。信头是必需的，信体是可选的，如果存在信体，空白行是必需的，如果只有信头，空白行就是可选的。在信体中，通常也有空白行，起的作用也是一样的。

这样设计的信件便于进行语法分析，提取信件的基本信息。

1. 信件体

首先描述信件体，在 RFC822 中，信件体比信件头简单，只是一系列的文本行，并没有附加的结构或含义。

可能读者要问，要是发送附件怎么办？遗憾的是在 RFC822 中，没有关于描述发送附件的标准，其实附件内容是包含在信体中的。

在 Internet 早期广泛使用的发送附件的约定是使用编码程序，把二进制数据转换成一个对电子邮件传送安全的格式。至今在许多电子邮件软件中可以看到 Uuencode 发送附件这一选项，因为 Uuencode 编码是当时常用的将二进制编码转译成可打印的 US-ASCII 字符的技术。遗憾的是，Uuencode 在 RFC822 中没有定义。这里着重介绍 RFC822 所规定的信件格式，对发送附件有兴趣的读者可以阅读后面的专门介绍有关邮件附件的内容。

2. 信头

（1）字段

信头的结构比较复杂，总的来说是由一些字段组成。这些字段为用户和程序提供了关于信件的信息。了解信头其实就是弄清楚信头的各个字段。

每个信头字段由一行或多行文字组成，对于跨多行的字段，附加行以一个空格开始作为续行。每个信头字段由以下部分组成：字段名称，可选的空格，一个冒号，可选的注解空格和一个可选的字段体。字段体也可以含有前导空格。

```
Field=field-name *WSP ":" [[CFWS] field-body] CRLF
```

例如一个 Subject 字段的描述：

```
subject:  say hello from busyzhong!
```

在字段名和冒号之间通常没有空格。

Field-name（字段名）由一系列可以打印的 US-ASCII 字符组成，除空格和“:”外，大多数字段的字段名称由一系列字母、数字组成，中间经常插入横线符。

```
From:
To:
X-Mailer:
```

RFC882 为信件定义了一些标准字段，表 5-5 中列出了这些字段。

表 5-5　　标准的信头字段

字段名称	描述
From	写信人
Sender	信件发信人
Reply-to	发送回复的地址
To	信件的主收信人
Cc	信件的辅收信人（抄送人）
Bcc	信件的密件抄送人
Message-ID	信件的唯一标识符
In-Reply-To	信件正被回复到
References	所有的信件来源
Date	信件的创建日期
Received	MTA 轨迹
Return-Path	发信人地址
Subject	信件主题
Comments	关于信件的其他说明
Keywords	与该信件有关的主题关键字
Encrypted	加密信息（过时）
Resent-*	重新分发时创建的字段
X-*	扩展字段

字段名称是相当直观的，但字段体就是另外一回事了。以上了解到的字段体由一系列 ACSII 字符组成，但是其空格、加括号的注释、引号和多行字段都比较复杂，上面没有进行描述。另外，字段体的概念依赖于字段名称，每个类型的字段有特定的格式。

（2）要求的字段

在 RFC822 中定义的 20 多个字段中，只有少数几个信头字段是实际要求必需的。信件必须使用 Date 或 Resent-Date 字段指定创建信件的日期，还必须使用 From 字段指定创建该信件的人或程序的信箱。

REC822 还要求有一个收信人的字段，可以是 To、Cc 或 Bcc，或者与它的 Resent-*等效的字段。

除了发送信件中要求的信头，处理信件的 MTA 必须在每个信头的开始增加一个 Received 字段，这就是我们通常在许多信件的开始部分看到多个 Received 字段的原因。

（3）字段的建议顺序

除了几个例外，信头中的字段不要求任何特定的顺序。这些例外是 Received、Return-Path 和 Resent-*字段。在信件经过一系列的 MTA 时，每个 MTA 在信件的开始增加一个 Received 字段。Return-Path 是只在最后一个 MTA 投递前在信件开始增加的字段。因为这些字段为诊断问题提供跟踪信息，所以它们的位置必须保持不变。给信件增加的任何 Resent-*字段也必须放在信件的开始。

RFC822 建议以下顺序：

```
Date
From
Subject
To
```

```
Cc
……
```

在实际使用中，没有强迫要求使用这种顺序。事实上，浏览任意数量的信件可以发现，许多信件并不是按照这个顺序进行的。尽管 MTA、邮件列表处理器和 MDA 插入的字段通常是放在信件的开始或最后，但其他字段的顺序通常不是固定。

（4）字段的多次出现

RFC822 允许任何字段多次出现。但除了 Received 和 Resent-*字段外，其他字段多次出现是没有必要的，所以字段最好只出现一次，否则给分析信件的语法结构增加困难。试想，如果一封信件中出现了两个 Subject 字段，那么无法确认发信者的用意是使用哪个作为主题。

一般来说，如果同一字段出现多次，处理方法有两种，可以使用第一个出现的字段内容，也可以使用最后一次遇到的字段内容。通常可以根据所涉及的字段，使用两个方法的混合。

（5）结构化字段和非结构化字段

由于每个字段所包含的信息是不一样的，下面会对一些较常用的字段进行详细说明。在详细说明前，先来看看字段的大致分类。

总的来说字段可以分为两大类：结构化字段和非结构化字段。

结构化字段有一个特定的格式，由语法分析程序检测。Sender 字段是一个很好的例子，它的字段内容是信箱，有一个离散的结构。

非结构化的字段含有任意的数据，没有固定的格式。例如，Subject 字段含有任意的文字，但这个文字没有固定格式。非结构化的字段只有 Subject、Comments、扩展字段、非标准字段和 In-Reply-To 和 References 的一些实例。所有其他字段都是结构化的。

3. 字段的元素

尽管信件的高层结构是非常简单的，但一些字段的结构是复杂的。下面简单介绍一下大多数字段共有的几个元素。

（1）空白

信件其实就是一个文本文件，与文本文件中常见的空白一样，主要有一个空格（ASCII32），或 Tab（ASCII9）组成。此外，Crlf 所形成的回车换行在这里也应算是空白。

在字段中使用空白对信件的格式化有着非常重要的作用。例如每个字段间用 Crlf 形成的空白进行分离。字段内有时利用空格形成的空白来分隔字段名和字段内容，例：

```
Subject: Say hello from busyzhong!
```

在 Subject 后面的冒号和内容之间插入空格字符，使得结构更加清晰。

在 E-mail 中，出现空白的地方很多，并且没有固定的规则，所以要正确地处理所有空白是很困难的事情。最好是在形成 E-mail 信件时，仅在需要时才形成空白，这样可以方便接收软件进行语法分析。

（2）注解

注解是由括号括起来的一系列字符。

```
(comments)
```

在注解被解释的地方，可以用一个空格字符代替，而且什么也不会破坏。这并不是说注解不含有对信件阅读者有用的信息，而是在考虑语义含义时，不需要注解。注解只是通常用于增加不是字段正式定义部分的附加信息。

注解仅在不是结构化字段中进行解释，它们可以出现在其他地方，但不作为注解解释。此外，在加引号的字符串中的注解也不必考虑。如：

```
"Say Hello (From busyzhong) !"
```

由于字符串是在引号里，所以没有注解。如果为：

```
Say Hello (From busyzhong)!
```

那么去掉注解后字段被解释为：

```
Say Hello !
```

注解也可以嵌套，这意味着注解里面可以含有其他注解。如：

```
Say Hello (From busyzhong(and busywxp))!
```

其中(and busywxp)为嵌套在注解里的注解。

嵌套注解会使分析 E-mail 信件语法结构变得困难，这里建议不要使用嵌套注解。

（3）字段折叠

为了增加可读性，对于比较长的字段，通常要分割成几行，形成折叠。在结构化和非结构化字段中都允许折叠。

折叠是由在字段中某些策略点插入的 Crlf 和一个或多个空白字符组成，第一行后的行称为续行。

下面是带折叠的一个例子：

```
To: busywxp@cocim.com, busyant@cocim.com,
 Busyzhong@cocim.com
```

当然，进行折叠的位置是随便的，上例也可以这样：

```
To: busywxp@cocim.com,
 busyant@cocim.com,Busyzhong@cocim.com
```

如果没有折叠，原句是这样的：

```
To: busywxp@cocim.com, busyant@cocim.com,Busyzhong@cocim.com
```

在分析字段折叠的语法时，要把一个折叠字段压缩成一行进行展开，这是通过将折叠空白符 Crlf 转换成空格字符实现的。

（4）字段大小写

字段名称是不区分大小写的，所以将 Subject 写成 subject 或 SUBJECT 都是一样的。

不过字段名称大小写是有常用形式的，如主题的大小写形式通常为 Subject，字段 Message-Id 常常规范为 Message-ID，字段 Mime-Version 则为 MIME-Version。不过字段名称不区分大小写，所以怎样写是无关紧要的。

与字段名称相比，字段体的大小写要稍微复杂点。一般来说，字段体的大小写是没有关系的，但在有些地方是必须注意的。如：

```
Subject 字段体
```

```
Comments 字段体
加引号的字符串
注解
地址的 Local-part 部分
```

4. 标准字段

在电子邮件中会发现许多不同字段名称，不过大多数字段在许多时候可以被忽略。标准字段在电子邮件标准中的数目是比较少的。

下面详细介绍 RFC822 中定义的字段，以及其他经常遇到的字段。

（1）From

From 字段用于表述产生这个信件的人，通常含有一个信箱。如：

From: busyzhong@cocim.com

如果字段中提供几个信箱，该信件必须包含一个 Sender 字段。

（2）Sender

Sender 字段用于指示信件发送者与信件创建者是否不同。如：

```
From: busyzhong@cocim.com
Sender: busyant@cocim.com
```

这个字段似乎是多余的，但其实并不是这样。例如，当信件由一个非原始信件创建者发送时，可以使用它。如秘书以经理的名义发送信件时可以用。

RFC822 规定，如果在 From 中有多个字段，必须使用 Sender 字段，以帮助标识信件的创建者。

（3）Reply-To

Reply-To 字段用于控制信件要回复的目的地。如：

```
From: busyzhong@cocim.com
Reply-To: busyzhong@holiway.com
```

该字段可以重设 From 字段，上面表示回复地址为 busyzhong@holiway.com，而不是 busyzhong@cocim.com。

以上是与发信方有关的字段。

（4）To

To 字段表示信件的主要收信人。如：

```
To: busywxp@cocim.com
```

也可以带多个地址：

```
To: busywxp@cocim.com , busywang@cocim.com
```

中间用逗号隔开。

（5）Cc

Cc 字段用于指定信件的辅收信人，也可以说是抄送人。该字段的用法与 To 字段的用法一致。

（6）Bcc

Bcc 为密件抄送（blind carbon copy）的缩写。与 Cc 类似，只是在 To 和 Cc 字段中指定的收信人不在 Bcc 字段中出现。

以上（4）～（6）点为与收信方有关的字段。

（7）Message-ID

Message-ID 字段用于表示一个信件唯一标识，该字段通常有 SMTP 服务器生成。如：

```
Message-ID: <MDAEMON30003200112041619.AA1902015@company.mail>
```

这个值通常是唯一的。形式根据使用的软件而定。通常左边是标识符，右边指定计算机名。

（8）Date

Date 字段含有电子邮件创建的日期和时间。

```
Date: Tue, 04 Dec 2001 16:19:02 +0800
```

（9）Received

Received 字段为信件的一个特定的邮件投递服务器的记录。

```
Received: from company.mail [127.0.0.1] by company.mail [127.0.0.1] with RAW
(MDaemon.v2.7.SP3.T) for <wang@company.mail>; Tue, 04 Dec 2001 16:19:02 +0800
```

处理邮件投递的每个服务器必须给它处理的每个信头的前面加一个 Received 字段，用以描述信件到达目的地所经过的路径。当跟踪各种电子邮件问题时，这个信息是非常有用的。

（10）重发字段

指把收到的信件重发给另一组收信人的重新分发技术。这种技术保持整个原始信件不变，只是简单地产生重发信件所要求的新版本的字段。添加到重新分发信件的字段都是以字符串 Resent-开始的，避免与以前的字段相混。下面是所有合法的 Resent-*字段：

```
Resent-From
Resent-Sender
Resent-To
Resent-Cc
Resent-Bcc
Resent-Date
Resent-Message-ID
Resent-Reply-To
```

（11）Subject

Subject 字段用于描述信件的主题

```
Subject: Say hello from busyzhong!
```

当回复信件时，通常在主题前面增加“Re:”前缀，标记该信件为回复信件：

```
Subject: Re:Say hello from busyzhong!
```

在信件被转发时，通常在主题文字前面加上“Fw:”、“Fwd:”这样的前缀：

```
Subject: Fw:Say hello from busyzhong!
```

（12）Comments

Comments 字段用于把一个注解添加到信件中。

```
Comments: This is a important email!
```

（13）扩展字段

如果要加入到信头中的信息在 RFC 中有规定的相应字段，这时需要创建新的非标准字段。RFC822 中规定了这种情况使用扩展字段解决，同时在字段名称前面加上前缀 X-。

扩展字段允许创建新的字段，而不会出现该字段名称在以后的标准被使用的矛盾。事实上，有许多扩展字段已经被广泛使用，但没有标准定义。

下面介绍两个相对常见的扩展字段：

◆ X-Loop

X-Loop 字段常常通过过滤器和邮件列表程序来防止邮件循环。过滤器或邮件列表处理程序，可以给它处理的每个信件增加一个 X-Loop 字段，如果它遇到在这个字段中含有特别值的信件，就假定该信件有一个循环，从而以不同的方式处理该信件。

◆ X-Mailer

X-Mailer 字段用于指示什么样的程序产生这个信件，它是使用最广泛的扩展字段。

```
X-Mailer: BigFox Mailer V1.0
```

邮件产生和发送软件的开发者应该让他们的软件为所有发出信件增加合适的 X-Mailer 字段，字段不仅要含有软件的名称，而且最好应该有版本号。例如设计的软件为 LittleFox Mailer，版本为 V1.0，可以将“X-Mailer: LittleFox Mailer V1.0”加到邮件信头中去。

5.7.2 构造符合 RFC822 的信件

1. 最简单的信件

从信件的格式和内容中我们了解了信件的基本构成，发送电子邮件程序的不仅要与 SMTP 服务器进行邮件会话，而且在发送之前还要进行邮件的构造。

从以上已经知道，信件主要分为两大部分：信头和信体。在两部分之间用空白行隔开。

先构造信头，信头的必需字段有：一个 Date 字段，一个 From 字段，最少一个收信人字段。

构造日期字段：

```
Date: Tue, 04 Dec 2001 16:19:02 +0800
```

构造发信人字段：

```
From:busyzhong@cocim.com
```

构造主题字段：

```
Subject: Say hello from busyzhong!
```

构造收信人字段：

```
To: busywxp@holiway.com
```

接着构造信体，RFC822 **指定的信体结构非常简单，是接收人能看到的信件正文。**

```
Busywxp:
       This is hello from busyzhong!
                Busyzhong
```

在信头和信体之间插入空白行，所得到的信件如下：

```
Date: Tue, 04 Dec 2001 16:19:02 +0800
From:busyzhong@cocim.com
Subject: Say hello from busyzhong!
To: busywxp@holiway.com
Busywxp:
       This is hello from busyzhong!
                Busyzhong
```

在接收到 SMTP **会话中的** Data **命令之后，将构造好的信件发送给服务器。**

2. **构造复杂信件**

其实知道了信件的基本结构，就可在此基础上扩展成复杂信件，信头可以根据需要加入字段。在每个字段中，也可以根据需要加入更多的内容。

下面给出一个例子：

```
Received: from company.mail [127.0.0.1] by company.mail [127.0.0.1] with RAW
(MDaemon.v2.7.SP3.T) for <wang@company.mail>; Tue, 04 Dec 2001 16:19:02 +0800
Date: Tue, 04 Dec 2001 16:19:02 +0800
From: MDaemon@company.mail
X-MDSend-Notifications-To: [trash]
Subject: Welcome to the email system for domain company.mail
To: wang@company.mail
Reply-To: MDaemon@company.mail
Message-ID: <MDAEMON30003200112041619.AA1902015@company.mail>
Content-Type: text/plain; charset=us-ascii
X-Actual-To: wang@company.mail
X-Actual-From: MDaemon@company.mail
Welcome wang!
```

请注意信件中哪些部分是由发送电子邮件程序（MUA**）产生的，哪些是** SMTP **服务器等投递邮件服务器产生的字段。**

5.7.3 RFC822 信件的语法分析

信件的语法分析可以说是构造信件的逆过程，发送 E-mail 时，发送邮件程序要构造、发送基本的要发送的信件，以符合规范。同样在接收邮件之后，接收邮件的程序要进行邮件结构和语法的分析，从中提取必要的信息，使用软件的用户最终看到的不是接收下来的原始信件，而是经过处理的有条理的信件内容。

下面使用 VB 实现一个用于分析 RFC822 信件语法的程序，该程序并未考虑得十分周全，只是用于处理一些比较常见的信件结构。

首先考虑的问题是将存在折叠的字段展开，将跨多行的字段去掉折叠字符合成一个完整的字段，并在信头将其与其他字段分隔开来。

在字段的折叠中，通常折叠的字符为一行的开头是一个或多个空白字符（如空格 ASCII32 或 tab ASCII09），我们可以用这样的标志来判断下一行是否字段内容的延续，如果是，将折叠字符以空格字符代替。

然后对字段进行处理，将字段头和字段体（字段内容）分割。

再次进行字段内容的显示，并不是要将所有字段的信息都显示出来，下面的例程中感兴趣的只是发信人、收信人、主题和日期，其他字段内容可以不去理会。

最后提取信件的正文内容（信件体）。信件体与信头之间以空白行分开，这个特点可以将信头与信体区分开来。

VB 程序定义保存主题、写信人、收信人、日期和正文的变量，代码如下。

```
Dim mSubject As String
Dim mFrom As String
Dim mTo As String
Dim mDate As String
Dim mContent As String
```

定义处理过程 HandMail，该过程将信件中的主题、写信人、收信人、日期和正文信息提取出来。

```
Private Sub HandMail()
Dim mOpenfile As String
Dim Fnum As Integer
Dim strLine, strHeadItem, strHeadName, strHeadContent As String
Dim iFlag, Pos As Integer
'iflag=0 字段头，iflag=1 折叠字段，iflag=2 正文
mOpenfile=App.Path & "\rawmail.txt"
Fnum=FreeFile()
Open mOpenfile For Input As #Fnum
If Not EOF(Fnum) Then
   Line Input #Fnum, strLine
   strHeadItem=strLine
End If
iFlag=1
```

```
While Not EOF(Fnum)
   Line Input #Fnum, strLine
   If strLine <> "" And iFlag <> 2 Then
       '如果读进的一行不是空行,表示还是信头
       If Left(strLine, 1) <> Chr(32) And Left(strLine, 1) <> Chr(9) Then
          '如果一行的开始不是空白字符 space 或 tab,表示是下一个字段
          iFlag=0
       Else
          '否则是字段的折叠
          iFlag=1
       End If
    Else
       iFlag=2
    End If
    If strLine <> "" Then
       Select Case iFlag
          Case 0
          ' 处理字段

          Pos=InStr(1, strHeadItem, ":")
          If Pos > 0 Then
             strHeadName=Left(strHeadItem, Pos - 1)
             strHeadContent=Mid(strHeadItem, Pos + 1)
             Select Case LCase(strHeadName)

                Case "subject"
                   mSubject=strHeadContent
                Case "from"
                   mFrom=strHeadContent
                Case "to"
                   mTo=strHeadContent
                Case "date"
                   mDate=strHeadContent
             End Select
          End If
          strHeadItem=strLine
          Case 1
          '处理字段折叠
          strHeadItem=strHeadItem & strLine
          Case 2
```

```
            '处理正文
               mContent=mContent & strLine
         End Select
      End If
   Wend
   Close (Fnum)
   End Sub
```

5.8 MIME 编码解码与发送附件

5.8.1 RFC822 的局限性

RFC822 作为电子邮件的文本格式标准，于 1982 年产生后，在 Internet 上得到广泛的应用。随着 Internet 技术的发展和电子邮件的日趋广泛应用，RFC822 的局限性也越来越明显。

RFC822 是对文本消息的格式说明，并没有提到非文本消息，如语音、图像等多媒体数据以及其他二进制数据文件。而且 RFC822 限制电子邮件的文本只能采用 US ASCII 字符集，不允许使用其他的字符集。

值得庆幸的是，电子邮件协议的标准也在改变和扩充。多媒体电子邮件扩展标准 MIME 的出现解决了这些问题。可以说，正是由于 MIME 的出现，电子邮件的应用才比现在广泛得多。

5.8.2 Uuencode 编码与解码

在介绍 MIME 之前，先来回顾一下一些先驱技术。

（1）Uuencode 的编码理论

在 MIME 出现并得到广泛应用之前，也有过一些不是特定标准的解决发送二进制文件的方案，早期的 Uuencode 和 Uudecode 的使用，就是应用比较广泛的解决方案之一。

Uuencode 的 Uu 指用于 UNIX 间传送二进制文件，就是 UNIX to UNIX。在 E-mail 或 News 中 Uuencode 经常用于 Attach 二进制文件。目前不仅 Netscape、Eudora、MS-mail，甚至包括 Hotmail、USA.NET 之类的 Webmail 在内的绝大多数 E-mail 程序都支持 Uuencode 的编码方式和自动解码方式。

Uuencode 是将二进制数据编码成适合于电子邮件传送的文本格式，Uudecode 是相反的过程，把经过编码的文本译码成原始的二进制格式。

就是在今天许多电子邮件软件包括 Outlook，Foxmail 都还提供 Uuencode 和 Uudecode。下面先来看看 Uuencode 编码结果：

```
begin 600 Index.txt
M,3DY.2#-];?&(-:[L*[$L,GZR,LZ#0H-"C`Q("”_JK6)LNC,_`T*,#(@(+6Q
```

```
MRK&UQ-3"P<$-"C`S("")TW\/?#0HP-"`@UKNPKL2PR?K(RPT*,#4@(+#9Q.JY
MPKS%#0HP-B`@NONU^PT*,#<@(+G)T=O4QM',#0HP."`@NZG2N\GYUJZZ\PT*
M,#D@(,W&MZT-"C$P("")^J[+*6[#9RL*_R<#6N>.XYB;NBLW[]M;WS.+'^ET-
M"C$Q("#*V,W[PO/,[UNPV<3JN<*\Q2`@N>.VJ[#F70T*,3(@(-/*LNY;(+K[
,M?L@("")YX[:KL.9=[
`
end
```

编码文件由一个起始行、编码的数据和一个结束行组成。

其中起始行由 3 **个空格分隔的字组成：字符串“** begin **”、**UNIX **许可值（由于最早** Uuencode **是在** UNIX **系统上使用的，该值指示译码文件应该有什么样的许可权，在** Windows **下的编程这个值几乎没有什么用处）和文件名。**

Uuencode **技术其实是相当简单的，它把** 3 **字节组映射成为** 4 **个可打印的** US-ASCII **字符组。**

Uuencode **算法的关键在于将** 3 **个字节组映射成** 4 **个字节组的可打印字符（**3 **字节的** 8 **位** bit **转换到** 4 **个字节的** 6 **个** bit**）。**

如下所示：

有 3 **个相邻的字节，其中第一个字节从** a0~a8 **一共** 8bit**，第二、三类似。**

```
A7a6a5a4a3a2a1a0  b7b6b5b4b3b2b1b0  c7c6c5c4c3c2c1c0
```

经过 Uuencode **后形成** 4 **个字节的数据：**

```
  0 0 a7a6a5a4a3a2   0 0 a1a0b7b6b5b4   0 0 b3b2b1b0c7c6   0 0 c5c4c3c2c1c0
```

然后将这些字节转换成可打印的字符，即加上 20h（32**）。**

注意：如果是字节的值为 0**，那么加上的不是** 20h**（字符“　”），而是** 60h**（字符“‘”）所以如果转换的是下面的字节：**

```
  00010100  00001111  10101000
```

由 3 **个字节变成** 4 **个字节后结果是** 25 60 5e 48 **（”%`^h”），表示成二进制为：**

```
101     00000000  00111110  00101000
```

但是第二个字节的值为 0**，这时需要加上** 60h**，其他三个字节加上** 20h**。**

因此，经过编码，数据所包含的字符为 21h**（字符“ !”）至** 60h**（字符“`”）。**

很明显，从 Uuencode **的编码过程来看，编码后的文件要比编码前的文件大** 33%**左右，所以只有必要时才编码，对于已经是** ASCII **文本的文件，就再没有必要使用** Uuencode **了。**

在介绍了 3 **个字符到** 4 **个字符间的映射变换后，下面介绍具体文件的编码。**

编码文件的第一行以字符串“ begin **许可值 文件名”开始，接着就是编码数据行。**

编码数据行每行的第一个字节的值是这一行编码的字节数（以编码前的文件字节计算）。在编码时这个字节也要编码，规则是加上 20h**，如果为** 0 **则加上** 60h**。一行的长度从** 0~45**（**45+32(20h)**所代表的字符为“** M **”，因此经过编码后的文件的每一行的数据的第一个字符都为**’M’**，编码后的数据除了最后一行可能不是其他的都是每行** 60 **个字符（除去表示长度值的第一个字符外）。**

编码文件的最后是

end

两行文本。

（2）Uuencode 编码的 VB 实现

EncodeByte 是从 3 个字节到 4 个字节的映射函数，其中参数 mInByte 是一个 3 维的 byte 数组存放映射前的 3 个字节，mOutByte 是一个 4 维的数组，存放映射后的 4 个字节。两个参数都是 ByRef 传递的。

```
Private Sub EncodeByte(mInByte() As Byte, mOutByte() As Byte)
Dim tbyte As Byte
Dim i As Integer
tbyte=mInByte(0) And &HFC
mOutByte(0)=tbyte / 4
tbyte=((mInByte(0) And &H3) * 16) + (mInByte(1) And &HF0) / 16
mOutByte(1)=tbyte
tbyte=((mInByte(1) And &HF) * 4) + ((mInByte(2) And &HC0) / 64)
mOutByte(2)=tbyte
tbyte=(mInByte(2) And &H3F)
mOutByte(3)=tbyte

For i=0 To 3
   If mOutByte(i)=0 Then
      mOutByte(i)=&H60
   Else
      mOutByte(i)=mOutByte(i) + &H20
   End If
Next i
End Sub
```

函数 Uuencode 是编码函数，参数 Infile 是编码前文件名，参数 Outfile 是编码后的文件，OutName 用于指定编码文件中第一行的“ begin 6xx 文件名 ”。其中调用了上面的函数 EncodeByte。

```
Public Sub UuEncode(infile As String, Outfile As String, OutName As String)
Dim Fnum, Fnum2, i, j As Integer
Dim myStr As Byte
Dim mInLine(45) As Byte '存放读入一行的数据
Dim mOutLine(61) As Byte
Dim mInLineLen As Integer '读入一行的字节数
Dim mInByte(3) As Byte
Dim mOutByte(4) As Byte
Dim mOutB As Byte
Dim FinishPercent As Long
Fnum=FreeFile()
Fnum2=Fnum + 1
```

```
Dim TotalB, k As Long
Open Outfile For Binary As #Fnum2
Open infile For Binary As #Fnum
TotalB=LOF(Fnum)
Put #Fnum2, , "begin 600 " & OutName & vbCrLf
    '读入一行并进行编码
    k=0
While Not EOF(Fnum)
    mInLineLen=0
    i=0
    j=0
    While (Not EOF(Fnum)) And i < 45
        Get #Fnum, , myStr
        mInLine(i)=myStr
        i=i + 1

    Wend
    mInLineLen=i
    k=k + 1
    FinishPercent=Fix(k * 45 * 100 / TotalB)
    Text1.Text="encoding " & k & " line" & vbCrLf
    Text1.Text=Text1.Text & "finish " & FinishPercent & " %" & vbCrLf
    DoEvents
    If mInLineLen > 0 Then
        '如果读入的大于 0
        '对一行进行编码
        mOutB=mInLineLen + &H20
        Put #Fnum2, , mOutB
        'Put #Fnum2, , 46 'mOutB
        While (j * 3 < mInLineLen)
            mInByte(0)=mInLine(j * 3 + 0)
            mInByte(1)=mInLine(j * 3 + 1)
            mInByte(2)=mInLine(j * 3 + 2)
            j=j + 1

            Call EncodeByte(mInByte, mOutByte)
            For i=0 To 3
                Put #Fnum2, , mOutByte(i)
            Next i
        Wend
```

```
        End If
        Put #Fnum2, , vbCrLf
    Wend
        Put #Fnum2, , "'" & vbCrLf & "end"
    Close (Fnum)
    Close (Fnum2)
    End Sub
```

（3）Uudecode 解码的 VB 实现

编码和解码是一对可逆过程，知道了编码的算法，解码就不难了。但是为了方便读者进行测试，在本书的例题中我们加入了 Uudecode 的程序。

在发送邮件中利用编码发送附件，先看在 Microsoft Outlook 中怎么发送附件，编码选择 Uuencode，如果使用 Windows 98 自带的 Outlook Express，则从菜单“工具->选项”弹出的对话框中选择“发送”选项卡，选择“纯文本”设置其属性，弹出图 5-11 的“纯文本设置”对话框，从中选择“Uuencode”选项。发送附件是 Outlook 采用 Uuencode 编码发送。

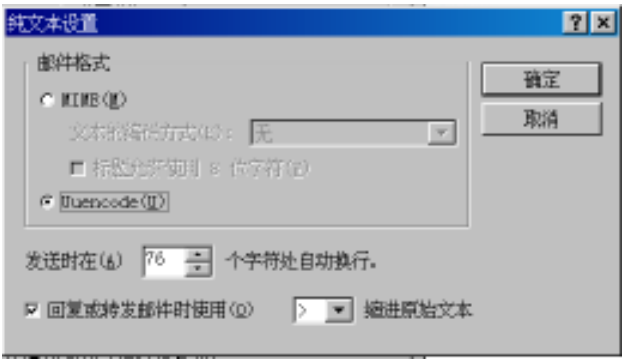


图 5-11　Outlook Uuencode 设置

下面是从 POP3 服务器接收回来的，并且用以上设置的 Outlook Express 发送带附件的信件。

```
    Received: from zhongjun [10.10.10.10] by ZHONG [10.10.10.10] with SMTP
(MDaemon.v2.1.rB.b1.32-T) for <1@zhong>; Fri, 07 Apr 100 20:31:01 -0000
    Message-ID: <000901bfa08d$22138240$0a0a0a0a@zhongjun.tsinghua.edu.cn>
    From: "zhong" <1@10.10.10.10>
    To: <busywxp@cocim.com>
    Subject: test attachment
    Date: Fri, 7 Apr 2000 20:31:00 +0800
    X-Priority: 3
    X-MSMail-Priority: Normal
    X-Mailer: Microsoft Outlook Express 4.72.3110.5
    X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3
    X-MDMail-Server: MDaemon v2.1 rB b1 32-T
    X-MDaemon-Deliver-To: busywxp@cocim.com
```

```
X-MBF-FILE: MDaemon Gateway to RFC822 (RFC822.MBF v1.0)

Wxp:
Hello,this is a mail with attachment !
               busyzhong

begin 600 1.txt
M35-$3B!,:6)R87)Y(,K'OZJWHLC+U+&UQ-;8TJJRSK^\U\K!SZ.LL/RZK,'+
MR-W!O\ZJ(#$@1T(@M<2QX+/,O+S*]=#%SZ*CK+#\P*C*OL#]M/K"ZZ&BSL2U
MM:&BO+S*]<[$U<*AHDUI8W)O<V]F=""_JK>BR,O4L=:JRK:_XJ&BTM2\L,3Z
MU-K*N=/#($UI8W)O<V]F=""YJ\N^M<2\O,KUP+2_JK>BO>*^]K>]L+C*L<OY
1T.C2JK7$QN3+^]?*P<^AHP"Y
'
end
```

其中，附件是一个文本文件，名称为“ 1.txt **”。**

进行编码后的编码结果为：

```
begin 600 1.txt
M35-$3B!,:6)R87)Y(,K'OZJWHLC+U+&UQ-;8TJJRSK^\U\K!SZ.LL/RZK,'+
MR-W!O\ZJ(#$@1T(@M<2QX+/,O+S*]=#%SZ*CK+#\P*C*OL#]M/K"ZZ&BSL2U
MM:&BO+S*]<[$U<*AHDUI8W)O<V]F=""_JK>BR,O4L=:JRK:_XJ&BTM2\L,3Z
MU-K*N=/#($UI8W)O<V]F=""YJ\N^M<2\O,KUP+2_JK>BO>*^]K>]L+C*L<OY
1T.C2JK7$QN3+^]?*P<^AHP"Y
'
end
```

可以从上面的例子看出来，对二进制文件进行编码后，将编码形成的文本与发送的信件体放在一起，放在正文之后即可。如果有多个附件怎么办？解决的方法是一样的，在第一个附件之后，附加上编码后第二个文件的编码文本。

参照上面的例子，读者使用本书的程序来进行编码，发送带附件的电子邮件相信不是一件难事。

5.8.3　MIME 及其编码

在理解 MIME **之前，首先来看一个从** POP3 **服务器接收下来的信件，该信件是用** Outlook Express **发送的。**

```
Received: from zhongjun [10.10.10.10] by ZHONG [10.10.10.10] with SMTP
(MDaemon.v2.1.rB.b1.32-T) for <1@zhong>; Wed, 05 Apr 100 16:17:46 -0000
Message-ID: <001001bf9ed7$6c80c980$0a0a0a0a@zhongjun.tsinghua.edu.cn>
From: "=?gb2312?B?1tO+/A==?=" <1@10.10.10.10>
To: <1@zhong>
Subject: =?gb2312?B?1tO+/MTjusM=?=
```

```
Date: Wed, 5 Apr 2000 16:17:45 +0800
MIME-Version: 1.0
Content-Type: multipart/mixed;
    boundary="----=_NextPart_000_000A_01BF9F1A.7A57BE40"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3110.5
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3
X-MDMail-Server: MDaemon v2.1 rB b1 32-T
X-MDaemon-Deliver-To: 1@zhong
X-MBF-FILE: MDaemon Gateway to RFC822 (RFC822.MBF v1.0)

This is a multi-part message in MIME format.
------=_NextPart_000_000A_01BF9F1A.7A57BE40
Content-Type: multipart/alternative;
    boundary="----=_NextPart_001_000B_01BF9F1A.7A57BE40"

------=_NextPart_001_000B_01BF9F1A.7A57BE40
Content-Type: text/plain;
    charset="gb2312"
Content-Transfer-Encoding: quoted-printable

=D6=D3=BE=FC=BA=CF=C0=ED=BF=A9

------=_NextPart_001_000B_01BF9F1A.7A57BE40
Content-Type: text/html;
    charset="gb2312"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>

<META content=3Dtext/html;charset=3Dgb2312 http-equiv=3DContent-Type>
<META content=3D'"MSHTML 4.72.3110.7"' name=3DGENERATOR>
</HEAD>
<BODY bgColor=3D#ffffff>
<DIV><FONT color=3D#000000 =
size=3D2>=D6=D3=BE=FC=BA=CF=C0=ED=BF=A9</FONT></DIV></BODY></HTML>
```

```
    ------=_NextPart_001_000B_01BF9F1A.7A57BE40--

    ------=_NextPart_000_000A_01BF9F1A.7A57BE40
    Content-Type: text/plain;
        name="Index.txt"
    Content-Transfer-Encoding: Base64
    Content-Disposition: attachment;
        filename="Index.txt"

    MTk5OSDN9bfGINa7sK7EsMn6yMs6DQoNCjAxICC/qrW9sujM/A0KMDIgILWxyrG1xNTCwcENC
jAz
    ICC038PfDQowNCAg1ruwrsSwyfrIyw0KMDUgILDZxOq5wrzFDQowNiAguvu1+w0KMDcgILn90
dvU
    xtHMDQowOCAgu6nSu8n51q668w0KMDkgIM3Gt60NCjEwICC+q7LKW7DZysK/ycDWueO45ibui
s37
    Jtb3zOLH+l0NCjExICDK2M37wvPM71uw2cTqucK8xSAgueO2q7DmXQ0KMTIgINPKsu5bILr7t
fsg
    ICC547arsOZd

    ------=_NextPart_000_000A_01BF9F1A.7A57BE40—
```

这是一封典型的采用 MIME **扩展信件规范的信件。前面已经介绍过，**MIME **是通过扩展** RFC822 **规范进行弥补其缺陷的。随着网络的飞速发展和电子邮件的日趋重要和适用广泛，可以说** MIME **已经成为得到广泛支持的电子邮件的标准。**

1. MIME **对信头字段的扩展**

MIME **对** RFC822 **提供的字段进行了扩充，补充了一些信头字段，这些字段通常是发送电子邮件软件在创建电子邮件时产生的,接收电子邮件软件提取其中的字段得到有用的信息。**

下面先介绍 MIME **补充的信头字段：**

（1**）**MIME-Version:

MIME-Version **字段用于标识使用的** MIME **版本号，这是为了将来增加版本号解决兼容性问题的。这个字段是** MIME **信件唯一必须要求出现的字段。**

目前只有一个 MIME **版本在使用，所以一般加入以下字段：**

```
    MIME-Version: 1.0
```

（2**）**Content-Type:

Content-Type **是** MIME **中的主要字段，描述特定** MIME **实体中包含的数据。**

这个字段有三部分：前 2 **个部分组成的媒体类型和** 1 **个可选分号分开的参数列表。如：**

```
    Content-Type: text/plain;charset="us-ascii"
```

其中 text **是媒体类型主类别，**plain **指示媒体类型的附加层次类别。这两个元素是不区分**

大小写的。charset="us-ascii"是可选的参数列表。又如：

```
Content-Type: text/html; charset=gb2312
```

指定媒体的主类型是 text，即文本，这与上面的是一样，但是文本的内容是 Html 格式的文件。使用的字符集为 gb2312。

参数用于控制媒体类型的解释，大多数媒体类型都有相应的参数。每个参数由一个属性名和值组成，使用“=”分开。

参数值最好放在引号之间，因为在参数值中不允许直接出现以下字符：

()<>@,;:\"/[]?=

如果有，应该使用引号。下面写法是比较好的编写习惯。

Content-Type: text/html; charset= "gb2312"

（3）Content-Transfer-Encoding:

许多数据格式可以包含在信件中允许的字符范围之外的字节值，而且含有超过允许长度的数据行。一些数据格式的定义甚至没有行的概念。Content-Transfer-Encoding 字段解决这些问题。

该字段有 5 个值：7bit、8bit、binary、Base64 和 quoted-printable。每个值都不区分大小写。符合 MIME 规范的邮件处理程序必须能对这些编码正确处理。

- 7bit

```
Content-Transfer-Encoding: 7bit
```

7bit 编码用于编码长度不超过 998 字节的数据，且该数据以 Crlf 结束的行组成，这通常等效于标准的 RFC822 文本行,行中的文字可以是小于 ASCII 128 的任何值,不包括 US-ASCII 0、CR 和 LF。这是缺省编码,如果不提供 Content-Transfer-Encoding 字段，就认为是 7bit 编码。

- 8bit

```
Content-Transfer-Encoding: 8bit
```

与 7bit 编码一样，8bit 编码用于编码长度不超过 998 字节数据，且该数据以 Crlf 结束的行组成，但与 7bit 不同的是，其字节长度允许大于 ASCII 127 的字节。另外不允许使用 CR 和 LF 字符，但允许 US-ASCII 0。

- binary

```
Content-Transfer-Encoding: binary
```

Binary 编码用于任意二进制 8 位数据，对行长度和允许的字符没有限制。在 MIME 信件实体里包含这种数据是不合法的，它仅仅是告诉指定数据的数据类型，而数据不是跟随邮件作为邮件内容发送的。

- Base64

Base64 编码是把二进制数据编码成适合通过 Iinternet 电子邮件传送的格式。在处理二进制数据作为邮件实体内容发送时，这种编码方式应用的最为广泛。可以说，Base64 编码是 MIME 的基础和核心。

下面将有一节内容专门地介绍 Base64 编码算法的原理,并给出 Base64 编码和解码的 VB 程序。

- quoted-printable

```
Content-Transfer-Encoding: quoted-printable
```

quoted-printable 编码用于编码主要由文本组成的数据，在行长度超过 80 个字符的情况下，这是非常有用的。RFC822 允许行的长度最大为 1000 个字符，但并不是所有的非 Internet 电子邮件程序都能正确处理它们。

后面有将一节专门介绍 Quoted-printable 算法及其程序实现，并给出了 VB 编写的编码和解码的函数。

- x-uuencode

```
Content-Transfer-Encoding: x-uuencode
```

尽管 MIME 几乎不再需要 Uuencode 进行编码的数据，但仍会见到 Uuencode。非标准的 x-uuencode 内容传输编码标记常常用在 MIME 实体中进行标识。

使用 x-uuencode 标记的实体的信件体含有传统的 Uuencode 编码的数据，且经常在 Content-Type 字段中包括一个 name 参数，与 Uuencode 编码数据中标识的文件名对应。

```
Content-Type: image/cursor;
name=" arrow1.cur"
Content-Transfer-Encoding: x-uuencode

begin 600 arrow1.cur
M‘‘‘"‘‘$‘("‘‘‘!‘‘"‘‘P‘0‘‘%@‘‘‘"@‘‘‘‘@‘‘‘‘0‘‘‘‘‘$‘‘0‘‘‘‘‘‘‘‘$‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘____‘‘‘‘"JH‘‘!55‘‘‘(‘@‘‘$‘$‘‘‘@"
M‘‘‘0@0‘‘",(‘‘!65‘‘‘)J@‘‘‘P‘‘‘$,‘‘‘!F‘‘‘‘?@‘‘‘’X‘‘‘!_P‘‘‘?X‘‘
M‘’\‘‘‘!^‘‘‘‘?!‘‘‘’@H‘‘!P;‘‘‘8((‘‘$!L‘‘‘‘*‘‘‘‘!‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘___@‘/__X‘#__^?\___G_/__YWS__^8<___F
M’/__X"#__V0‘__\X?___&’___P#___\‘____‘‘___P‘?__\‘/___‘’___P#_
M__\![___‘\?__P>#__\/‘?__’X/__S_’____[_______________________
,______________\‘
‘
end
```

(4) Content-ID

Content-ID 字段提供一种方法唯一标识 MIME 实体，与 Message-ID 唯一标识信件类似。它为媒体类型提供一种方法引用其他的 MIME 实体，且对于除 message/external-body 之外的所有媒体类型是可选的。

(5) Content-Description

Content-Description 字段允许一个文本描述与 MIME 实体相关联。

(6) Content-Disposition

Content-Disposition 字段用于控制一个实体是作为内插数据还是作为外部附件处理。

Inline 处置类型指示实体首选在信件体内插显示，可能的例子包括小的图像和 Html 材料。

Attachment 处置类型指示在常规信件的外部显示，常见的例子是大小不适合于常规邮件软件显示的大图像。

这个字段有几个参数，最常见的参数是 filename，它提供了为实体建议一个文件名的方法。

```
Content-Disposition: attachment;   filename="Index.txt"
```

不要求一定使用提供的文件名，它只是为了方便用户的，其他的参数如表 5-6 所示。

表 5-6　　处置参数

参数	描述
filename	提供的文件名
Size	实体（附件的大小）
Creation-date	文件被创建的日期
Modification-date	文件最后修改的日期
Read-date	文件读的日期

2. MIME 媒体类型

Content-Type 是用于在信头中指定媒体类型的字段。它是 MIME 对 RFC822 扩展的最主要信头字段，包含着很丰富的信息。下面就对这个字段的一些参数和值进行详细的说明。

用一般表达式表示 Content-Type 如下：

Content-Type: 媒体类型主类别 / 媒体类型的附加层次类别 + 由可选分号分开的参数列表。

在 RFC2045 中定义了 8 个顶层的媒体类型：text、image、audio、video、application、message 和 multipart。这些主要类型为它们各自的数据提供了一个结构。

（1）Text

Text 类型用于指示基于文本的内容。这里类别中任何子类型都不需要程序格式化数据而进行约定。

参数有 charset，用于指定 text 所使用的字符集，缺省时为 US-ASCII。

如：Content-Type: text/plain; charset=US-ASCII

与 Content-Type: text/plain 表示是一样的。

对于汉字，较常用的 charset 是 GB2312，通常由汉字内容的实体指定字符集：

Content-Type: text/plain; charset= GB2312

不过编写邮件程序时只要能将相应的文本数据内容提取出来，操作系统会自动解释并显示，而不需要额外的处理。如果操作系统不能正确处理，则需要程序处理了，但这种处理过程是十分复杂的。如使用中文 Windows 98 接收到德语写的信件，如果邮件程序不经过特别处理，是无法正确的显示信件内容的。

Text 的附加层次类别有：plain、enrich、html 等

- Plain

```
Content-Type: text/plain
```

text/plain 媒体类型标识由普通可打印字符组成，可以是 US-ASCII，也可以是汉字，这些数据没有经过格式化，直接浏览阅读就可以了。

- Enrich

```
Content-Type: text/enrich
```

text/enrich 为创建有对齐、颜色和字体变化功能的文本提供了一个简单的标记语言。如：

```
本书的<bold>主要内容</bold>为：介绍网络协议并使用<italic>VB</italic>实现
```

应该显示为：

```
本书的主要内容为：介绍网络协议并使用 VB 实现
```

这种格式与 Html 控制显示格式的方式非常相似，如果熟悉 Html 的读者相信不难理解。表 5-7 列出了可用的标签。

表 5-7　　Text/enrich 的标签

标签	描述
bold	文字加粗
italic	文字倾斜
Underline	文字加下划线
param	提供属性参数
smaller	缩小文字
bigger	增大文字
fixed	使用定宽文字
fontfamily	指定字体
color	指定颜色
center	文字居中
flushleft	文字左对齐
flushright	文字右对齐
flushboth	文字两端对齐
nofill	不带段落填充地显示文字
paraindent	控制页边界位置
excerpt	作为节录材料显示文字
lang	设置语言

不过随着 Html 的出现，enrich 类型的使用很快被 Html 所代替，无论是显示格式的种类或是标准上，Html 都有 enrich 无法比拟的优越性。

- Html

```
Content-Type: text/html
```

Html 相信大家都很熟悉了，我们在 Web 上冲浪时，所看到的绚丽多彩的网上世界就是 Html 在后面控制的。不过要是在自己实现的邮件程序中能够正确的显示 Html 内容的文字可不是一件容易的事。最好是使用第三方提供的浏览器控件显示。

（2）Image、Audio 和 Video

这几个媒体类型主类别是相当直观的，数据通常是二进制的，采用 Base64 编码。

```
Content-Type: image/jpeg
Content-Transfer-Encoding: Base64
```

（3）Application

这个媒体类型主类别含有与特定应用文件格式有关的任何媒体类型。主要有以下几种附加层次类别：

- octet-stream

```
Content-Type: Application/ octet-stream
```

Application/ octet-stream 用于指示一个实体中含有任意数据。当内容类型未知或者对数据没有定义媒体类型时，通常使用它描述。

任何未识别的媒体类型也应作为一个 Application/octet-stream 实体处理，所有的邮件软件必须能够理解这个字段。

- Postscript

```
Content-Type: Application/postscript
```

Application/postscript 媒体类型用于标识一个实体的内容是 postscript 代码。

因为 Application/postscript 实体含有可执行代码，它的内容必须毫无损失地到达目的地。由于这个原因，通常使用 quoted-printable 编码，使它不至于因为长度限制或在通过 Internet 网关时被破坏。

- Applefile，mac-binhex40

这两个附加层次类别都是为 Macintosh 操作系统专用的，这里就不再详述。

（4）Message

这个媒体类型主类别指定信体组成的实体。

- RFC822

```
Content-type: message/RFC822
```

message/RFC822 类型提供在一个信件中打包另外一个信件的简单方法。所有符合 MIME 的邮件处理程序都必须支持这种类型的分析。

如下面的例子：

```
Date: Tue, 04 Dec 2001 16:29:02 +0800
From:busyzhong@cocim.com
Subject: Say hello from busyzhong!
To: busywxp@holiway.com
MIME-Version:1.0
Content-type: message/RFC822
From: gongliang@holiway.com
To: busyzhong@cocim.com
Date: Tue, 03 Dec 2001 16:00:02 +0800
hello:
       Say hello to you
```

这种情况通常发生在转发邮件时，将原来的信件原封不动打包转发给另外一个收信人。对于这种情况，Content-Transfer-Encoding 的值只能设置为 7bit、8bit 和 binary。原因是转发时原信件已经经过编码，信件的内容可能经过了 quoted-printable 和 Base64 编码，是符合 RFC822 传输的文本，故经过再次编码是多余的。

- partial

```
Content-type: message/partial
```

message/partial 媒体类型指定了一种方法，该方法把大的实体分成多个部件，每个部件可以分开传输，在接收端组装，在发送和传递邮件的服务器之间有信件尺寸限制的情况下，这

种方法可以发送比较大的信件。

这个类型有 3 个参数：id、total 和 number。

其中 id 参数与 Message-Id 类似，用于标识一组 message/partial 实体集合的每个部分，每个部分的该参数值是一样的。

Total 参数标识有多少个 message/partial 实体（部分），这个参数不是必须在最后一个实体出现，在前面的实体中也可以出现。

Number 参数标识是第几个实体。

在收到信件后，可以根据这几个参数组装信件。例：下面是一个有两部件的两封信件。

第一封信件内容：

```
Date: Tue, 04 Dec 2001 16:19:02 +0800
From:busyzhong@cocim.com
Subject: Say hello from busyzhong!
To: busywxp@holiway.com
MIME-Version:1.0
Content-type: message/ partial;
id="21313131@cocim.com";
number=1; total=2
Content-Type: video/mpeg;
    name="first.mpeg"
Content-Transfer-Encoding: Base64
Content-Disposition: attachment
MIME-Version: 1.0
（第一部分经过编码的数据）
```

第二封信件内容：

```
Date: Tue, 04 Dec 2001 16:19:02 +0800
From:busyzhong@cocim.com
Subject: Say hello from busyzhong!
To: busywxp@holiway.com
MIME-Version:1.0
Content-type: message/ partial;
id="21313131@cocim.com";
number=2; total=2
（第二部分经过编码的数据）
```

在接收到信件后，进行组装合并。得到的信件如下：

```
Date: Tue, 04 Dec 2001 16:29:02 +0800
From:busyzhong@cocim.com
Subject: Say hello from busyzhong!
To: busywxp@holiway.com
MIME-Version:1.0
Content-Type: video/mpeg;
```

```
        name="first.mpeg"
Content-Transfer-Encoding: Base64
Content-Disposition: attachment
MIME-Version: 1.0
（第一部分经过编码的数据）
（第二部分经过编码的数据）
```

值得注意的是，描述部件的信息（编码方式等）是放在第一封信件中进行的，因此第一封信件的结构比第二封信件的结构要稍微复杂一些。

- External-body

Content-type: message/External-body

message/External-body 表示实体含有的数据并不在信件内容里，而仅仅是描述了一种从外部取出数据的方式。如：

```
Content-Type: message/External-body;
        Access-type=local-file;
        name="first.mpeg"
Content-Type: video/mpeg;
Content-Transfer-Encoding: Base64
Content-ID:<234243@cocim.com>
```

其中有两个 Content-Type，第一个说明接下来的实体媒体类型是以外部文件的形式存放的，并指明了存放的地址是本地文件，和相应的文件名。第二个 Content-Type 说明这个外部文件的媒体类型，编码方式等信息。

Content-Type: message/External-body 使用的参数如下：

- Access-type 参数是最基本的，也是必需的，它指明用于访问外部数据的方法，又如：

Access-type=ftp

- Expiration 参数指明外部数据何时到达，如：
- Expiration: Tue, 06 Dec 2001 16:29:02 +0800
- Size 参数指明数据文件的大小。

其中 expiration 和 size 这两个参数都是可选的。Access-type 参数包含最基本的信息如表 5-8 所示。

表 5-8　　Access-type 参数的值

参数值	描述
FTP	访问方法是 FTP
ANON-FTP	访问方法是匿名的 FTP
TFTP	访问方法是 TFTP
Local-File	引用的是本地机器上的一个文件
Mail-Server	通过邮件服务器取得
URL	指明文件的 URL 地址，通常是通过 http 取得

```
Access-type=FTP
```

Access-type=FTP 类型指明数据可以通过 ftp 的方式取得，至少要求的参数有两个：site 和 name。其中 site 指定 ftp 站点，name 指定数据文件名。如：

```
Content-Type: message/External-body;
```

```
        Access-type=FTP;
        Site=ftp.cocim.com;
        name="first.mpeg"
```

此外，还有以下几个参数：directory 用于指定文件所在的目录，mode 用于指定 FTP 传输模式是“ image ”（二进制），还是“ ASCII ”（文本）。又如：

```
Content-Type: message/External-body;
        Access-type=FTP;
        Site=ftp.cocim.com;
        name="first.mpeg";
        directory="/pub/video";
        size=23234362;
```

根据各参数提供的信息登录到相应的 ftp 站点就可以获得数据文件，不过因为安全性的问题，访问 ftp 的账号和密码不在该字段参数中。

```
Access-type=ANON-FTP
```

Access-type=ANON-FTP 类型与 Access-type=FTP 一样，唯一不同的地方是通过匿名 FTP 登录获得文件。（匿名 ftp 的账号为 anonymous，密码为登录者的电子邮件信箱名）

```
Access-type=TFTP
```

Access-type=TFTP 类型是指明数据文件是通过 TFTP（普通文件传输协议）的方式取得的，这种方式并不多见，与 FTP 类似。

```
Access-type=Local-File
```

Access-type=Local-File 用于引用本地机器上的一个文件，这种类型有两个常用的参数：name 和 site。

name 参数用于指定文件名，site 参数是指定在哪里访问文件是合法的。如：

```
Content-Type: message/External-body;
        Access-type=local-file;
        name="/pub/vedio/first.mpeg";
        site="10.11.111.119"
```

上面的例子中表明只有地址为 10.11.111.119 的机器可以访问该文件，另外 site 也可以写成掩码的形式：site=“ 10.11.111.* ”，这样地址在 10.11.111.1 至 10.11.111.254 范围之内的机器都可以访问。

```
Access-type=Mail-Server
```

Access-type=Mail-Server 类型指明可以通过邮件服务器访问到数据文件。必需的参数为 server，用于指定把邮件服务器请求发送到的电子邮件地址。如：

```
Content-Type: message/External-body;
        Access-type=local-file;
        Server="zhjun@cocim.com"
Access-type=URL
```

根据 URL 的意义，其实许多访问类型都可以使用 Access-type=URL 的。这种类型需要一个参数 url，用于指定文件地址及获取方式。如：

```
Content-Type: message/External-body;
```

```
        Access-type=url;
        url="http://www.xh-expo.com/index.htm"
```

- Multipart

```
Content-Type: multipart/mixed
```

Multipart 媒体类型主类表示多个实体打包到单个实体，如图 5-12 所示。

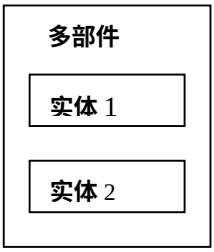


图 5-12　多部件实体

多部件实体由一个可选的前导、一个或多个信件体部件（每个前面有一个边界行）和一个可选的结尾部件组成。边界行由两个横线后跟一个边界字符串组成。这个边界字符串由 Content-Type 字段中参数 boundary 的值指定。如：

```
Content-Type: multipart/mixed; boundary=bigfox982340284
-- bigfox982340284（边界行）
实体 1 字段
实体 1 内容
--bigfox982340284（边界行）
实体 2 字段
实体 2 内容
--bigfox982340284--（结尾边界行）
```

其中信件体中所含的部件（实体）可以不提供实体字段，但需要提供一个空白行，使得其他处理电子邮件的软件可以辨别没有信头字段。如：

```
Content-Type: multipart/mixed; boundary=bigfox982340284
-- bigfox982340284（边界行）
（空白行）
实体 1 内容
--bigfox982340284（边界行）
（空白行）
实体 2 内容
--bigfox982340284--（结尾边界行）
```

multipart 的附加层次类别有许多：mixed、alternative、digest、parallel、report 等，其中 mixed 和 alternative 是最常用的，以下只介绍这两个。

mixed 和 alternative 的主要区别是：mixed 表示各个部件（实体）之间没有特定的关系，通常是各自意义分离的。Alternative 表示各部件之间的原始数据都是相同的，只是格式不一样，一般来说，首选的部件在最后。

下面是使用 Outlook 发送的多部件的信件。

```
    Received: from zhongjun [10.10.10.10] by ZHONG [10.10.10.10] with SMTP
(MDaemon.v2.1.rB.b1.32-T) for <1@zhong>; Wed, 05 Apr 100 16:17:46 -0000
    Message-ID: <001001bf9ed7$6c80c980$0a0a0a0a@zhongjun.tsinghua.edu.cn>
    From: "=?gb2312?B?1tO+/A==?=" <1@10.10.10.10>
    To: <1@zhong>
    Subject: =?gb2312?B?1tO+/MTjusM=?=
    Date: Wed, 5 Apr 2000 16:17:45 +0800
    MIME-Version: 1.0
    Content-Type: multipart/mixed;
        boundary="----=_NextPart_000_000A_01BF9F1A.7A57BE40"
    X-Priority: 3
    X-MSMail-Priority: Normal
    X-Mailer: Microsoft Outlook Express 4.72.3110.5
    X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3110.3
    X-MDMail-Server: MDaemon v2.1 rB b1 32-T
    X-MDaemon-Deliver-To: 1@zhong
    X-MBF-FILE: MDaemon Gateway to RFC822 (RFC822.MBF v1.0)

    This is a multi-part message in MIME format.

    ------=_NextPart_000_000A_01BF9F1A.7A57BE40
    Content-Type: multipart/alternative;
        boundary="----=_NextPart_001_000B_01BF9F1A.7A57BE40"

    ------=_NextPart_001_000B_01BF9F1A.7A57BE40
    Content-Type: text/plain;
        charset="gb2312"
    Content-Transfer-Encoding: quoted-printable

    =D6=D3=BE=FC=BA=CF=C0=ED=BF=A9

    ------=_NextPart_001_000B_01BF9F1A.7A57BE40
    Content-Type: text/html;
        charset="gb2312"
    Content-Transfer-Encoding: quoted-printable

    <!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
    <HTML>
    <HEAD>
```

```
    <META content=3Dtext/html;charset=3Dgb2312 http-equiv=3DContent-Type>
    <META content=3D'"MSHTML 4.72.3110.7"' name=3DGENERATOR>
    </HEAD>
    <BODY bgColor=3D#ffffff>
    <DIV><FONT color=3D#000000 =
    size=3D2>=D6=D3=BE=FC=BA=CF=C0=ED=BF=A9</FONT></DIV></BODY></HTML>

    ------=_NextPart_001_000B_01BF9F1A.7A57BE40--

    ------=_NextPart_000_000A_01BF9F1A.7A57BE40
    Content-Type: text/plain;
        name="Index.txt"
    Content-Transfer-Encoding: Base64
    Content-Disposition: attachment;
        filename="Index.txt"

    MTk5OSDN9bfGINa7sK7EsMn6yMs6DQoNCjAxICC/qrW9sujM/A0KMDIgILWxyrG1xNTCwcENC
jAz
    ICC038PfDQowNCAg1ruwrsSwyfrIyw0KMDUgILDZxOq5wrzFDQowNiAguvu1+w0KMDcgILn90
dvU
    xtHMDQowOCAgu6nSu8n51q668w0KMDkgIM3Gt60NCjEwICC+q7LKW7DZysK/ycDWueO45ibui
s37
    Jtb3zOLH+l0NCjExICDK2M37wvPM71uw2cTqucK8xSAgueO2q7DmXQ0KMTIgINPKsu5bILr7t
fsg
    ICC547arsOZd
    ------=_NextPart_000_000A_01BF9F1A.7A57BE40--
```

其中在信头字段中的 Content-Type **是这样描述的：**

```
    Content-Type: multipart/mixed;
        boundary="----=_NextPart_000_000A_01BF9F1A.7A57BE40"
```

表明这个信体是多部件的。边界行为：

```
    ----=_NextPart_000_000A_01BF9F1A.7A57BE40
```

这样我们得出信件体的初步结构一共有两个部件：

部件 1 **是第一个边界行“** ----=_NextPart_000_000A_01BF9F1A.7A57BE40 **”和第二个边界行的数据，部件** 2 **是第二个边界行和第三个之间的数据。**

从以上得出的部件 1 **的信头字段：**

```
    Content-Type: multipart/alternative;
        boundary="----=_NextPart_001_000B_01BF9F1A.7A57BE40"
```

由此可知，部件 1 **嵌套着一个多部件实体，媒体类型是** multipart/alternative**，表明部件** 1 **中的每个实体的原始数据都是一样，只是数据的格式不一样。根据其边界行得出它包含的两**

个实体。这里暂且称为部件 1-1，部件 1-2。

```
部件 1-1：
Content-Type: text/plain;
    charset="gb2312"
Content-Transfer-Encoding: quoted-printable

=D6=D3=BE=FC=BA=CF=C0=ED=BF=A9
部件 1-2：
Content-Type: text/html;
    charset="gb2312"
Content-Transfer-Encoding: quoted-printable
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>
<META content=3Dtext/html;charset=3Dgb2312 http-equiv=3DContent-Type>
<META content=3D'"MSHTML 4.72.3110.7"' name=3DGENERATOR>
</HEAD>
<BODY bgColor=3D#ffffff>
<DIV><FONT color=3D#000000 =
size=3D2>=D6=D3=BE=FC=BA=CF=C0=ED=BF=A9</FONT></DIV></BODY></HTML>
```

很明显，部件 1-1 与部件 1-2 的原始数据是一样的，都为经过 Quoted-printable 后的字符串：=D6=D3=BE=FC=BA=CF=C0=ED=BF=A9，格式却不一样，一个为 text/plain，为直接可读的文本，另一个为 text/html，必须经过浏览器解释后才显现效果。

从部件 2 的信头字段：

```
Content-Type: text/plain;
    name="Index.txt"
Content-Transfer-Encoding: Base64
Content-Disposition: attachment;
    filename="Index.txt"
```

可知部件 2 的数据是一个采用 Base64 编码的名称为 Index.txt 的附件文件。

3. MIME 用于信头字段

在前面给出的使用 Outlook 发出的信件的原始内容，信头字段中有这样的内容。

```
From: "=?gb2312?B?1tO+/A==?=" <1@10.10.10.10>
To: <1@zhong>
Subject: =?gb2312?B?1tO+/MTjusM=?=
```

字段 From 和 Subject 的内容是一些零乱的字符，也许有些读者已经意识到了，这是采用了编码方式。但是其具体结构和编码方式是什么？下面就介绍 MIME 用于信头字段的情况。

编码的字有一系列的初始字符、使用字符集的名字、使用的编码方案、实际编码的文本和一系列终止符组成：

```
"=?"charset"?"encoding"?"encoded-text"?="
```

以“=?”开始，接下来是编码使用的字符集、编码方案和编码后的文本，使用“?”分开每个值，以字符“?=”结束。像以下的写法：

```
=?gb2312?B?1tO+/MTjusM=?=
```

使用的字符集为汉字的 gb2312，编码方式采用 Base64 编码，编码文本为“1tO+/MTjusM=”，解码后的文本为“钟军你好”。

编码方式有两种，一种是 Q 编码，即 Quoted-printable 编码，另一种是 B 编码，也就是 Base64 编码，这两种编码方法我们都已经介绍过，这里不再重复。

信头字段的编码目的和信体的编码目的一样，都是为了 E-mail 数据经过 Internet 网关时不会被破坏。

不是所有的信头字段都可以编码，通常只在 Subject 字段、From 字段进行编码。处理电子邮件的程序必须能够识别这些编码方法，并进行正确的译码，还原数据。

4. Base64 编码与解码算法及其实现

（1）Base64 编码理论

Base64 的编码方式也是类似于 Uuencode。

第一步是从要编码的数据中每次依次取 3 个字节的数据作为一组。并把它们转换成 4 个 6 位的二进制数字，这 4 个数字也是分别用字节来表示，每个字节的前两位都为 0，这样转换后的 4 个字节的值范围限制在 0~63 之间。这一步是与 Uuencode 一样的。

第二步是按照表 5-9 中对应关系将数值转换成对应的 ASCII 字符。之所以选择这些字符是因为它们经过电子邮件网关时不会被破坏。

这两步对要编码的数据字节序列重复进行，直到所有的数据编码完成为止。

表 5-9　　编码字符表

值	字符	值	字符	值	字符	值	字符	值	字符	值	字符	值	字符	值	字符
0	A	8	I	16	Q	24	Y	32	g	40	o	48	w	56	4
1	B	9	J	17	R	25	Z	33	h	41	p	49	x	57	5
2	C	10	K	18	S	26	a	34	i	42	q	50	y	58	6
3	D	11	L	19	T	27	b	35	j	43	r	51	z	59	7
4	E	12	M	20	U	28	c	36	k	44	s	52	0	60	8
5	F	13	N	21	V	29	d	37	l	45	t	53	1	61	9
6	G	14	O	22	W	30	e	38	m	46	u	54	2	62	+
7	H	15	P	23	X	31	f	39	n	47	v	55	3	63	/

此外，编码过程中还有两点需要注意：

- 关于编码后的文本每行的长度

邮件信件内容的文本行应该控制在 76 个字符以内。所以编码时要注意，如果每行经过编码的文本控制在 76 个字符以内，那么超过的需要插入 Crlf（回车换行）。插入的 Crlf 不需要经过编码，而原来的要编码的数据中可能有 CR（0A）和 LF（0D），则与其他字符一样需要编码的。在译码时将碰到的 Crlf 删除即可。

- 填充字符

编码由于是采用三个作为一组进行的。如果被编码的数据长度不是 3 的整数倍，则编码

的最后一组不是完整的 3 字节组，就有可能剩下 1 个或 2 个字节。

当剩余 1 个字节时，追加两个 0 值来产生 3 字节组。产生的 4 个编码数据只有前两个带有剩余的那个字节的位。所以编码后后两个字节设置为“ = ”字符。

剩余 2 个字节时，追加 1 个 0 值产生 3 字节组。产生的 4 个编码最后一个不包含剩余的 2 个字节的位。所以将最后一个编码后的字节设置为“ = ”字符。

“MSDN Library 是开发人员的重要参考资料，包含了容量为 1 GB 的编程技术信息，包括示例代码、文档、技术文章、Microsoft 开发人员知识库、以及您在使用 Microsoft 公司的技术来开发解决方案时所需要的其他资料。”

上面的这段文字经过 Base64 编码后，成为如下的文本，该文本的每一行的最大长度，控制在 76 个字符之内。

```
TVNETiBMaWJyYXJ5IMrHv6q3osjL1LG1xNbY0qqyzr+818rBz6OssPy6rMHLyN3Bv86qIDEg
R0IgtcSx4LPMvLzK9dDFz6KjrLD8wKjKvsD9tPrC66GizsS1taGivLzK9c7E1cKhok1pY3Jv
c29mdCC/qreiyMvUsdaqyra/4qGi0tS8sMT61NrKudPDIE1pY3Jvc29mdCC5q8u+tcS8vMr1
wLS/qreiveK+9re9sLjKscv50OjSqrXExuTL+9fKwc+how==
```

（2）Base64 编码的 VB 实现

下面的程序代码是用于 Base64 编码的函数，其中 Base64EncodeByte 是用于将三个字符映射成四个字符并转换成相应的 ASCII 码的函数，Base64Encode 为外部接口函数，可以通过该函数将指定文件 Infile 的内容编码，输出到文件 Outfile 指定的文件中。具体代码如下：

```
Public Function Base64Encode(Infile As String, Outfile As String)
Dim FnumIn As Integer, FnumOut As Integer
Dim mInByte(3) As Byte, mOutByte(4) As Byte
Dim myByte As Byte
Dim i As Integer, LineLen As Integer, j As Integer
FnumIn=FreeFile()
Open Infile For Binary As #FnumIn
FnumOut=FreeFile()
Open Outfile For Binary As #FnumOut
While Not EOF(FnumIn)
    i=0
    Do While i < 3
    Get #FnumIn, , myByte
    If Not EOF(FnumIn) Then
        mInByte(i)=myByte
        i=i + 1
    Else
        Exit Do
    End If
    Loop
```

```
        Base64EncodeByte mInByte, mOutByte, i
        For j=0 To 3
            Put #FnumOut, , mOutByte(j)
        Next j
        LineLen=LineLen + 1
        If LineLen * 4 > 70 Then
            Put #FnumOut, , vbCrLf
            LineLen=0
        End If
    Wend
    Close (FnumOut)
    Close (FnumIn)
    End Function

    Private Sub Base64EncodeByte(mInByte() As Byte, mOutByte() As Byte, Num As
Integer)
    Dim tByte As Byte
    Dim i As Integer

    If Num=1 Then
        mInByte(1)=0
        mInByte(2)=0
    ElseIf Num=2 Then
        mInByte(2)=0
    End If

    tByte=mInByte(0) And &HFC
    mOutByte(0)=tByte / 4
    tByte=((mInByte(0) And &H3) * 16) + (mInByte(1) And &HF0)/16
    mOutByte(1)=tByte
    tByte=((mInByte(1) And &HF) * 4) + ((mInByte(2) And &HC0)/64)
    mOutByte(2)=tByte
    tByte=(mInByte(2) And &H3F)
    mOutByte(3)=tByte
```

```
    For i=0 To 3
        If mOutByte(i) >= 0 And mOutByte(i) <= 25 Then
            mOutByte(i)=mOutByte(i) + Asc("A")
        ElseIf mOutByte(i) >= 26 And mOutByte(i) <= 51 Then
            mOutByte(i)=mOutByte(i) - 26 + Asc("a")
```

```
    ElseIf mOutByte(i) >= 52 And mOutByte(i) <= 61 Then
        mOutByte(i)=mOutByte(i) - 52 + Asc("0")
    ElseIf mOutByte(i)=62 Then
        mOutByte(i)=Asc("+")
    Else
        mOutByte(i)=Asc("/")

    End If
Next i

If Num=1 Then
    mOutByte(2)=Asc("=")
    mOutByte(3)=Asc("=")
ElseIf Num=2 Then
    mOutByte(3)=Asc("=")
End If
End Sub
```

（3）Base64 解码的 VB 实现

下面的程序代码是用于 Base64 解码的函数，其中 Base64DecodeByte 是用于将编码的 4 个字符映射成 3 个字符的函数，Base64Decode 为外部接口函数，可以通过该函数将指定文件 Infile 的内容解码，解码后的内容输出到文件 Outfile 指定的文件中。具体代码如下：

```
Public Function Base64Decode(Infile As String, Outfile As String)
Dim FnumIn As Integer, FnumOut As Integer
Dim mInByte(4) As Byte, mOutByte(3) As Byte
Dim myByte As Byte
Dim i As Integer, LineLen As Integer, j As Integer
Dim ByteNum As Integer
FnumIn=FreeFile()
Open Infile For Binary As #FnumIn
FnumOut=FreeFile()
Open Outfile For Binary As #FnumOut

While Not EOF(FnumIn)
    i=0
    Do While i < 4
    Get #FnumIn, , myByte
    If Not EOF(FnumIn) Then
        If myByte <> &HA And myByte <> &HD Then
        '把回车符和换行符去掉
```

```
            mInByte(i)=myByte
            i=i + 1
         End If
      Else
         Exit Do
      End If
      Loop
      Base64DecodeByte mInByte, mOutByte, ByteNum

      For j=0 To 2 - ByteNum
         Put #FnumOut, , mOutByte(j)
      Next j
      'LineLen=LineLen + 1
   Wend
   Close (FnumOut)
   Close (FnumIn)
   End Function

   Private Sub Base64DecodeByte(mInByte() As Byte, mOutByte() As Byte, ByteNum
As Integer)
   Dim tByte As Byte
   Dim i As Integer
   ByteNum=0
   For i=0 To 3
      If mInByte(i) >= Asc("A") And mInByte(i) <= Asc("Z") Then
         mInByte(i)=mInByte(i) - Asc("A")
      ElseIf mInByte(i) >= Asc("a") And mInByte(i) <= Asc("z") Then
         mInByte(i)=mInByte(i) - Asc("a") + 26
      ElseIf mInByte(i) >= Asc("0") And mInByte(i) <= Asc("9") Then
         mInByte(i)=mInByte(i) - Asc("0") + 52
      ElseIf mInByte(i)=Asc("+") Then
         mInByte(i)=62
      ElseIf mInByte(i)=Asc("/") Then
         mInByte(i)=63
      Else '"="
         ByteNum=ByteNum + 1
         mInByte(i)=0
      End If
   Next i
   '取前六位
```

```
tByte=(mInByte(0) And &H3F) * 4 + (mInByte(1) And &H30) / 16
'0 的六位和 1 的前两位
mOutByte(0)=tByte
tByte=(mInByte(1) And &HF) * 16 + (mInByte(2) And &H3C) / 4
'1 的后四位和 2 的前四位
mOutByte(1)=tByte
tByte=(mInByte(2) And &H3) * 64 + (mInByte(3) And &H3F)
mOutByte(2)=tByte
'2 的后两位和 3 的六位
End Sub
```

5. Quoted-Printable 编码算法及其实现

如果被编码数据的值在 33（字符!）到 60（字符<）之间，或 62（字符>）到 126（字符~）之间，则该数据可用其对应的 ASCII 字符来表示。这部分字符是可打印的。编码后的数据即为 7bit 的 ASCII 码。

数据 9 和 32 对应的 ASCII 码字符为制表符 tab 和空格 space。tab 和 space 必须按规则 1 编码。对于一 Quoted-Printable 编码理论

先看一个编码后的字符串，如下：

```
=D6=D3=BE=FC=BA=CF=C0=ED=BF=A9
```

Quoted-Printable 编码算法由 5 条规则组成：

- 被编码的数据以 8bit 的字节为单位，每个字节都可以用等号“ = ”与相应的十六进制形式来表示。如值 12 被编码为“ =0C ”，值 61 被编码为“ =3D ”。十六进制数据的表示用字母表“ 0 1 2 3 4 5 6 7 8 9 A B C D E F ”，即必须用大写字母，除了符合以下 4 条规则的数据外，其余数据都必须按这种方法编码。
- 如果被编码数据的值在 33（字符!）到 60（字符<）之间，或 62（字符>）到 126（字符~）之间，则该数据可用其对应的 ASCII 字符来表示。这部分字符是可打印的。编码后的数据即为 7bit 的 ASCII 码。
- 数据 9 和 32 对应的 ASCII 码字符为制表符 tab 和空格 space。tab 和 space 必须按规则 1 编码。对于一个编码行末尾的 space 和 tab，不同的邮件传输系统的处理方法是不同的：有的系统在编码行末尾增加一些空格，有的系统在编码行末尾的空格都删除掉。
- 编码行的结束用回车符 CR 和换行符 LF 表示。Quoted-Printable 编码允许回车符和换行符单独出现，也可以 LF CR 或 CRLF 的顺序相邻出现。回车符被编码为“ =0D ”，换行符被编码为“ =0A ”。
- 规则 4 的编码行是指所含字符少于 1000 个的、适用于 SMTP 传输协议的编码行，而 Quoted-Printable 编码要求次编码行别分割成若干段，每段不超过 76 个字符。这种行分割符要用等号“ = ”表示，此等号也被包含在 76 个字符中。

例如：

```
Now's the time for all folk to come to the aid of their country.
```

经过编码后将一行较长的文本变成较短的形式：

```
Now’s the time =
for all folk to come=
to the aid of their country.
```

6. Quoted-Printable 编码的 VB 实现

下面的函数 QpEncode 是用于将指定文件 Infile 的内容编码，并输出到指定的文件 OutFile 中。具体代码如下：

```
Private Sub EncodeByte(mInByte As Byte, mOutStr As String)
If (mInByte >= 33 And mInByte <= 60) Or (mInByte >= 62 And mInByte <= 126)
Then
    mOutStr=Chr(mInByte)
Else
    If mInByte <= &HF Then
        mOutStr="=0" & Hex(mInByte)
    Else
        mOutStr="=" & Hex(mInByte)
    End If
End If
End Sub

Public Sub QpEncode(infile As String, Outfile As String)
Dim Fnum, Fnum2 As Integer
Dim myB As Byte
Dim mOutStr As String
Dim FinishPercent As Long
Fnum=FreeFile()
Fnum2=Fnum + 1
Dim TotalB, k As Long
Open Outfile For Output As #Fnum2
Open infile For Binary As #Fnum
TotalB=LOF(Fnum)
    k=0
Do While Not EOF(Fnum)
    Get #Fnum, , myB
    If EOF(Fnum) Then Exit Do
        EncodeByte myB, mOutStr
        Print #Fnum2, mOutStr;
Loop
Close (Fnum)
```

```
Close (Fnum2)
End Sub
```

Quoted-Printable 解码的 VB 实现

下面的函数 QpDecode 是用于将指定文件 Infile 的内容解码，并输出到指定的文件 OutFile 中。具体代码如下：

```
Private Sub DecodeByte(mInByte1 As Byte, mInByte2 As Byte, mOutByte As Byte)
Dim tbyte1 As Integer, tbyte2 As Integer
If mInByte1 > Asc("9") Then
   tbyte1=mInByte1 - Asc("A") + 10
Else
   tbyte1=mInByte1 - Asc("0")
End If
If mInByte2 > Asc("9") Then
   tbyte2=mInByte2 - Asc("A") + 10
Else
   tbyte2=mInByte2 - Asc("0")
End If
mOutByte=tbyte1 * 16 + tbyte2
End Sub

Public Sub QpDecode(infile As String, Outfile As String)
Dim Fnum, Fnum2 As Integer
Dim myB As Byte
Dim myByte1 As Byte, myByte2 As Byte
Dim mOutByte As Byte
Dim FinishPercent As Long
Fnum=FreeFile()
Fnum2=Fnum + 1
Dim TotalB, k As Long
Open Outfile For Binary As #Fnum2
Open infile For Binary As #Fnum
TotalB=LOF(Fnum)
   k=0
Do While Not EOF(Fnum)
   Get #Fnum, , myB
   If EOF(Fnum) Then Exit Do
   If myB=Asc("=") Then
      Get #Fnum, , myByte1
      If myByte1=&HA Then
```

```
            '如果是回车,继续
        Else
            '取第二个字节
            Get #Fnum, , myByte2
            Call DecodeByte(myByte1, myByte2, mOutByte)
            Put #Fnum2, , mOutByte
        End If
    Else
        mOutByte=myB
        Put #Fnum2, , mOutByte
    End If
Loop
Close (Fnum)
Close (Fnum2)
End Sub
```

5.8.4 构造 MIME 信件

前面说过，MIME 对 RFC822 信头扩展了许多字段，可以充分利用这些字段构造自己的信件，让接收并分析 E-mail 的程序获得尽可能多的信息。但是如果 RFC822 能够满足信件的需求，尽量使用 RFC822 规范构造信件，因为对于每个 E-mail 客户端程序，都要求起码能够解释分析 RFC822。只有在发送附件和其他 RFC822 局限而满足不了要求时，才采用 MIME 信件。假如要发送的正文是一段文本和一个附件文件（附件文件可以是图片或其他文件，这里为讲解方便，用一个文本文件作为附件发送）。

要发送的信件正文为：

```
busywxp:
   这是一个来自 busyzhong 的问候。
This is a multi-part message in MIME format.
```

要发送的附件是一个名为“index.txt”的文本文件，内容为：

```
“MSDN Library 是开发人员的重要参考资料，包含了容量为 1 GB 的编程技术信息，包括示例代码、文档、技术文章、Microsoft 开发人员知识库、以及您在使用 Microsoft 公司的技术来开发解决方案时所需要的其他资料。”
```

考虑到汉字的第 8 位字节在通过某些 Internet 网关时会被破坏以及需要发送附件，采用 MIME 来构造并发送邮件是必要的。

首先构造信件头：

```
From: <zhjun@10.11.111.111>
To: <busywxp@cocim.com>
Subject: Say Hello From zhjun
Date: Wed, 5 Apr 2000 16:17:45 +0800
MIME-Version: 1.0
```

```
Content-Type: multipart/mixed;
    boundary="----=Bigfox983283432424--001"
X-Mailer: BigFox Mailer V1.0
```

如果字段 Subject 的内容包含汉字，可采用编码方式来构造，如 Subject 的内容为：

```
来自 busyzhong 的问候
```

按照 MIME 用于信头字段的规范，构造的 Subject 的字段值如下：

```
=?gb2312?Q? =C0=B4=D7=D4busyzhong=B5=C4=CE=CA=BA=F2?=
```

指定了采用的字符集为 GB2312，字段内容采用 Quoted-printable 编码。

信头字段中除了包含基本的 To、Subject 等字段，还包含了 MIME-Version 等字段，特别是字段的 Content-Type 的内容表明了信体是多部件结构以及边界行的符号。

接下来构造信件体：

在通常的 MIME 信件中，信头和信体之间用空白行隔开，并且有这样的一段文字：

```
This is a multi-part message in MIME format.
```

在这里把它加到信头和信体之间。

在每个实体之间使用边界行隔开，并且在每个边界行内的实体加上相应的信头字段描述实体的媒体类型。

如部件 1 的构造：

```
------=Bigfox983283432424--001
Content-Type: text/plain;
    charset="gb2312"
Content-Transfer-Encoding: quoted-printable
busywxp:=0D=0A=20=20=20=D5=E2=CA=C7=D2=BB=B8=F6=C0=B4=D7=D4=
busyzhong=B5=C4=CE=CA=BA=F2=A1=A3
```

指定媒体类型为 text/plain，采用字符集为 GB2312，编码方式为 Quoted-printable。

部件 2 的构造：

```
------=Bigfox983283432424--001
Content-Type: text/plain;
    name="Index.txt"
Content-Transfer-Encoding: Base64
Content-Disposition: attachment;
    filename="Index.txt"
TVNETiBMaWJyYXJ5IMrHv6q3osjL1LG1xNbY0qqyzr+818rBz6OssPy6rMHLyN3Bv86qIDEg
R0IgtcSx4LPMvLzK9dDFz6KjrLD8wKjKvsD9tPrC66GizsS1taGivLzK9c7E1cKhok1pY3Jv
c29mdCC/qreiyMvUsdaqyra/4qGi0tS8sMT61NrKudPDIE1pY3Jvc29mdCC5q8u+tcS8vMr1
wLS/qreiveK+9re9sLjKscv50OjSqrXExuTL+9fKwc+how==
------=Bigfox983283432424--001–
```

指定媒体类型为 text/plain，名称为 index.txt，编码方式为 Base64。Content-Disposition 字段指明该部件的内容是以附件的形式解释的。虽然实际上解码后该部件的内容可以作为文本直接阅读，但是大多数附件的类型是不可阅读的，要将其解码后使用其他相应的软件打开。

综上所述，这个构造出来的信件如下：

```
From: <zhjun@10.11.111.111>
To: <busywxp@cocim.com>
Subject: Say Hello From zhjun
Date: Wed, 5 Apr 2000 16:17:45 +0800
MIME-Version: 1.0
Content-Type: multipart/mixed;
    boundary="----=Bigfox983283432424--001"
X-Mailer: BigFox Mailer V1.0
This is a multi-part message in MIME format.
------=Bigfox983283432424--001
Content-Type: text/plain;
    charset="gb2312"
Content-Transfer-Encoding: quoted-printable
busywxp:=0D=0A=20=20=20=D5=E2=CA=C7=D2=BB=B8=F6=C0=B4=D7=D4=
busyzhong=B5=C4=CE=CA=BA=F2=A1=A3
------=Bigfox983283432424--001
Content-Type: text/plain;
    name="Index.txt"
Content-Transfer-Encoding: Base64
Content-Disposition: attachment;
    filename="Index.txt"
TVNETiBMaWJyYXJ5IMrHv6q3osjL1LG1xNbY0qqyzr+818rBz6OssPy6rMHLyN3Bv86qIDEg
R0IgtcSx4LPMvLzK9dDFz6KjrLD8wKjKvsD9tPrC66GizsS1taGivLzK9c7E1cKhok1pY3Jv
c29mdCC/qreiyMvUsdaqyra/4qGi0tS8sMT61NrKudPDIE1pY3Jvc29mdCC5q8u+tcS8vMr1
wLS/qreiveK+9re9sLjKscv50OjSqrXExuTL+9fKwc+how==
------=Bigfox983283432424--001—
```

5.8.5 MIME 信件的语法分析

编写一个既能编码发送附件又能接收并分析电子邮件的客户端程序是比较困难的。通过上面讲解的内容，相信读者都可以实现发送和接收 E-mail 了，但是，编写电子邮件软件的难点在于对从 POP3 服务器接收下来的原始信件进行语法和结构分析，从中提取出包含的信息。

笔者曾经利用国内比较流行的 E-mail 软件 Foxmail 发送一封信件，使用 Windows 下的 Outlook 进行接收，发送有时会出现乱码的情况。出现这种情况的原因当然是标准不太统一的结果。如果利用相同的软件发送和接收，是不会出现这样的问题的。

以上所介绍的 MIME 只是涉及了一些比较常用的内容。要编写一个能够考虑周全、兼容于所有软件发送的符合标准的信件语法分析程序是不太可能的。在这里由于篇幅的限制，就不再给出 MIME 信件的语法分析程序了。

5.9　E-mail 客户端高级编程

5.9.1　建立工程项目

在本工程项目中，将讲解使用 VB 实现发送多个附件的 E-mail 程序的高级开发实例。由于在前面我们已经介绍了一个发送无附件的 E-mail 实例，所以本实例所用到的许多内容都是在前面实例中讲解过的，在此只是分析关键代码，通过比较两个实例的不同之处，深入了解发送 E-mail 附件的机制。

建立的工程项目一共由以下几个模块组成：

- 模块 ModBase64
- 窗体 FrmSetup
- 窗体 frmAtt
- 窗体 Smtp

1. 窗体 Smtp

该窗体为发送 E-mail 的界面。如图 5-13 所示。

图 5-13　Smtp 界面

该窗体的许多代码与前面发送无附件 E-mail 的实例有许多相同之处，因此就不再重复，这里只分析一些实现重要功能的关键代码。

（1）用于生成信件的内容的函数

函数 GenMail 用于根据写好的信件内容及附件生成符合 MIME 的格式内容，并将其写到临时文件 mail.tmp 中。

```
'该函数用于构造信件内容
Private Sub GenMail(bAttachment As Boolean)
```

```
    Dim fnum As Integer, FAttin As Integer
    Dim strLine As String
    strSendName=txtSName.Text
    strReceiveName=txtRName.Text
    strFromMail=txtFrom.Text
    strToMail=txtTo.Text
    m_Date=Format(Date,"Ddd")&","&Format(Date,"dd Mmm YYYY")&""&Format(Time,
"hh:mm:ss") & "" & " -0600"
    strSubject=txtSubject.Text
    strContent=txtContent.Text
    fnum=FreeFile()
    Open App.Path & "\mail.tmp" For Output As fnum
    '构造信件标题字段
    Print #fnum, "From:" & Chr(32) & strSendName
    Print #fnum, "Date:" & Chr(32) & m_Date
    Print #fnum, "X-Mailer: BigAnt Smtp Mailer V1.0"
    Print #fnum, "To:" & Chr(32) & strReceiveName
    Print #fnum, "Subject:" & Chr(32) & strSubject
    If bAttachment=False Then
       Print #fnum, ""
       Print #fnum, strContent
       Exit Sub
    End If
    Print #fnum, "MIME-Version: 1.0"
    Print #fnum, "Content-type:multipart/mixed;"
    Print #fnum, " boundary =""----=_NextPart_000_000A_01BF9F1A"""
    Print #fnum, ""
    '书写信件的正文内容
    Print #fnum, "--" & "----=_NextPart_000_000A_01BF9F1A"
    Print #fnum, "Content-Type: text/plain;"
    Print #fnum, "    Charset=""gb2312"""
    Print #fnum, "Content-Transfer-Encoding: 8bit"
    Print #fnum, ""
    Print #fnum, strContent
    '附件内容
    Dim i As Integer
    For i=0 To cobAtt.ListCount - 1
       Base64Encode cobAtt.List(i), App.Path & "\attachment" & i & ".tmp"
       Print #fnum, "--" & "----=_NextPart_000_000A_01BF9F1A"
```

```
        Print #fnum, "Content-Type: Application/octet-stream"
        Print #fnum, "  name=" & cobAtt.List(i)
        Print #fnum, "Content-Transfer-Encoding: Base64"
        Print #fnum, "Content-Disposition: attachment;"
        Print #fnum, "  FileName=" & cobAtt.List(i)
        Print #fnum, ""
        FAttin=FreeFile
        Open App.Path & "\attachment" & i & ".tmp" For Input As #FAttin
        While Not EOF(FAttin)
            Line Input #FAttin, strLine
            Print #fnum, strLine
        Wend
        Close FAttin
    Next i
    Print #fnum, "--" & "----=_NextPart_000_000A_01BF9F1A" & "--"
    Close fnum
    End Sub
```

（2）发送信件的过程

过程 cmdSend_Click **用于响应发送按钮单击事件，发送的过程是遵循** SMTP **会话进行的。将临时文件** mail.tmp **的内容按照正文发送。**

```
    Private Sub cmdSend_Click()
    If cobAtt.ListCount > 0 Then
        GenMail True
    Else
        GenMail False
    End If
    '设置 Winsock
    Wsock.Close
    Wsock.RemoteHost=ServerIp
    Wsock.RemotePort=ServerPort
    '连接 SMTP 服务器
    Wsock.Connect
    If Not WaitForResponse("220", 10) Then
        txtMsg.Text="邮件服务器连接不上......"
        Exit Sub
    End If
    '打开对话
    Wsock.SendData "HELO" & " " & Wsock.LocalHostName & vbCrLf
    If Not WaitForResponse("250", 10) Then
```

```
    txtMsg.Text=txtMsg.Text & "无法打开邮件发送对话" & vbCrLf
    Exit Sub
End If
'发送发送方地址
Wsock.SendData "MAIL FROM:" & " " & strFromMail & vbCrLf
If Not WaitForResponse("250", 10) Then
    txtMsg.Text=txtMsg.Text & "无法发送发送方地址" & vbCrLf
    Exit Sub
End If
'发送接收方地址
Wsock.SendData "RCPT TO:" & " " & strToMail & vbCrLf
If Not WaitForResponse("250", 10) Then
    txtMsg.Text=txtMsg.Text & "无法发送接收方地址" & vbCrLf
    Exit Sub
End If
'发送消息体
Wsock.SendData "DATA" & vbCrLf
If Not WaitForResponse("354", 10) Then
    txtMsg.Text=txtMsg.Text & "无法发送消息体" & vbCrLf
    Exit Sub
End If
Dim fnum As Integer
fnum=FreeFile
Open App.Path & "\mail.tmp" For Input As #fnum
'Wsock.SendData mData & vbCrLf
While Not EOF(fnum)
    Line Input #fnum, strContent
    Wsock.SendData strContent & vbCrLf
Wend
Close #fnum
Wsock.SendData "." & vbCrLf
If Not WaitForResponse("250", 20) Then
    txtMsg.Text=txtMsg.Text & "消息体发送不成功" & vbCrLf
    Exit Sub
End If
'结束邮件发送对话
Wsock.SendData "QUIT" & vbCrLf
If Not WaitForResponse("221", 10) Then
    Exit Sub
```

```
End If
Wsock.Close
txtMsg.Text=txtMsg.Text & "邮件发送成功"
End Sub
```

2. **窗体** frmSetup

该窗体用于设置 SMTP **服务器，显示的窗体如图** 5-14 **所示。**

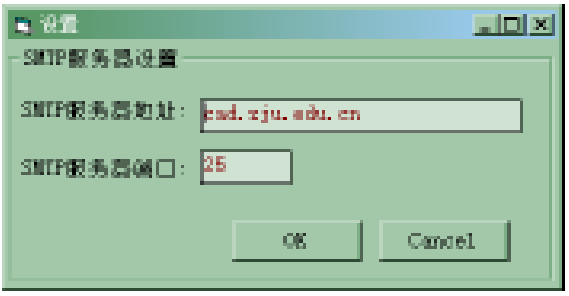


图 5-14　**设置** SMTP **服务器**

该窗体代码请参看源程序。

3. **窗体** frmAtt

该窗体用于选择加入的附件文件，显示的窗体如图 5-15 **所示。**

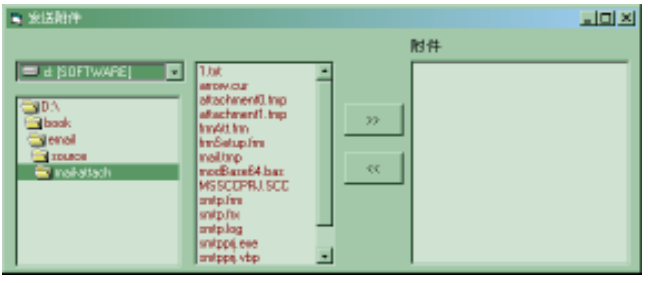


图 5-15　**加入附件文件**

该窗体代码请参看源程序。

4. **模块** modBase64

该窗体定义了用于 Base64 **编码的函数。程序代码请参看源程序。**

5.10　E-mail **乱码**

5.10.1　**乱码的常见形式及形成原因**

计算机以及很多计算机网络协议，特别是 E-mail **协议的制定都是建立在** ASCII **码的基础上的，但是用** ASCII **码表示计算机信息有很大的不足，主要表现在多国文字、图形、声音等二进制文件的信息压缩、信息保密等很多方面。**

而编码和解码可以使得在 ASCII 的基础上，用一定的规则定义一些新的信息表达形式，使原来非纯 ASCII 的信息也能用 ASCII 表示。但是编码和解码必须成对出现才能对信息进行正确的转换和还原。乱码的产生是信息编码和解码不能够统一的结果。比如一个使用 Quoted-printable 编码的文本：

```
=C0=B4=D7=D4busyzhong=B5=C4=CE=CA=BA=F2
```

使用 Quoted-printable 解码得到正确的文字：

```
来自 busyzhong 的问候
```

如果使用不正确的解码方式 Base64 得到以下内容：

```
¨??n??'?  °@
```

经过编码的信息没有得到解码或解码方法错误，就会产生乱码。

解决乱码的过程就是找到和编码相统一的解码方法。在 E-mail 中获取信件信头字段的字段信息是很重要的。因为编码方式通常在信头字段中会有提示。

相信读者在阅读过本章的有关编码和解码的内容后，对 E-mail 出现的乱码不至于手足无措。不过与 E-mail 有关的内容非常丰富，加上 Internet 包罗万象，所以还是会产生或这或那的问题。

值得一提的是，有些乱码的出现不是由于不能正确理解所用的 E-mail 编码方式并对其进行解码。最常见的汉字之间的乱码出现在 Big5 和 GB 编码之间。

GB 和 Big5 的转换也是如此。假如一段 Big5 的文字使用在简体中文 Windows 98 下，看到的就是一段乱码，相反在繁体中文 Windows 98 下看 GB 的文字，看到的也不是我们想看的信息。

需要说明的是，简体和繁体这两个概念和 GB、Big5 并没有逻辑上的联系，GB 的定义是简体字，Big5 采用的是繁体字，但是为了阅读方便，在各自的编码中再做一个内部字形或字体的映射，就形成了所谓 GB 繁体或 Big5 简体之类的概念，它们仅仅是一些汉字软件提供的方便功能，如南极星等。上面所说的繁体中文一般是指采用 Big5 编码，简体中文采用 GB 编码。

相同情况的又如：使用日文操作系统的字符，在中文操作系统下直接阅读看到的也是乱码。这些情况的乱码的解决已经超出本书的范围，在此就不再详述了。

5.10.2 避免乱码的方法

现在已经知道乱码产生的原因和一些解决办法，避免乱码不仅仅是接收阅读 E-mail 程序的责任。避免乱码最好的方法在于发送及接收 E-mail 的程序都是符合相应规范的。

以下是使用时的一些要求。

（1）选用大众化的电子邮件收发程序

由于不同的电子邮件收发程序支持的编码有所不同，收件人和发件人自己定制的一些选项也会各不相同，所以在收到编码的信件后，系统不一定能识别出邮件所用的编码方法。识别不出编码方法，系统自然无法自动解码，这样当查看信件内容时，就会出现所谓的乱码，使收信人无法阅读该文件。选用大众化的电子邮件收发程序则可以在一定程度上避免不同的编码方法。

（2）使用“附件”功能发送文件

一般电子邮件收发程序的“附件”功能可以自动对邮件先进行编码，然后再发送。如果收信人的电子邮件收发程序（如 Netscape mail、Outlook Express、Eudora、Pegasus 等）能够区别邮件的编码方式，则可以自动将邮件解码。

（3）编写电子邮件程序

编写的发送电子邮件程序要符合相应的规范，尽量提供足够的字段信体说明信件的结构、编码等信息。

接收电子邮件的软件尽量多测试，多参考相应的 RFC 以增加对标准字段及内容的支持。

（4）不使用电子邮件收发程序中特别的编辑功能

如 Outlook Express 邮件编辑器是个功能很强的 Html 编辑器，你可以编辑五颜六色、各种字体的电子邮件。不过，如果接收方不使用 Outlook Express 来接收邮件，可能只看到很难看清楚的 Html 源码。注意将邮件发送格式设置为“纯文本”，这样其他软件接收 Outlook 发送的 E-mail 就不会出现 Html 格式了。

5.11　MAPI 概述

5.11.1　Windows 的 MAPI 介绍

不知道读者在了解到 E-mail 的核心内容之后，是否对 E-mail 有一种既简单又难懂的感觉，简单的是能够自己开发发送和接收 E-mail 的程序，难点在于要制作一个能够运行良好、能够处理多数邮件格式的 E-mail 处理程序也就不是很容易的事了，如协议、Winsock 编程、编码等。不过，如果不想研究 E-mail 的核心内容，但是又想设计一个符合自己应用或加在其他 Application 中的 E-mail 软件，MAPI 或许能够帮忙。

从 Windows 95 开始，Windows 就提供了一个消息处理应用程序接口（MAPI），通过这个接口可用于创建具有电子邮件功能的应用程序，而不需要懂得 E-mail 的一些协议问题。MAPI 是一系列的核心系统部件，它们可将任何用于电子邮件或工作组的应用程序，和适应 MAPI 的消息服务天衣无缝地连接起来。例如，通过使用 MAPI 驱动程序，Microsoft Exchange 消息系统可被连接到绝大多数私用或公用电子邮件系统中。

后面我们将会给出一个使用 VB 实现 MAPI 的实例，可以用于收发电子邮件。

5.11.2　在 VB 中使用 MAPI

首先，使用 MAPI 有一个前提条件，就是 Windows 下安装了 Microsoft Exchange 或 Outlook 电子邮件组件。如果要运行使用了 MAPI 控件的程序，则可以保证已经正确安装了 32 位的 MAPI DLL，否则就不能完成像 SignOn 之类的 MAPI 功能。

MAPI DLL 提供了一系列的 API 函数，让设计人员能够实现电子邮件系统的一般功能，

使用这些 API 函数是比较繁琐的事情，读者可以打开 VB 自带的 API Viewer 将 Mapi32.txt 加载查看 MAPI 函数。

所幸的是，在 VB6 中提供了两个控件 MAPISession 和 MAPIMessages。MAPISession 控件用以登录并建立 MAPI 会话。它也可用于退出 MAPI 会话。MAPIMessages 控件包含了完成上面所说的消息系统功能所需的所有属性和方法。这两个控件在程序运行时不可见，但可以通过它们进行邮件会话及操作。

在利用 MAPISession 建立邮件对话之前，首先要了解其属性和方法。

1. MAPISession 控件及使用

MAPISession 的常用属性：

（1）Action 属性

该属性决定激活 MAPI Session 控件时执行什么动作，如表 5-10 所示。

表 5-10 Action 属性

常数	值	描述
MapSignOn	1	将用户加载到根据 Usre name 和 Password 属性指定的账户下，并对当前的消息系统提供了一个会话句柄。会话句柄保存在 SessionID 属性中。依 NewSession 属性的值，会话句柄可能参考一个新创建的会话或已有的会话。
MapSignOff	2	结束消息会话，并使用户退出指定的账户。

（2）DownloadMail 属性

指定用户从邮件服务器中下载新消息的时间，如表 5-11 所示。

表 5-11 DownloadMail 属性

设置值	描述
True	（缺省值）在启动过程中来自邮件服务器的所有新消息将进入用户的收件箱。
False	服务器中的新消息不是立即顺利进入用户的收件箱中，而是在用户设定的时间间隔内进入新消息。

（3）LogonUI 属性

指定是否为启动会话提供一个对话框，如表 5-12 所示。

表 5-12 LogonUI 属性

设置值	描述
True	（缺省值）对话框提示用户输入用户名和密码（除非已经存在一个有效的消息会话）。
False	不显示对话框。

✍ 要开始一个邮件会话但又不希望用户干预，并且已经有了该用户的用户名和密码，设置成 False 很有用。然而，如果没提供足够的信息或提供了错误的值时，将产生错误。

（4）NewSession 属性

指定是否建立一个新的邮件会话，即使现在已存在一个有效的会话，如表 5-13 所示。

表 5-13　　NewSession 属性

设置值	描述
True	无论是否已有一个有效会话，都建立一新的消息会话。
False	（缺省值）使用用户已建立的会话。

（5）UserName 属性与 PassWord 属性

UserName：指定账户用户名，或建立会话所使用的配置文件。

PassWord：指定和 UserName 属性相关的账户密码。

✍ 在安装了 Microsoft Exchange 或 Microsoft Outlook 的计算机上，UserName 属性设置了当建立会话时所使用的配置文件。配置文件包括用户实际使用的名称和口令，使得 Password 属性不再使用。通过单击 Windows 控制面板上的 Mail and Fax 图标可以建立新的配置文件。

（6）SessionID 属性

在用 SignOn 方法（详见后面章节）建立了消息处理会话后，该属性将返回一个唯一的消息处理会话句柄。SessionID 的值可被 MAPIMessages 控件用来创建与合法消息处理会话的关联。

MAPISession 的方法：

◆ SignOn 方法

登录用户到由 UserName 和 Password 属性指定的账户中，并且提供会话句柄给当前的消息系统。

◆ SignOff 方法

结束消息会话并且将由 UserName 和 Password 属性指定的用户从账户中退出。

SignOn 方法与 SignOff 方法的作用与设置相应的 Action 属性执行的动作一样。

在用 MAPISession 和 MAPIMessages 控件进行收发电子邮件前，必须用 MAPISession 控件建立邮件对话，然后将建立邮件对话返回的句柄交给控件 MAPIMessages，由 MAPIMessages 进行邮件的管理和操作。下面用简单的子程序说明建立对话的过程。

```
Private Sub BuildMailDialog()
MAPISess.UserName="username"
MAPISess.Password="userPassword"
MAPISess.NewSession=TRUE
MAPISess.LogonUI=TRUE
MAPISess.DownLoadMail =TRUE
MAPISess.SignOn
If Err <> 0 Then
    MsgBox "登录失败:" + Error$
End If
```

```
End Sub
```

2. MAPIMessages 控件及使用

在用 MAPISession 控件建立一个消息会话后，设置 MAPIMessages 的属性 SessionID 与 MAPISession 的属性 SessionID 相关联，这样，就可以用 MAPIMessages 控件执行各种消息系统功能。

使用 MAPIMessages 控件可以执行以下的操作：

- 访问当前收件箱中的消息。
- 构成一条新消息。
- 添加及删除消息收件人和附件。
- 发送消息（无论有无支持的用户接口）。
- 保存、复制、和删除消息。
- 显示“通信簿”对话框。
- 显示“详细资料”对话框。
- 访问附件，包括对附件连接和嵌入（OLE）附件。
- 在寻址过程中，分析一个收件人的名字。
- 对消息执行应答、全应答和转发操作。

在利用 MAPIMessages 控件进行各种操作之前，熟悉其属性和方法是很有必要的。

MAPIMessages 控件的大部分属性可分为 4 个功能区：通信簿、文件附件、消息和收件人属性。文件附件、消息和收件人属性分别由 AttachmentIndex、MsgIndex 和 RecipIndex 属性控制。

（1）有关通信簿的属性（如表 5-14 所示）

表 5-14　　通信簿的属性

通信簿属性	意 思
AddressCaption	指定通信簿中出现的标题
AddressEditFieldCount	指定在通信簿中为用户显示哪些编辑控件
AddressLabel	指定通信簿中“To”编辑控件的外观
AddressModifiable	指定用户是否可以修改通信簿
AddressResolveUI	在指定 ResolveName 方法后，指定在寻址期间是否为收件人名的分析显示一对话框。

（2）有关文件附件的属性（如表 5-15 所示）

表 5-15　　文件附件的属性

文件附件的属性	意　　思
AttachmentCount	返回和当前索引的消息相关联的附件的总数
AttachmentIndex	设置值当前索引的附件
AttachmentName	指定当前索引附件文档的名字
AttachmentPathName	指定当前索引附件的全部的路径名
AttachmentPosition	指定消息体内当前索引附件的位置
AttachmentType	指定当前索引文件附件的类型

在要发送的消息中添加文件附件，只要在属性 AttachmentPathName 中指定要附加的文件的名称和路径。如下所示：

```
MAPIMess.AttachmentPathName="C:\sendtext.txt"
```

如果路径名为空，则会产生错误。

AttchmentName 属性可为文件附件指定不同的名称，如果没有指定这个属性，在消息体中将显示该文件的真实名称。

AttchmentPosition 属性用来指定附件在消息体中的位置。缺省值为 0，这时附件将被放在消息体开始的位置。如果需要将附件放在消息体的最后，则要计算消息正文的长度，如果发送的消息的字符数为 10，则要设置该属性的值为 10（其中 0～9 的位置被消息体的正文字符所占用）。

在同一个消息中，两个不同的附件不能放在相同的位置上。也不能将附件放在消息体的结尾或结尾之后。可以在消息体的末尾加上一个多余的空格或 vbCrlf 字符，然后将 AttchmentPosition 的属性设置为比 MsgNoteText 的属性长度少 1 的值。

（3）有关消息的属性（如表 5-16 所示）

表 5-16　　有关消息的属性

通信簿属性	意思
MsgCount	用于在消息会话期间返回存在于消息设置中的消息总数
MsgDateReceived	返回收到的当前索引消息的时间
MsgID	返回当前编号消息的标识字符串
MsgIndex	指定当前索引消息的索引数
MsgNoteText	指定消息的文本部分
MsgOrigAddress	返回当前索引消息的原始发件人的邮件地址
MsgOrigDisplayName	返回当前索引消息的原始发件人的名字
MsgRead	标识消息是否已经被读过，该值为 TRUE 时标识当前索引的消息已经被读过，为 FALSE 时表示当前索引的消息未被读过
MsgReceiptRequested	指定对当前索引消息是否要求一返回收条
MsgSent	指定对当前索引消息是否已被发送给邮件服务器并等待发送
MsgSubject	当前索引消息的主题
MsgType	为当前索引的消息指定类型

利用这些有关消息的属性，就很容易进行消息的管理和阅读，如在建立基本的邮件对话时，从邮件服务器下载了邮件消息后，就可阅读指定的消息了，如下面子程序所示：

```
Private Sub ViewMessage()
MAPIMess.MsgIndex=1
'将当前的消息索引设置为 1，阅读第二条消息

Dim strMsg as String
strMsg=MAPIMess.MsgSubject
'取得当前消息的主题
strMsg=strMsg & MAPIMess.MsgNoteText
'取得当前消息的正文内容
```

```
MsgBox strMsg
End Sub
```

（4）有关收信人的属性（如表 5-17 所示）

表 5-17　　有关收信人的属性

通信簿属性	意思
RecipAddress	指定当前索引的收信人的电子邮件地址
RecipCount	用一长整型返回当前索引消息的的收信人的总数
RecipDisplayName	指定当前索引收信人的名字
RecipIndex	设置当前索引的收信
RecipType	设置当前索引收信人类型

在发送新邮件消息之前，设置收信人的属性是很重要的，下面举一个简单的例子说明这些属性的使用方法。假设邮件的“发送”收信人为字符串数组中的第一个数 receiver(0)，“转发”收信人为其余的数。

```
Private Sub SetMailReceivers()
Dim Rcount as Integer
'记录收信人的总数的变量

Rcount=MAPIMess.RecipCount
'取得收信人的数量

Dim I as Integer
MAPIMess.RecipIndex=0

'设置当前收信人的索引为 0
MAPIMess.RecipDisplayName=receiver( 0 )
MAPIMess.RecipType=mapToList
'
收信人接收“发送”的邮件。
For I=0 to Rcount - 1
    MAPIMess.RecipIndex=I
    MAPIMess.RecipDisplayName=receiver( I )
    MAPIMess.RecipType=mapCcList
    '其余收信人接收“抄送”的邮件。

Next I
End Sub
```

MAPIMessages 的常用的方法，如表 5-18 所示。

表 5-18　　MAPIMessages 控件的常用的方法

方法	功能
Compose	书写一条消息
Copy	将当前索引的消息复制到书写消息的缓冲区
Delete	删除一消息、收件人、或附件
Fetch	由收件箱中选择的消息创建一消息集合
Forward	转发一个消息。该方法将当前索引的消息作为一个转发消息复制到构成缓冲区中，并且将“FW:”加到主题的开始。同时将 MsgIndex 属性置为：-1
Reply	答复消息
ReplyAll	对所有消息收件人的响应
ResolveName	分析当前索引的收件人的名字。该方法搜索通信簿以寻找当前索引的收件人名字的一个匹配。如果未找到匹配，则返回一个错误。它并不对消息发送者的名字和地址提供附件的分析方法
Save	保存当前构成缓冲区中的消息（MsgIndex=-1 的条件下）
Send	发送一条消息
Show	显示“邮件通信簿”对话框或当前索引收件人的细节

5.12　MAPI 高级编程

5.12.1　建立工程项目

整个工程由两个窗体构成：frmMain 和 frmRead，其中 frmMain 是程序的主窗体，如图 5-16 所示。

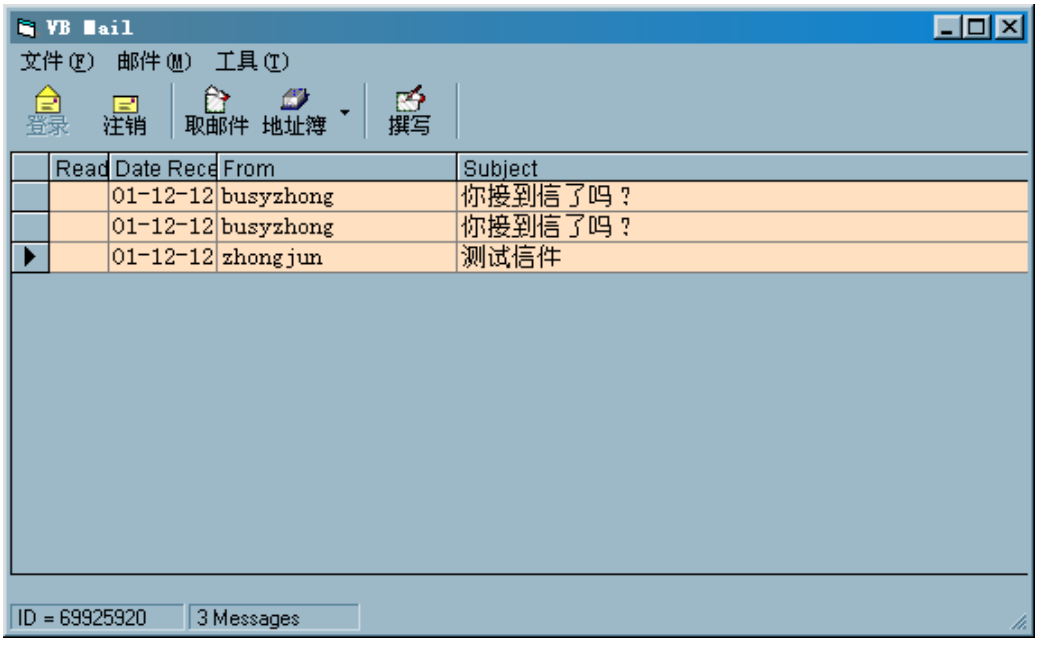


图 5-16　程序主界面

5.12.2 代码分析

1. 窗体 frmMain

该窗体上存在一个 MAPISession 和 MAPIMessage 控件，用于进行邮件会话和消息管理，其他的控件如 DataGrid 用于显示消息（信件），此外还有工具条、状态条等，请读者打开相应的工程项目。

```
Option Explicit
Private bNewSession As Boolean          '登录状态的标志。
'每当执行一个导致 DataGrid 控件的 BeforeColEdit 事件发生的操作设置 gbIgnoreEvent 为真，这样
'此事件将不再执行其他操作而退出。
Private gbIgnoreEvent As Boolean
Private WithEvents rsUnread As ADODB.Recordset
Private gbGridConfigured As Boolean     '配置 DataGrid 标志
Private gbRSalreadyPopulated As Boolean     '给出 RS 被组织的信号的标志
```

上面的代码定义窗体模块变量，其中 rsUnread 定义成为 ADO 记录集，用于管理消息内容。读者可以体会到这样定义在程序的设计过程中是非常方便的，当然也可以定义成变量数组的形式。

```
Private Sub Form_Load()
sbrMapi.Panels(1).Key="SessID"          '配置状态栏
sbrMapi.Panels.Add , "MsgCnt"           '添加窗格
End Sub
```

上面代码用于增加状态条的窗格，可以在设计时定义。

```
Private Sub tbrMail_ButtonClick(ByVal Button As MSComCtlLib.Button)
'使用 Select Case 语句来决定那个按钮被单击，然后作出适当的反应。
On Error GoTo ButtonErr
Select Case Button.Key
Case "LogOn"
'登录并取未读信息。CheckRS 将检查 ADORecordset 是否已经被组织了
'以及是否需要组织。DoGrid 配置窗格并且用记录集进行组织。
   If LogOn=True Then
      MyFetchMail
      CheckRS
      DoGrid
   Else
      Exit Sub
   End If
Case "logOff" '注销
   LogOff
```

```
Case "fetch"
    MyFetchMail
    '检查 ADO Recordset 是否已经被组织。然后组织窗格
    CheckRS
    DoGrid
Case "compose" '创建新的信息
    ComposeMessage
Case "address"
    Debug.Print "something else"
Case Else
    Debug.Print Button.Key
End Select
Exit Sub
ButtonErr:
Debug.Print Err.Number, Err.Description
Resume Next
End Sub
```

上面的代码是响应工具条按钮单击事件的过程，工具条的大小、位置、图标等都是在设计时定义的。一共有 5 个工具条按钮（“登录”、“注销”、“取邮件”、“地址”和“撰写”）。其中地址下有两个工具条菜单，用于显示不同的地址簿。

```
Private Sub mnuAddress_Click()
'显示地址簿。
On Error GoTo AddressErr
mapMess.Show True
Exit Sub
AddressErr:
Debug.Print Err.Number, Err.Description
Resume Next
End Sub

Private Sub mnuCheck_Click()
MyFetchMail             '取未读信息。
CheckRS             '检查记录集并且填充。
DoGrid              '设置窗格 DataSource 为记录集。
End Sub

Private Sub mnuExit_Click()
'如果没有完成则取消，并且卸载此窗题。
If mapSess.SessionID <> 0 Then mapSess.SignOff
Unload Me
```

```
End Sub

Private Sub mnuLogOff_Click()
LogOff
End Sub

Private Sub mnuLogOn_Click()
If LogOn=True Then
   MyFetchMail
   CheckRS
   DoGrid
Else
   Exit Sub
End If
End Sub
Private Sub tbrMail_ButtonMenuClick(ByVal ButtonMenu As
MSComCtlLib.ButtonMenu)
On Error GoTo btnClickErr
Select Case ButtonMenu.Key
Case "global"
   mapMess.Show False
Case "recepient"
   mapMess.Show True
End Select
Exit Sub
btnClickErr:
If Err.Number=mapUserAbort Then
   Resume Next
Else
   MsgBox Err.Number & ": " & Err.Description
End If
End Sub
```

上面的代码是用于处理与对应工具条按钮的菜单事件。

```
Private Function LogOn() As Boolean
'创建名为 rsUnread 的 Recordset 对象
CreateRS
'如果会话已经启动,退出此函数。
If mapSess.NewSession Then
   MsgBox "会话已经被建立"
   Exit Function
```

```
    End If
    On Error GoTo errLogInFail
    With mapSess
        '设置 DownLoadMail 为 False 来防止直接下载。
        .DownLoadMail=False
        .LogonUI=True '使用下列的电子邮件系统的登录 UI。
        .SignOn 'Signon method.
        ' 如果成功，则返回 True 。
        LogOn=True
        ' 设置 NewSession 为真并且 set0
        ' 变量的标志也为真。
        .NewSession=True
        bNewSession=.NewSession
        mapMess.SessionID=.SessionID '您必须在继续前设置它
        sbrMapi.Panels("SessID")="ID=" & .SessionID
    End With
    ' 使按钮可用或禁用。
    ToggleButtonEnabled
    Exit Function
    errLogInFail:
    Debug.Print Err.Number, Err.Description
    If Err.Number=32003 Then
        MsgBox "取消登录"
        LogOn=False
    End If
    Exit Function
    End Function
```

以上代码是登录时启动邮件会话的。如果没有指定 MAPISession 的 UserName 属性的值，登录时系统会弹出“选择配置文件”对话框，如图 5-17 所示。

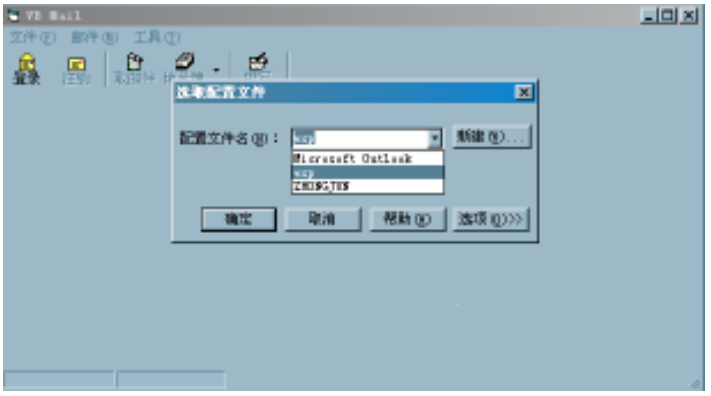


图 5-17　配置文件对话框

由于配置文件中保存了进行收发邮件的一些设定，所以指定配置这一步是必须的，从图上可以看到，有 3 个配置文件可以选择。一个是“ Microsoft Outlook ”，这是在 Windows 98 下使用的 Outlook 程序使用的当前配置，另外还有两个配置文件“ wxp ”、“ ZHONGJUN ”是用户自设定的配置文件。选择配置文件这一步也可以在程序中实现，可以指定 MAPISession 的 UserName 属性值为“ wxp ”，在登录时就不会弹出上面的对话框，直接按照配置文件“ wxp ”指定的设置登录。

关于配置文件的创建和设置，可以从程序登录时弹出的“ 选择配置文件 ”对话框中“ 新建… ”按钮来实现，也可以从控制面版中选择“ 电子邮件 ”项目，如图 5-18 所示。

图 5-18　控制面版中选择电子邮件

下面就从程序的“ 选择配置文件 ”对话框中创建配置文件介绍其内容。

第 1 步：首先选择 Internet Mail 信息服务，如图 5-19 所示，这个是最常用的服务，直接从指定的 Internet E-mail 服务器上下载信件。

图 5-19　创建配置文件向导

第 2 步：选择所要创建的配置文件名，如图 5-20 所示。

图 5-20　创建配置文件向导

第 3 步：选择连接 Internet Mail 服务器通过局域网或是 Modem 连接，如图 5-21 所示。

图 5-21　创建配置文件向导（c）

第 4 步填写 POP3 邮件服务器的地址，如图 5-22 所示。

图 5-22　创建配置文件向导（d）

第 5 步：选择传输邮件方式是“脱机”还是“自动”，这里选择“自动”就行了，如图 5-23 所示。

图 5-23　创建配置文件向导（e）

第 6 步：填写电子邮件地址和全名，如图 5-24 所示。

图 5-24　创建配置文件向导（f）

第 7 步：填写邮箱名和密码，如图 5-25 所示。

图 5-25　创建配置文件向导（g）

接下来将提示输入“个人通讯录”路径，如图 5-26 所示。

图 5-26　创建配置文件向导（h）

最后选择个人文件夹和通信录的文件路径，如图 5-27 所示。

图 5-27　创建配置文件向导（j）

```
    Private Sub grdMess_MouseUp(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    On Error Resume Next
    If mapSess.SessionID=0 Or mapMess.MsgCount=0 Then Exit Sub
    'grdMess.Row=grdMess.RowContaining(Y)
    If gbIgnoreEvent Then Exit Sub
    On Error GoTo RowERR
    mapMess.MsgIndex=grdMess.Columns(0).Value '设置 MsgIndex.
    Load frmRead ' 加载此窗体。
    '对信息及标题并且将它们放在适当的文本框中
    With frmRead
        .lblFrom=mapMess.MsgOrigDisplayName
        .lblCC=GetList(mapCcList)
        .lblTo=GetList(mapToList)
        .lblSubject=mapMess.MsgSubject
        .txtRead=mapMess.MsgNoteText
    End With
    grdMess.Columns("Read").Text="X"
    frmRead.Show
    Exit Sub
    RowERR:
    Debug.Print Err.Number, Err.Description
    Resume Next
    End Sub
```

上面的代码响应 DataGrid 控件 grdMess 鼠标单击事件，从 MAPIMessage 中取出消息主题、发送人、消息正文等信息，弹出窗体 frmRead 显示这些信息。

```
    Private Sub CheckRS()
    '在继续运行前检查标志 gbRSalreadyPopulated
    '在重新组织它之前清除记录集
```

```
If gbRSalreadyPopulated Then
    ClearRS
    PopulateRS
Else
    PopulateRS
    gbRSalreadyPopulated=True
End If
End Sub
```

以上的代码用于根据 gbRSalreadyPopulated 检查 DataGrid 是否已经重新设置了有关的邮件消息的数据。如果没有，更新并设置 DataGrid 显示内容。

```
Private Sub DoGrid()
' 在继续运行前检查标志 gbGridConfigured,如果已经被配置，则清除它
If gbGridConfigured Then
    gbIgnoreEvent=True
    grdMess.HoldFields
    Set grdMess.DataSource=rsUnread
    gbIgnoreEvent=False
Else
    ConfigureGrid
End If
End Sub

Private Sub ComposeMessage()
On Error GoTo ComposeErr
Dim strMessage As String
'使用 Compose 方法并激活 Send 方法。
mapMess.Compose
mapMess.Send True
Exit Sub
ComposeErr:
Debug.Print Err.Number, Err.Description
Resume Next
End Sub
```

以上代码用于调用系统编辑电子邮件的窗体编辑并进行发送。如果 MAPIMessage 的 Send 方法使用 False，则必须自己提供编写电子邮件的窗体并发送。

```
Private Sub CreateRS()
'创建 ADO 记录集并且添加字段。每个字段在 DataGrid 控件中形成一列。
Set rsUnread=New ADODB.Recordset
With rsUnread.Fields
    .Append "ID", adSmallInt
```

```
    .Append "Read", adBSTR
    .Append "Date Received", adDate
    .Append "From", adBSTR
    .Append "Subject", adBSTR
End With
rsUnread.Open
End Sub
```

以上代码用于创建保存消息内容的记录集。

```
Private Sub MyFetchMail()
With mapMess
    '将所有信息全部读取出来，然后在状态栏中显示信息数。
    .FetchUnreadOnly=False
    .Fetch
    sbrMapi.Panels("MsgCnt").Text=.MsgCount & " Messages"
End With
End Sub
```

以上代码用于从 MAPIMessage 控件中读取登录后的消息。

```
Private Sub ClearRS()
'清除所有行的记录集。
If rsUnread.RecordCount=0 Then Exit Sub
Dim i As Integer
gbIgnoreEvent=True
rsUnread.MoveFirst
For i=1 To rsUnread.RecordCount
    rsUnread.Delete adAffectCurrent
    DoEvents
Next i
gbIgnoreEvent=False
End Sub
```

以上代码用于清除记录集中的消息。

```
Private Sub PopulateRS()
gbIgnoreEvent=True '作防止 RowColChanged 事件发生的标志。
Dim i As Integer
For i=0 To mapMess.MsgCount - 1
    mapMess.MsgIndex=i
    rsUnread.AddNew
    rsUnread!ID=i
    rsUnread![date received]=mapMess.MsgDateReceived
    rsUnread!From=mapMess.MsgOrigDisplayName
```

```
    rsUnread!subject=mapMess.MsgSubject
Next i
gbIgnoreEvent=False '重新设置标志。
End Sub
```

以上代码用于根据 MAPIMessage 控件中的消息设置记录集。

```
Private Sub ConfigureGrid()
'在设置 DataSource 到记录集前，设置窗格列的宽度。
gbIgnoreEvent=True
With grdMess
    Set .DataSource=rsUnread '激活事件
    .Columns("ID").Width=0   '隐藏 ID 列。
    .Columns("Read").Width=500
    .Columns("Date Received").Width=900
    .Columns("From").Width=2000
    .Columns("Subject").Width=5100
End With
Dim fmtdate As StdDataFormat
'使用 Format 对象来格式化日期列。
Set fmtdate=New StdDataFormat
With fmtdate
    .Type=fmtCustom
    .Format="Short Date"
 End With
Set grdMess. _
Columns("Date Received").DataFormat=fmtdate
gbIgnoreEvent=False
gbGridConfigured=True '设置标志使我们知道我们不必再执行此过程。
End Sub
```

以上代码用于设置并调整 DataGrid 控件的布局。

```
Private Sub LogOff()
' 注销 MapSessions 控件。
With mapSess
    .SignOff ' 关闭会话。
    .NewSession=False ' 为新的会话设置标志。
    bNewSession=.NewSession ' 重新设置标志。
End With
' 禁用或可用按钮。
ToggleButtonEnabled
rsUnread.Close ' 关闭 ADO 记录集并且设置变量为没有。
```

```
    Set rsUnread=Nothing
    gbRSalreadyPopulated=False
    Unload frmRead ' 卸载此窗体。
    grdMess.ClearFields ' 清除窗格。
    End Sub
```

以上代码用于注销邮件会话过程，将各控件、变量及对象恢复到初始状态。

```
    Private Sub ToggleButtonEnabled()
    '切换各个按钮的 Enabled 属性。
    With tbrMail
        .Buttons("LogOn").Enabled=Abs(.Buttons("LogOn").Enabled) - 1
        .Buttons("logOff").Enabled=Abs(.Buttons("logOff").Enabled) - 1
        .Buttons("fetch").Enabled=Abs(.Buttons("fetch").Enabled) - 1
        .Buttons("compose").Enabled=Abs(.Buttons("compose").Enabled) - 1
        .Buttons("address").Enabled=Abs(.Buttons("address").Enabled) - 1
        .Buttons("address").ButtonMenus(1).Enabled=Abs(.Buttons("address").But
tonMenus(1).Enabled) - 1
        .Buttons("address").ButtonMenus(2).Enabled=Abs(.Buttons("address").But
tonMenus(2).Enabled) - 1
    End With
    '切换菜单的可用属性。
    mnuLogOn.Enabled=Abs(mnuLogOn.Enabled) - 1
    mnuLogOff.Enabled=Abs(mnuLogOff.Enabled) - 1
    mnuTools.Enabled=Abs(mnuTools.Enabled) - 1
    mnuCheck.Enabled=Abs(mnuCheck.Enabled) - 1
    mnuAddress.Enabled=Abs(mnuAddress.Enabled) - 1
    '切换 DataGrid 的可见属性
    grdMess.Visible=Not (mnuLogOn.Enabled)
    End Sub
```

以上代码用于切换菜单及按钮的可用属性。

```
    Private Function GetList(ListType As Integer) As String
    '这个函数获得信息的所有接收者并且分别对他们进行处理。
    Dim i As Integer
    Dim strList As String
    For i=0 To mapMess.RecipCount - 1
        mapMess.RecipIndex=i
        If mapMess.RecipType=ListType Then
            strList=strList & mapMess.RecipDisplayName & "; "
        End If
    Next i
```

```
    If strList="" Then
       GetList=""
       Exit Function
    End If
    '从最后接收者的名称中分离出分号。
    GetList=Left(strList, Len(strList) - 2)
    End Function
```

使用函数可获得信息的所有接收者并且分别对它们进行处理

2. **窗体** frmRead

窗体 frmRead **是用于阅读信件消息内容的。在窗体** frmMain **的** grdMess_MouseUp **事件中弹出，在该窗体上，有“回复”、“全部回复”、“转发”三个按钮。代码非常简单，请读者自己分析。程序运行时显示的窗体如图** 5-28 **所示。**

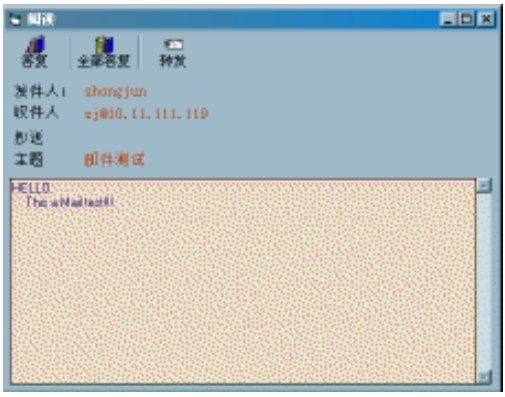


图 5-28　**阅读邮件**

```
    Option Explicit
    Private intMsgIndex As Integer

    Private Sub Form_Load()
    '在发生错误的情况下存储在这。
    intMsgIndex=frmMain.mapMess.MsgIndex
    End Sub

    Private Sub tbrRead_ButtonClick(ByVal Button As MSComCtlLib.Button)
    On Error GoTo SendErr
    Select Case Button.Key
    Case "reply"    '答复
       With frmMain.mapMess
           .Copy
           .Reply
           .AddressResolveUI=True
```

```
        .Send True
    End With
Case "ReplyAll" '全部答复
    With frmMain.mapMess
        .ReplyAll
        .Send True
    End With
Case "forward"  '转发
    With frmMain.mapMess
        .Forward
        .AddressResolveUI=True
        .Send True
    End With
Case Else
    Debug.Print Button.Key
End Select
Exit Sub
SendErr:
'出错后的处理
If Err.Number=mapInvalidComposeBufferAction Then
    frmMain.mapMess.MsgIndex=intMsgIndex '重新设置阅读缓冲区
    Resume
Else '其他情况
    MsgBox Err.Number & ": " & Err.Description
End If
End Sub
```

第 6 章　Telnet 协议及高级编程

远程登录是 Internet 上最广泛的应用之一。“远程登录”就是用户把本地计算机与远程计算机连结起来，然后使用远程计算机系统的资源或其他服务，包括在本地计算机不能获得而从远程计算机上可以获得的其他 Internet 服务。我们可以先登录到一台主机，然后再通过网络远程登录到任何一台网络主机上去，而不需要为每一台主机连接一个主机终端（注意要能成功登录远程主机必须要有登录账号）。

在 TCP/IP 网络上，可以通过两种方法来实现远程登录的功能。

- 通过 Telnet 方法实现远程登录。在 Internet 上，通常使用 Telnet 工具软件实现远程登录，所以习惯上把远程登录称为 Telnet。 Telnet 是提供远程功能的应用标准，几乎每一个 TCP/IP 的实现都有这个功能。它能够运行在不同的操作系统的主机之间。
- 通过 Rlogin 方法实现远程登录。Rlogin 起源于伯克利 UNIX，开始智能工作在 UNIX 操作系统之间，现在可以在其他操作系统上运行。

6.1　Telnet 简介

Telnet 是一种最古老的 Internet 应用，起源于 1969 年的 ARPANET。它的名字是“电信网络协议（Telecommunication Network Protocol）”的缩写。Telnet 是支持远程登录的通信协议，它属于 TCP/IP 通信协议的终端协议部分。Telnet 软件使用 TCP/IP 在用户计算机和远程宿主计算机之间建立一条通信线路，使终端设备通过线路与远程主机连接，提供虚拟终端服务。通过这条临时线路，用户如果在远程系统拥有账号，就可以从本地系统的终端对远程系统登录。这时 Telnet 使本地主机变成远程主机的虚终端，用户可以像使用本地计算机一样地使用远程计算机。Telnet 提供一组命令，作为用户远程登录中使用的工具。

Telnet 同 Internet 上其他服务软件一样，也采用 Client/Server 工作模式。Telnet 应用由两部分软件组成：一部分是客户机程序，运行在提出服务请求的计算机上；而另一部分是服务器程序，运行在提供服务的计算机上。网络则在两者之间提供通信媒介，使用的是 TCP（Transmission Control Protocol）或 UDP（User Datagram Protocol）的服务。

客户机程序在你键入 Telnet 命令时开始执行，它必须完成以下工作：

- 建立与一个远端服务器的 TCP 网络连接。
- 接受本地系统的特定形式的输入。
- 按特定标准对输入的内容重新格式化并发送给远程服务器。
- 从一个服务器接收某种标准形式的输出。
- 对接收的输出内容重新格式化，并在本地系统显示出来。

服务器端程序是服务的提供者。它们是一些总在后台运行的系统进程，一旦该服务器启动以后，就会在特定的端口（默认为 23）侦听是否有客户发出请求。如果服务器程序没有启

动，就不能提供服务。一个典型的服务器程序在准备接受服务请求时，要完成以下工作：

- 告诉联网软件，服务器本身已作好进行连接的准备。
- 侦听一个标准协议格式的客户端请求。
- 对客户请求进行所需的服务。
- 以标准协议格式向客户端机器发送结果。
- 等待下一次请求的出现。

一个服务器必须能够处理多种多样的客户机请求。例如，Telnet 的客户请求有的是来自同一类计算机，有的是来自 IBM/PC、Macintosh 或其他类型的计算机。这就要求服务器采用一组在应用程序之间进行通信的规则，即应用协议（Application Protocol）。这就是 Telnet 所遵从的协议，也是 Internet 应用层网络间通信的标准协议之一。任何人在任何类型的计算机上都能编写客户机程序，只要程序能正确表达协议，就能访问服务器。

6.2　使用 Windows 的 Telnet 程序登录远程服务器

无论是 Windows 98、Windows NT 或者是 Windows 2000 都提供了这样一个程序来实现远程登录的功能。可以远程登录到网络上的其他主机，典型的就是可以通过该程序来连接 BBS 系统，不过，通过该程序来实现 BBS 登录是不明智的，因为它提供的功能有限，而且界面比较难看。这里为了演示其远程登录的功能，用该程序登录到浙江大学 BBS 上。可以在命令行模式输入“telnet 10.13.21.88”（其中 10.13.21.88 为远程 IP，默认为 23 端口）。登录界面如图 6-1 所示，其中账号和密码已经输入正确。

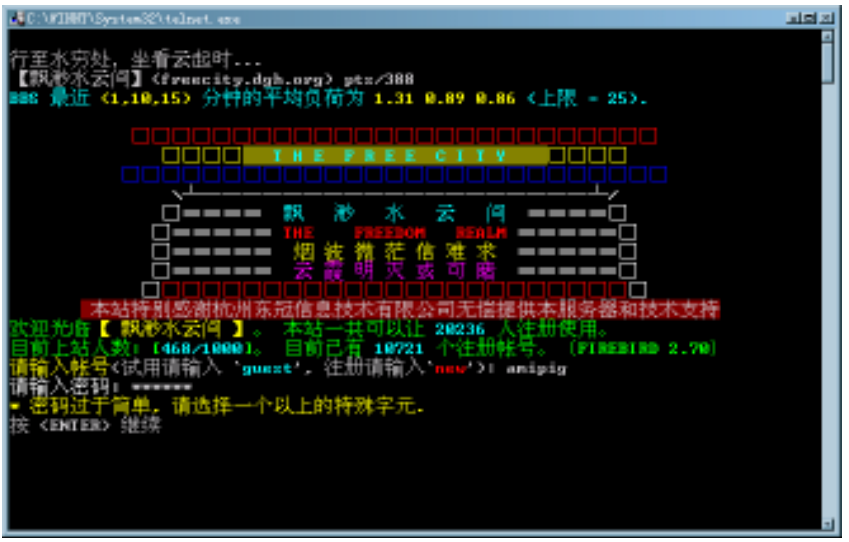


图 6-1　BBS 登录成功

登录成功以后，就可以进入工作页面。此时，用户可以利用系统提供的服务，检索各种信息、发表文章、聊天等等，具体功能相信读者已经知道，这里不再多说。工作界面如图 6-2 所示。

图 6-2　登录成功后工作界面

6.3　深入 Telnet 协议

Telnet 协议可以工作在任何主机、操作系统或者终端之间。该协议是由 RFC854 给出的规范。Telnet 协议建立在 3 个思想基础上。第一是该协议中定义了一种通用字符终端——网络虚拟终端 NVT（Network Virtual Terminal）。NVT 是虚拟设备，连接的双方即客户端和服务器端都必须把它们的物理终端和 NVT 进行相互转换。不管客户终端是什么类型的，操作系统必须把它们转换成 NVT 格式。同理，不管服务器终端是什么类型，操作系统也必须把 NVT 格式转换成为终端所能够支持的格式。第二是具体的命令和协议。第三是终端和进程的均衡性。

6.3.1　NVT ASCII 字符集

术语 NVT ASCII 代表 7bit 的 ASCII 字符集，网络间的协议族都是使用 NVT ASCII 的。每个 7bit 的字符都以 8bit 发送，但最高 bit 为 0。

通常在命令后面以 2 个字符结束，即回车（VBCR）和换行（VBLF）符。单独的一个 VBCR 回车符也是以 2 个字符序列来表示，即回车符后面紧跟着一个空字节 NULL（0 字节）。该字符集的使用是非常广泛的，在 FTP、SMTP、Finger 和 Whois 协议都是以该字符集来描述客户端命令和服务器端响应的。

6.3.2　Telnet 命令

所有的 Telnet 命令至少包括 2 个字节序列。第一个字节为 IAC 字节，即字节 0XFF（10 进制数为 255）。IAC 为（Interpret As Command）的缩写，意思是作为命令来解释，也就是表示后面紧跟着的字节是命令字节，而该字节只不过是先导字节。如果要发送数据 255，则必须发送 2 个连续的字节 255。这里要注意的是，由于 Telnet 是基于 NVT ASCII 字符集，只是 7bit 的格式，因此 255 这个字节不能在 Telnet 上传输。但是在 Telnet 协议中还有一个二进制选项，该选项允许技能型 8bit 的传输。Telnet 协议中的命令并不多，因此在下面详细说明这些命令的使用。

- 命令“BOF”，代码为“236”（如果不作说明，代码均为 10 进制）

该命令表示的是文件结束符。

- 命令“ SUSP ”，代码为“ 237 ”
  该命令表示的是挂起当前的进程。
- 命令“ ABORT ”，代码为“ 238 ”
  该命令表示的是非正常终止当前进程。
- 命令“ EOR ”，代码为“ 239 ”
  该命令表示的是记录结束符。
- 命令“ SE ”，代码为“ 240 ”
  该命令表示的是子选项结束。
- 命令“ NOP ”，代码为“ 241 ”
  该命令表示的是空操作。
- 命令“ DM ”，代码为“ 242 ”
  该命令表示的是数据标记。
- 命令“ BRK ”，代码为“ 243 ”
  该命令表示的是中断。
- 命令“ IP ”，代码为“ 244 ”
  该命令表示的是中断进程。
- 命令“ AO ”，代码为“ 245 ”
  该命令表示的是异常终止进程。
- 命令“ AYT ”，代码为“ 246 ”
  该命令表示的是对方是否还在运行。
- 命令“ EC ”，代码为“ 247 ”
  该命令表示的是转义字符。
- 命令“ EL ”，代码为“ 248 ”
  该命令表示的是删除行。
- 命令“ GA ”，代码为“ 249 ”
  该命令表示的是继续进行。
- 命令“ SB ”，代码为“ 250 ”
  该命令表示的是子选项开始。
- 命令“ WILL ”，代码为“ 251 ”
  该命令表示的是协商选项。
- 命令“ WONT ”，代码为“ 252 ”
  该命令表示的是协商选项。
- 命令“ DO ”，代码为“ 253 ”
  该命令表示的是协商选项。
- 命令“ DONT ”，代码为“ 254 ”
  该命令表示的是协商选项。
- 命令“ IAC ”，代码为“ 255 ”
  该命令表示的是命令开始标记，即该字节后面的当作命令来解释。

以上列出了大部分的 Telnet 的命令，这些命令都是来自 RFC。关于 RFC，本书附的光盘里面有详细的英文原稿。RFC 版本不断地升级，因此协议的内容也在不断地增加。如果想对此编程有深入的了解，可以去研究 RFC，那样能理解得更深刻。

6.3.3 协商选项

在前一节介绍 Telnet 命令的时候，后面有几个命令被解释成为协商选项。顾名思义，也就是需要客户端和服务器端进行协商，然后进行其他的处理。通常这些协商选项都是在刚开始连接服务器的时候使用的。

通常可以默认通过 Telnet 连接的双方都是 NVT，但是实际上 Telnet 连接双方都是先互相发送协商选项数据的。协商选项是相互的和对称的，无论是客户还是服务器都能够主动发送协商选项给对方，以等待对方的回答。接下来对上面提到的 4 个协商选项作一个说明。

- WILL 协商选项：该选项表示的是发送方想主动激活某个子选项。
- DO 协商选项：该选项表示的是发送方想让对方激活某个子选项。
- WON'T 协商选项：该选项表示的是发送方想主动禁止某个子选项。
- DON'T 协商选项：该选项表示的是发送方想让对方禁止某个子选项。

发送协商选项以及接收方的回复都是需要遵从一定的规则的。根据 Telnet 协议的规定，对于发送方的各种激活协商选项的请求，接收方有权接受请求或者拒绝请求。而对于请求禁止某个协商选项的请求，接收方必须同意。因此对于上面的这 4 个协商选项，有 6 种请求和应答方式，如表 6-1 所示。

表 6-1　　6 种协商选项的情况

发送方（协商请求命令）	接收方（应答命令）	发送和应答选项说明
WILL	DO	发送方想主动激活选项，接收方同意
WILL	DON'T	发送方想主动激活选项，接收方不同意
DO	WILL	发送方想让对方激活选项，接收方表示同意
DO	WON'T	发送方想让对方激活选项，接收方表示不同意
WON'T	DON'T	发送方想主动禁止选项，接收方必须同意
DON'T	WON'T	发送方想让对方禁止选项，接收方必须同意

通常进行协商选项的时候，一般要包括 3 个字节。第一个字节也就是字节 255（IAC），让接收方知道发出的是命令而不是普通的数据。第二字节是发送协商请求命令，如 WILL、DO、DON'T、WON'T 等。第 3 个字节表示的是要激活或者禁止的子选项。在目前的 RFC 规则里面，已经有很多可以协商的子选项，因此如果读者想开发一个完整的服务器或者客户端的 Telnet 程序，需要对这些子选项有所了解。表 6-2 列出了一些常用的协商子选项。

表 6-2　　部分协商子选项

子选项代码（10 进制）	子选项说明
1	表示服务器是否回显
3	抑制继续进行
5	状态
6	定时标记

续表

子选项代码（10 进制）	子选项说明
24	终端类型
31	窗口大小
32	终端速率
33	远程流量控制
34	执行方式
36	环境变量

以上的这些子选项都是从 RFC 中获得的。由于 RFC 在不断地补充，因此这些子选项可能不是来自同一个 RFC 文档。

注意，虽然这些协商选项是对称的，双方都可以提出请求，但是由于 Telnet 协议也是客户/服务器模式，双方运行的是不同的进程，因此有些子选项仅适合服务器端，而有些子选项仅适合客户端。

6.3.4　子协商选项

通常进行协商选项的时候，可能还需要进行下一级的协商。因此有些协商请求可能不仅仅是要激活或者禁止一个选项，而是需要对方返回相应的数据。如指定终端类型，如果用户发出要求终端类型的请求，则服务器端不能简单地返回同意或者不同意，如果接收请求，还需要发送具体的终端类型给对方，这就需要有子协商机制。在 RFC109 中就定义了这些机制。下面就获得终端类型这个例子来作一个说明。

首先发送方发送请求，表示想激活某个选项，可以用如下 3 个字节表示：

```
<IAC,WILL,24>
```

其中 24 表示的是终端类型子选项号。如果接收端表示同意激活该选项，则会响应数据表示同意。格式如下：

```
<IAC,DO,24>
```

其中 DO 表示的是对方同意发送方激活该选项。通常当接收方返回这些数据以后，会接着发送下面的数据：

```
<IAC,SB,24,1,IAC,SE>
```

其中 SB 表示的是子选项的开始，而 SB 后面的 24 表示的子选项代码即终端类型。紧接着是 1，1 表示的是让发送方发送自己的终端类型过来。子选项结束一般也要发 IAC 字节，然后是子选项结束命令即 SE。此时如果发送方接收到上述数据，则会响应相应的终端类型，其格式如下：

```
<IAC,SB,24,0, "A", "B", "C",IAC,SE>
```

以上的格式就是发送方发出的提交终端类型的格式。同理，IAC 是命令起始字节，接着是协商起始命令 SB，0 表示的是“终端类型是”，然后“ABC”表示的是具体的终端类型，根据协议规定，终端类型在发送的时候用大写表示。

6.3.5　Telnet 操作方式

在 Telnet 的服务器和客户进行通信的过程中，通常可以采用 4 种操作方式。下面就这 4

种工作方式作一个简单的介绍。

- 半双工方式

该种方式是 Telnet 默认的操作方式，现在的系统已经很少采用这种方式了。前面介绍了 NVT 即网络虚拟终端，它默认是一个半双工的设备。通常在接收用户输入之前，必须从服务器端获得 GO AHEAD(GA)命令。服务器端不负责回显，而回显是由客户端自己实现的，一般是从 NVT 键盘到 NVT 打印机或者 NVT 显示器。客户端进程到服务器进程只能发送整行的数据。虽然这种方式适用于所有的终端类型，但是显而易见效率是很低的，不能充分发挥目前支持全双工的终端设备的功能。

- 一次一个字符方式

该种方式是一种单个字符发送的操作方式，也就是客户键入的每一个字符都单独发送到服务器，而服务器同时回显部分的数据，当然回显可以通过客户端和服务器进行协商来决定。这种方法，也有一个很大的缺点，就是速度比较慢，因为字符是单个发送的，而且客户端的显示是由服务器负责的，因此当网络拥挤时，回显会很慢，不过这种方法比较安全，因为可以确信数据已经到达了服务器。尽管该种方法速度上有缺陷，但是目前这种方法已经替代第一种方法而成为默认的 Telnet 操作方式。

实现这种方式，只需要发送命令 SUPPRESS GO AHEAD（抑制继续进行）进行协商就可以了。客户端发送 DO SUPPRESS GO AHEAD（客户请求激活）或者服务器端发送 WILL SUPPRESS GO AHEAD（服务器主动激活）都可以实现这种操作方式。

- 准行方式

该方式又称为一次一行方式，是在 RFC 858 中定义的。该规则规定，如果要实现远程回显的一次一个字符方式，回显选项和继续进行选项必须同时有效。而如果要以准行方式工作，只要让其中的一个失效就可以了。

- 行方式

该操作方式是在 RFC 1184 中定义的。这个选项通过客户和服务器的协商而确定，它纠正了准行方式的一些缺陷。目前比较新的 Telnet 实现支持这种方式。

6.4 BBS 客户端高级开发

BBS 常用的客户端软件有许多，目前比较流行的是 CTerm，该系统使用比较方便，功能非常强大，是由国人自己开发的。而在以前常用的软件有 NetTerm，但是由于使用不是很方便，现在使用的人已经不多了。

BBS 应用是 Telnet 协议的一种应用，它的默认端口也是 23。在这个实例中，主要介绍如何和 BBS 服务器系统进行通信，如何向服务器发送数据、接收数据等。在本例题中使用的操作方式是一次一个字符方式，也就是现在运用比较多的一种方式。

在本例题中，读者将会学习到以下关于 BBS 的应用：

- 分析服务器端发来的命令
- 向服务器发送命令
- 设置字体颜色

- 设置坐标位置
- 汉字处理
- 自动登录
- 快速离站

当然这个项目还有许多功能都没有做，读者可以自己扩充，相信只要花一定的时间，还是可以做到像 CTerm 那样好的。

6.4.1　建立工程项目

本实例对 VB 知识的要求比较高，需要有一定的基础。尤其是在字节操作方面，要知道汉字和一般字符的处理和显示、颜色处理、字符和数字的转换等等。这些是 VB 的基础知识，同样在后面的代码分析中也不多做说明，关键的地方在代码中注释。

该应用由以下两个部分组成：

- 窗体 frmBBS，本例题的主窗体。
- 窗体 frmAddress，主要是进行远程服务器地址及相关信息设置。

例题包含的代码和对象不多，共有两个窗体。一个是窗体的主界面，另外一个是辅助界面即地址簿界面。有些菜单中的功能并没有做进去，读者可以自己加进去。

6.4.2　关键代码分析

这部分同其他的例题一样，也是对关键的代码进行分析，主要是对涉及协议底层的知识进行解释说明，而对于 VB 的运用，只是在代码注释中做解释，不单独说明。

1. 窗体 frmAddress

该窗体的代码和功能相对简单，用到的只是 VB 的知识，并没有涉及到具体的协议编程。在窗体中主要是增加、删除地址，并可以设置端口、自动登录字符串和站点描速等。其工作界面如图 6-3 所示。

图 6-3　地址簿工作界面

接下来看具体代码。

（1）变量声明

```
Option Explicit
'定义一数组，来存放远程地址信息
```

```
    Dim SiteInfo(100, 3) As String

    '定义变量，来存放远程服务器的数量
    Dim SiteNum As Integer

    '定义属性变量，该变量可以被外部访问
    'IP 为要连接的远程地址，Port 表示远程服务器端口号，mvarAction 表示的是用户按“ 取消 ”按钮还是“ 连接 ”按钮
    Dim IP As String, Port As Integer, mvarAction As Command

    '定义自动登录字符串，也就是连接到服务器的时候自动登录用户名和密码
    Dim LoginString As String

    '定义枚举变量，表示用户是按了“ 连接 ”按钮还是“ 取消 ”按钮
    Public Enum Command
        comdOK
        comdCancel
    End Enum
    '设置操作属性值，表示是取消还是连接
    Public Property Let Action(ByVal vData As Command)
         mvarAction = vData
    End Property

    '获取操作属性值
    Public Property Get Action() As Command
        Action = mvarAction
    End Property

    '得到 IP 地址属性值
    Public Property Get IPAddress() As String
        IPAddress = IP
    End Property

    '得到端口号属性值
    Public Property Get PortNum() As Integer
        PortNum = Port
    End Property

    '设置 IP 地址属性值
```

```
    Public Property Let IPAddress(ByVal tempIP As String)
        IP = tempIP
    End Property

    '设置端口号属性值
    Public Property Let PortNum(ByVal tempPort As Integer)
        Port = tempPort
    End Property

    '获得自动登录字符串属性值
    Public Property Get LoginStr()
        LoginStr = LoginString
    End Property
```

（2）加入新地址

```
    '按钮 cmdAdd 事件，功能是将新的地址加入到地址簿中
    Private Sub cmdAdd_Click()
    Dim FileNum As Integer
    Dim I As Integer
    Dim Founded, Changed As Boolean

    '首先调用函数 CheckTxt 判断加入地址的一些信息是否合法
    If CheckTxt Then

    '查找该站点是否已经加入到地址簿，
    For I = 0 To SiteNum - 1
                        '循环检查，如果找到，则表示已经加入过，站点名存放在数组 SiteInfo(I,
0)中
        If SiteInfo(I, 0) = Trim(txtSite.Text) Then
            Founded = True
    '然后判断其他的字段是否相同，如 IP 地址、端口号，自动登录字符串
    '如果都相等，表示和以前一样，不需要修改，否则需要修改
    If SiteInfo(I, 1) <> Trim(txtAdd.Text) Or SiteInfo(I, 3) <> txtAuto.Text Or
SiteInfo(I, 2) <>
    Trim(txtPort.Text) Then
        '设置变量，表示站点名没改变，但是其他信息改变
                Changed = True
                '修改相关项，SiteInfo(I, 1)存放远程地址，SiteInfo(I, 2)存放端口号，
SiteInfo(I, 3)存放登
```

```
   '录字符串
            SiteInfo(I, 1) = Trim(txtAdd.Text)
            SiteInfo(I, 2) = Trim(txtPort.Text)
            SiteInfo(I, 3) = txtAuto.Text
            ListAdd.ListIndex = I
         End If
         Exit For
      End If
   Next I

   '如果以前没有加入过，则增加一个新项
   If Not Founded Then
      '首先在列表框中加入该项
      ListAdd.AddItem Trim(txtSite.Text)
      '然后在数组中增加该项
      SiteInfo(SiteNum, 0) = txtSite.Text
      SiteInfo(SiteNum, 1) = txtAdd.Text
      SiteInfo(SiteNum, 2) = txtPort.Text
      SiteInfo(SiteNum, 3) = txtAuto.Text
      '并在列表框中让新增加的项获得焦点
      ListAdd.ListIndex = SiteNum
                  '将站点数增加
      SiteNum = SiteNum + 1

      '获得一个空文件号
      FileNum = FreeFile
      '以增加模式打开该文件
      Open App.Path & "\address.txt" For Append As FileNum
      '向文件写入刚加入的站点信息，以便下次打开程序时能保留信息，每项之间用“——”隔开
      Print #FileNum, Trim(txtSite.Text) & "--" & Trim(txtAdd.Text) & "--" &
Trim(txtPort.Text) & "--" & txtAuto.Text
      '关闭文件
      Close #FileNum
   '如果以前有该站点信息，但是有些项被修改，则需要重新把修改信息写入文件
   ElseIf Founded And Changed Then
      '获得文件号
      FileNum = FreeFile
      '以 output 方式打开文件，这样原来的数据被清空
      Open App.Path & "\address.txt" For Output As FileNum
```

```
    '循环加入每一个站点的信息
        For I = 0 To ListAdd.ListCount - 1
            Print #FileNum, SiteInfo(I, 0) & "--" & SiteInfo(I, 1) & "--" & SiteInfo(I,
2) & "--" & SiteInfo(I, 3)
        Next I
        '关闭文件
        Close #FileNum
    End If
    End If
    End Sub
```

（3）连接服务器。

```
    '该事件是设定一些属性值，让外部程序调用
    Private Sub cmdConnect_Click()
        IP = Trim(txtAdd.Text)
        Port = CInt(txtPort.Text)
        LoginString = Trim(txtAuto.Text)
        mvarAction = comdOK
        '卸载窗体
        Unload Me
    End Sub
```

（4）删除一个地址

```
    '该事件表示删除一个地址
    Private Sub cmdDel_Click()
    '调用函数 DelSite 删除一项
    Call DelSite
    '将站点数减少一
    SiteNum = SiteNum - 1
    '在列表框中删除一项
    ListAdd.RemoveItem (ListAdd.ListIndex)
    '将地址簿定位到第一项
    If ListAdd.ListCount > 0 Then
        ListAdd.ListIndex = 0
        '设置相关文本框的值
        SetTxt (0)
    Else
        ClearTxt
    End If
```

```
End Sub
```

（5）退出窗体

```
'该事件退出窗体
Private Sub cmdExit_Click()
    mvarAction = comdCancel
    Unload Me
End Sub
```

（6）载入窗体

```
'该事件是载入窗体
Private Sub Form_Load()
Dim FileNum As Integer
Dim Address, tempPort, SiteDesc, AutoLogin, TempStr As String

SiteNum = 0
'获得文件号
FileNum = FreeFile
'分析存放地址文件存不存在
If Dir(App.Path & "\address.txt") <> "" Then
    Open App.Path & "\address.txt" For Input As FileNum
    '循环读取每一行
    Do While Not EOF(FileNum)
        Line Input #FileNum, TempStr
        '调用函数 GetValue 来获得每一个具体项
        GetValue (TempStr)
    Loop
    Close #FileNum
End If
End Sub
```

（7）列表框单击事件

```
'设置相应文本框的值
Private Sub ListAdd_Click()
    SetTxt (ListAdd.ListIndex)
End Sub
```

（8）清空文本框中的值

```
'该函数的功能是清空相应文本框的值
```

```
Private Sub ClearTxt()
    txtSite.Text = ""
    txtAdd.Text = ""
    txtAuto.Text = ""
    txtPort.Text = ""
End Sub
```

（9）设置文本框的值

```
'该函数的功能是根据选中的地址设置相应的文本框的值
Private Sub SetTxt(Index As Integer)
    txtSite.Text = SiteInfo(Index, 0)
    txtAdd.Text = SiteInfo(Index, 1)
    txtPort.Text = SiteInfo(Index, 2)
    txtAuto.Text = SiteInfo(Index, 3)
End Sub
```

（10）删除站点函数

```
'该函数的功能是删除一个站点
Private Sub DelSite()
Dim I, FileNum As Integer
FileNum = FreeFile
'以增加模式打开文件
Open App.Path & "\address.txt" For Output As FileNum
'循环写入每一个地址
For I = 0 To SiteNum - 1
    '如果索引结果不等于的索引号，则直接写入文件
    If I <> ListAdd.ListIndex Then
        Print #FileNum, SiteInfo(I, 0) & "--" & SiteInfo(I, 1) & "--" & SiteInfo(I,
2) & "--" & SiteInfo(I, 3)
    End If
    '如果索引大于被删除的索引，则要改变数组相应的值
    If I > ListAdd.ListIndex Then
        ChangeValue (ListAdd.ListIndex)
    End If
Next I
Close #FileNum
End Sub
```

（11）删除记录后，改变相应数组的值

```
'该函数的功能是改变相应的数组值，即向前挪动一个数据
Private Sub ChangeValue(I As Integer)
    SiteInfo(I, 0) = SiteInfo(I + 1, 0)
    SiteInfo(I, 1) = SiteInfo(I + 1, 1)
    SiteInfo(I, 2) = SiteInfo(I + 1, 2)
    SiteInfo(I, 3) = SiteInfo(I + 1, 3)
End Sub
```

（12）获得从文件读取的数据项

```
'该函数的功能是分析字符串，并拆开字符串，将数据项写入到数组中，同时加入到列表框
Private Sub GetValue(TempStr As String)
Dim info As Variant
If Trim(TempStr) <> "" Then
    info = Split(TempStr, "--", , vbTextCompare)
    SiteInfo(SiteNum, 0) = info(0)
    SiteInfo(SiteNum, 1) = info(1)
    SiteInfo(SiteNum, 2) = info(2)
    SiteInfo(SiteNum, 3) = info(3)
    ListAdd.AddItem info(0)
    SiteNum = SiteNum + 1
End If
End Sub
```

（13）检查输入的远程地址信息是否正确，如地址、端口、登录字符串等

```
'该函数是检查文本框输入
Private Function CheckTxt() As Boolean
If Trim(txtAdd.Text) = "" Then
    MsgBox "请输入远程服务器地址！"
    txtAdd.SetFocus
    CheckTxt = False
    Exit Function
End If

If Trim(txtPort.Text) = "" Then
    MsgBox "请输入远程服务器端口号！"
    txtPort.SetFocus
    CheckTxt = False
    Exit Function
```

```
    End If
    If Trim(txtSite.Text) = "" Then
        MsgBox "请输入远程服务器站点描速！"
        txtSite.SetFocus
        CheckTxt = False
        Exit Function
    End If
    If Trim(txtAuto.Text) = "" Then
        txtAuto.Text = " "
    End If
    CheckTxt = True
    End Function
```

（14）双击连接服务器。

```
'该事件激活连接按钮单击事件，用以连接服务器
Private Sub ListAdd_DblClick()
Call cmdConnect_Click
End Sub
```

以上介绍了窗体 frmAddress **的工作代码，注释可以参照光盘实例，因此不单独解释了。**

2. **窗体** frmBBS

该窗体主要是连接服务器，然后显示从服务器发送来的数据或者分析服务器发来的命令，同时接受客户的输入并发送到服务器端。由于例题采用的是一次一个字符方式，回显由服务器完成，因此所有显示在界面上的信息都是由服务器发送过来的。这一点希望读者能够了解。当然读者可以改编程序，使其工作在其他的操作模式下。

当在地址簿中，采用“ 缥缈 ”的账号进行登录后，会自动连接到服务器并显示欢迎信息，并且自动登录，如图 6-4 **所示。**

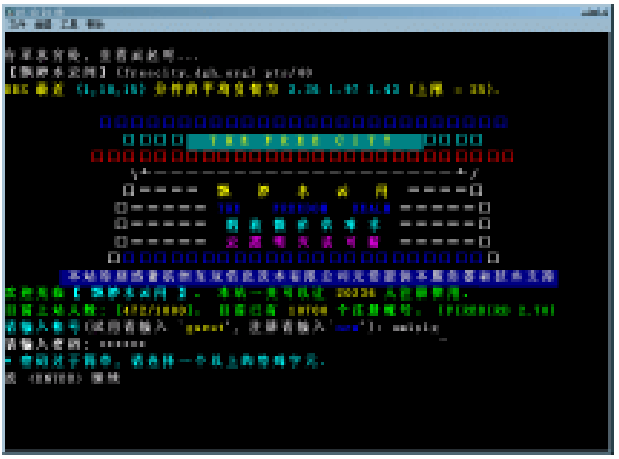


图 6-4　**自动登录到** BBS **站点**

登录成功后，即会自动进入好友名单，如图 6-5 **所示。**

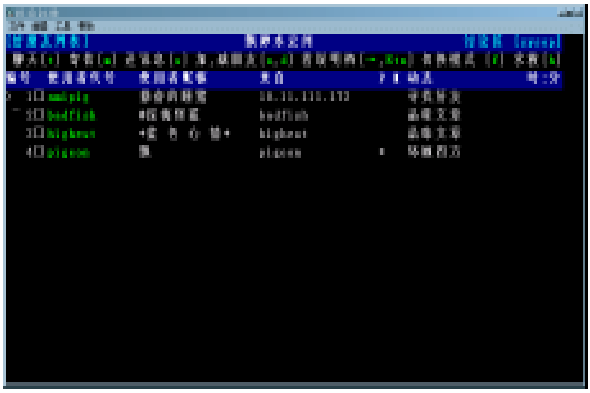


图 6-5　登录成功后显示好友界面

当用户按下“F”键，则会显示所有的用户，如图 6-6 所示。

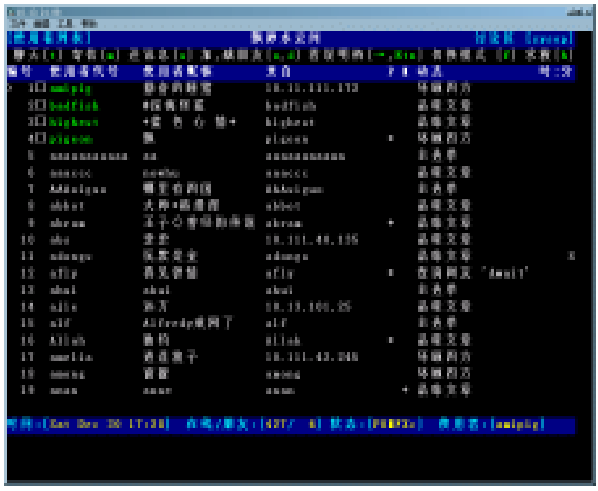


图 6-6　显示所有用户界面

从上面的工作界面看，该程序完全可以实现 BBS 客户端的功能。当然还有其他的许多 BBS 的功能，读者可以自己去测试。

下面就逐一来分析说明这些代码。

（1）变量声明

```
Option Explicit
'下面的声明表示所有以 A 到 Z 开始的函数都是返回整型值
DefInt A-Z
'定义一些基本变量，入是否粗体、反色、前景色、背景色、坐标等
Dim Bold, Reverse, Bcolor, Fcolor, Css, ox, oy
'定义最近一次登录的远程站点地址
Dim LastLoginSite As String
'定义最近一次登录的远程站点的端口号
Dim LastLoginSitePort As Integer
```

```
'定义是否还同服务器连接上
Dim Connected As Boolean
'以下为定义字的宽度和高度
Const WordWidth = 22
Const LetterWidth = 11
Const WordHeight = 22
Const VerSpace = 2
Const HorSpace = 16
'下面是一个 API 声明，表示让程序暂停多少毫秒
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

在上面的变量声明中，注意几个常数的声明，由于在本程序中，显示的字体大小已经确定好，并没有让字体随这窗体的大小变化而变化，或者让用户自定义大小，因此字的高度和宽度都定义为常数。如果读者要改编程序，扩充功能，就定义成变量。而且注意由于 BBS 客户端必须支持汉字的显示，因此汉字的宽度通常为普通字符的两倍，定义的时候也要注意。同时如果要改变字体的大小，这些字的宽度和高度都会改变，读者可以按照一定的比例来缩放或者编程来获得当前字体的高度和宽度，这样显示出来的信息就不会乱，否则显示的信息可能会很乱。

（2）窗体的 KeyDown 事件，用来捕获方向键

```
'在该事件中，如果捕获到是方向键，则发送相应的字节到服务器
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
'判断是否还是连接到服务器
If Connected Then
   Select Case KeyCode
      '上方向键
Case 38: SendChars "27;91;65"

'下方向键
      Case 40: SendChars "27;91;66"

'左方向键
      Case 37: SendChars "27;91;68"

      '右方向键
      Case 39: SendChars "27;91;67"
   End Select
End If
End Sub
```

在该事件中，程序只是简单的发送一些字节就可以了。因为 BBS 系统都是有相关的规定

的，在 BBS 的实际操作中，虽然我们按方向键来左右上下移动光标，但是至于光标的位置，以及进入到下一个页面是哪一个页面，都不需要客户端来处理。因为在服务器端保存了客户端的所有信息，包括现在光标的位置、所在的讨论区或者在哪一篇文章。所以客户只需要操纵相应的方向键，同时发送该方向键对应的字节序列，服务器就能读懂客户的意思，对于客户来说，这是比较简单的。在 CTERM 中，可以使用鼠标，其实使用鼠标除了在服务器端可以处理以外，在客户端也能够处理。在 CTERM 程序的使用过程中可以看到，对于有些 BBS 系统可以使用鼠标，而对于另外的一些系统并不能使用鼠标，因此可以看出这是由服务器来支持的。而如果单纯在客户端实现这种功能，也并不复杂，只需要在程序中记录当前光标的位置，以及鼠标的单击位置，并计算两个位置之间的差距，然后连续发送多个方向键以达到实现鼠标的功能。有兴趣的读者可以自己开发这样的程序。在上面应该注意，不同的方向键发出的字节序列是不同的，如左键发送的是“ 27;91;65 ”，注意经过函数“ SendChars ”处理以后，“ 27 ”、“ 91 ”、“ 65 ”都是以字节连续发送出去的。可以在函数“ SendChars ”中看到具体的发送过程。

（3）捕获除方向键以外的其他输入

```
'窗体的 KeyPress 事件
Private Sub Form_KeyPress(KeyAscii As Integer)
Dim L&
'判断是否和服务器保持连接
If Connected Then
    '然后判断输入字符的 ascii 值，大于等于 0 表示的是单字节字符
    If KeyAscii >= 0 Then
        SendChars Str$(KeyAscii)
    '否则为汉字
Else
        L& = KeyAscii + 65536
        SendChars Str$(L& \ 256) + ";" + Str$(L& Mod 256)
    End If
Else
    '如果没有连接上服务器，则连接最近连接的服务器
    Call mnu_file_connect_Click
End If
End Sub
```

在这个事件中主要是处理除了方向键以外的其他输入，因为客户显示的数据都是通过服务器回显而得到的，因此在用户输入的同时并不需要往客户屏幕上写，而是等待服务器返回数据再显示在屏幕上。当然并不是所有的数据都回显的，一般是在用户密码、用户名或者发表文章的时候才会回显，而其他的时候服务器则以功能键处理，如果是不认识的功能键，则不作处理。如在显示用户列表的时候，如果读者发送字符“ F ”，则会只列出好友名单，而

这个时候“F”并不回显，此时如果按下字符“U”，则不会有任何效果，虽然程序已经将该字符发送到了服务器。

在这个事件中，还有一个必须要注意到的地方就是汉字与字符的区别。通常普通字符是占有一个字节的，而汉字是占用两个字节，因此在发送的时候要作不同的处理。在汉字输入的时候，该事件返回的“KeyAscii”是负数，因此可以通过这个属性来判断是汉字还是普通字符，而且在发送到服务器的时候，也需要拆分成两个字节来处理。通常如果要将“KeyAscii”转化成汉字，只要用“KeyAscii + 65536”就可以转换成汉字的内码，然后分开成高低两个字节发送到服务器，其中“L& \ 256”获得高八位，而“L& Mod 256”获得的低八位。

（4）载入窗体进行初始化设置

```
'窗体载入事件
Private Sub Form_Load()
                    '设置前景色变量
    Fcolor = 7
    '设置背景色变量
    Bcolor = 0
    BackColor = 0

    '获得上次登录的站点信息
    LastLoginSite = GetSetting("MyBBS", "Login", "Site", "")
    LastLoginSitePort = CInt(GetSetting("MyBBS", "Login", "Port", "0"))

    '关闭定时器
    Timer1.Enabled = False

    '如果最近一次连接的站点信息为空，则使菜单项 mnu_file_connect 失效
    If LastLoginSite <> "" Then
        mnu_file_connect.Enabled = True
    End If

End Sub
```

在窗体载入事件。其中没有重要的代码而且比较简单，读者自己看看就可以了，主要是进行一些初始的设置而已。

（5）弹出地址簿窗体，并连接服务器

```
'菜单事件，弹出地址簿窗体
Private Sub mnu_file_book_Click()
'定义一个地址簿窗体对象
```

```
Dim myConnect As New frmAddress
'模态显示该地址簿窗体
myConnect.Show 1
'分析用户是单击连接服务器按钮还是取消按钮，如果是连接服务器，则继续下面的代码
If myConnect.Action = comdOK Then
    '首先关闭当前连接
Winsock1.Close
'连接服务器后面的两个参数分别是远程地址和端口号
    Winsock1.Connect myConnect.IPAddress, myConnect.PortNum

    '修改一些属性值
    Connected = False
    Me.Caption = myConnect.IPAddress
    mnu_file_connect.Enabled = True
    LastLoginSite = myConnect.IPAddress
    LastLoginSitePort = myConnect.PortNum

    '将连接服务器的信息存入注册表，以方便下次连接
    SaveSetting "MyBBS", "Login", "Site", LastLoginSite
    SaveSetting "MyBBS", "Login", "Port", LastLoginSitePort
    '启动定时器
    Timer1.Enabled = True

    '循环等到连接成功或者连接失败
    Do While Not Connected Or Winsock1.State = sckClosing
        DoEvents
    Loop
        '如果连接成功，则调用函数自动登录
    If Connected Then
        AutoLogin (myConnect.LoginStr)
    End If
End If

End Sub
```

在这段代码中，读者要了解到 Telnet 应用也是一种 Socket 通信，因此也是调用 Winsock 对象来连接服务器，如果连接成功，服务器会自动发送数据到客户端。在判断是否成功连接到服务器的方法上可以通过直接判断 Winsock 控件的状态来实现。在该程序中是通过变量来实现的。如果获得服务器端传来的数据，则表示连接成功。

（6）连接上次登录的服务器

```
'菜单事件，用来连接上一次登录的站点，这主要是方便用户操作
Private Sub mnu_file_connect_Click()
    Winsock1.Close
    Winsock1.Connect LastLoginSite, LastLoginSitePort
    Connected = False
    Timer1.Enabled = True
    Me.Caption = LastLoginSite
End Sub
```

（7）快速离站

```
'菜单事件，主要是方便用户，能够进行快速离开站点
Private Sub mnu_file_leave_Click()
Dim I As Integer
On Error Resume Next
Connected = False
'发送五次左方向键来退出系统
For I = 0 To 5
    SendChars "27;91;68"
    '为了让服务器能及时响应，这里调用 sleep 函数暂停执行 50 毫秒
    Sleep (50)
Next I

'对于某些系统，还需要发送几个回车才能够退出系统
For I = 0 To 2
    SendChars "13"
    Sleep (50)
Next I

End Sub
```

上面的代码采用的是连续发送左键的方法快速退出 BBS，因为正常次序是一直按左方向键直到退出登录,因此在快速离站的时候可以利用这个道理连续向服务器发送该指令来退出。而且在快速退出的时候，虽然每次向服务器发送，服务器都会发回相应的数据，但是由于并不需要处理显示，因此可以节省不少时间，已达到快速离站的目录。如果多发了回车键，服务器端并不作处理，所以此操作对速度影响不大。通常要完全地退出登录，还需要加上两个回车，当用户输入回车符的时候，服务器才真正地关闭 Socket 连接。

（8）断开连接

```
'菜单事件，强行断开与服务器的连接
Private Sub mnu_file_off_Click()
    frmBBS.Picture = LoadPicture("")
    frmBBS.Cls
    Winsock1.Close
End Sub
```

断开同快速离站的原理刚好相反。通常，如果用户非正常退出，客户端的一些信息仍然保存在服务器端。采用断开的方法也就是非正常退出，在某用户断开后，其他用户仍然可以看到该用户在站上，服务器在经过一段时间后才删除该用户的信息。快速离站是服务器端主动关闭 SOCKET 连接，而断开是客户端强行调用 Winsock 对象的关闭方法来断开连接，是被动的。在实际的使用过程中，建议用户都能按照正常次序离站，这样可以节省服务器的资源。

（9）定时处理

```
'时间控件的 Timer 事件
Private Sub Timer1_Timer()
Dim X, Y, C As Integer
    '调用 Main 函数处理来自服务器端的数据，该函数在后面介绍
Main

    '下面获得光标的颜色和显示光标的位置
    Css = (Css + 1) Mod 10
    'X、Y 为当前的横、纵坐标
    X = CurrentX
    Y = CurrentY
    '判断上次光标的位置和现在相同不，如果不同，则在原来位置画背景色覆盖原来的光标
    If X <> ox Or Y <> oy Then
        Line (ox, oy + WordHeight)-Step(LetterWidth, 0), QBColor(Bcolor)
        '保留此次的坐标
        ox = X
        oy = Y
    End If

    '根据变量 Css 的值来设置光标的颜色
    If Css < 5 Then
        C = Bcolor
    Else
        C = Fcolor
    End If
```

```
    '然后在新的位置重新画线显示光标
    Line (X, Y + WordHeight)-Step(LetterWidth, 0), QBColor(C)
    CurrentX = X
    CurrentY = Y

End Sub
```

（10）获得服务器端的一个字节

```
'该函数的功能是一次获得一个字节
Function Inkey() As Byte
Dim b As Byte
'如果没有数据到达，则程序作空循环
While Winsock1.BytesReceived = 0
    Nop
Wend
'直到有数据，并获得该数据
Winsock1.GetData b
'返回该字节
Inkey = b
Debug.Print b & "--" & Chr(b)

End Function

'该过程是空循环，等待服务器数据
Sub Nop()
    DoEvents: DoEvents: DoEvents: DoEvents: DoEvents: DoEvents: DoEvents:
    DoEvents: DoEvents
End Sub

'该函数是将字符串转换成数据
Function VVV(D$)
    '通过 val 函数将字符串转换成数据
    VVV = Val(D$)
    '将被转换成数据的字符串去掉
    D$ = Mid$(D$, InStr(D$ + ";", ";") + 1)
End Function
```

（11）向服务器发送数据

```
'循环向服务器发送数据，直到字符串为空
Sub SendChars(D$)
Dim b As Byte
   While D$ <> ""
      b = VVV(D$)
      '发送字节数据
      Winsock1.SendData b
   Wend
End Sub
```

该函数的功能是向服务器发送数据。注意这里不能直接将数据当作字符串发送出去，而必须转换成数值。比如字符串“255;253;1”的发送，是要发送三个数据，如果当作字符串发送，就当作“2、5、5、2、5、3、1”7 个字节发送出去了，但这是不对的。因此可以调用函数 VVV 来每次发送一个数据，即分成 3 次发送，3 个字节依次是 255、253、1。

（12）在客户端显示服务器端发送的数据

```
'该函数负责在客户屏幕上显示数据，
Sub Main()
Dim b As Byte, LL
Dim C, D, V, xx, yy As Byte
Dim X, Y, L, F, T As Integer
Dim s$, dat$, p&
'循环获得数据，直到数据缓冲区中没有数据
While Winsock1.BytesReceived > 0
   '获得一个字节
   b = Inkey
   '然后分析该字节的值，因为不同的值会对应着不同含义
   Select Case b
   Case 255
   '255 为命令的先导字符，即表示后面紧跟着的是命令
      '然后获得后两个字节，来分析具体是什么命令
      C = Inkey
      D = Inkey

      '如果 C 为 253，表示服务器发出 DO 命令，让客户激活某个选项
      '1 表示要回显，24 表示终端类型
      If C = 253 And (D = 1 Or D = 24) Then
         SendChars "255;251;" & D
         GoTo L2
```

```
        End If

        '如果 C 为 254，则表示的是服务器要求客户禁止某个选项，这个时候客户必须同意禁止
        '因此返回码 253，表示同意禁止
        If C = 254 And D = 1 Then
            SendChars "255;252;1"
            GoTo L2
        End If

        '如果 C 为 251，则表示服务器愿意自己激活某个选项，1 表示回显
        '我们可以发送禁止和同意，对于有些 BBS 系统，在登录的时候可能返回两次回显
        '因此可以禁止服务器回显，发出命令 254
        If C = 251 And D = 1 Then
            SendChars "255;254;1"
            GoTo L2
        End If

        '如果为 250，表示子选项开始
        If C = 250 Then
            '循环读取，直到子协商结束，240 表示子协商结束
            '在这个程序中，并没有具体分析子选项，通常是终端类型
            While D <> 240
                D = Inkey
            Wend
            '当子协商结束以后，客户发送自己的终端类型到服务器，这里发送的终端类型为
            'VT100 类型
            SendChars "255;250;24;0;118;116;49;48;48;255;240"
            GoTo L2
        End If

    Case 27
    '如果接收到的字节是 27
        s$ = ""
        '接着获得下一个字节
        C = Inkey
        '如果不等于 91，即字符“]”，则退出这一轮循环，表示不处理这几位数据
        '通常服务器发送客户端的颜色设置或者坐标设置都是以字节 27 和 91 开始的
        If C <> 91 Then
            GoTo L2
```

```
        End If
        '如果 C 为 91，则自行 L1 的代码
L1:
        '读取后面的数据，然后分析获得的数据表示的是什么含义
        '将读取的字节转换成字符
        dat$ = Chr$(Inkey)
        '如果该字符为数字符号或者分号，则将字符累加成字符串
        If InStr(" 0123456789;", dat$) > 1 Then

            s$ = s$ + dat
            '然后继续循环读取数据，直到读到的字符不在以上的那 11 个字符之列
            GoTo L1
        End If

        '然后分析第一个不在上面那 11 个字符之列的字符，根据不同的值作不同的处理
        Select Case dat$
            '如果为字符 m
            Case "m"
                '如果上面获得字符串为空则赋为字符“0”
                If s$ = "" Then
                    s$ = "0"
                End If

                '循环获得字符串中的值，其中字符串中的值都是以分号隔开的
                While s$ <> ""
                    V = VVV(s$)
                    '获得一个值
                    '改变前景颜色，这个颜色可以按照一定的规则来自定义
                    If V > 29 And V < 38 Then
                        Fcolor = V - 30 + Bold * 8
                    End If

                    '设置背景颜色
                    If V > 39 And V < 48 Then
                        Bcolor = V - 40
                    End If

                    '恢复一些基本设置
                    If V = 0 Then
```

```
                Bold = 0
                Reverse = 0
                Fcolor = 7
                Bcolor = 0
            End If

            '改变背景色
            If Bcolor = 4 Then
                Bcolor = 1
            End If

            '如果为 1，设置背景色
            If V = 1 Then
                Bold = 1
                Fcolor = Fcolor Mod 8 + 8
            End If

            '如果为 7，则表示要反色
            If V = 7 Then
                Reverse = 1
            End If
            '改变窗体的前景颜色
            ForeColor = QBColor(Fcolor)
        Wend

    '如果为字符 K，表示要画一条以 Bcolor 背景的矩形区域
    Case "K"
        '保存当前的坐标位置
        X = CurrentX
        Y = CurrentY
        '画出举行区域
        Line (X, Y)-Step(1000, WordHeight), QBColor(Bcolor), BF
        '恢复到原来的坐标，因为画完上述区域后，CurrentX 和 CurrentY 都被改变
        CurrentX = X
        CurrentY = Y

    '如果字符为 C，表示要改变当前的横坐标
    Case "C"
        '获得横坐标的改变量
```

```
            xx = VVV(s$)
            '然后增建坐标值，注意这里要加上字符宽度
            CurrentX = CurrentX + xx * LetterWidth

        '如果字符为 H，表示重新设置横坐标和纵坐标
        Case "H"
            '获得纵坐标值，在服务器发送的过程中，先发送纵坐标值
            yy = VVV(s$)
            '获得横坐标值，注意这里获得值是没有加字符宽度的
            xx = VVV(s$)
            If xx > 0 And yy > 0 Then
                CurrentX = (xx - 1) * LetterWidth
                CurrentY = (yy - 1) * WordHeight
            End If

        '如果为字符 J，表示要清除整个屏幕
        Case "J"
            '清屏
            frmBBS.Picture = LoadPicture()
            frmBBS.Cls
    End Select

Case 7
'如果接收到的是字节 7，则表示发出声音，不同的系统会发送不同个数的字节 7
    Beep

Case 8
'如果是 8，表示将横坐标减少一
    If CurrentX > 0 Then
        CurrentX = CurrentX - LetterWidth
    End If

Case 13
'如果是 13，表示回车，将横坐标设置为 0
    CurrentX = 0

Case 0

Case 10
```

```
        '如果是 10，表示换行，将纵坐标的值增加一
            CurrentY = CurrentY + WordHeight
            '如果整个窗体的宽度是小于当前的坐标值，则要将屏幕滚动
            If CurrentY >= 600 Then
                CurrentY = CurrentY - WordHeight
                frmBBS.Picture = frmBBS.Image

                PaintPicture frmBBS.Picture, 0, -WordHeight
                '并将光标的纵坐标减小一
                oy = oy - WordHeight
            End If

        Case Else
        '如过为其他的字节，则表示是要限制在屏幕上的数据
            p& = -1
            '如果 b 小于 128 表示是普通字符，LL 变量标记第一个字符是否大于 128
            '如果后面紧跟着的字节仍然大于 128，表示是汉字的内码
            If b < 128 Then
                LL = 0
                p& = b
            End If

            '如果 b 大于 128 同时 LL 为 0，表示得到汉字的第一个字节，因此将变量 LL 保存为第一个
字节
            If b >= 128 And LL = 0 Then
                LL = b
            Else
            '如果 LL 不等于 0，则获得汉字内码，并将 LL 清 0
                p& = LL * 256& + b
                LL = 0
            End If

            '接着分析是汉字还是普通字符，来决定字符的宽度
            If p& > 256 Then
                L = WordWidth
            Else
                L = LetterWidth
            End If
```

```
        X = CurrentX
        Y = CurrentY
        F = Fcolor
        b = Bcolor
        '如果反色，则将前景色和背景色互换，注意是变量前景和背景，而不是显示屏的背景和前景
        If Reverse Then
           T = F
           F = b
           b = T
        End If

        ForeColor = QBColor(F)

        If p& >= 0 Then
           Line (X, Y)-Step(L - 1, WordHeight), QBColor(b), BF
           CurrentX = X
           CurrentY = Y
           Print Chr$(p&);
           CurrentX = X + L
        End If
    End Select
L2: Wend
End Sub
```

上面的 Main 函数是该程序的核心之一，其功能是显示服务器端发送的各种数据或者处理服务器发送的各种命令。该函数显示完数据后，就可以让用户和服务器进行交互的交流。要正确的显示并处理程序，就必须对这些数据的含义了解，否则就不能实现这个功能。其实对于客户端的 BBS 程序而言，需要编程的并不是很多，大部分的功能都是由服务器来提供的，而客户端所要完成的就是读懂服务器端的数据同时把客户端的输入提交到服务器。由于单纯地看上面的代码不容易理解，下面以实际的例子来说明。当连接到服务器以后，我们读取所有服务器端发来的数据并分析，然后显示在屏幕上。总结后面的数据为程序登录过程中捕获到的从服务器端发送的数据，用户可以对照该数据、图 6-4 和下面的数据以及 Main 函数的代码来学习，这样就容易许多了。

（13）Socket 对象的事件

```
'当连接关闭的时候，清屏，通常连接关闭可以是客户主动关闭也可以是服务器先关闭
Private Sub Winsock1_Close()
    frmBBS.Picture = LoadPicture("")
    frmBBS.Cls
    Timer1.Enabled = False
```

```
    Connected = False
End Sub

'当有数据到达时，说明连接已经建立，当然可以有其他方法来实现
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
Connected = True
End Sub

'出现错误，通常当不能连接服务器的时候，会激发该事件
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext
As Long, CancelDisplay As Boolean)
    frmBBS.Cls
    Timer1.Enabled = False
    frmBBS.Picture = LoadPicture("")
    MsgBox "无法连接服务器！"
End Sub
```

在连接服务器和断开服务器的时候，一般通过 Winsock 对象的事件来判断状态比较方便。因为通过 TCP 连接是安全的行为，建立连接的双方，如果有一方断开，都会引起对方的连接关闭。因此如果 Winsock 对象的 CLOSE 事件触发，可以表示连接断开，然后做一些处理如清屏或者提示同服务器连接已经断开。如果通过正常次序退出 BBS 系统，则首先是服务器端关闭连接，然后引起客户端关闭。当然客户端也可以强行断开连接，即调用 Winsock 对象的 CLOSE 方法就可以实现。另一方面，如果服务器没有开，或者输入的地址和端口无效，则会让 Winsock 无法连接到服务器。此时会产生连接错误事件，因此可以通过这个事件来判断是否能连上服务器，或者可以设置一个时间限制，如果超时，则表示无法连接服务器。

（14）自动登录

```
'该函数的功能时自动向服务器发送用户名和密码
Private Sub AutoLogin(LoginStr As String)
Dim I As Integer
Dim tempChar As String
Dim tempChar1 As String

LoginStr = Trim(LoginStr)
'首先自动登录字符串是否为空
If Len(LoginStr) <> 0 Then
    '当客户接收服务器数据完毕以后发送密码和用户名
    Do While Winsock1.BytesReceived <> 0
        DoEvents: DoEvents
```

```
    Loop
    '发送每一个字节
    For I = 1 To Len(LoginStr)
        '为了能让服务器响应，这里每发送一个字节可以暂停程序执行
        Sleep (100)
        '获得一个字符
        tempChar = Mid(LoginStr, I, 1)

        '分析字符是否为“\”，因为回车是以“\n”表示的
        If tempChar <> "\" Then
            '然后分析是否为“n”，如果为“n”，还要判断前一个字符是否为“\”
            If tempChar <> "n" Or (tempChar = "n" And _
                Mid(LoginStr, IIf((I - 1) > 0, I - 1, I), 1) <> "\") Then
                '发送该字符，注意要发送该字符的 ASCII 码
                SendChars (CStr(Asc(tempChar)))
            End If
        ElseIf Mid(LoginStr, I + 1, 1) = "n" Then
            '发送回车
            SendChars ("13")
        Else
            '发送字符“\”
            SendChars (CStr(Asc("\")))
        End If
    Next I
End If

End Sub
```

该自动登录采用的是当服务器发送完数据，并被处理完毕后，开始向服务器发送用户名和密码。实际上服务器可以保留客户发送的字节，即当用户发送用户名和密码的时候，还没有发送完的数据，但是，服务器会保留发送的数据，一旦发送完登录数据后，会自动处理该用户名和密码。这里采用的是连续发送用户名和密码，而不是每发送完一个字节然后等待回显，如果回显成功再发送第二个字节（Cterm 软件采用的就是这种方法）。但是该方法有一个不好的地方就是如果用户在自动登录的过程中，按下回车键，则会首先发送了回车键，这样可能会导致登录失败。而本程序采用的算法是每隔一个很短的时间发送一个字节登录，所以即使用户按下回车键，对登录并没有影响。

3. 总结

在这个例题中，说明了如何开发 BBS 客户端程序。该程序可以实现 BBS 客户端的基本操作，作为一个客户端程序是绰绰有余了。读者当然可以作许多的修改，而需要修改的地方主要是界面和操作的方便性，因为本书是介绍学习网络编程，这些设计并不是重点，因此有

兴趣的读者可以自己开发。还有一个值得改进的地方，就是在程序处理的时候，由于是每次获得一个字节然后来处理，这样极大的影响了显示速度，读者也可以在这方面加以改进。关于 Telnet 的协议有许多的内容，想深入了解的读者，可以自己查阅 RFC 文档，里面有非常详细的说明和解释。

第 7 章　HTTP 协议及高级编程

HTTP 协议是当前 Internet 上最为流行的网络协议我们平时上网冲浪总是习惯地在浏览器的地址栏中敲入“ http://…… ”的地址，但是许多人可能没有意识到是 HTTP 协议的出现使得整个网络变得活跃起来。Internet 的迅速发展与 HTTP 协议的应用以及基于 HTTP 协议的浏览器的出现是密切相关的。

在本章中，主要介绍 WWW（Word Wide Web）的核心协议——HTTP 协议的内容以及相应的一些高级开发实例。

本章的主要内容如下：

- HTTP协议。
- 断点续传下载高级开发。

网页服务器高级开发：

- 网站下载高级实例开发。
- HTTP API高级程序开发。
- HTTP代理服务器简单程序。

7.1　HTTP 协议介绍

HTTP 是 Hypertext Transfer Protocal（超文本传输协议）的简称，与 E-mail 的 SMTP 和 POP3 协议、FTP 协议等一样，HTTP 协议也是基于 TCP/IP 的客户机/服务器协议，用以支持客户机和服务器进行对话以及事务处理。我们所熟悉的 Internet Explorer 和 Netsacape Navigator 浏览器就是客户机，当我们在浏览器的地址栏中敲入“ http://…… ”再按下回车键时，作为客户机的浏览器向提供服务的远程的 Web Server 请求服务。对于它们之间的对话规则 HTTP 协议做了规定。开发基于 HTTP 的应用程序，首先应该了解 HTTP 协议的内容。

7.1.1　HTTP 背景

HTTP 最初的原型是在 1990 年出现的，刚开始只是在实验室应用，用于提供一种新型的信息组织方法，便于将信息组织成为 Web 文档，就是所谓的超链接文档。当这种方法被公开应用到 Internet 上之后，得到了广泛的应用，事实证明 HTTP 比以前的任何一种协议都能将信息很好地组织起来，让人能方便地、直接地从 Internet 上检索和获取所需的信息。

与其他流行的 Internet 协议一样，HTTP 协议的发展也经历了一个不断完善的，功能不断增加的过程。从 HTTP 出现到制定相应的 RFC，HTTP 协议的版本经历过了 HTTP 0.9，HTTP 1.0 以及 HTTP 1.1。

在介绍已经出现的几个 HTTP 版本时，读者可能对文中的一些术语和知识还不甚了解，

可以先暂时跳过这一小节接着往下阅读，后面的内容是结合几个实例详细讲解 HTTP 1.1 的具体内容和一些应用。在掌握了 HTTP 1.1 协议后再比较几个版本之间的区别。

1. HTTP 0.9

HTTP 0.9 是 HTTP 第一次出现时制定的原始协议它是目前使用的 HTTP 协议的子集，以后的 HTTP 都对其兼容。

HTTP 0.9 是一个面向消息的简单协议，该协议描述了 Client 和 Server 间请求和响应的过程：Client 在特定地址向 Server 请求连接（Connection），然后调用 GET 请求，访问 Server 端对象（一般为 HTML 文件），Server 在终止连接前将该对象（或一个出错消息）返回给 Client，结束响应过程。

2. HTTP 1.0

HTTP 1.0 以 HTTP 0.9 为基础，增加了在复杂的网络连接下访问不同类型的对象的功能，包括：

- 增加请求类型。如HEAD、POST请求。
- 请求和响应消息的协议版本。如响应消息第一行以HTTP/1.0开始，表示Server使用HTTP 1.0协议。
- Server响应码，表示请求、响应消息是否成功。如响应消息第一行以200 OK结束，表示请求成功。
- 用MIME（Multipurpose Internet Mail Extension）的消息标题字段（Header）和消息体（Body）格式来描述访问对象的数据类型和附加在后面的元信息。如MIME标题有Content-Type: text/html的字段，表示响应消息实体为HTML文件。
- 用询问/响应（Challenge/Response）实现访问认证。如在访问某些主页时要求Client用户输入用户名和口令。

在 HTTP 0.9 中，客户机和服务器间的相互作用只能直接进行，HTTP 1.0 对此进行了扩充，允许通过中间实体，如代理（Proxy）进行连接。

HTTP 1.0 用 MIME 描述对象的数据类型，通过有力的扩充方法，可以处理几乎所有的数据类型，既可以处理简单的纯文本 HTML，也可以处理更复杂的多媒体信息，如声音、图像和视频等。

3. HTTP 1.1

HTTP 1.1 是 HTTP 1.0 的一次飞跃，它主要强调解决后者的性能、安全、数据类型处理和缓冲等方面的缺陷。HTTP 1.1 定义非常严格，如缓冲和代理 Server（Proxy Server）的操作，误解的可能性减少了，协议实现更加可靠。

HTTP 1.1 主要改进包括：

- 使用永久连接（Persistent Connection）作为缺省连接，提高性能。
- 使用摘要认证（Digest Authentication），提高安全性，克服基本认证方法中“显式”传递用户名和密码的缺陷。
- 使用内容协商机制（Content Negotiation mechanism），允许客户机和服务器以最佳方式描述对象。

HTTP 1.1 提出在 Server 方缓冲对象，通过一种 Client/Server 协议操作缓冲对象进一步提高性能的思想，目的是减少请求、往返次数，并且是在确实需要时才返回完整的响应。

另外，HTTP 1.1 突破了 HTTP 1.0 中 Server 和 IP 一对一的限制，允许使用 Host 标题字段 Server 的名字来决定由哪个 Server 对请求进行服务，而不用 IP 地址来决定。

4. HTTPng

HTTPng 是下一代的 HTTP 协议，和 HTTP 1.x 完全不同，并且试图不继承这些协议。虽然 HTTP 1.1 和 HTTPng 由类似的特性：永久连接、请求/响应流水，但 HTTPng 在提高效率和性能上走得更远。

对 HTTPng 改进概括如下：

- 和 HTTP 1.1 相似，在 Client 和 Server 间处理请求的地方建立一条永久连接，不同的是每个 HTTPng 连接实际上被分割到多个不同通道（虚对话）上——一个控制通道（用于发送和接收命令）、每个请求对象各对应的一条通道。
- 异步请求和响应。Client 在等待以前的请求响应时，可启动另一个请求；Server 响应顺序不定，响应时间也不定。

5. HTTP 与 HTML

读者从 HTML 很容易联想到网页。不错，在使用浏览器上网冲浪时看到的多彩的网上信息背后都有 HTML 的支持。HTML（超文本 Markup 语言）是一种能够使文档显示在 Web 浏览器中的语言。使用 HTML 可以创建出能在浏览器中显示的 .htm 文件。HTML 是一种 HTTP Client 和 Server 都可以理解的一种数据格式。而 HTTP 协议是进行网络通信的 Internet 协议，它规定了 Client/Server 之间的信息传输和会话标准。除了 HTML 能通过 HTTP 传输并且被 Client 和 Server 所理解外，其他许多格式的数据，如图形、声音、PostScript 等也能通过 HTTP 传输。

下面的 HTML 程序控制了浏览器中的显示，其结果如图 7-1 所示。

```
    <html>
    <head>
    <STYLE type=text/css>
    .td1 {FONT-FAMILY: 宋体, Arial, Times New Roman; FONT-SIZE: 9pt; FONT-WEIGHT:
strong;COLOR: red}
    .td2 {FONT-FAMILY: 宋体, Arial, Times New Roman; FONT-SIZE: 9pt; FONT-WEIGHT:
normal}
    </STYLE></head>
    <body bgcolor=AntiqueWhite><p><font size=3 face=幼圆>BigFox Web Server V1.0
（大尾狐 Web 服务器）</font></p>
    <table border=0 width=500 cellPadding=1 cellSpacing=1>
    <tr   bgcolor=blue><td   height=5><td></tr><tr><td><div   align=center><b
class=td1>Welcome to BigFox Server</b></div></td></tr>
    <tr><td class=td2>目前提供的服务：</td></tr>
```

```
<tr><td class=td2>1.<a href=/upload.htm>软件/文件上载</a></td></tr>
<tr><td class=td2>2.<a href=/uploadfile/>软件/文件下载</a></td></tr>
<tr><td class=td2>3.<a href=/html/index.htm>网页浏览</a></td></tr>
<tr bgcolor=blue><td height=5><td></tr><tr><td class=td2><p></td></tr>
<tr><td class=td1>注：有时对中文的连接不太支持</td></tr>
<tr><td class=td1><div align=right class=td1>(不需要密码)管理员：busyzhong</div>
</td></tr></table></body></html>
```

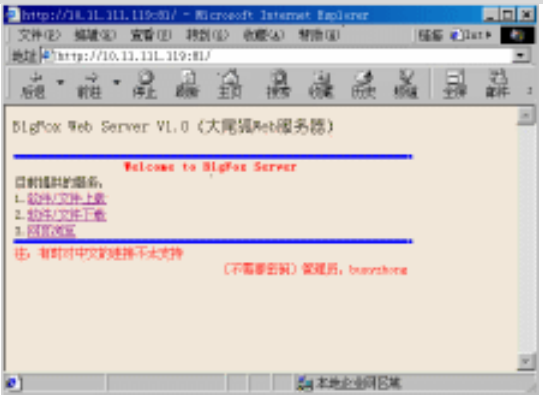


图 7-1　HTML 控制网页的显示

7.1.2　HTTP 的内容

最新的 HTTP 1.1 在 RFC 2068 有详细的描述，在这里我们不想逐个讲解其规范，只是介绍一些基本的概念和较常用的内容。

1. 从浏览器的使用和服务器的响应角度了解 HTTP

浏览器作为一个客户端程序（Client），每当我们在地址栏输入地址按下回车或者点击网页上相应的链接时，Client 向目标服务器（Web Server）发送请求，如果请求正确，Server 接到 Client 的请求之后，响应 Client 的请求并发送回来相应的消息。

对于服务器，要知道客户机（浏览器）请求的是什么服务，以什么样的数据格式返回。对于浏览器，则要知道怎么连接服务器，以什么样的格式发送命令，浏览器怎么返回并根据什么来判断返回的数据格式。浏览器的请求有可能是非法的或者是不合理的，服务器以什么样的形式告诉浏览器，这些问题都必须有一个事先协商好的规则来作为标准判断和辨别。与众多的 Internet 协议一样，HTTP 协议用于规定 Client/Server 之间通信请求、响应的问题。

2. HTTP 的请求周期

HTTP 的请求周期由 4 个阶段组成：连接（connection）、请求（request）、响应（response）和断开（disconnection）。

■　连接

HTTP 基本上是建立在 TCP/IP 协议之上的，以 TCP 作为传输协议，这就意味着我们在 VB 中可以通过 Winsock 控件进行连接通信。HTTP 的客户端（如 Web 浏览器）在地址栏中给定了一个地址和端口（缺省时端口是 80）和目标资源的 Server 进行 TCP/IP 连接。

■ 请求

Server 侦听并接收到连接后，客户端发出请求消息，消息中含有资源在 Server 上的位置。

■ 响应

Server 响应 Client 的请求，返回状态码，表示请求是否完成，并在消息标题中进一步描述响应和请求的对象（一般为 HTML 文件）。对象本身作为一个实体，由两部分组成：描述对象的实体标题和对象数据的实体体。

■ 断开

一旦响应消息发出，Server 将关闭 TCP/IP 会话，完成事务处理全过程。HTTP 的一个重要特点就是每个请求和其他请求是独立的。

7.1.3 消息（Message）

发送和接收消息是 HTTP 协议的核心，它有两类消息：请求消息和响应消息。请求消息是由客户端（如浏览器）向服务器发送的，用于请求服务器提供某个类型服务的。响应消息是服务器接到请求消息之后，返回给客户端的信息，表明服务器所作出的动作即响应。总而言之，客户端与服务器之间的信息传递是通过消息来进行的。

本节着重讲解消息的一般性的知识，由于消息都用于请求和响应中，具体的消息及含义在讲解请求和响应时再作详细说明。

消息通常可分为两个部分：消息头和消息体。

下面是一个完整的请求标题的消息，在这个消息中不含消息体。

```
GET /BestESong.htm HTTP/1.1
Host: 10.11.111.119
Accept: */*
Referer: http://10.11.111.119/Head.htm
User-Agent: Downjet/1.0
Connection: close
```

从以上可以看出，消息头一般由消息的标题字段和字段值构成，它们之间使用冒号“：”隔开，如：

```
Host: 10.11.111.119
```

一个消息头可能包含着多个标题字段和字段值。

为什么要有标题字段呢？很显然，这是在客户端与服务器以此种方式相互通告对方一些信息。上面的请求的消息我们暂且不作讲解，在后面的讲解客户端请求时将有详细的介绍。

消息标题的分类可以分为几种：普通标题、请求或响应标题、实体标题。标题的名字不区分大小写。

请求和响应的消息都可以使用普通标题。普通标题的信息与消息本身相关，而与消息内的实体无关。通常使用的普通标题有两种：时间和附注标题，如：

```
Date: 25 Dec 2000 15:02:24 GMT
Pragma: no-cache
```

时间标题记录 Server 或 Client 创建消息的时间。响应消息总是需要时间标题的，而请求消息只有在消息中含有一个实体时才需要，如 POST 请求向服务器提交数据时。

附注标题（pragma）的值只能为 no-cache，表示客户端和服务器端的数据请求/响应不使用缓冲区的数据。

7.1.4　请求（Request）

请求是由客户端向服务器发送消息来实现的，通常要求服务器提供一定的服务。在 HTTP 协议中，最常用的请求就是要求服务器返回一个文件的内容。

请求分为两个部分：请求的方法和请求消息。

1. 请求方法（request method）

请求的方法常用的有两种：GET 和 POST。

■ GET

GET 方法通常只是用于请求指定服务器上的资源。这种资源可以是静态的 HTML 页面或其他文件，也可以是由 CGI 程序生成的结果的数据。如：

```
GET /BestESong.htm HTTP/1.1
Host: 10.11.111.119
Accept: */*
Referer: http://10.11.111.119/Head.htm
User-Agent: Downjet/1.0
Connection: close
```

该请求使用了 GET 方式要求服务器 10.11.111.119 返回相对 URL 为/BestESong.htm 的文件。这种方式不仅仅能够返回 HTML 文件的内容，而且能够请求返回任何类型的文件类型，如下载 ZIP 类型的文件。

```
GET /resume.zip HTTP/1.1
Host: 10.11.111.119
Accept: */*
Referer: http://10.11.111.119/Head.htm
User-Agent: Downjet/1.0
Connection: close
```

值得注意的是，使用 GET 方法请求文件也可以向服务器传送一些数据。如：

```
GET /song.asp?starid=14&type_id=25 HTTP/1.1
Host: 10.11.22.133
Accept: image/jpg
Referer: http://10.11.22.133/song.html
User-Agent: Mozilla/4.0
Connection: close
```

GET 方法将要上传的数据附在 URL 之后，以问号“？”隔开，数据以 name=value 对偶的形式出现，如果有多个数据对，使用符号‘ & ’隔开。在上面的例子中请求 ASP 文件 song.asp，传递数据给服务器，作为执行 ASP 文件所需的参数。ASP 根据提供的参数值执行得到的结果，由服务器返回给客户端。

这种方法在下载 Web 邮箱中的附件时很有用。不知道读者们有没有因为网络的速度慢而使用 Web 浏览器打不开 Web 邮箱上的附件，如果观察仔细，发现 E-mail 信箱中的附件都提供了一个链接如：

```
    http://202.96.44.25/cgi/ldmsapp?funcid=readp&websid=FALfJkui&mid=SM18XV&p
art=2
```

所有的附件都是请求 http://202.96.44.25/cgi/ldmsapp 文件，只是对于不同的附件后面提供的参数值不一样。可以使用 GET 方式将请求的字符串发送给服务器，服务器将会根据上传的数据返回指定的附件内容。如果一次请求不能接收完成，可以通过断点续传再次下载剩余的数据。

由于 GET 方式是将数据附在 URL 后作为 URL 的一部分传递的，用此受 URL 的最大长度为 1024 Byte 的限制，GET 方式上传的数据不能太大，如果要向服务器上载比较多的数据比如一个文件，那么就要使用 POST 方式了。

■ POST

POST 方式提交的请求消息中通常含有消息体，通过消息体将数据传递给服务器。一般是传递用户输入的数据。

如下面的 HTML 页面。

```
    <html><head><title>贵宾留言簿</title></head>
    <body><h3>留言簿</h3>
    <form action="guestbook.exe" method="post" id=form1 name=form1>
    <TABLE border=0 cellPadding=1 cellSpacing=1 width=75%>
    <TR>
                  <TD width=150>您的姓名</TD>
                  <TD> <input name="name" ></TD></TR>
    <TR>
                  <TD>您的 Email 信箱</TD>
                  <TD> <input name="email" ></TD></TR>
    <TR>
                  <TD>您的主页的 URL</TD>
                  <TD> <input name="URL" ></TD></TR>
    <TR>
                  <TD colSpan=2>您的建议</TD></TR>
    <TR>
                  <TD colSpan=2>
                  <TEXTAREA cols=30 name=URL_Comment rows=4 style="HEIGHT: 121px;
WIDTH: 313px">
                  </TEXTAREA></TD></TR>
    <TR>
                  <TD><input type="submit" value=" 留言 "> 
                  <input type="reset" name="Reset" value="重填" style="HEIGHT:
24px; WIDTH: 53px">
```

```
                </TD><TD><A href="guestbook.exe">所有留言</A></TD></TR>
    </TABLE>
    </form></body></html>
```

该 HTML 文件在 IE 浏览器的显示如图 7-2 所示。

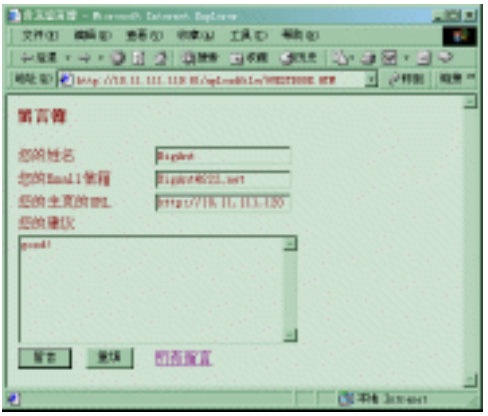


图 7-2　在 HTML 表单中填写数据

在表单中填写了相应的内容之后，单击“留言”提交按钮，浏览器连接服务器，向服务器发送请求消息，同时在请求消息中加入了填写的数据，如下所示：

```
    POST /uploadfile/guestbook.exe HTTP/1.1
    Accept:     image/gif,     image/x-xbitmap,     image/jpeg,     image/pjpeg,
application/vnd.ms-excel,
    application/msword, */*
    Referer: http://10.11.111.119:81/uploadfile/GUESTBOOK.HTM
    Accept-Language: zh-cn
    Content-Type: application/x-www-form-urlencoded
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
    Host: 10.11.111.119:81
    Content-Length: 83
    Connection: Keep-Alive
    name=BigAnt&email=BigAnt@222.net&URL=http%3A%2F%2F10.11.111.120&URL_Comme
nt=good%21
```

POST 方式提交的数据是通过消息体的形式进行传送的，能够提交的数据可以说不受 URL 长度的限制。

2. 请求消息

（1）构造最小的请求消息

RFC 2068 中规定的最小 HTTP 1.1 请求必须由两行文本组成，请求行和 HOST 标题字段。如：

```
GET /index.htm HTTP/1.1
HOST: 10.11.111.119
```

请求行的一般格式为：

请求方法（GET|POST|HEAD 等）相对 URL|绝对 URL HTTP 版本号。

HOST 标题字段的值指明请求资源的 HOST 地址。其实 HOST 的出现有点奇怪，按理说在使用 TCP 连接远程 Web 服务器时就已经确定了主机的地址，这里为什么还需要一个 HOST 字段表明请求资源所属的主机地址呢？其实这一点并不奇怪，在 HTTP 1.1 中允许连接的主机地址与资源所在的主机地址不相同。例如连接的主机地址为 10.11.111.119，但是请求的资源可以放在其他主机上如 10.11.111.200。这时 HOST 标题字段就很有必要了。不过在大多数情况下，TCP 连接的服务器主机地址与资源所在的主机地址是一致的。如果指定了 HTTP 的版本为 1.1，该标题字段是必须的，否则将会返回 400 响应码。若发送的请求消息中没有 HOST 标题字段，如：

```
GET /index.htm HTTP/1.1
```

则响应的第一行为：

```
HTTP/1.1 400 Bad Request
```

客户端发送不合法的请求消息时，如果把 HTTP 版本定义为 1.0，则不会出现 400 响应，请读者自己试验。

（2）其他请求标题字段的使用

除了 HOST 标题字段之外，请求消息还可以发送其他标题字段，用于加入描述请求的附加信息和客户端的信息。

RFC 2068 规定的请求消息的标题字段如表 7-1 所示。

表 7-1　　请求的标题字段

字段名	含义
Accept	客户端期望接受的媒体类型
Accept-Charset	客户端期望接受数据所用的字符集
Accept-Encoding	客户端期望接受数据的编码方式
Authorization	用于向服务器提供身份验证的字段
From	说明客户端提供的邮件源地址
Host	表明请求资源所在的主机地址
If-Modified-Since	访问某个时间之后修改过的对象
If-Match	访问的对象需要符合的条件
If-None-Match	访问的对象需要不符合的条件
If-Range	请求对象指定范围的数据是否存在
If-Unmodified-Since	访问的对象在指定时间后没有被修改过
Proxy-Authorization	用于向代理服务器发送认证的字段
Range	用于要求服务器返回部分数据的字段
Referer	用于记录客户端获得 URL 资源的地址
User-Agent	用于标明客户端身份的字段

表 7-1 只是简单地介绍了请求消息中的标题字段的意思，标题字段的运用是比较复杂的，请读者自行参考 RFC 2068 有关内容。在请求消息中巧妙地应用标题字段，能够实现不少意想不到的功能。下面就介绍其中的一些标题字段在 HTTP 请求时的应用。

7.1.5　响应（Response）

响应是服务器对于客户端请求返回的结果。在 HTTP 协议中，服务器的响应也是以消息的形式出现的。下面先看看一段 Web 服务器的响应数据：

```
HTTP/1.1 200 OK
Date: Wed, 22 Nov 2000 02:44:34 GMT
Server: Apache/1.3.9 (Unix)
Last-Modified: Tue, 18 Apr 2000 01:45:00 GMT
ETag: "3e300-79-38fbbe1c"
Accept-Ranges: bytes
Content-Length: 121
Connection: close
Content-Type: image/gif
(请求的 GIF 文件的二进制数据)
```

与请求消息类似，响应消息由两大部分组成：消息头和消息体，两者之间使用一个空白行分开。划分得细一点则由响应的状态行、响应的标题字段和响应的实体数据组成。

1.　状态行

响应的状态行处于响应消息的第一行，由 HTTP 版本号、响应码和响应描述文本构成，中间使用空格相隔。

如 HTTP/1.1 200 OK

响应码由三个数字构成，是响应中最重要的部分，客户端主要是从服务器返回的响应码中了解对客户端所作的响应意思。首先看响应码的分类，如表 7-2 所示。

表 7-2　　响应码的分类

响应码	类别	含义
1××	信息	仅用于试验目的
2××	成功	请求已经被成功的接收、理解和接受
3××	重定向	客户端需要根据返回的信息做进一步请求
4××	客户端出错	客户端的请求不能被理解或满足
5××	服务器出错	服务器不能满足或完成客户端的合法请求

由于响应码比较多，表 7-3 只简要解释常见的响应码。如果读者要了解所有的响应码及具体意思，请参考 RFC2068。

表 7-3　　常见的响应码

响应码	含义
200	请求成功，发出正确响应
201	服务器（POST）创建了一个新资源
202	服务器收到请求，但尚未处理完
204	请求成功，但无数据返回
300	请求结果有多个，返回列表供客户端选择
301	请求资源已移到新的永久 URL 上
302	请求资源暂时移到另一个 URL 上
304	请求资源未更新

续表

响应码	含义
400	错误的请求
401	请求未认证（需认证）
403	服务器接受请求，但拒绝访问资源（Client 或 Server 无权访问该资源）
404	未找到请求资源
405	不允许的请求方法
500	服务器出错
501	服务器尚未实现请求的方法
502	错误的网关
503	服务器太忙

HTTP 响应码是可以扩展的，HTTP 应用程序不必要理解并对所有的响应码进行处理，通常只是对认为有必要的响应码进行处理。如果程序想设计得完善一些，最好能够高度识别响应码类别并处理响应码，可简单地认为类似 xxx 的响应码等价于 x00 的响应码意思，例如服务器返回 431 响应码（扩展的响应码），但是在 RFC 2068 中没有相应的解释，可简单地认为其等价于 400 响应码，即服务器判断是客户端（应用程序）的请求出错了。

2. 标题字段

响应消息中标题字段可以分成两类：响应标题和实体标题。

（1）响应标题

利用响应标题服务器可以给客户端返回除响应行之外的其他信息。

在 HTTP 1.1 中定义了 9 种响应标题：Age、Location、Proxy-Authenticate、Public、Retry-After、Server、Vary、Warning 和 WWW-Authenticate。下面介绍常用的几个标题字段的意思。

■ Locaiton 标题

该响应标题表示请求 URL 的实际位置，使用的是绝对的 URL。通常用于 3××的响应之中，如下面的 302 响应的消息头，表明请求的资源已经被移到另一个 URL 了：

```
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/4.0
Date: Mon, 27 Nov 2000 07:33:20 GMT
Connection: close
Location: http://202.109.73.177/~netboy/good_night/daniel05.mp3
Content-Type: text/html
Cache-control: private
Transfer-Encoding: chunked
```

■ Server 标题

该字段用于指定服务器使用的软件名称及版本信息。如：

```
Server: Microsoft-IIS/4.0
Server: Apache/1.3.14 (Unix) PHP/4.0.3
```

■ WWW-Authenticate 标题

当客户端请求受限资源时，服务器要求客户端进行身份认证，响应的消息中含有该标题，用于表明资源所在的区域。该标题必须包含在服务器的 401 响应消息中，并至少含有一个指示请求 URL 可用的认证方法。下面的标题字段及内容表示采用基本认证方法，受保护的资源所在的区域名称为“ Busyzhong’s BigFox Server1.0 ”。

```
WWW-authenticate: basic realm="Busyzhong’s BigFox Server1.0"
```

（2）实体标题

实体标题用于描述消息中所含有的实体数据的信息。在响应消息中，服务器一般都要返回一定的实体数据以响应客户端的请求。值得注意的是，如果在请求消息中有实体数据提交给服务器（如使用 POST 方法提交数据），下面介绍的这些用于描述实体信息的标题字段也是适用的。

在 HTTP 1.1 中定义了 12 个实体标题：Allow、Content-Type、Content-Length、Content-Encoding、Expires、Last-Modified、Content-Base、Content-Language、Content-Location、Content-MD5、Content-Range 和 Etag。下面就介绍常见的几个实体标题。

■ Allow

在请求的 URL 中，用 Allow 标题指示客户端对资源可以使用的方法，该标题字段的形式如下：

```
Allow: GET, HEAD
```

这个字段一定要与 405 响应码一起出现，用于提示客户端请求的资源所允许的请求方法。

■ Content-Type

这个字段用于描述消息内所包含的实体的数据类型。一般 Web 浏览器检查该标题，然后调用相应的应用程序查看实体。如：

```
Content-Type: text/html; charset=US-ASCII
```

上面的 Content-Type 字段指明实体是文本类型的 HTML 格式的数据，使用的字符集为 US-ASCII。

关于媒体类型，详见介绍 E-mail 协议的一章有关 MIME 的内容。

■ Content-Length

Content-Length 标题与 Content-Type 一起使用，用于描述消息内任何类型的实体大小（字节数）。例如下面的响应消息头：

```
HTTP/1.1 200 OK
Date: Wed, 22 Nov 2000 02:44:34 GMT
Server: Apache/1.3.9 (Unix)
Last-Modified: Tue, 18 Apr 2000 01:45:00 GMT
ETag: "3e300-79-38fbbe1c"
Accept-Ranges: bytes
Content-Length: 121
Connection: close
Content-Type: image/gif
```

表明服务器的响应消息中的实体数据为 image/gif 类型，实体长度为 121 个字节。

■ Expires

该标题影响资源的缓冲方式，为资源指定一个过期日期和时间。这对于那些规则的更新、产生数据的资源，如 CGI 过程，以及那些在某个时间后消失的资源很有用。这些资源在某个时间后不应该缓冲。类似的，如果资源的丢弃日期和 Date 标题的时间相同或还要早，则资源不必缓冲。

■ Last-Modified

该标题说明资源最近修改的时间，计算时间的方法依资源的类型而定。例如资源为一个文件，Last-Modified 应该是该文件最近修改的时间；客户端资源副本比 Last-Modified 指定的时间要早，则客户端放弃本地副本，请求更新；指示的时间比响应消息的创建时间（Date 标题指定的时间）要晚，则 Last-Modified 时间被设定为创建消息的时间。

3. 实体数据

实体数据是在消息数据的后部分，用一个空白行与消息头分隔开。所以在编写程序时，可以判断连续有两个 Crlf 的位置之后，就是实体数据开始的地方。这也说明消息头的标题字段间不能存在空白行，以免引起客户端程序的误解。

7.1.6 访问认证

HTTP 认证方法可以分成两种：基本认证和摘要认证。基本认证应用比较广泛，是比较常见的认证方法。下面只是介绍这种认证方法的内容和过程。

下面就一个具体的 IE 浏览器与 Web Server 认证过程描述其原理。

（1）客户端请求访问保护资源

首先在浏览器的地址栏或显示页面的链接中请求访问受保护资源：http://10.11.111.119:81/up.jpg。浏览器在连接主机 10.11.111.119:81 后发送的请求消息如下：

```
    GET /up.jpg HTTP/1.1
    Accept:     image/gif,     image/x-xbitmap,     image/jpeg,     image/pjpeg,
application/vnd.ms-excel, application/msword, */*
    Referer: http://10.11.111.119:81/
    Accept-Language: zh-cn
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
    Host: 10.11.111.119:81
    Connection: Keep-Alive
```

（2）服务器在接到请求之后，向客户端发送 401 响应消息通知客户端该请求为未认证请求

服务器在接到客户端对受保护资源的请求后，检查到客户端中没有进行身份验证的标题字段 Authorization 及其内容，于是向客户端发送带有 WWW-Authenticate 的字段 401 响应消

息，通知客户端发送身份验证。WWW-Authenticate 的格式及意思如下：

```
WWW-Authenticate: basic realm="Busyzhong’s BigFox Server1.0"
```

Basic 说明需要的是基本认证，realm 后面的字符串表示受保护的资源名称。

如下面的消息：

```
HTTP/1.1 401 Unauthorized
Server: BigFox Server1.0
Connection: close
Date: Mon, 1999-12-27 16:08:11 12T
WWW-authenticate: basic realm="Busyzhong’s BigFox Server1.0"
Content-Type: text/html
Content-Length: 782
<html><head><STYLE type=text/css>
.td1 {FONT-FAMILY: 宋体, Arial, Times New Roman; FONT-SIZE: 9pt; FONT-WEIGHT:
strong;COLOR: red}
.td2 {FONT-FAMILY: 宋体, Arial, Times New Roman; FONT-SIZE: 9pt; FONT-WEIGHT:
normal}
.A:link {FONT-SIZE: 9pt; TEXT-DECORATION: none}
.A:visited {COLOR: #0000ff; FONT-SIZE: 9pt; TEXT-DECORATION: none}
.A:active {COLOR: #0000ff; FONT-SIZE: 9pt; TEXT-DECORATION: none}
.A:hover {COLOR: red; TEXT-DECORATION: none}
</STYLE></head><body bgcolor=AntiqueWhite>
<p><font size=3 face=幼圆>BigFox Web Server V1.0（大尾狐 Web 服务器）管理者：
busyzhong</font></p><table border=0 width=500 cellPadding=0.5 cellSpacing=0>
<tr bgcolor=blue><td height=5><td></tr>
<tr><td class=td1>需要用户输入密码进行认证</td></tr>
</table>
</body></html>
```

消息中往往带有媒体类型为 TEXT/HTML 的实体数据，用作没有通过验证的提示。

（3）客户端接到 401 响应后，发送认证的信任状

客户端接到 401 响应后，发送认证的信任状，如果是使用 IE 浏览器，则会弹出如图 7-3 所示的请求输入认证用户名和密码的对话框。

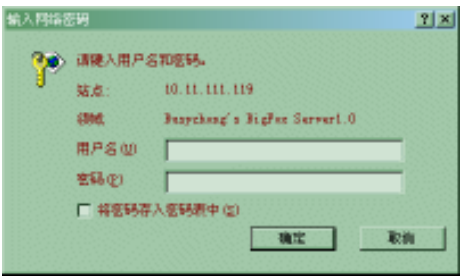


图 7-3　IE 用于输入身份验证的对话框

在输入用户名和密码后，按下“确定”按钮，浏览器将会发送带有身份认证的标题字段的消息给服务器，如果在该对话框中填写用户名为 zj，密码为 1，则浏览器发送的请求如下所示：

```
GET /up.gif HTTP/1.1
Accept: */*
Referer: http://10.11.111.119:81/uploadfile/
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 10.11.111.119:81
Connection: Keep-Alive
Authorization: Basic emo6MQ==
```

从上面的 Authorization 的字段内容得知，该字段开始首先用字符串 basic 表示是基本认证，然后 basic 之后的内容“emo6MQ==”为用于认证的信息，值得肯定的是，这个字符串包含了输入的用户名为 zj、密码为 1 的信息。

在基本认证中，认证的用户名和密码是经过 base64 编码才加入到 Authorization 字段中的。如上面所说的情况就是将用户名和密码组成的字符串“zj:1”（用户名和密码之间使用冒号“:”隔开）经过 base64 编码而形成编码后字符串“emo6MQ==”。

关于 base64 编码解码，在 E-mail 协议一章有详细论述，是属于 MIME 内容，在这里就不再重复了。

（4）服务器再次响应

服务器接到有认证信任状的请求后，判断身份是否正确，如果正确，发送 200 响应码及请求资源的消息。服务器的响应如下：

```
HTTP/1.1 200 OK
Server: BigFox Server1.0
Connection: close
Date: Mon, 1999-12-27 16:11:36 12T
Content-Type: image/gif
Accept-Ranges: bytes
Last-Modified: Tue, 2000-04-25 13:26:40
Content-Length: 878
……
```

值得注意的是，如果使用 IE 等浏览器，在提示输入了用户名和密码通过了服务器的认证之后，发送的每个请求都包含了相同的 Authorization 字段用于身份验证，直到打开使用一个新浏览器窗口为止。

7.1.7 URL 编码

由于 Web 服务器和浏览器不能正确处理一些特殊的字符，Web 服务器和浏览器之间可

能会因此而产生某种程度的误会，因此在数据被传送之前，浏览器都要对表单内客户输入的数据中的特殊字符进行 URL 编码。

例如，Web 系统用“ = ”分解表单各元素的 NAME 和 VALUE 属性，用“ & ”分解不同表单元素的输入数据。如果在表单的输入数据中包含这些特殊的字符，并且表单的数据在传送给 Web 服务器前不作任何处理，则 Web 服务器将无法知道哪一个“ = ”、“ & ”是用户输入的，哪一个是浏览器加上的。在由表单属性 ACTION 定义的 URL 中，也可能会出现一些特殊的字符，这也会影响数据的正确传送。

URL 编码（URL Encoding）就是将 Web 服务器所不能正确处理的特殊字符转换成它的十六进制数的形式，比如将“ % ”转换成“ %25 ”、“ = ”转换成“ %3D ”等等。这些特殊的字符通常被称作 Web 系统的保留字符。在 Web 系统上无论是用 GET 方法还是用 POST 方法传送的数据都要进行 URL 编码。CGI 程序要想处理表单传送来的数据，还必须对浏览器 URL 编码过的数据进行解码。因此，理解 URL 编码对于我们进行 CGI 编程是非常重要的。URL 编码一般包括以下步骤：

（1）浏览器将所传送的数据根据表单所包含的元素分解成“ NAME=VALUE ”形式，NAME 和 VALUE 分别是表单元素的属性。其中，VALUE 属性中存储客户机在表单中输入的数据，如果客户机没有输入数据，则 VALUE 存储的是表单定义的缺省值；如果缺省值也没有定义，则 VALUE 值为空。

（2）代表表单中各元素的各个“ NAME=VALUE ”对被浏览器用“ & ”连接起来。

（3）VALUE 属性中存放的数据若含有空格，则被转换成“ + ”。

（4）URL 和输入数据中所包含的 Web 系统的保留字符必须被编码成其相应的十六进制数形式。

（5）被编码后的字符被表示成一个“ % ”和它们的十六进制数形式（即%HH）。

CGI 程序从环境变量“ QUERY_STRING ”或标准输入中读入的数据是经过浏览器 URL 编码过的，故在使用这些数据以前还必须对它们进行 URL 解码。解码的目的是将数据还原成客户端用户在 Web 页面上输入时的形式。前面已经介绍了 URL 编码过程，URL 解码过程与它正好相反，它一般包括以下步骤：

（1）从浏览器用 GET 或 POST 方法所传送来的数据中找出代表各个表单元素所储存数据的“ NAME=VALUE ”对。

（2）VALUE 属性中所存放的数据若含有“ + ”，则被转换成空格。

（3）将 VALUE 属性中所存放的数据的十六进制数“ %HH ”转换成相应的字符。

Web 系统将汉字当成特殊的字符，对它也要进行 URL 编码。对于一个特殊的单字节字符（比如“ / ”），浏览器通常将它编码成十六进制数的形式（比如%2F），“ % ”表示它后面跟的是两位十六进制数。

7.1.8　HTTP 协议的应用

HTTP 协议对编程有什么用？能够实现什么样实用的程序？这些问题使我们不得不回到 HTTP 的本质——超文本传输协议，主要是用于传输文件的协议。虽然 RFC2068 对 HTTP 协议描述早就超出了文件传输的范围。但是传输文件的作用还是最主要的。在这里我们提出几个问题，这些问题都可以使用 HTTP 协议编程实现。比如

- 基于 HTTP 的文件断点续传的程序
- 使用代理服务器下载的程序
- Web 服务器程序
- 能够通过身份认证而下载文件的程序
- 接受浏览器网页上载文件的程序

这些功能的实现都要求用户对 HTTP 比较了解，下面的几个结合 HTTP 协议的 VB 实例将让读者领略 HTTP 能够实现的一些有趣但很实用的功能。

7.2 断点续传

7.2.1 建立工程项目

大多数上过网的读者都使用过网络蚂蚁（NetAnts）或网际快车（FlashGet）下载软件，在这一节里我们要介绍一个类似于网络蚂蚁、网际快车的下载工具 DownJet（下载引擎），但其功能要简单一点。

- 能够下载基于 HTTP 协议传输的文件。
- 支持断点续传。
- 能同时启动两个文件进行下载。
- 支持通过代理服务器进行下载。

在这个工程里，使用了许多 VB 中的控件如 ListView、SSTab 等，如果读者对这些比较基本的控件还不太会使用，建议先看看 MSDN 的帮助和例子，在后面的解释中假定读者都能熟练地掌握了这些基本控件的使用。

该工程一共由 3 个窗体、一个类模块和一个模块组成：

- 窗体 frmDown
- 窗体 frmAdd
- 窗体 MinForm
- 类模块 clsDown
- 模块 ModDown

下面的代码分析中并不是严格地按照各个窗体和模块的顺序依次独立的讲解，而是从工程的整体结构和功能实现上讲解关键性的代码。图 7-4 是程序运行时的界面。

图 7-4　程序运行界面

图 7-5 是添加新的下载任务显示的界面。

图 7-5　加入下载任务界面

7.2.2　代码分析

在程序中，使用 Winsock 控件实现 HTTP 的 TCP/IP 连接和交换数据，由于前面已经有章节使用了 Winsock 控件进行基于 Client/Server 之间的程序的开发和介绍，这里对于 Winsock 控件的具体使用就不再多说了。

1. 模块 ModDown

该模块主要声明一些要使用的 API 函数和定义保存信息的变量类型。

```
    Public Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal
hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal
cy As Long, ByVal wFlags As Long) As Long
```

以上定义的是该工程项目唯一的 API 函数，作用是使窗体 minForm 在所有窗口之上，便于激活整个程序到前台。在 NetAnts 和 FlashGet 中也是采用这样的形式。

```
'定义保存下载任务信息的类型
Public Type DownInfo
    mIndex As Integer        '位置索引
    mUrl As String           '地址
    mFile As String          '文件名
    mSize As Long            '文件的大小
    mGetSize As Long         '已获得的字节数
    mUseProxy As Boolean     '是否使用 proxy
    mProxy As String         '使用的代理服务器地址
    mProxyPort As Integer    '代理服务器端口
    mProxyId As String       '代理服务器的用户名
    mProxyPass As String     '代理服务器的密码
End Type

'用于把下载任务信息读写到随机文件的类型
Public Type DownInfoSave
    mIndex As Integer        '位置索引
    mUrl As String * 180     '地址
    mFile As String * 50     '文件名
    mSize As Long            '文件的大小
    mGetSize As Long         '已获得的字节数
    mUseProxy As Boolean     '是否使用 proxy
    mProxy As String * 50    '使用的代理服务器地址
    mProxyPort As Integer    '代理服务器端口
    mProxyId As String * 20  '代理服务器的用户名
    mProxyPass As String * 20    '代理服务器的密码
End Type
'记录每个下载任务信息的变量数组
Public mDownInfo() As DownInfo
```

以上首先定义了两个自定义的变量类型：DownInfo 和 DownInfoSave，DownInfo 类型是用于程序运行时保存下载任务的一些信息的，如下载 URL、保存文件的位置、文件长度、使用代理服务器等。DownInfoSave 只是在程序运行时能够将保存在文件中的下载任务信息读取出来，在程序运行结束时保存所有的下载任务信息。这两个变量类型里变量都是一样的，唯一不同的是，在 DownInfo 中，String 类型的变量都是不定长的，而在 DownInfoSave 中是定长的。这是为了方便使用随机型文件以记录的形式保存，而 String 类型的数据如果不是以 String*N 的形式声明，可能在保存时每个记录写入的长度不一样，下次读出时数据会无法准确得到。由于下载任务可能不止一个，因此还声明了一个变量数组 mDownInfo。

```
'当前正在下载的任务的索引
Public CurrentDown(3) As Integer
'当前在 ListView 中选择的任务的索引
```

```
    Public SelectDown As Integer
```

以上是为了方便程序设计而定义的变量，从索引中可以知道哪个任务正在下载。

```
    'base64 编码函数：Base64Encode
    'Instr1     编码前字符串
    'Outstr1    编码后字符串
    Public Function Base64Encode(InStr1 As String, OutStr1 As String)
    Dim mInByte(3) As Byte, mOutByte(4) As Byte
    Dim myByte As Byte
    Dim i As Integer, LenArray As Integer, j As Integer
    Dim myBArray() As Byte
    myBArray() = StrConv(InStr1, vbFromUnicode)
    LenArray = UBound(myBArray) + 1
    For i = 0 To LenArray Step 3
        If LenArray - i = 0 Then
            Exit For
        End If
        If LenArray - i = 2 Then
            mInByte(0) = myBArray(i)
            mInByte(1) = myBArray(i + 1)
            Base64EncodeByte mInByte, mOutByte, 2
        ElseIf LenArray - i = 1 Then
            mInByte(0) = myBArray(i)
            Base64EncodeByte mInByte, mOutByte, 1
        Else
            mInByte(0) = myBArray(i)
            mInByte(1) = myBArray(i + 1)
            mInByte(2) = myBArray(i + 2)
            Base64EncodeByte mInByte, mOutByte, 3
        End If
        For j = 0 To 3
            OutStr1 = OutStr1 & Chr(mOutByte(j))
        Next j
    Next i
    End Function

    Private Sub Base64EncodeByte(mInByte() As Byte, mOutByte() As Byte, Num As
Integer)
    Dim tByte As Byte
    Dim i As Integer
    If Num = 1 Then
```

```
        mInByte(1) = 0
        mInByte(2) = 0
    ElseIf Num = 2 Then
        mInByte(2) = 0
    End If
    tByte = mInByte(0) And &HFC
    mOutByte(0) = tByte / 4
    tByte = ((mInByte(0) And &H3) * 16) + (mInByte(1) And &HF0) / 16
    mOutByte(1) = tByte
    tByte = ((mInByte(1) And &HF) * 4) + ((mInByte(2) And &HC0) / 64)
    mOutByte(2) = tByte
    tByte = (mInByte(2) And &H3F)
    mOutByte(3) = tByte
    For i = 0 To 3
        If mOutByte(i) >= 0 And mOutByte(i) <= 25 Then
            mOutByte(i) = mOutByte(i) + Asc("A")
        ElseIf mOutByte(i) >= 26 And mOutByte(i) <= 51 Then
            mOutByte(i) = mOutByte(i) - 26 + Asc("a")
        ElseIf mOutByte(i) >= 52 And mOutByte(i) <= 61 Then
            mOutByte(i) = mOutByte(i) - 52 + Asc("0")
        ElseIf mOutByte(i) = 62 Then
            mOutByte(i) = Asc("+")
        Else
            mOutByte(i) = Asc("/")
        End If
    Next i
    If Num = 1 Then
        mOutByte(2) = Asc("=")
        mOutByte(3) = Asc("=")
    ElseIf Num = 2 Then
        mOutByte(3) = Asc("=")
    End If
    End Sub

    '编码函数：EncodeStr
    'Str1   编码前字符串
    '返回值 编码后字符串
    Public Function EncodeStr(Str1 As String) As String
        Dim OutStr1 As String
        Call Base64Encode(Str1, OutStr1)
```

```
    EncodeStr = OutStr1
End Function
```

以上是 base64 编码函数，在向服务器（特别是代理服务器，一般都要使用账号和密码进行身份认证）发送的认证账号和密码要经过 base64 编码，关于 base64 编码的知识，在介绍 E-mail 协议的一章有详细的算法介绍和程序实现，不过哪一章的程序函数的编码输入和输出都是针对文件的，这里给出输入的是未编码字符串和输出的是编码后的字符串的函数，以方便该工程的使用。

2. 窗体 frmDown

在程序运行时的界面 frmDown 如图 7-6 所示。主要由一个 ListView 管理下载任务的显示，Sstab 中的“信息提示”用文本框显示下载时的 Winsock 通信和其他信息提示，“下载情况”中使用了 3 个 PictureBox，用于显示下载任务的下载情况的显示，类似于 Netants 中用小圆点直观地表示文件大小和已下载的数量。其中 Index 为 0 的图片框用于显示选中的没有正在下载的任务的文件情况，Index 为 1 和 2 图片框用于显示正在下载的任务的文件大小情况，如图 7-4 所示。

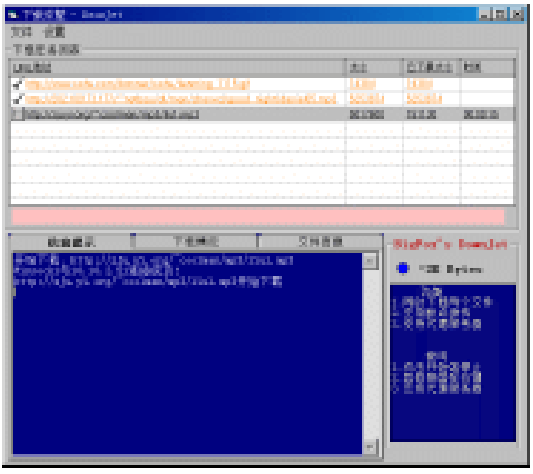


图 7-6　窗体 frmDown

另外还有用于 TCP 通信的 Winsock 控件，用于提示保存下载文件的 CommonDialog 控件等，这些读者可以打开工程文件自己了解。

程序代码分析：

```
Dim Xitem As ListItem
Dim mDownInfoSave As DownInfoSave
'声明正在下载的任务的类变量
Dim DownJet(1 To 2) As clsDown
```

以上的程序代码声明了类变量数组 DownJet，用于下载文件并保存下载数据，每个变量对应一个正在下载的任务。

```
Private Sub add_Click()
frmAdd.Show vbModal
End Sub
```

```
Private Sub Form_Load()
MinForm.Show
CDlg.FileName = App.Path & "\TEMP"
'加载用于下载的 Winsock 控件，可同时执行两个任务
Load Wsock(1)
Load Wsock(2)
'创建两个类对象，每个对象负责一个下载任务
Set DownJet(1) = New clsDown
Set DownJet(2) = New clsDown
'初始化 ListView 控件
LView.ColumnHeaders.Clear
LView.ColumnHeaders.Add,,"URL 地址",LView.Width - 240 * Screen.TwipsPerPixelX
LView.ColumnHeaders.Add , , "大小", 80 * Screen.TwipsPerPixelX
LView.ColumnHeaders.Add , , "已下载大小", 80 * Screen.TwipsPerPixelX
LView.ColumnHeaders.Add , , "时间", 80 * Screen.TwipsPerPixelX
'从数据文件中读取下载任务的信息，加入到 ListView 中
Dim i
Dim Fnum As Integer
Dim mFname As String
'保存下载任务信息的文件
mFname = App.Path & "\Downjet.djt"
Fnum = FreeFile
Open mFname For Random As #Fnum Len = Len(mDownInfoSave)
i = 1
While Not EOF(Fnum)
    ReDim Preserve mDownInfo(i)
    Get #Fnum, i, mDownInfoSave
    If Not EOF(Fnum) Then
        mDownInfo(i).mFile = Trim(mDownInfoSave.mFile)
        mDownInfo(i).mGetSize = mDownInfoSave.mGetSize
        mDownInfo(i).mIndex = mDownInfoSave.mIndex
        mDownInfo(i).mProxy = Trim(mDownInfoSave.mProxy)
        mDownInfo(i).mProxyId = Trim(mDownInfoSave.mProxyId)
        mDownInfo(i).mProxyPass = Trim(mDownInfoSave.mProxyPass)
        mDownInfo(i).mProxyPort = mDownInfoSave.mProxyPort
        mDownInfo(i).mSize = mDownInfoSave.mSize
        mDownInfo(i).mUrl = Trim(mDownInfoSave.mUrl)
        mDownInfo(i).mUseProxy = mDownInfoSave.mUseProxy
        If mDownInfo(i).mGetSize + 1 < mDownInfo(i).mSize Then
```

```
            If Dir(mDownInfo(i).mFile) <> "" And mDownInfo(i).mFile <> "" Then
                mDownInfo(i).mGetSize = FileLen(mDownInfo(i).mFile)
            Else
                mDownInfo(i).mGetSize = 0
            End If
        End If
        AddUrl mDownInfo(i).mUrl, mDownInfo(i).mSize, mDownInfo(i).mGetSize,
mDownInfo(i).mFile
        i = i + 1
    End If
Wend
Close Fnum
SelectDown = -1
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
Dim i As Integer, j As Integer
Dim Fnum As Integer
Dim mFname As String
'往文件 DownJet.djt 中保存在 ListView 中的下载任务
mFname = App.Path & "\Downjet.djt"
If Dir(mFname) <> "" Then
    Kill mFname
End If
Fnum = FreeFile
j = 1
Open mFname For Random As #Fnum Len = Len(mDownInfoSave)
For i = 1 To UBound(mDownInfo) - 1
    If mDownInfo(i).mUrl <> "" And LView.ListItems(i).SmallIcon <> "delete" Then
        mDownInfoSave.mFile = mDownInfo(i).mFile
        mDownInfoSave.mGetSize = mDownInfo(i).mGetSize
        mDownInfoSave.mIndex = mDownInfo(i).mIndex
        mDownInfoSave.mProxy = mDownInfo(i).mProxy
        mDownInfoSave.mProxyId = mDownInfo(i).mProxyId
        mDownInfoSave.mProxyPass = mDownInfo(i).mProxyPass
        mDownInfoSave.mProxyPort = mDownInfo(i).mProxyPort
        mDownInfoSave.mSize = mDownInfo(i).mSize
        mDownInfoSave.mUrl = mDownInfo(i).mUrl
        mDownInfoSave.mUseProxy = mDownInfo(i).mUseProxy
        Put #Fnum, j, mDownInfoSave
```

```
            j = j + 1
        End If
    Next i
    Close Fnum
    End
    End Sub

    '向 ListView 添加 Item 的过程
    Public Sub AddUrl(myUrl As String, Optional ByVal mSize As String, Optional
ByVal mGetSize As String, Optional ByVal mFile As String)
    If myUrl <> "" Then
        If Val(mGetSize) + 1 < Val(mSize) Or Val(mSize) = 0 Then
            Set Xitem = LView.ListItems.Add(, "", myUrl, "stop", "stop")
        Else
            Set Xitem = LView.ListItems.Add(, "", myUrl, "ok", "ok")
        End If
        Xitem.Tag = mFile
        Xitem.ListSubItems.Add , "size", mSize
        Xitem.ListSubItems.Add , "getsize", mGetSize
    End If
    End Sub
```

程序开始运行时，在 Form_Load 事件中加入一些代码，以达到以下目的：

- 加载 Winsock 控件数组，Index 为 1 和 2 的 Winsock 每个负责一个任务的下载，可以同时下载两个任务。
- 初始化 ListView 控件，在该控件中显示下载的 URL 和文件大小，以及已下载的文件大小和下载时间。
- 从保存上次下载信息的文件（App.Path & "\Downjet.djt"）中通过中间变量 mDownInfoSave 读取下载信息。

在 Form_QueryUnload 事件中加入把下载信息保存到文件 Downjet.djt 的代码。值得注意的是，下载任务的已经下载文件的大小在没有完成或刚开始时，并没有按照保存了的已获得文件的大小计算，而是通过保存了部分下载数据的文件得到已下载的数据大小。这样，即使未完成的文件被删除，也能重新下载得到准确恢复。

```
    Private Sub Form_Resize()
    '如果主窗体最小化，显示浮动小窗体
    If Me.WindowState = vbMinimized Then
        MinForm.Show
    End If
    End Sub
    Private Sub LView_DblClick()
```

```
    '如果选中的任务处于下载状态则停止，否则开始下载
    Dim i As Integer
    Dim mSel As Integer
    mSel = LView.SelectedItem.Index
    '如果选中的任务正在下载，则停止该下载
    For i = 1 To 2
        If Wsock(i).State <> sckClosed And CurrentDown(i) = mSel Then
            DownJet(i).bCancel = True
            CurrentDown(i) = 0
            Exit Sub
        End If
    Next i
    '如果选中的已经下载完毕，显示下载完毕提示
    If   mDownInfo(mSel).mGetSize   +   1   >=   mDownInfo(mSel).mSize   And
mDownInfo(mSel).mSize > 0 Then
        lblTishi.Caption = mDownInfo(mSel).mFile & "已经下载完毕!!!"
        txtInfo.Text = txtInfo.Text & lblTishi.Caption & vbCrLf
        Exit Sub
    End If
    '检查是否有空闲的 Winsock
    For i = 1 To 2
        If Wsock(i).State = sckClosed And CurrentDown(i) = 0 Then
            'Winsock 已经关闭，处于空闲状态，或者处于连接请求状态
            Dim mSel2 As Integer
            mSel2 = mSel
            Set DownJet(i) = Nothing
            Set DownJet(i) = New clsDown
            DownJet(i).DownUrl = LView.SelectedItem.Text
            '分析下载的 Url 是否合法
            If DownJet(i).AnalyzeUrl = False Then
                Exit For
            End If
            '如果第一次下载，选择路径保存下载文件
            If mDownInfo(mSel).mFile = "" Then
                '如果取消保存则退出该过程，取消下载
                On Error GoTo err1
                CDlg.CancelError = True
                CDlg.Flags = cdlOFNOverwritePrompt
                CDlg.FileName = DownJet(i).mFile
```

```
            CDlg.ShowSave
            DownJet(i).mFile = CDlg.FileName
        Else
            DownJet(i).mFile = mDownInfo(mSel).mFile
        End If
        Pic(i).Cls
        '下载任务索引
        DownJet(i).WhichDown = mSel
        '下载使用的 Winsock 索引
        DownJet(i).WhichSocket = i
        '下载的文件总长度
        DownJet(i).mFlen = mDownInfo(mSel2).mSize
        '已经下载的文件的大小
        DownJet(i).ReceiveBytes = mDownInfo(mSel2).mGetSize
        '****设置代理服务器选项
        DownJet(i).mProxy = mDownInfo(mSel2).mProxy
        DownJet(i).mProxyPort = mDownInfo(mSel2).mProxyPort
        DownJet(i).mProxyId = mDownInfo(mSel2).mProxyId
        DownJet(i).mProxyPass = mDownInfo(mSel2).mProxyPass
        '****
        LView.SelectedItem.SmallIcon = "start"
        '表明当前下载的任务索引
        CurrentDown(i) = LView.SelectedItem.Index
        '根据文件长度和已下载的文件长度在图片框画表示下载情况的圆点
        DrawDownPic       i,       0,       mDownInfo(CurrentDown(i)).mSize,
mDownInfo(CurrentDown(i)).mGetSize
        txtInfo.Text=txtInfo.Text&"开始下载："&mDownInfo(mSel2).mUrl&vbCrLf
        '开始下载，如果 StartDown 返回 True 表示连接服务器成功，发送请求
        If DownJet(i).StartDown() = False Then
            LView.ListItems(DownJet(i).WhichDown).SmallIcon = "error"
            lblTishi.Caption = "Wisock" & i & "连接服务器失败!!"
            txtInfo.Text=txtInfo.Text&"Wisock"&i&"连接服务器失败!!"&vbCrLf
            CurrentDown(i) = 0
        Else
            '开始下载成功，下载文件的路径保存到任务变量中
            mDownInfo(mSel2).mFile = DownJet(i).mFile
        End If
        Exit For
    Else
```

```
            lblTishi.Caption = "Wisock" & i & "已经有文件在下载了!!!"
            txtInfo.Text=txtInfo.Text&"Wisock"&i&"已经有文件在下载了!!!"&vbCrLf
        End If
    Next i
    err1:
    End Sub
```

以上代码中的 LView_DblClick **事件中的代码是比较重要的。双击下载任务，首先判断下载任务是否正在下载，如果是就设置** Downjet **的** bCancel **属性为** True**，下载管理是由类对象** Downjet **实现的，后面我们还要详细介绍它的功能。如果选中的任务未在下载，检查当前下载线程是否满了两个，如果未满足，则利用空闲的** Winsock **和** DownJet **进行下载，分析下载的** URL **得到机器名、文件名等在类方法** AnalyzeUrl **中实现，开始下载在类方法** StartDown **中实现。这两个方法和其他类方法在** clsDown **中的会有详细说明。**

```
    Private Sub LView_MouseUp(Button As Integer, Shift As Integer, x As Single,
y As Single)
    Static mIndex As Integer
    SelectDown = LView.SelectedItem.Index
    If SelectDown < 1 Then Exit Sub
    '在 txtfile 中显示选中的下载任务信息
    txtFile.Text = "URL:        " & mDownInfo(SelectDown).mUrl & vbCrLf
    txtFile.Text = txtFile.Text & "文件名:     " & mDownInfo(SelectDown).mFile &
vbCrLf
    txtFile.Text = txtFile.Text & "大小:       " & mDownInfo(SelectDown).mSize &
"字节" & vbCrLf
    txtFile.Text = txtFile.Text & "已下载大小:  " & mDownInfo(SelectDown).mGetSize
& "字节" & vbCrLf
    txtFile.Text = txtFile.Text & "代理服务器:  " & mDownInfo(SelectDown).mProxy
& vbCrLf
    '在 picturebox 控件中显示当前选中的下载任务的 block 信息
    '其中 pic(0)显示选中的没有在下载的任务信息
    'pic(1)和 pic(2)显示第一个和第二个 Winsock 的下载信息
    If SelectDown = CurrentDown(1) Then
        PicVisible (1)
    ElseIf SelectDown = CurrentDown(2) Then
        PicVisible (2)
    Else
        Pic(0).Cls
        PicVisible (0)
        DrawDownPic         0,         0,         mDownInfo(SelectDown).mSize,
mDownInfo(SelectDown).mGetSize
```

```
    End If
    mIndex = SelectDown
    '如果按了鼠标右键弹出删除菜单
    If Button = 2 Then
        PopupMenu menusetup
    End If
    End Sub
```

以上代码的功能是当点击了ListView中的任务，将任务的详细信息显示在窗体下面的“下载情况”、“文件信息”中。如果单击鼠标右键，将弹出删除的菜单，可以选择“删除”命令，给相应的任务打上“删除”的标志。但有一点要注意的是，选择删除了之后并没有将下载的URL立即删除，而是等到了程序退出时，才通过不保存该下载任务的信息实现删除的，如果已经下载了一部分文件数据就存放在磁盘上，并没有删除那些没有下载完成的文件。这是程序不太完善的地方之一，读者可以自己加上几句代码加以完善。

```
    '接收拖放的信息，如果是可下载的Url，加到ListView中
    Private Sub LView_OLEDragDrop(Data As MSComctlLib.DataObject, Effect As Long,
Button As Integer, Shift As Integer, x As Single, y As Single)
    If Data.GetFormat(vbCFText) Then
        If vbOK = MsgBox("确定要下载" & Data.GetData(vbCFText), vbOKCancel) Then
            AddNewUrl Data.GetData(vbCFText)
        End If
    End If
    End Sub
    Private Sub menuadd_Click()
    '加入下载任务
    frmAdd.Show vbModal
    End Sub
    Private Sub menuDel_Click()
    '删除下载任务
    LView.ListItems(LView.SelectedItem.Index).SmallIcon = "delete"
    End Sub
    Private Sub menuquit_Click()
    '退出程序
    Unload Me
    End Sub
    '加入新的下载任务
    Public Function AddNewUrl(myUrl As String)
        Dim i As Integer
        For i = 1 To LView.ListItems.Count
            If LView.ListItems(i).Text = myUrl Then
                MsgBox "该URL已经在下载队列中了！"
```

```
            Exit Function
        End If
    Next i
    AddUrl myUrl
    Dim kk As Integer
    kk = UBound(mDownInfo)
    ReDim Preserve mDownInfo(kk + 1)
    mDownInfo(kk).mUrl = myUrl
End Function
```

以上代码可以看出，加入下载任务可以通过菜单选择显示 frmAdd 实现，也可以通过拖放的技术实现。拖放实现是不能配置下载选项的，而从 frmAdd 加入可以配置代理服务器的各个选项，此外还有退出、删除的菜单项。

```
Private Sub Timer1_Timer()
Dim i As Integer
Static Count(1 To 2) As Integer
Static mColor(1 To 2) As Long
'定时显示下载时的状态
For i = 1 To 2
    If CurrentDown(i) > 0 Then
        LView.ListItems(CurrentDown(i)).SubItems(3) = _
Format(DateAdd("s", DateDiff("s", DownJet(i).StartTime, Time()), #12:00:00
AM#), "hh:mm:ss")
    End If
    If DownJet(i).bBusy = True Then
        Count(i) = Count(i) + 1
    Else
        Count(i) = 0
    End If
    If Count(i) > 6 Then
        Count(i) = 0
        If mColor(i) = vbRed Then
            mColor(i) = vbGreen
        Else
            mColor(i) = vbRed
        End If
    End If
    MinForm.FillColor = mColor(i)
    MinForm.Circle ((Count(i) + 1) * 120, i * 120 + 60), 50
Next i
End Sub
```

定时器 Timer1 用于定时显示下载时的状态。

```
    Private Sub Wsock_Close(Index As Integer)
    CloseSocket Index, "Winsock 关闭"
    End Sub

    Private Sub Wsock_Connect(Index As Integer)
    txtInfo.Text = txtInfo.Text & "Winsock" & Index & " 与 " &
Wsock(Index).RemoteHostIP & "连接成功！" & vbCrLf
    End Sub

    Private Sub Wsock_DataArrival(Index As Integer, ByVal bytesTotal As Long)
    Dim ByteData1() As Byte
    Dim ByteData2() As Byte
    '根据 Winsock 的索引接收和保存数据
    If Index = 1 Then
        '文件总长度的变量
        Dim Flen1 As Long
        '请求服务器返回的响应码
        Dim ReCode1 As String
        Wsock(Index).GetData ByteData1, vbByte
        '下载数据保存数据，如果是连接后第一次返回的数据，返回服务器的响应码
        ReCode1 = DownJet(Index).SaveData(bytesTotal, ByteData1(), Flen1)
        Select Case ReCode1
        Case "200"
            '响应码为 200 表示成功
            txtInfo.Text = txtInfo.Text & DownJet(Index).DownUrl & "开始下载" & vbCrLf
        Case "206"
            '响应码 206 表示断点续传成功
            txtInfo.Text = txtInfo.Text & DownJet(Index).DownUrl & "开始从" &
DownJet(Index).mFlen _
    & "断点续传下载" & vbCrLf
        Case "404"
            '响应码为 404 表示请求的下载的文件未找到
            txtInfo.Text = txtInfo.Text & DownJet(Index).DownUrl & "文件不存在" &
vbCrLf
            CloseSocket Index, "文件未找到！"
        Case "error"
            '其他响应码视为错误
            CloseSocket Index, "请求时出错"
            txtInfo.Text = txtInfo.Text & DownJet(Index).DownUrl & "出错了" & vbCrLf
```

```
        Case "cancel"
            '用户取消
            CloseSocket Index, "用户取消"
            Exit Sub
        End Select
        If Flen1 > 0 Then
            '如果任务第一次下载，则保存后得到文件长度
            mDownInfo(DownJet(Index).WhichDown).mSize = Flen1
            LView.ListItems(DownJet(Index).WhichDown).SubItems(1) = Flen1
        End If
    Else
        Dim Flen2 As Long
        Dim ReCode2 As String
        Wsock(Index).GetData ByteData2, vbByte
        ReCode2 = DownJet(Index).SaveData(bytesTotal, ByteData2(), Flen2)
        Select Case ReCode2
        Case "200"
            txtInfo.Text = txtInfo.Text & DownJet(Index).DownUrl & "开始下载" & vbCrLf
        Case "206"
            txtInfo.Text = txtInfo.Text & DownJet(Index).DownUrl & "开始从" &
DownJet(Index).mFlen _
    & "断点续传下载" & vbCrLf
        Case "404"
            txtInfo.Text = txtInfo.Text & DownJet(Index).DownUrl & "文件不存在" &
vbCrLf
            CloseSocket Index, "文件未找到！"
        Case "error"
            CloseSocket Index, "请求时出错"
            txtInfo.Text = txtInfo.Text & DownJet(Index).DownUrl & "出错了" & vbCrLf
        Case "cancel"
            CloseSocket Index, "用户取消"
            Exit Sub
        End Select
        If Flen2 > 0 Then
            mDownInfo(DownJet(Index).WhichDown).mSize = Flen2
            LView.ListItems(DownJet(Index).WhichDown).SubItems(1) = Flen2
        End If
    End If
    End Sub
```

```
    '控制描绘下载情况 block 的 PictureBox 的可见
    Public Sub PicVisible(Index As Integer)
    Dim i As Integer
    For i = 0 To Pic.Count - 1
       Pic(i).Visible = False
    Next i
    Pic(Index).Visible = True
    End Sub

    Private Sub Wsock_Error(Index As Integer, ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As String,
ByVal HelpContext As Long, CancelDisplay As Boolean)
    CloseSocket Index, "Winsock 出错"
    End Sub

    '关闭 socket 后做的一些处理
    Public Sub CloseSocket(Index As Integer, ClsStr As String)
    If mDownInfo(DownJet(Index).WhichDown).mGetSize + 1 >= _
     mDownInfo(DownJet(Index).WhichDown).mSize                          And
mDownInfo(DownJet(Index).WhichDown).mSize _
    >  0 Then
       LView.ListItems(DownJet(Index).WhichDown).SmallIcon = "ok"
       txtInfo.Text = txtInfo.Text & mDownInfo(DownJet(Index).WhichDown).mUrl &
_
    "的下载完成了" & vbCrLf
    Else
       LView.ListItems(DownJet(Index).WhichDown).SmallIcon = "stop"
       txtInfo.Text = txtInfo.Text & mDownInfo(DownJet(Index).WhichDown).mUrl & _
    "的下载因为" & ClsStr & "被关闭了" & vbCrLf
    End If
    DownJet(Index).bBusy = False
    Wsock(Index).Close
    CurrentDown(Index) = 0
    End Sub
```

Winsock 的事件 Close、Connect、Error 中添加了代码，其作用是显示一些提示信息、处理错误和关闭。Public 类型的函数 CloseSocket 对关闭 Winscok 连接后做了一些处理，恢复一些描述下载情况的变量。

接收从服务器返回的下载文件数据是在 Winsock 事件 DataArrival 中进行的，但对数据进

行处理和保存是在类对象的方法 SaveData 中进行的，并有可能返回相应的响应码字符串，可根据这些响应码的字符串进行处理。在这里对于使用两段几乎完全一样的代码对两个同时下载的线程进行数据处理的原因，没有太多的理论根据，是依经验这样做的，否则其中的一个线程会将经常等待另外的下载任务下载完成之后才进行。这样可以使得两个下载的线程同时在下载数据。不过使用多线程（不同于上面所说的“下载线程”）可以很方便的实现，每个下载任务使用一个线程下载，每个下载线程由于分析数据、保存数据或等待连接等情况需要在程序中循环，会使得整个程序好象被挂起，无法响应其他事件和执行程序。在后面讲到类 clsDown 的方法 StartDown 中也存在这样的情况：程序因为循环等待 Winsock 连接，如果处理不当会引起挂起。在没有使用多线程时碰到这种情况，往往用 DoEvents 语句使得程序能够响应其他事件，不致于会出现程序挂起“假死”的状态。

```
    '根据接收到的文件长度，已经下载长度的信息在 Pic 画 Block 图
    'mflen：         文件长度
    'mNum：          接收到的字节数
    ' ReceiveBytes：已经接收到的字节数
    Public Sub DrawDownPic(Index As Integer, mNum As Long, Optional mFlen As Long,
Optional ReceiveBytes As Long)
    If mNum > 0 Then
       mDownInfo(DownJet(Index).WhichDown).mGetSize =  _
    mDownInfo(DownJet(Index).WhichDown).mGetSize + mNum
       LView.ListItems(DownJet(Index).WhichDown).SubItems(2) = ReceiveBytes + mNum
    End If
    Dim Getnum As Long
    Getnum = ReceiveBytes
    Dim TGetNum As Long
    Dim i, j As Long
    Dim kk1 As Long, kk2 As Long
    If mNum = 0 Then
       Getnum = 0
    End If

    If Getnum = 0 Then
       Pic(Index).FillColor = vbWhite
       kk1 = mFlen / 4096
       j = 0
       For i = 1 To mFlen / 4096
          Pic(Index).Circle ((i - j * 50) * 120 + 0, j * 120 + 100), 50, vbBlack
          j = Fix(i / 50)
       Next i
       Pic(Index).FillColor = &HFF0000
```

```
End If

TGetNum = Getnum
If Getnum = 0 And ReceiveBytes > 0 Then
    '加上以前已经接收到的
    Getnum = ReceiveBytes
End If
Getnum = Getnum + mNum
kk1 = Fix(TGetNum / 4096)
kk2 = Fix(Getnum / 4096)
j = Fix(kk1 / 50) + 1
For i = kk1 To kk2
    Pic(Index).Circle ((i - j * 50) * 120 + 0, j * 120 + 100), 50, vbRed
    j = Fix(i / 50)
Next i
End Sub
```

以上的代码是根据接收到下载的数据在 PictureBox **中画圆点来描述下载情况的。要注意的是：要将** Picturebox **的** AutoRedraw **属性设置为** True**，否则用** Circle **方法画的圆要在** PictureBox **刷新之后才会显示。**

3. **窗体** frmAdd

该窗体是用于加入下载任务的，可以设置的有代理服务器的使用、代理服务器的端口（缺省是 80**）、代理服务器的认证账号和密码（在大多数情况下是必须的）。**

```
Private Sub chkAuth_Click()
If chkAuth.Value = 1 Then
    txtID.Enabled = True
    txtPass.Enabled = True
Else
    txtID.Enabled = False
    txtPass.Enabled = False
End If
End Sub
Private Sub chkProxy_Click()
If chkProxy.Value = 1 Then
       txtProxy.Enabled = True
       txtPort.Enabled = True
       chkAuth.Enabled = True
Else
       txtProxy.Enabled = False
       txtPort.Enabled = False
```

```
        chkAuth.Enabled = False
        txtID.Enabled = False
        txtPass.Enabled = False
End If
chkAuth.Value = 0
End Sub
Private Sub cmdAdd_Click(Index As Integer)
If Index = 0 Then
    Dim i As Integer
    For i = 1 To frmDown.LView.ListItems.Count
        If frmDown.LView.ListItems(i).Text = txtUrl.Text Then
            MsgBox "该 URL 已经在下载队列中了！"
            txtUrl.Text = ""
            Exit Sub
        End If
    Next i
    frmDown.AddUrl txtUrl
    Dim kk As Integer
    kk = UBound(mDownInfo)
    ReDim Preserve mDownInfo(kk + 1)
    mDownInfo(kk).mUrl = txtUrl.Text
    If chkProxy.Value = 1 Then
        mDownInfo(kk).mUseProxy = True
        mDownInfo(kk).mProxy = txtProxy.Text
        mDownInfo(kk).mProxyPort = Val(txtPort.Text)
        If chkAuth.Value = 1 Then
            mDownInfo(kk).mProxyId = txtID.Text
            mDownInfo(kk).mProxyPass = txtPass.Text
        End If
    End If

End If
    SelectDown = 0
    Unload Me

End Sub
Private Sub Form_Activate()
Dim mStr1 As String
mStr1 = Clipboard.GetText
If InStr(1, mStr1, "http://") = 1 And Len(mStr1) < 180 Then
```

```
        txtUrl.Text = mStr1
    End If
    End Sub
```

下载任务信息保存在 mDownInfo 数组变量中，该数组变量根据任务的数量制定数组的大小，重新定义数组大小是用 ReDim 语句实现的（在前面的模块 modDown 中已经声明了数组），并且使用关键字 Preserver 可以使原来数组的值保存下来。如：

```
    kk = UBound(mDownInfo)
    ReDim Preserve mDownInfo(kk + 1)
```

上面的两行代码就是将原来数组的大小加 1，而保持原数组前面的数值。

4. 窗体 MinForm

该窗体是一个总是在所有程序最上方的浮动式小窗体，用来实现一些方便使用的功能：

- 双击可以显示下载的主窗体 frmDown。
- 简单地查看下载是否正在进行。
- 接受拖放下载的 URL。

```
    Private Sub Form_DblClick()
    If frmDown.WindowState = vbNormal Then
        frmDown.WindowState = vbMinimized
    Else
        frmDown.WindowState = vbNormal
    End If
    End Sub

    Private Sub Form_Load()
    SetWindowPos Me.hwnd, -1, 0, 0, 0, 0, 3
    End Sub

    Private Sub Form_OLEDragDrop(Data As DataObject, Effect As Long, Button As
Integer, Shift As Integer, x As Single, y As Single)
    If Data.GetFormat(vbCFText) Then
        If vbOK = MsgBox("确定要下载" & Data.GetData(vbCFText), vbOKCancel) Then
            frmDown.AddNewUrl Data.GetData(vbCFText)
            frmDown.WindowState = vbNormal
        End If
    End If
    End Sub
```

5. 类模块 clsDown

程序中最重要的是类模块 clsDown，这个类模块负责分析下载的 URL 地址，向服务器发送请求和接收分析服务器返回的数据并存储数据等作用。在这里之所以要使用类模块来管理

下载任务，是因为使用类能够很好地进行程序设计，使整个程序的设计思路变得更清晰。由于类模块在整个程序中处于核心地位，了解这个类模块成为了解整个程序的关键所在。首先介绍一下类使用模块的好处。

当前流行的编程方式由原来的面向过程转为面向对象的程序设计，许多新的编程语言如 C++，Java 等也是面向对象设计语言的，Visual Basic 6.0 虽然还不是完全的面向对象设计语言，但已经提供了类、对象等来实现面向的对象设计。类将用户定义类型和过程组织在一起类模块和标准模块的不同点在于存储数据的方法不同。标准模块的数据只有一个备份。这意味着标准模块中一个公共变量的值改变以后，在后面的程序中再读取该变量时，它将得到同一个值。而类模块的数据，是相对于类实例（也就是，由类创建的每一对象）而独立存在的。同样的，标准模块中的数据在程序作用域内存在，也就是说，它存在于程序的存活期中；而类实例中的数据只存在于对象的存活期，它随对象的创建而创建，随对象的撤消而消失。最后，当变量在标准模块中声明为 Public 时，则它在工程中任何地方都是可见的。而类模块中的 Public 变量，只有当对象变量含有对某一类实例的引用时才能访问。

类变量的这些优点能使得程序能够方便地处理多个对象的情况。在该工程中，实现了两个文件同时下载，每个类变量对应着一个下载任务，下载时彼此的信息是不会发生干涉的，并且使用的是一样的变量和程序代码。类变量的优点从这里可以体现出来。

首先定义类模块的公有变量，这些共有变量在类模块之外的程序代码可以通过类对象（实例）访问。严格地来说，这样定义是不好的。因为这些公有变量之中，有些是只允许外部程序访问而不允许修改的。如果以 Public 的方式声明，则外部程序也可以修改了。这里只是为了写程序的时候方便才这样定义。

```
Public bBusy As Boolean     '表示正在下载一个任务
Public DownUrl As String        '要下载的 URL 地址
Public WhichSocket As Integer   '使用的 Winsock 的索引
Public WhichDown As Integer '下载任务的索引
Public ReceiveData As String    '接收到的下载数据（字符串类型）
Public StartTime As Date        '下载的开始连接时间
Public ReceiveBytes As Long '已下载的文件数据字节数
Public mFlen As Long            '已下载文件数据的长度
Public bCancel As Boolean       '用户是否取消下载
Public mProxy As String     '代理服务器地址
Public mProxyPort As Integer    '代理服务器端口
Public mProxyId As String       '代理服务器的认证账号及密码
Public mProxyPass As String
Public mFile As String          '保存的文件路径

接着声明私有变量，这些变量只有类方法才能使用：


Private mHost As String     '连接的主机名和端口
Private mPort As Integer
Private mRelativeUrl As String  '下载的相对 URL
```

代码分析：

```
'分析下载的 URL
Public Function AnalyzeUrl() As Boolean
Dim pos1, pos2 As Integer
Dim mUrl As String
mUrl = DownUrl
If InStr(1, mUrl, "http://") > 0 Then
    '得到端口号
    mPort = 80
Else
    AnalyzeUrl = False
    Exit Function
End If
pos1 = InStr(1, mUrl, "http://")
pos2 = InStr(8, mUrl, "/")
If pos2 = 0 Then
    AnalyzeUrl = False
    Exit Function
Else
    '得到主机地址
    mHost = Mid(mUrl, 8, pos2 - 8)
    '得到相对路径
    mRelativeUrl = Mid(mUrl, pos2)
End If
pos2 = InStrRev(mUrl, "/")
If pos2 > 8 Then
    '得到文件名
    mFile = Mid(mUrl, pos2 + 1)
Else
    AnalyzeUrl = False
    Exit Function
End If
AnalyzeUrl = True
End Function
```

函数 AnalyzeUrl **是对下载** URL**（在类变量** DownUrl**）分析，得到下载的服务器名称、下载的文件名和相对** URL**。在这里需要** URL **形式是这样的：**

```
http://10.11.111.119/download/zhang.mp3
```

经过 AnalyzeUrl **函数进行分析后，得到主机地址为** 10.11.111.119**，相对的** URL **路径为** /download/zhang.mp3**，文件名为** zhang.mp3**。**URL **的是以“** http:// **”开始的，设置端口为** 80**，**

这是 Web Server 默认的端口号。需要注意的是有些个别的 Web 服务器端口不是 80，这时下载的 URL 有如下的形式：

```
http://10.11.111.119:81/download/zhang.mp3
```

在主机地址加上一个冒号接着一个端口号，这时必须取得正确的端口号，否则将会连接不成功，在以下的函数中没有加入相应的代码判断，读者可以自己进一步完善。

经过分析后的 URL 如果符合要求，该函数返回 True，否则返回 False。

```
'根据代理的设置使用不同的函数连接服务器
'连接成功返回 true，否则返回 false
Public Function StartDown() As Boolean
bBusy = True
If mProxy <> "" And mProxyPort > 0 Then
    '使用代理服务器下载
    StartDown = StartDownProxy()
Else
    '直接下载
    StartDown = StartDownNoProxy()
End If
End Function
```

以上的函数 StartDown 是根据下载任务是否设置了代理服务器而调用不同的函数。

```
'直接连接 Url 指定的服务器下载
Public Function StartDownNoProxy() As Boolean
StartTime = Time()
'设置 Winsock 属性并连接服务器
frmDown.Wsock(WhichSocket).RemoteHost = mHost
frmDown.Wsock(WhichSocket).RemotePort = mPort
frmDown.Wsock(WhichSocket).Connect
'使用循环等待连接服务器成功
Do While frmDown.Wsock(WhichSocket).State <> sckConnected
    DoEvents: DoEvents: DoEvents: DoEvents
    '连接时间超过 20 秒或取消下载，退出该过程并返回 false
    If DateDiff("s", StartTime, Time()) > 20 Or bCancel = True Or bBusy = False Then
        frmDown.CloseSocket WhichSocket, "连接服务器时间过长"
        StartDownNoProxy = False
        Exit Function
    End If
Loop
'向服务器发送下载文件请求
Dim Getstr As String
Getstr = Getstr & "GET " & mRelativeUrl & " HTTP/1.1" & vbCrLf
```

```
    Getstr = Getstr & "Accept: */*" & vbCrLf
    Getstr = Getstr & "Accept -Language: zh -cn" & vbCrLf
    Getstr = Getstr & "Accept -Encoding: gzip , deflate" & vbCrLf
    Getstr = Getstr & "User-Agent: DownJet1.0" & vbCrLf
    Getstr = Getstr & "Host: " & mHost & vbCrLf
    If mFlen > 0 Then
        '如果以前已经下载了一部分数据，发送断点续传请求
        Getstr = Getstr & "Range: bytes=" & ReceiveBytes & "-" & vbCrLf
    End If
    Getstr = Getstr & "Connection: close" & vbCrLf
    Getstr = Getstr & vbCrLf
    frmDown.Wsock(WhichSocket).SendData Getstr
    StartDownNoProxy = True
    End Function
```

StartDwonNoProxy 是直接请求 Web Server 下载文件的，下载的具体步骤如下：

（1）在指定的地址和端口连接 Server。

（2）服务器接受 Client 的连接请求后，Client 向服务器发送请求下载文件的字符串。

（3）Client 接收并整理保存 Server 返回的数据。

前面有关 HTTP 的内容中也讲过，根据 HTTP 协议，客户端（Client）向服务器（Web Server）请求文件的方法一共有两种：GET 和 POST，而 POST 方法是用于向服务器提交数据时才用的，因此要下载文件只要通过 GET 方法就可以了。StartDownNoProxy 函数就是完成步骤 1 和 2 的，后面的 SaveData 完成步骤 3。向服务器发送的请求字符串的形式到底应该怎么写呢？下面先看看经常使用的 IE 4.0 浏览器向服务器请求文件的字符串：

```
    GET /index.htm HTTP/1.1
    Accept:            application/vnd.ms-excel,           application/msword,
application/vnd.ms-powerpoint, */*
    Accept-Language: zh-cn
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
    Host: localhost
    Connection: Keep-Alive
```

从以上得知：首先在第一行中指明获取文件的方法 GET、POST 或 HEAD 等，接着指明文件的相对 URL，最后是 HTTP 的版本号，中间要用空格相隔。通用的写法为：

GET|POST|HEAD (空格) 相对 URL(空格) HTTP 版本号

接着从第二行开始，使用标题字段表明一些请求信息，如 Accept 后面接着的标题字段内容表示可以接受的媒体类型，如果如下：

```
    Accept: */*
```

表明可以接受所有类型的文件。其他字段的意思在前面都介绍过，这里就不再啰嗦了，值得注意的，在 IE 的请求中，Connetion 标题字段往往设置为 Keep-Alive，在请求文件结束

以后用以保持与服务器连接，下次请求时不用再启动 Winsock 连接，从而增加连接效率。如果只是要下载一个文件，可将 Connetion 设置为 Close：

```
Connection: close
```

在下载完成之后，远程的 Web Server 会自动地将 TCP 连接关闭。

所以可以构造请求下载文件的字符串（假如 Web Server 为 10.11.111.119）：

```
GET /zhang.mp3 HTTP/1.1
Accept: */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Downjet1.0
Host: 10.11.111.119
Connection: close
```

在连接远程 Web Server 后，将以上字符串发送给服务器，如果服务器上存在指定的资源文件，服务器将会返回响应字符串和文件的数据。注意发送的请求字符串必须以两个 Crlf 结尾，否则服务器不知道接到的请求字符串已经结束。

如果文件已经下载了一部分，下次开始下载时就要指定所需文件数据的开始位置，以实现断点续传，断点续传在文件传输协议中特别是 HTTP 和 FTP 协议中较常用，不过需要服务器的支持，只有服务器支持断点续传下载文件的客户端才能正确地取得返回的部分数据，目前许多 Web 服务器都是支持断点续传的。进行断点续传时需要使用标题字段 Range 指定文件的起始位置（以字节计算）。如：

```
GET /zhang.mp3 HTTP/1.1
Accept: */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Downjet1.0
Host: 10.11.111.119
Range: bytes=2666-
Connection: close
```

如果文件大小超过 2666Bytes，服务器返回的数据从文件 zhang.mp3 的 2666 字节开始。

也可以指定中间的一段数据：

```
GET /zhang.mp3 HTTP/1.1
Accept: */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Downjet1.0
Host: 10.11.111.119
Range: bytes=2666-3888
Connection: close
```

服务器返回的数据从文件 zhang.mp3 的第 2666 字节开始，第 3888 个字节结束。如果要

设计可以多个 Winsock 同时下载一个文件的数据，这样请求返回数据是很有用的。

```
    '通过代理服务器连接下载
    Public Function StartDownProxy() As Boolean
    StartTime = Time()
    '设置 Winsock 属性并连接代理服务器
    frmDown.Wsock(WhichSocket).RemoteHost = mProxy
    frmDown.Wsock(WhichSocket).RemotePort = mProxyPort
    frmDown.Wsock(WhichSocket).Connect
    Do While frmDown.Wsock(WhichSocket).State <> sckConnected
        DoEvents: DoEvents: DoEvents: DoEvents
        If DateDiff("s", StartTime, Time()) > 10 Or bCancel = True Or bBusy = False Then
            frmDown.CloseSocket WhichSocket, "连接代理服务器时间过长"
            StartDownProxy = False
            Exit Function
        End If
    Loop
    '向代理服务器发送下载文件请求
    Dim Getstr As String
    Getstr = Getstr & "GET " & DownUrl & " HTTP/1.1" & vbCrLf
    Getstr = Getstr & "Accept: */*" & vbCrLf
    Getstr = Getstr & "Accept -Language: zh -cn" & vbCrLf
    Getstr = Getstr & "Accept -Encoding: gzip , deflate" & vbCrLf
    Getstr = Getstr & "User-Agent: DownJet1.0" & vbCrLf
    Getstr = Getstr & "Host: " & mHost & vbCrLf
    If mProxyId <> "" Then
        '如果使用身份验证，编码后加入到请求字符串中
        Getstr = Getstr & "Proxy-Authorization: Basic " & EncodeStr(mProxyId & ":"
& mProxyPass) & vbCrLf
    End If
    If mFlen > 0 Then
        '如果以前已经下载了一部分数据，发送断点续传请求
        Getstr = Getstr & "Range: bytes=" & ReceiveBytes & "-" & vbCrLf
    End If
    Getstr = Getstr & "Connection: close" & vbCrLf
    Getstr = Getstr & vbCrLf
    frmDown.Wsock(WhichSocket).SendData Getstr
    StartDownProxy = True
    End Function
```

以上的函数 StartDwonProxy 是通过代理服务器下载文件的。与直接连接服务器请求下载文件类似，但有以下的几点不同：

- Winsock 连接的是代理服务器，向代理服务器发送请求字符串；
- 请求下载文件使用完整的 URL；
- 通常需要发送代理服务器的认证账号及密码。

如通过代理服务器下载 http://10.11.111.119/zhang.mp3 的文件，在连接代理服务器之后向服务器发送下面字符串。

```
GET http://10.11.111.119/zhang.mp3 HTTP/1.1
Accept: */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Downjet1.0
Proxy-Authorization: Basic YnVzeXpob25nOjE=
Connection: close
```

在这里还使用到了代理服务器的身份验证，验证的账号和密码经过 base64 编码后作为标题字段 Proxy-Authorization 的内容发送给代理服务器。以上发送的就是账号为 busyzhong 密码为 1 的经过编码后的字符串。

```
'分析并保存 Winsock 得到服务器响应的数据
'入口变量
'ByteNum：  接收到数据的字节数
'ByteData： 接收数据的 Byte 类型的数组
'出口变量：
'Flen：     文件长度
'函数返回值：表示一定意思的字符串
Public Function SaveData(ByteNum As Long, ByteData() As Byte, Flen As Long)
As String
Dim Tfile As String
Dim Fnum As Integer
Static m3Byte(3) As Byte
Static bAppend As Boolean
Dim StartPos As Long
Dim i As Long
If bAppend = False Then
   ReceiveData = ReceiveData & StrConv(ByteData(), vbUnicode)
   If (InStr(1, ReceiveData, "HTTP/1.0 200 OK") Or InStr(1, ReceiveData,
"HTTP/1.1 200 OK")) Then
      '表示请求下载文件成功
      SaveData = "200"
   ElseIf (InStr(1, ReceiveData, "HTTP/1.0 206 ") Or InStr(1, ReceiveData,
"HTTP/1.1 206")) Then
      '表示请求断点续传成功
      SaveData = "206"
```

```
        ElseIf (InStr(1, ReceiveData, "HTTP/1.0 404 ") Or InStr(1, ReceiveData,
"HTTP/1.1 404")) Then
            '表示服务器未找到请求的资源
            SaveData = "404"
        Else
            '请求错误
            SaveData = "error"
            Exit Function
        End If
        '如果服务器响应的字符串有指定文件大小的标题字段，取得文件大小
        If InStr(1, ReceiveData, "Content-Length:") > 0 And mFlen = 0 Then
        Dim pos1 As Long, pos2 As Long
        pos1 = InStr(1, ReceiveData, "Content-Length:")
        pos2 = InStr(pos1 + 16, ReceiveData, vbCrLf)
            If pos2 > pos1 Then
                mFlen = Mid(ReceiveData, pos1 + 16, pos2 - pos1 - 16)
                Flen = mFlen
            End If
        End If
        '从服务器响应返回的数据中查找下载文件的起始位置
        For i = 0 To UBound(ByteData()) - 3
            If ByteData(i) = 13 And ByteData(i + 1) = 10 And _
    ByteData(i + 2) = 13 And ByteData(i + 3) = 10 Then
                StartPos = i + 4
                bAppend = True
                Exit For
            End If
        Next i
    End If
    '如果取消，则退出该过程，并返回字符串“cancel”
    If bAppend = False Then
        If bCancel = True Then
            SaveData = "cancel"
        End If
        Exit Function
    End If
    '在调用 frmDown 的 Public 函数 DraoDownPic 反映下载情况
    frmDown.DrawDownPic WhichSocket, ByteNum - StartPos, mFlen, ReceiveBytes
    ReceiveBytes = ReceiveBytes + ByteNum - StartPos
    Tfile = mFile
```

```
Fnum = FreeFile()
'向二进制文件中加入下载文件的数据
Open Tfile For Binary Lock Write As #Fnum
If LOF(Fnum) > 0 Then
   Seek #Fnum, LOF(Fnum) + 1
End If
If StartPos > 0 Then
   For i = StartPos To UBound(ByteData())
      Put #Fnum, , ByteData(i)
   Next i
Else
   Put #Fnum, , ByteData()
End If
Close #Fnum
End Function
```

函数 SaveData 用于保存下载数据，并且根据下载数据返回如响应码、文件大小等信息。在接收下载数据并保存之前，首先要理解服务器接受文件请求之后返回信息的格式和内容。

```
HTTP/1.1 200 OK
Date: Thu, 30 Nov 2000 11:29:41 GMT
Server: Apache/1.3.3 (Unix)  (Red Hat/Linux) PHP/4.0.1
Last-Modified: Thu, 05 Oct 2000 04:26:38 GMT
ETag: "19338c-3c7000-39dc02fe"
Accept-Ranges: bytes
Content-Length: 3960832
Connection: close
Content-Type: audio/mpeg
(文件数据)
```

服务器返回的数据由两部分组成，第一部分为表示响应信息的字符串，第二部分为文件的二进制数据，两部分之间的是用两个连续的 Crlf 隔开的。由于要下载的文件通常都不是纯文本的文件，接收数据（在 Winsock 的 Data_Arrival 事件中）时也无法立即把响应的第一部分信息和第二部分信息分割开。因此先使用 Byte 类型的数组变量接收数据，然后将数组按引用传递（ByRef）给函数 SaveData 进行处理分析。

数据的第一部分可能会有不少标题字段是用于描述返回的数据和其他信息的。对我们有用的是有响应码的第一行和有文件实体长度的标题字段 Content-Length。在前面 HTTP 的内容已经有响应码的介绍，这里就不再详细介绍了。下面是所用到的一些响应信息。

```
HTTP/1.1 200 OK
```

表示请求文件正确，服务器返回文件内容。

```
HTTP/1.1 206 Partial content
```

表示请求文件断点续传正确，服务器返回文件部分内容。

```
HTTP/1.1 404 Not Found
```

表示根据请求的 URL 没找到文件。

此外，服务器可能返回其他信息，如没有通过代理服务器的身份验证（401）、禁止访问（403）、服务器错误（500）等。

特别是 302 响应，表示所要请求的 URL 已经被移动，所以碰到这种情况时，要重新根据 Location 字段指定的地址下载，增加成功率。如：

```
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/4.0
Date: Mon, 27 Nov 2000 07:33:20 GMT
Connection: close
Location: http://10.11.111.120/mp3/zhang.mp3
Content-Type: text/html
Cache-control: private
Transfer-Encoding: chunked
```

其他响应的意思请参照前面介绍的 HTTP 内容，读者可以自己增加对服务器不同响应的处理。

另外要注意的是，接收数据是使用 Byte 类型的数组，如果要检查接收到的数据中是否有指定的字符串如：“ HTTP/1.1 200 ”等，可以使用函数 StrConv，将 Byte 数组转换为 String 类型的数据。

7.3 网页服务器高级开发

7.3.1 Web Server 的一些理论

设计一个完善的 Web Server 是比较困难的。因为一个成熟的 Web Server 涉及的问题有安全、效率、容错性强等诸多方面。在这里只是通过设计一个简单的 Web Server 达到对 HTTP 协议的认识，而前面是通过 HTTP 协议通信的客户端角度进行了解的。通过该实例，可以从 Web Server 的角度进行了解 HTTP 协议。

典型的 Web Server 结构采用“ 请求/响应 ”的模型，请求与响应由如下阶段组成：

（1）接收 Client 程序（如 Web Browser）的 HTTP 请求消息。

（2）用消息请求标题的 URL 定位资源。

（3）用指定的请求方法访问资源。

（4）创建一个 HTTP 响应消息，消息中包含状态信息和资源文件内容。

（5）发送消息。

（6）关闭与 Client 的连接。

请求的资源一般是 HTML 文件、GIF 图像或者多媒体对象（声音、视频等等），也可能是产生数据的对象，如网关程序（CGI）等，它用请求消息中的数据请求数据库（或其他资源）的信息，将其作为响应的消息体送回 Client。

所以 Web Server 的操作和实现可以详细描述成以下的步骤：

首先 Server 接受来自客户端的连接请求，接收 Client 发出的 HTTP 请求消息。Server 接收请求后，解释请求消息的每个标题；对于受保护的资源，Server 用认证的方法要求 Client 认证（一般用 HTTP 基本认证方法），然后用标题信息访问资源（需要时，首先将相对 URL 转化成实际的 URL）。

请求标题中的方法说明资源访问方法，最常用的方法有 GET、HEAD 和 POST。请求 URL 指定 Server 资源的位置，如果 GET 方法将 HTML 表的内容发送给 Server，则 URL 中还可能含有编码的 name=Value 对。如果是 POST 请求，name=Value 表作为请求消息体而发送，在后面的实例中，使用的是 POST 方式将表单的内容（文件上传）提交给 Server。

然后，Server 检查 URL 形式，形成资源类型。如果是静态的，文件资源就检查扩展名，并形成相应的媒体类型，Server 用该媒体类型填写响应消息中的 Content-Type 标题字段。这是最重要的标题字段，因为它决定 Client 应该如何解释消息。对于浏览器，将决定调用哪个查看程序或帮助应用程序。

接着，Server 创建响应消息，消息中包括：

- 状态行：指示请求成功或其他状态。例如成功的响应以“HTTP/1.1 200 OK”开始；
- 普通标题说明消息的属性，如时间、实体大小等；
- 含与响应有关信息的响应标题，如 Server 标题等；
- 描述资源的实体标题，如 Content-Type 标题等。

最后，读入静态资源作为消息体一起发送给 Client，发送完成，关闭连接。

7.3.2　建立工程项目

在下面要介绍的例子中，要实现一个简单的 Web Server。为什么要介绍这样的一个程序？虽然读者自己开发 Web Server 应用在 Internet 上的情况很少，因为这是一项庞大的工程，并且现在已经有许多现成的 Web 服务器软件，如运行在 Windows NT 下的 IIS 和运行在 Linux 下的 Apache 等，这些服务器软件提供了许多可以使用的开发方法，如 ISAPI、ASP、PHP 等进行功能上的扩展，而且这些软件都非常成熟，稳定性、效率、安全等都很好。但是如果是自己在局域网内部要进行一些网络上的信息交流，而没必要使用一些用于大型系统的 Web Server，小型的 Web Server 不能满足自己的实际需要，那设计自己的 Web Server 还是很必要的。并且随着网络的发展，根据 HTTP 协议设计的浏览器已经形成了一定的标准，只要是符合 HTTP 协议的通信过程和数据内容，可以使用浏览器作为客户端进行连接和传递。但是有读者可能也会认为，使用 FTP 服务不也可以进行资源的共享和信息的交流吗？与 FTP 服务的不同在于使用浏览器与 Web Server 进行资源共享与交流的直观性和方便性超过其他任何一种 Internet 资源服务。

在这个 HTTP 应用中，构造的 Web Server 有以下的功能：

- 能响应浏览器的请求，提供浏览网页（HTML 文件）功能；
- 能让浏览器浏览服务器根目录以下的目录和文件，并能对文件进行下载；
- 能让浏览器上传文件；
- 能够进行身份验证，限制浏览器访问特定的资源。

因为在其他 VB 的书上所讲的实例中很少涉及这方面的内容，所以这个例子还是比较新

颖的，希望读者能够仔细的阅读并掌握其中的原理。当然它的功能不是很多，程序运行的效率和稳定性不十分好，但是能够具有 Web Server 的一些常用的功能。

工程中使用的 VB 控件比较少，Web Server 也是基于 TCP 与客户端连接的。所以在工程中使用了 Winsock 控件进行通信。可能有多个浏览器请求连接或一个浏览器有多个 TCP 请求连接，因此将 Winsock 控件定义成数组控件的形式，Index 为 0 的控件负责侦听来自浏览器的 TCP 连接请求，数组的其他控件负责连接，分别与之进行通信。

该工程一共由 3 个窗体、一个类模块和一个模块组成：

- 窗体 frmServer
- 窗体 frmSet
- 窗体 frmSetDir
- 模块 modBase64
- 模块 modServer
- 类模块 clsClient

其中窗体 frmServer 是程序的主窗体，运行时如图 7-7 所示。

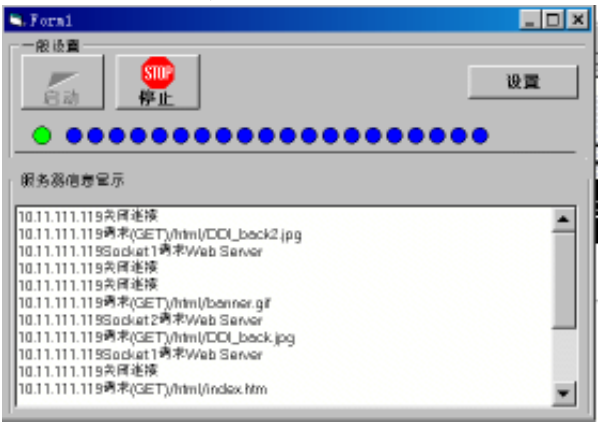


图 7-7　窗体 frmServer

窗体 frmSet 用于设置服务器的选项，运行时显示如图 7-8 所示。

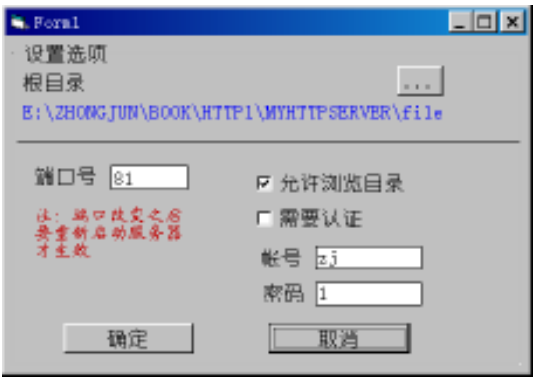


图 7-8　设置窗体

FrmSetDir 用于选择服务器的根目录，运行时显示的窗体如图 7-9 所示。

图 7-9　选择根目录

服务器起始根目录下，缺省为 App.Path & “\file”应有的目录和文件：

- \temp 目录，用于存放 Winsock 通信的数据的临时文件，程序会自动生成。
- \uploadfile 目录，上传和下载的文件放在这个目录中。
- \html 及该目录下的 Index.htm，是服务器提供网页浏览的起始连接网页。
- upload.htm 文件，用于提供上载文件的网页。
- logo.htm 文件，服务器提供的缺省网页。
- nofound.htm 文件，服务器找不到请求资源返回的网页。
- pass.htm 文件，浏览器没有或缺少认证身份时服务器返回的网页。

此外还有 new.gif，note.gif 和 up.gif 三个 gif 图。

7.3.3　代码分析

1. 模块 modServer

该模块定义工程使用到的公有变量和 API 函数。

```
    Public Declare Function GetPrivateProfileString Lib "kernel32" Alias
"GetPrivateProfileStringA" ( _
    ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault
As String, _
    ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As
String) As Long
    Public Declare Function WritePrivateProfileString Lib "kernel32" Alias
"WritePrivateProfileStringA" ( _
    ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpString As
Any, _
    ByVal lpFileName As String) As Long
```

以上定义的两个 API 函数是用于读写配置文件配置信息的，使得在程序退出后能够保存上次的配置信息，下次启动时不必重新进行配置。

```
Public Const MaxConnect = 20        '定义最大的连接数
Public ClientInfo(20) As ClsClient  '存储客户端的信息的类数组
Public ServerInfoStr As String      '存储提示信息
Public RootPath As String           '服务器的根目录
Public ServerPort As Integer            '服务器侦听的端口
Public bDir As Boolean          '是否允许浏览目录
Public NeedPass As Boolean      '是否需要身份验证
Public UserName As String           '验证账号
Public UserPass As String           '验证密码
Public MaxTime As Integer           '最大连接时间
Public PassStr As String            '经编码后的验证字符串
```

以上定义的是一些方便程序设计的全局变量，其中ClientInfo是一个类数组，每个负责与一个TCP客户端连接进行通信，处理来自客户端的请求并响应。在介绍到类模块clsClient时将会详细说明其结构。

2. 模块 modBase64

该模块定义的函数或过程EncodeStr、Base64EncodeByte和Base64Encode是用于base64编码的，这些函数的使用原理在上面的工程DownJet有说明，这里就不在重复了。

3. 窗体 frmServer

该窗体运行时的界面如图 7-7 所示。

以下为窗体的程序代码：

```
'加载的 Winsock 个数
Dim SockNum As Integer
```

以上变量用于存储加载 Winsock 的个数，事实上 Winsock 的个数也可以通过 Winsock 的属性 Count 来取得。

```
Private Sub Form_Load()
Dim RetStr As String * 50
Dim RetInt As Long
'从配置文件中读取根目录的值
RetInt = GetPrivateProfileString("Busyzhong's BigFox Server", "根目录",
App.Path, RetStr, 50, App.Path & "\bigfox.ini")
RootPath = RetStr
If RetInt > 0 Then
   RootPath = Mid(RootPath, 1, RetInt)
End If
'读取服务器侦听端口的值
GetPrivateProfileString "Busyzhong's BigFox Server", "端口", "80", RetStr, 50,
```

```
App.Path & "\bigfox.ini"
    ServerPort = Val(RetStr)
    '读取验证账号
    RetInt = GetPrivateProfileString("Busyzhong's BigFox Server", "用户名", "zhong",
RetStr, 50, App.Path & "\bigfox.ini")
    UserName = RetStr
    If RetInt > 0 Then
        UserName = Mid(UserName, 1, RetInt)
    End If
    '读取验证密码
    RetInt = GetPrivateProfileString("Busyzhong's BigFox Server", "密码", "1",
RetStr, 50, App.Path & "\bigfox.ini")
    UserPass = RetStr
    If RetInt > 0 Then
        UserPass = Mid(UserPass, 1, RetInt)
    End If
    '是否允许目录浏览
    RetInt = GetPrivateProfileString("Busyzhong's BigFox Server", "允许目录浏览
", "1", RetStr, 50, App.Path & "\bigfox.ini")
    If Val(RetStr) = 1 Then
        bDir = True
    Else
        bDir = False
    End If
    '是否需要密码
    RetInt = GetPrivateProfileString("Busyzhong's BigFox Server", "需要认证", "1",
RetStr, 50, App.Path & "\bigfox.ini")
    If Val(RetStr) = 1 Then
        NeedPass = True
    Else
        NeedPass = False
    End If
    RootPath = App.Path & "\file"
    MaxTime = 40
    Call DrawSocket
    End Sub
    Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    '将设置保存到配置文件中
    Dim iii As String
    iii = Str(ServerPort)
```

```
    WritePrivateProfileString "Busyzhong's BigFox Server", "根目录", RootPath,
App.Path & "\bigfox.ini"
    WritePrivateProfileString "Busyzhong's BigFox Server", "端口", iii, App.Path
& "\bigfox.ini"
    WritePrivateProfileString "Busyzhong's BigFox Server", "用户名", UserName,
App.Path & "\bigfox.ini"
    WritePrivateProfileString "Busyzhong's BigFox Server", "密码", UserPass,
App.Path & "\bigfox.ini"
    If bDir = True Then
       iii = "1"
    Else
       iii = "0"
    End If
    WritePrivateProfileString "Busyzhong's BigFox Server", "允许目录浏览", iii,
App.Path & "\bigfox.ini"
    If NeedPass = True Then
       iii = "1"
    Else
       iii = "0"
    End If
    WritePrivateProfileString "Busyzhong's BigFox Server", "需要认证", iii,
App.Path & "\bigfox.ini"
    End Sub
```

以上的代码是在程序运行开始或结束时执行，目的是设置服务器，进行读取或保存配置文件操作。配置文件为 App.Path & "\bigfox.ini"**，下面是一个配置文件的格式：**

```
    [Busyzhong's BigFox Server]
    端口= 81
    用户名=zj
    密码=1
    允许目录浏览=1
    需要认证=0
    根目录=E:\zhongjun\book\http1\myHttpServer
```

使用 API **函数** WritePrivateProfileString**，**GetPrivateProfileString **进行读写。如果要进行程序扩展，可以自己加入更多的设置。**

```
    Private Sub cmdSet_Click()
    frmSet.Show vbModal
    End Sub
    Private Sub cmdStart_Click(Index As Integer)
    If Index = 0 Then
       '服务器启动
```

```
        If Winsock1(0).State = sckClosed Then
            '服务器侦听客户端请求
            Winsock1(0).LocalPort = ServerPort
            Winsock1(0).Listen
        End If
        cmdStart(0).Enabled = False
        cmdStart(1).Enabled = True
        ServerInfoStr = "服务器于" & Format(Time, "hh:mm:ss") & "开始运行" & vbCrLf
& ServerInfoStr
    Else
        '服务器停止
        If Winsock1(0).State <> sckClosed Then
            Winsock1(0).Close
        End If
        cmdStart(1).Enabled = False
        cmdStart(0).Enabled = True
        ServerInfoStr = "服务器于" & Format(Time, "hh:mm:ss") & "停止服务" & vbCrLf
& ServerInfoStr
    End If
    DrawSocket
    txtInfo.Text = ServerInfoStr
    End Sub
```

上面的代码是响应点击启动、停止和设置服务器的。服务器用于侦听客户端所用的是 Index 为 0 的 Winsock 控件，从启动服务器开始，该控件始终处于 Listen 状态，如果有客户端请求，就加载或利用已有的 Winsock 控件进行连接。

```
    Private Sub Winsock1_ConnectionRequest(Index As Integer, ByVal requestID As
Long)
    Dim i As Integer
    Dim WhichSocket As Integer
    If Index <> 0 Then Exit Sub
    For i = 1 To SockNum
    '检查已经加载的 winscok 是否有未连接的
      If Winsock1(i).State = sckClosed Then
         WhichSocket = i
         Exit For
       End If
    Next i
    If WhichSocket = 0 Then
    '连接的客户数大于 socket 个数,加载 socket
       SockNum = SockNum + 1
```

```
        Load Winsock1(SockNum)
        WhichSocket = SockNum
    End If
    Winsock1(WhichSocket).Accept requestID
    Set ClientInfo(WhichSocket) = New ClsClient
    ClientInfo(WhichSocket).WinsockIndex = WhichSocket
    '开始时间
    ClientInfo(WhichSocket).StartTime = Time()
    '清除该 client 的临时文件
    ClientInfo(WhichSocket).ClearData
    '客户端的 ip
    ClientInfo(WhichSocket).ClientIP = Winsock1(WhichSocket).RemoteHostIP
    ServerInfoStr = Winsock1(WhichSocket).RemoteHostIP & "Socket:" & WhichSocket
& "请求 Web Server" & vbCrLf & ServerInfoStr
    txtInfo.Text = ServerInfoStr
    Call DrawSocket
    End Sub
```

以上的 Winsock 事件 ConnectionRequest 是在 Index 为 0 的控件侦听到客户端的请求后触发的，在事件中加入接收客户端连接请求的代码，在这里限定了使用 Winsock 连接的最大个数为 20，当然可以使用更多的连接。此外 20 个连接并不是代表着可以让 20 浏览器同时请求，因为实际情况是一个浏览器可能启动多个 TCP 请求连接下载多个文件。在接受来自客户端的请求后，创建相应的类 clsClient 的对象，以负责每个连接请求。

```
    Private Sub Winsock1_DataArrival(Index As Integer, ByVal bytesTotal As Long)
    Dim strData As String
    Dim Bytedata() As Byte
    '接收来自客户端的数据
    Winsock1(Index).GetData Bytedata(), vbByte
    '类对象保存数据
    ClientInfo(Index).SaveData bytesTotal, Bytedata()
    ClientInfo(Index).ReceiveData = ClientInfo(Index).ReceiveData & strData
    '类对象处理数据
    ClientInfo(Index).HandleData
    txtInfo.Text = ServerInfoStr
    End Sub
```

Winsock 的 DataArrival 事件用于接收来自客户端的请求信息。处理请求都在类模块 clsClient 中有定义。具体见模块 clsClient 的说明。

```
    Private Sub Winsock1_Error(Index As Integer, ByVal Number As Integer,
Description As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile
As String, ByVal HelpContext As Long, CancelDisplay As Boolean)
    Winsock1(Index).Close
```

```
End Sub
Private Sub Winsock1_SendComplete(Index As Integer)
'发送数据完毕，关闭 TCP 连接
Winsock1(Index).Close
ServerInfoStr = ClientInfo(Index).ClientIP & "关闭连接" & vbCrLf & ServerInfoStr
txtInfo.Text = ServerInfoStr
Call DrawSocket
End Sub
```

以上代码是在 Winsock 发送完响应的数据之后关闭连接的，每次客户端请求都要重新进行连接，在响应完成之后关闭。不过在响应标题字段 Connection 中要指明为 close，这样设计是符合 HTTP 1.0 规范的，程序设计上比较容易实现，但代价是服务器对于浏览器的每个请求都要重新启动 Winsock，效率性变差。

```
'根据 Winsock 的 State 属性描绘其工作状态
Private Sub DrawSocket()
Dim i As Integer
Dim intR As Integer
intR = 80
Pic.FillStyle = vbFSSolid
If Winsock1(0).State <> sckClosed Then
    Pic.FillColor = vbGreen
Else
    Pic.FillColor = vbYellow
End If
Pic.Circle (1 * 6 * intR / 2, 3 * intR / 2), intR * 5 / 4, vbBlack
For i = 1 To MaxConnect
    If i > Winsock1.Count - 1 Then
        Pic.FillColor = vbBlue
    Else
        If Winsock1(i).State <> sckClosed Then
            Pic.FillColor = vbRed
        Else
            Pic.FillColor = vbBlue
        End If
    End If
    Pic.Circle ((i + 1.5) * 6 * intR / 2, 3 * intR / 2), intR, vbBlack
Next i
End Sub
```

上面的程序只为监控 Winsock 的状态提供一些直观的图像。

4. 窗体 frmSet

以下的程序代码用于设置服务器的选项，如起始根目录、是否允许浏览目录、身份验证的账号和密码、端口等。具体程序代码比较简单，读者可以自己理解。

```
'该窗体用于设置服务器选项
Private Sub cmdCancel_Click()
Unload Me
End Sub
Private Sub cmdOk_Click()
If chkDir.Value = 1 Then
    bDir = True
Else
    bDir = False
End If
If chkPass.Value = 1 Then
    NeedPass = True
Else
    NeedPass = False
End If
ServerPort = Val(txtPort.Text)
Dim Str1 As String
Str1 = txtUser.Text & ":" & txtPsw.Text
PassStr = EncodeStr(Str1)
UserName = txtUser.Text
UserPass = txtPsw.Text
Unload Me
End Sub

Private Sub CmdSetDir_Click()
frmSetDir.Show vbModal
End Sub

Private Sub Form_Load()
lblRoot.Caption = RootPath
If bDir = True Then
    chkDir.Value = 1
End If
If NeedPass = True Then
    chkPass.Value = 1
End If
```

```
txtPort.Text = ServerPort
txtUser.Text = UserName
txtPsw.Text = UserPass
End Sub

Private Sub txtPort_LostFocus()
If IsNumeric(txtPort) = False Then
   MsgBox "你填写的服务器端口须为整数!!"
   txtPort.SetFocus
End If
End Sub
```

5. **窗体** frmSetDir

该窗体用于设置服务器的根目录。

```
Private Sub cmdOk_Click()
RootPath = Dir1.Path
If Right(RootPath, 1) = "\" Then
   RootPath = Mid(RootPath, 1, Len(RootPath) - 1)
End If
frmSet.lblRoot.Caption = RootPath
Unload Me
End Sub

Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub
```

6. **类模块** clsClient

类模块负责接受来自客户端浏览器的请求并响应，是整个程序的核心部分。

首先接收客户端的请求消息，前面已经介绍过，在窗体 frmServer **的** Winsock **在接受请求后，接收来自客户端浏览器的请求数据，在** Winsock **的** DataArrival **事件中调用类对象的** SaveData **保存数据。将请求字符串信息写入到** temp **目录中，文件名与** Winsock **的** Index **或** clsClient **类对象数组索引相对应，定义为“** 1.tmp **”、“** 2.tmp **”等。以下是保存在***.tmp **文件中来自** IE **浏览器的一个请求信息：**

```
GET /html/banner.gif HTTP/1.1
Accept: */*
Referer: http://10.11.111.119:81/html/index.htm
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
If-None-Match: "0967d7838c5bf1:ae1"
```

```
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
Host: 10.11.111.119:81
Connection: Keep-Alive
```

有些读者可能会问，如果只是要从浏览器的请求字符串提取信息定位资源用以响应请求，直接把数据存在一个字符串变量不就可以了吗？这种想法是正确的，所以我们同时定义了一个字符串变量 ReceiveData，用函数 StrConv 将接收的数据（因为数据以 Byte 形式从 Winsock 缓冲区接收的）转换成字符串的形式赋值给该变量，可以从该变量中直接得到相应的请求信息。在程序中还实现了通过浏览器（主要是针对 IE 4.0 以上浏览器）实现文件上载的功能。这时浏览器上传的数据（通常是二进制数据）和请求信息混在了一起，必须以 Byte 类型的数组接收，再以二进制的形式保存数据到临时文件中，接着从临时文件中提取相应的上载文件数据出来写到另外的文件中，实现文件上传功能，显然数据不能以字符串的形式用变量保存，也不能以文本文件的方式对数据进行读写，这时创建临时文件就是一种很好的解决方法。

```
Option Explicit
'记录客户端请求的类
Public ClientIP As String           '客户端的 IP 地址
Public WinsockIndex As Integer      'Winsock 的序号
Public RequestUrl As String         '请求的地址(资源)
Public ReceiveData As String        '接收到的数据(字符串类型)
Public StartTime As Date            '连接的开始时间
Private SendDataB() As Byte         '发送给客户端的数据
```

以上是定义类变量，其中 RequestUrl 是从请求字符串中分析得到用于定位服务器上的资源的。

```
Public Sub SaveData(ByteNum As Long, Bytedata() As Byte)
Dim Tfile As String
Dim Fnum As Integer
'如果没有临时目录，创建一个
If Dir(RootPath & "\temp", vbDirectory) = "" Then
   MkDir (RootPath & "\temp")
End If
Tfile = RootPath & "\temp\" & WinsockIndex & ".tmp"
Fnum = FreeFile()
'向临时文件中加入二进制的数据
Open Tfile For Binary As #Fnum
If LOF(Fnum) > 0 Then
   Seek #Fnum, LOF(Fnum) + 1
End If
Put #Fnum, , Bytedata()
Close #Fnum
'将接到的数据转换成 String 类型的数据
```

```
    ReceiveData = ReceiveData & StrConv(Bytedata(), vbUnicode)
    End Sub
    Public Sub ClearData()
    Dim Tfile As String
    ReceiveData = ""
    Tfile = RootPath & "\temp\" & WinsockIndex & ".tmp"
    If Dir(Tfile) <> "" Then
        '删除临时文件
        Kill (RootPath & "\temp\" & WinsockIndex & ".tmp")
    End If
    End Sub
```

以上的 SaveData **用来保存客户端浏览器请求时发送过来的数据，并以二进制的形式保存在临时文件和字符串变量** ReceiveData **中。用** ClearData **过程清除临时文件。**

```
    '处理接收到的数据,从中提取信息
    Public Sub HandleData()
    Dim pos1, pos2 As Integer
    Dim StrTemp As String
    If ReceiveData = "" Or RequestUrl <> "" Then Exit Sub
    '分析取得的字符串,得到浏览器请求的信息
    If InStr(1, ReceiveData, "GET", vbTextCompare) > 0 Then
        '浏览器以 GET 方式请求响应
        pos1 = InStr(1, ReceiveData, "GET", vbTextCompare)
        pos2 = InStr(pos1 + 4, ReceiveData, " ")
        If pos2 > pos1 Then
            '取得资源定位的字符串（相对 URL）
            RequestUrl = Mid(ReceiveData, pos1 + 4, pos2 - pos1 - 4)
            ServerInfoStr = ClientIP & "请求(GET)" & RequestUrl & vbCrLf &
ServerInfoStr
            '响应浏览器的请求
            Call ResponseUrl
            ReceiveData = ""
        End If
    ElseIf InStr(1, ReceiveData, "POST", vbTextCompare) > 0 Then
        '浏览器以 POST 方式请求响应
        pos1 = InStr(1, ReceiveData, "POST", vbTextCompare)
        pos2 = InStr(pos1 + 5, ReceiveData, " ")
        If pos2 > pos1 Then
            '取得资源定位的字符串（相对 URL）
            RequestUrl = Mid(ReceiveData, pos1 + 5, pos2 - pos1 - 5)
            ServerInfoStr = ClientIP & "请求(POST)" & RequestUrl & ServerInfoStr
```

```
            '处理浏览器提交数据
            Call HandlePost
            '响应请求
            Call ResponseUrl
            ReceiveData = ""
        End If
    End If
    End Sub
```

以上 HandleData 是从客户端的请求字符串中提取相应的信息的。前面说过浏览器请求的方法一共有 GET、POST、HEAD 等，这里只是考虑 GET 和 POST 这两种情况，GET 方式是浏览器请求一个资源文件的，除了请求的标题字段外的，后面往往不带其他数据。而 POST 方式除了标题字段外往往是带有其他数据的，上传文件的功能浏览器就是利用我们提供的网页，使用 POST 方式上传文件的。所以针对这两种方式使用不同的处理方法。处理 GET 方式得到用于资源定位的字符串（存放在变量 RequestUrl）后，直接调用 ResponseUrl 响应请求。处理 POST 方式得到资源定位的字符串后，调用 HandlePost 处理上传数据，再调用 ResponseUrl 响应请求。

```
    '根据 url 响应请求的过程
    Private Sub ResponseUrl()
    If RequestUrl = "" Then
        Exit Sub
    End If
    Dim FileContent As String '文件数据(字符型)
    Dim ByteContent() As Byte '文件数据(二进制字节型)
    Dim FileSize As Long '文件大小
    Dim FileName As String '返回的文件名
    Dim FileType As String '文件类型
    Dim FileTime As String '文件的创建或修改时间

    If Right(RequestUrl, 1) = "/" Then
        '请求目录浏览
        If bDir = False Or RequestUrl = "/" Then
            '禁止目录浏览
            FileName = "\"
            FileType = "text/html"
        Else
            '允许目录浏览
            FileType = "text/html"
            FileName = RootPath & RequestUrl
        End If
    Else
```

```
        '请求返回文件
        FileName = RootPath & RequestUrl
        '根据文件扩展名设置文件类型
        If InStr(1, RequestUrl, ".htm", vbTextCompare) > 0 Or _
    InStr(1, RequestUrl, ".html", vbTextCompare) > 0 Then
            FileType = "text/html"
        ElseIf InStr(1, RequestUrl, ".jpg", vbTextCompare) > 0 Or _
    InStr(1, RequestUrl, ".jpeg", vbTextCompare) > 0 Then
            FileType = "image/jpeg"
        ElseIf InStr(1, RequestUrl, "gif", vbTextCompare) > 0 Then
            FileType = "image/gif"
        Else
            FileType = "application/" & Right(RequestUrl, 3)
        End If
    End If

    If FileName = "\" Then
        '显示进站画面
        FileContent = AddLogoHTML()
        FileContent = AddHeader("200", FileType, , FileTime) & vbCrLf & FileContent
        frmServer.Winsock1(WinsockIndex).SendData FileContent
        Exit Sub
    End If

    '如果需要用户输入密码及账号进行认证
    If NeedPass = True Then
            Dim bPass As Boolean
            '验证密码
            bPass = CheckPass()
            If bPass = False Then
                If ReadTxtFile(RootPath & "\pass.htm", FileContent, FileSize,
FileTime) = True Then
                    FileContent = AddHeader("401", FileType, FileSize, FileTime) &
vbCrLf & FileContent
                    frmServer.Winsock1(WinsockIndex).SendData FileContent
                Else
                    '这个信息具体要编写 404 响应信息
                    frmServer.Winsock1(WinsockIndex).SendData AddHeader("404")
                End If
                Exit Sub
```

```
            End If
        End If

        If InStr(1, FileType, "text") > 0 Then
            '处理文本数据类型
            If Right(FileName, 1) <> "/" Then
                If ReadTxtFile(FileName, FileContent, FileSize, FileTime) = True Then
                    FileContent=AddHeader("200",FileType,FileSize,FileTime)&vbCrLf
& FileContent
                    frmServer.Winsock1(WinsockIndex).SendData FileContent
                Else
                    '这个信息具体要编写 404 响应信息
                    frmServer.Winsock1(WinsockIndex).SendData AddHeader("404")
                End If
            Else
                '目录浏览
                FileContent = myDir(FileName)
                FileContent   =   AddHeader("200",   FileType,   ,   Format(Now(),
"ddd,yyyy-mm-dd hh:mm:ss")) _
        & FileContent
                frmServer.Winsock1(WinsockIndex).SendData FileContent
            End If
        ElseIf InStr(1, FileType, "image", vbTextCompare) > 0 Then
            '二进制数据文件
            If ReadBinFile(FileName, ByteContent, FileSize, FileTime) = True Then
                frmServer.Winsock1(WinsockIndex).SendData AddHeader("200", FileType,
FileSize, FileTime)
                frmServer.Winsock1(WinsockIndex).SendData SendDataB()
            Else
                '这个信息具体要编写 404 响应信息
                frmServer.Winsock1(WinsockIndex).SendData AddHeader("404")
            End If
        Else
            If ReadBinFile(FileName, ByteContent, FileSize, FileTime) = True Then
                frmServer.Winsock1(WinsockIndex).SendData AddHeader("200", FileType,
FileSize, FileTime)
                frmServer.Winsock1(WinsockIndex).SendData SendDataB()
            Else
                '这个信息具体要编写 404 响应信息
                frmServer.Winsock1(WinsockIndex).SendData AddHeader("404")
```

```
    End If
End If
End Sub
```

以上的过程是根据 RequestUrl 向客户端浏览器发送响应消息的。程序流程简单介绍如下：首先判断 RequestUrl 的最后一个字符是否“/”，如果是浏览器请求的是目录浏览。注意直接在浏览器中敲入：http://10.11.111.119:81（假设 Web Server 地址为 10.11.111.119，使用的端口为 81），那么浏览器的请求字符串是“\”，这时可以定位到根目录浏览或者响应起始文件可定为 Index.htm 等。如果不允许目录浏览，资源文件名为“\”，在下一步中将响应进站页面（在函数 AddLogoHTML 中定义），如果允许浏览目录，则根据 RequestUrl 确定进行映射资源为本地硬盘实际路径。如果请求的是文件，则先映射相对 URL 得到资源在本地硬盘的路径，根据文件的后缀名设定返回的媒体类型。这里只是判断了三种媒体类型：text/html、image/gif 和 image/jpeg，其他文件都按照 application 类型处理。

接着如果设置要进行身份认证，使用函数 CheckPass 判断请求字符串中是否含有身份验证的用户名和密码。通过身份验证则往下执行程序，否则发送 401 响应，通知客户端没有身份验证或验证不通过。

然后根据资源文件的媒体类型构造响应消息。根据三种媒体类型 text、image、其他构造，其中目录浏览的要使用函数 myDir，将指定目录下的文件和文件夹全部搜索出来，生成标准的 HTML 页面，HTML 页面中给每个文件或文件夹提供链接，可以通过浏览器进行下一次的请求。如图 7-10 所示。

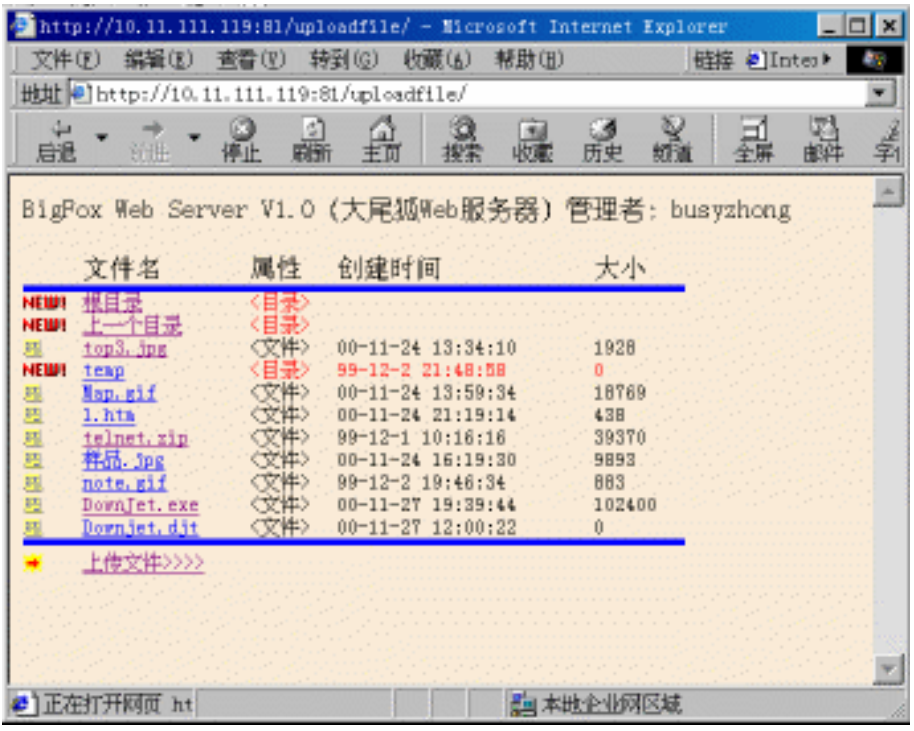


图 7-10　浏览目录

```
Private Function myDir(FileName As String) As String
Dim RetStr As String
```

```
    Dim Tstr As String
    '返回目录及文件
    RetStr = "<html><head>"
    RetStr = RetStr & "<STYLE type=text/css>"
    RetStr = RetStr & ".td1 {FONT-FAMILY: 宋体, Arial, Times New Roman; _
    FONT-SIZE: 9pt; FONT-WEIGHT: strong;COLOR: red}" & vbCrLf
    RetStr = RetStr & ".td2 {FONT-FAMILY: 宋体, Arial, Times New Roman; _
    FONT-SIZE: 9pt; FONT-WEIGHT: normal}" & vbCrLf
    RetStr = RetStr & ".A:link {FONT-SIZE: 9pt; TEXT-DECORATION: none}" & vbCrLf
    RetStr = RetStr & ".A:visited {COLOR: #0000ff; FONT-SIZE: 9pt; _
    TEXT-DECORATION: none}" & vbCrLf
    RetStr = RetStr & ".A:active {COLOR: #0000ff; FONT-SIZE: 9pt; TEXT-DECORATION:
none}" & vbCrLf
    RetStr = RetStr & ".A:hover {COLOR: red; TEXT-DECORATION: none}" & vbCrLf
    RetStr = RetStr & "</STYLE></head><body bgcolor=AntiqueWhite>" & vbCrLf
    RetStr = RetStr & "<p><font size=3 face=幼圆>BigFox Web Server V1.0（大尾狐
Web 服务器）_
    管理者：busyzhong</font></p>"
    RetStr = RetStr & "<table border=0 width=500 cellPadding=0.5 cellSpacing=0>" & vbCrLf
    RetStr = RetStr & "<tr><td></td><td>文件名</td><td>属性</td><td>创建时间
</td><td>大小</td>"
    RetStr = RetStr & "<tr bgcolor=blue><td height=5 colspan=5><td></tr>"
    Tstr = Dir(FileName, vbDirectory)
    If Len(FileName) <> Len(RootPath) + 1 Then
       RetStr = RetStr & "<tr class=td1><td><img src=/./new.gif></td><td><a
href=/>根目录</a></td>"_
     & vbCrLf
       RetStr = RetStr & "<td><目录></td>" & vbCrLf
       RetStr = RetStr & "<tr class=td1><td><img src=/./new.gif></td><td><a href="
& RequestUrl _
     & "../>上一个目录</a></td>" & vbCrLf
       RetStr = RetStr & "<td><目录></td>" & vbCrLf
    End If
    While Tstr <> ""
       '文件名
       If Tstr <> "." And Tstr <> ".." Then
          If (GetAttr(FileName & Tstr) And vbDirectory) = vbDirectory Then
             '判断是否目录
             RetStr = RetStr & "<tr class=td1><td><img src=/./new.gif></td><td><a
```

```
href=""" _
    & RequestUrl & Tstr & "/"">" & Tstr & "</a></td>" & vbCrLf
            RetStr = RetStr & "<td><目录></td>" & vbCrLf
        Else
            '文件
            RetStr     =     RetStr     &     "<tr     class=td2><td><img
src=/./note.gif></td><td><a href=""" _
    & RequestUrl & Tstr & """>" & Tstr & "</a></td>"
            RetStr = RetStr & "<td><文件></td>" & vbCrLf
        End If
        RetStr = RetStr & "<td>" & FileDateTime(FileName & Tstr) & "</td>" &
vbCrLf
        RetStr = RetStr & "<td>" & FileLen(FileName & Tstr) & "</td></tr>" &
vbCrLf
    End If
    Tstr = Dir()
Wend
RetStr = RetStr & "<tr bgcolor=blue><td height=5 colspan=5><td></tr>"
RetStr = RetStr & "<tr><td height=5 colspan=5><td></tr>"
RetStr = RetStr & "<tr><td><img src=/up.gif></td><td class=td1 colspan=4> _
<a href=/upload.htm>上传文件>>>></a><td></tr>"
RetStr = RetStr & "</table></body></html>"
myDir = RetStr
End Function
```

以上代码为将指定路径的文件及文件夹全部搜索出来，生成标准 HTML 页面的形式，关于 HTML 的语法，请读者参考其他书籍。

```
'增加响应头信息
Private Function AddHeader(ResponseCode As String, Optional MimeType As String, _
Optional FileSize As Long, Optional FileTime As String) As String
Dim Tstr As String
Dim ResponseLine As String
Select Case ResponseCode
    Case "200"
        '200 请求正确响应标题
        ResponseLine = "HTTP/1.1 200 OK"
        Tstr = ResponseLine
        Tstr = Tstr & vbCrLf & "Server: BigFox Server1.0"
        Tstr = Tstr & vbCrLf & "Connection: close"
        Tstr = Tstr & vbCrLf & "Date: " & Format(Now, "ddd, yyyy-mm-dd hh:mm:ss
```

```
GMT")
            Tstr = Tstr & vbCrLf & "Content-Type: " & MimeType
            Tstr = Tstr & vbCrLf & "Accept-Ranges: bytes"
            Tstr = Tstr & vbCrLf & "Last-Modified: " & FileTime
            If FileSize <> 0 Then
                Tstr = Tstr & vbCrLf & "Content-Length: " & FileSize
            End If
            Tstr = Tstr & vbCrLf & vbCrLf
        Case "404"
            '404 文件未找到响应标题
            ResponseLine = "HTTP/1.1 404 Not Found"
            Tstr = ResponseLine
            Tstr = Tstr & vbCrLf & "Server: BigFox Server1.0"
            Tstr = Tstr & vbCrLf & "Connection: close"
            Tstr = Tstr & vbCrLf & "Date: " & Format(Now, "ddd, yyyy-mm-dd hh:mm:ss
GMT")
            Tstr = Tstr & vbCrLf & "Content-Type: text/html"
            Tstr = Tstr & vbCrLf & vbCrLf
            Dim ErrorInfo As String
            If ReadTxtFile(RootPath & "\Nofound.htm", ErrorInfo) = True Then
                Tstr = Tstr & ErrorInfo
            Else
                Tstr = Tstr & "Sorry!请求的文件不存在" & vbCrLf
            End If
        Case "401"
            '401 需要身份验证响应标题
            ResponseLine = "HTTP/1.1 401 Unauthorized"
            Tstr = ResponseLine
            Tstr = Tstr & vbCrLf & "Server: BigFox Server1.0"
            Tstr = Tstr & vbCrLf & "Connection: close"
            Tstr = Tstr & vbCrLf & "Date: " & Format(Now, "ddd, yyyy-mm-dd hh:mm:ss GMT")
            Tstr = Tstr & vbCrLf & "WWW-authenticate: basic realm=""Busyzhong's
BigFox Server1.0"""
            Tstr = Tstr & vbCrLf & "Content-Type: " & MimeType
            If FileSize <> 0 Then
                Tstr = Tstr & vbCrLf & "Content-Length: " & FileSize
            End If
            Tstr = Tstr & vbCrLf & vbCrLf
    End Select
```

```
    AddHeader = Tstr
    End Function
```

以上函数是根据响应码构造响应的标题字段信息，其中主要针对三个常用状态：正确响应（200 响应）、未找到请求资源（404 响应）和请求须认证（401 响应）。

其中在给客户端发送 401 需要进行用户身份验证后，客户端浏览器（IE 4.0 以上）会弹出输入身份验证的对话框要求输入账号和密码。如图 7-11 所示。

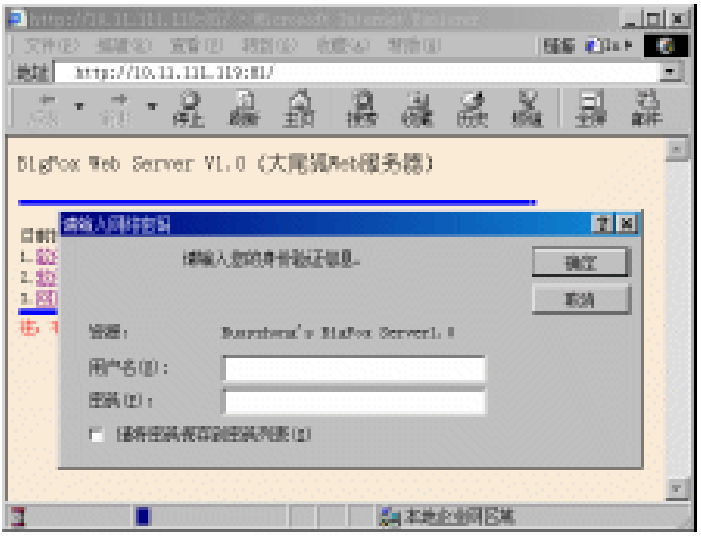


图 7-11　要求用户输入身份验证信息

```
    '读入文本文件
    Public Function ReadTxtFile(FileName As String, FileContent As String, _
    Optional FileSize As Long, Optional FileTime As String) As Boolean
    If Len(Dir(FileName)) = 0 Then
        ReadTxtFile = False
        Exit Function
    End If
    FileTime = Format(FileDateTime(FileName), "ddd, yyyy-mm-dd hh:mm:ss")
    Dim Fnum As Integer
    Fnum = FreeFile()
    Open FileName For Input As #Fnum
    FileSize = LOF(Fnum)
    FileContent = StrConv(InputB(LOF(Fnum), #Fnum), vbUnicode)
    Close Fnum
    ReadTxtFile = True
    End Function

    '读入二进制文件
    Public Function ReadBinFile(FileName As String, ByteContent() As Byte, FileSize
As Long, _
```

```
    FileTime As String) As Boolean
    If Len(Dir(FileName)) = 0 Then
        ReadBinFile = False
        Exit Function
    End If
    FileTime = Format(FileDateTime(FileName), "ddd, yyyy-mm-dd hh:mm:ss")
    Dim Fnum As Integer
    Fnum = FreeFile()
    Open FileName For Binary As #Fnum
    FileSize = LOF(Fnum)
    ReDim SendDataB(FileSize)
    Get #Fnum, , SendDataB()
    Close Fnum
    ReadBinFile = True
    End Function
```

以上的函数是用于读取文本文件（ReadTxtFile）或二进制文件（ReadBinFile）的，成功则返回 True，否则返回 False，同时作为参数，返回文件的修改时间、文件大小和文件数据等。在响应客户端的请求，这两个函数取得资源文件的数据用于响应请求。

```
    '处理 POST 数据内容的文件
    Public Sub HandlePost()
    Dim pos1, pos2 As Integer
    Dim Boundary As String
    Dim ContentType As String
    Dim Tstr As String
    While Boundary = "" And Abs(DateDiff("s", Time(), StartTime)) < MaxTime
        DoEvents: DoEvents: DoEvents
        If pos1 = 0 Then
            pos1 = InStr(1, ReceiveData, "Content-Type", vbTextCompare)
        End If
        If pos1 > 0 Then
            pos2 = InStr(pos1 + 1, ReceiveData, vbCrLf, vbTextCompare)
        End If
        If pos1 > 0 And pos2 > 0 Then
            Tstr = Mid(ReceiveData, pos1, pos2 - pos1)
            If InStr(1, Tstr, "multipart") > 0 Then
                pos1 = InStr(1, Tstr, "boundary=")
                '将 boundary 的内容取出来
                If pos1 > 0 Then
                    pos2 = InStr(pos1, Tstr, " ")
                    If pos2 = 0 Then
```

```
                    pos2 = InStr(pos1, Tstr, vbCrLf)
                End If
                If pos2 > 0 Then
                    Boundary = Mid(Tstr, pos1 + Len("boundary="), pos2 - pos1 -
Len("boundary="))
                Else
                    Boundary = Mid(Tstr, pos1 + Len("boundary="))
                End If
                If Boundary <> "" Then
                    Boundary = Replace(Boundary, """", "")
                End If
            End If
        End If
    End If
Wend
'等待数据传送完毕
Do While True And Abs(DateDiff("s", Time(), StartTime)) < MaxTime
    DoEvents: DoEvents: DoEvents
    If InStr(1, ReceiveData, Boundary & "--") > 0 Then
        Exit Do
    End If
Loop
'数据传送完毕,从保存的临时文件中取出 Browser 提交的信息进行分析
Dim Tfile As String
Dim Fnum As Integer
Dim i As Integer
Dim myByte As Byte
Dim OutArray() As Byte
Dim LineStr As String
Dim SectData As String
Dim PostName As String, PostResume As String
Dim StPos As Long, EndPos As Long
Dim FileExtN As String
Dim TmpStr As String
Tfile = RootPath & "\temp\" & WinsockIndex & ".tmp"
Fnum = FreeFile()
If Dir(Tfile) <> "" Then
    Open Tfile For Input As #Fnum
    While Not EOF(Fnum)
        Line Input #Fnum, LineStr
```

```
        LineStr = LineStr & vbCrLf
        '遇到分隔字符串
        If LineStr = vbCrLf Then
            If SectData <> "" Then
                If InStr(1, SectData, "newfile") > 0 Then
                '得出当前文件的位置
                    If InStr(1, SectData, "filename=") > 0 Then
                        '下面是提取文件名
                        pos1 = InStr(1, SectData, "filename=")
                        pos2 = InStr(pos1 + 10, SectData, """")
                        If pos2 = 0 Then
                            pos2 = InStr(pos1 + 10, SectData, vbCrLf)
                        End If
                        If pos2 > pos1 Then
                            TmpStr = Mid(SectData, pos1 + 10, pos2 - pos1 - 10)
                        Else
                            TmpStr = Mid(SectData, pos1 + 10)
                        End If
                        pos1 = InStrRev(TmpStr, "\")
                        If pos1 > 0 Then
                            TmpStr = Mid(TmpStr, pos1 + 1)
                        End If
                    End If
                    StPos = Seek(Fnum)
                    EndPos = LOF(Fnum) - Len("--" & Boundary & "--") - 3
                End If
            End If
        End If

        If InStr(1, LineStr, "--" & Boundary) > 0 Then
            If SectData <> "" Then
                If InStr(1, SectData, "txtname") > 0 Then
                    pos1 = InStr(1, SectData, "txtname")
                    pos2 = InStr(pos1, SectData, vbCrLf & vbCrLf)
                    If pos2 > 0 Then
                        PostName = Mid(SectData, pos2 + 4)
                        PostName = Left(PostName, Len(PostName) - 2)
                    End If
                End If
                If InStr(1, SectData, "txtresume") > 0 Then
```

```
                pos1 = InStr(1, SectData, "txtresume")
                '找出数据与字段的分界
                pos2 = InStr(pos1, SectData, vbCrLf & vbCrLf)
                If pos2 > 0 Then
                    PostResume = Mid(SectData, pos2 + 4)
                    '删除 vbcrlf
                    PostResume = Left(PostResume, Len(PostResume) - 2)
                End If
            End If
        End If
        SectData = ""
    Else
        SectData = SectData & LineStr
    End If
  Wend
  Close (Fnum)
  '将上载的文件写到 Uploadfile 目录中
  If EndPos > StPos Then
    ReDim OutArray(EndPos - StPos - 2) As Byte
    Open Tfile For Binary As #Fnum
      Seek #Fnum, StPos
      Get #Fnum, , OutArray()
    Close (Fnum)
    Open RootPath & "\uploadfile\" & TmpStr For Binary As #Fnum
      Put #Fnum, , OutArray()
    Close (Fnum)
  End If
End If
End Sub
```

在以上的过程 HandlePost **中，用于处理客户端浏览器以** POST **方式提交的数据。在整个工程中只有软件上载的功能使用** POST **方式提交文件数据。在了解** HandlePost **前，首先了解软件上载的工作原理。其中提交数据的表单是由服务器提供的，具体的** HTML **如下：**

```
<html><head>
<STYLE type=text/css>
.td1 {FONT-FAMILY: 宋体, Arial, Times New Roman; FONT-SIZE: 9pt; FONT-WEIGHT:
strong;COLOR: red}
.td2 {FONT-FAMILY: 宋体, Arial, Times New Roman; FONT-SIZE: 9pt; FONT-WEIGHT: normal}
.A:link {FONT-SIZE: 9pt; TEXT-DECORATION: none}
.A:visited {COLOR: #0000ff; FONT-SIZE: 9pt; TEXT-DECORATION: none}
```

```
    .A:active {COLOR: #0000ff; FONT-SIZE: 9pt; TEXT-DECORATION: none}
    .A:hover {COLOR: red; TEXT-DECORATION: none}
    </STYLE></head><body bgcolor=AntiqueWhite>
    <form     method="post"     name="form1"     enctype="multipart/form-data"
action="uploadfile/">
    <p><font size=3 face=幼圆>
    BigFox Web Server V1.0（大尾狐 Web 服务器）管理者：busyzhong</font></p>
    <table border=0 width=500 cellPadding=0.5 cellSpacing=0>
    <tr bgcolor=blue><td height=5><td></tr>
    <tr><td>
                <div align="center">文件上传</div>
                </td></tr>
    <tr>
                <td>软件名称:
    <input type="text" name=txtname></td></tr>
    <tr>
        <td>软件说明:
        <textarea name="txtresume" cols="35" rows="5"></textarea></td></tr>
    <tr><td>
                <div align="right">
        <input type="submit" name="Submit" value="上传">
        <input type="reset" name="Reset" value="重填">
        </div>
        </td></tr>
    <tr><td><input type="file" name="newfile"><br></td></tr>
    <tr bgcolor=blue><td height=5><td></tr>
    <tr><td class=td2><p> </td></tr>
    <tr><td class=td1>注：请不要上载超过 1M 的文件</td></tr>
    </table>
    </form>
    </body></html>
```

除去其中控制结构和显示格式的 CSS **和** HTML**，以下的几个要素是必须的。**

- **在** Form **元素的语法中，**EncType **表明提交数据的格式，**Method **表明提交的方法**(Get /Post)**。在** IE 4.0 **及以后的版本中，都支持“** multipart/form-data **”这种格式，相应的** Method **方法必须是** Post**，表明要上载文件到服务器。**
- Form **中必须有一个** input **元素，而且** input **的属性** type="file"**。（说明：如果要上载多个文件，有多个** input **元素就可以了，但至少有一个有效文件，否则会出错。）系统会自动产生一个文本区域和一个“** browse... **” 或“浏览... ”按钮，也可以直接在文本区域内输入文件路径名称，或按“** browse... **”或“浏览... ”按钮，从文件对话框中选择一个文件。**

- Form 中必须有一个 submit 按钮，即 input 的属性 type="submit"，此按钮即为上传按钮。或者在其他相关事件中调用此 Form 的 Submit 方法。但两种方法实际上本质相同，只不过用方法调用还可以在上载前加上其他处理代码，如数据的有效性检查等。
- 如图 7-12 所示的是用于上载文件的页面显示。

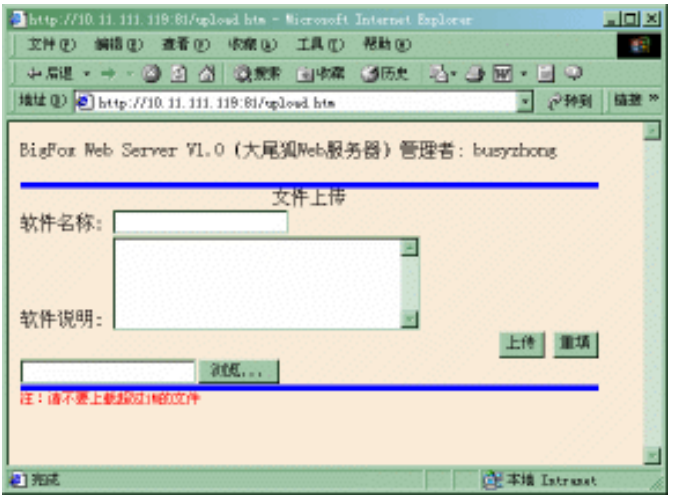


图 7-12　文件上载页面

如果使用的是 IE 4.0 以上的浏览器，上载后的数据结构如下所示：

```
    POST /uploadfile/ HTTP/1.1
    Accept:           application/vnd.ms-excel,           application/msword,
application/vnd.ms-powerpoint, */*
    Referer: http://10.11.111.119:81/upload.htm
    Accept-Language: zh-cn
    Content-Type:                                          multipart/form-data;
boundary=---------------------------7cf1901ff80
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
    Host: 10.11.111.119:81
    Content-Length: 1386
    Connection: Keep-Alive

    -----------------------------7cf1901ff80
    Content-Disposition: form-data; name="txtname"

    图片文件
    -----------------------------7cf1901ff80
    Content-Disposition: form-data; name="txtresume"

    图片
    -----------------------------7cf1901ff80
```

```
    Content-Disposition: form-data; name="Submit"

    上传
    -----------------------------7cf1901ff80
    Content-Disposition:              form-data;              name="newfile";
filename="C:\WINDOWS\Desktop\note.gif"
    Content-Type: image/gif

    (上传文件的二进制数据)
    -----------------------------7cf1901ff80--
```

与其他的浏览器请求一样，服务器接到的数据先是一段请求字符串，请求字符串中包含了请求方法、请求的资源、请求的数据内容等标题字段。其中较重要的标题字段为 Content-Type，它的值‘ multipart/form-data ’表示下面提交的数据是一个多部件（有关详细概念见 E-mail 协议一章）的结构，属性 boundary 及其值给出了各部件之间的分隔字符串。每个部件又分为两部分：描述部件（实体）部分和实体数据部分。HandlePost 过程就是要取得上传文件实体部分的文件数据。

此外，提交数据的格式与 MIME 比较类似，可参看“ E-mail 协议 ”一章的描述。

```
    '验证身份
    Private Function CheckPass() As Boolean
    If InStr(1, ReceiveData, "Authorization:") > 0 And _
       InStr(1, ReceiveData, PassStr) > 0 Then
           CheckPass = True
    Else
        CheckPass = False
    End If
    End Function
```

以上函数检查接收的数据中是否存在标题字段 Authorization 及含有编码字符串的信息。如身份认证账号为 zj 密码为 1 时，浏览器发送的在 Authorization 标题字段中为“ zj:1 ”经过 base64 编码后的字符串“ emo6MQ== ”。如以下字符串为浏览器发送过来的带有身份验证标题字段的请求字符串：

```
    GET /upload.htm HTTP/1.1
    Accept:     image/gif,     image/x-xbitmap,     image/jpeg,     image/pjpeg,
application/vnd.ms-excel, application/msword, */*
    Referer: http://10.11.111.119:81/
    Accept-Language: zh-cn
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
    Host: 10.11.111.119:81
    Connection: Keep-Alive
    Authorization: Basic emo6MQ==
```

CheckPass 验证方法基本能够满足用户需求了，但是如果用户输入的是 wxpzj 和密码 1，该函数还是认为是正确的。读者可以自己修改一下这个函数，使之更完善。

```
    '生成进站时的 html 字符串
    Private Function AddLogoHTML() As String
    Dim RetStr1 As String
    RetStr1 = RetStr1 & "<html><head><STYLE type=text/css>"
    RetStr1 = RetStr1 & ".td1 {FONT-FAMILY: 宋体, Arial, Times New Roman;
    FONT-SIZE: 9pt; FONT-WEIGHT: strong;COLOR: red}"
    RetStr1 = RetStr1 & ".td2 {FONT-FAMILY: 宋体, Arial, Times New Roman;
    FONT-SIZE: 9pt; FONT-WEIGHT: normal}"
    RetStr1 = RetStr1 & ".A:link {FONT-SIZE: 9pt; TEXT-DECORATION: none}"
    RetStr1 = RetStr1 & ".A:visited {COLOR: #0000ff; FONT-SIZE: 9pt; TEXT-DECORATION:
none}"
    RetStr1 = RetStr1 & ".A:active {COLOR: #0000ff; FONT-SIZE: 9pt; TEXT-DECORATION:
none}"
    RetStr1 = RetStr1 & ".A:hover {COLOR: red; TEXT-DECORATION: none}"
    RetStr1 = RetStr1 & "</STYLE></head><body bgcolor=AntiqueWhite>"
    RetStr1 = RetStr1 & "<p><font size=3 face=幼圆>BigFox Web Server V1.0
    (大尾狐 Web 服务器)</font></p>"
    RetStr1 = RetStr1 & "<table border=0 width=500 cellPadding=1 cellSpacing=1>"
    RetStr1 = RetStr1 & "<tr bgcolor=blue><td height=5><td></tr>"
    RetStr1 = RetStr1 & "<tr><td><div align=center><b class=td1>
    Welcome to BigFox Server</b></div></td></tr>"
    RetStr1 = RetStr1 & "<tr><td class=td2>目前提供的服务：</td></tr>"
    RetStr1 = RetStr1 & "<tr><td class=td2>1.<a href=/upload.htm>软件/文件上载
</a></td></tr>"
    RetStr1 = RetStr1 & "<tr><td class=td2>2.<a href=/uploadfile/>软件/文件下载
</a></td></tr>"
    RetStr1 = RetStr1 & "<tr><td class=td2>3.<a href=/html/index.htm>网页浏览
</a></td></tr>"
    RetStr1 = RetStr1 & "<tr bgcolor=blue><td height=5><td></tr><tr><td
class=td2><p></td></tr>"
    RetStr1 = RetStr1 & "<tr><td class=td1>注：有时对中文的连接不太支持</td></tr>"
    RetStr1 = RetStr1 & "<tr><td class=td1><div align=right class=td1>"
    If NeedPass = True Then
        RetStr1 = RetStr1 & "(需要密码)"
    Else
        RetStr1 = RetStr1 & "(不需要密码)"
```

```
    End If
    RetStr1 = RetStr1 & "管理员：busyzhong</div></td></tr></table></body></html>"
    AddLogoHTML = RetStr1
    End Function
```

以上代码的作用是生成 Web Server 的起始页面，在浏览器显示效果如图 7-13 所示。

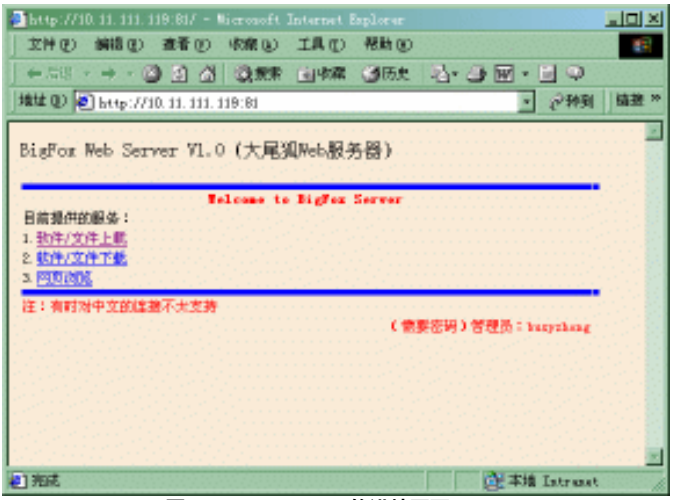


图 7-13　Web Server 的进站页面

7.4　网站下载程序高级开发

7.4.1　实例介绍

可能许多读者都使用过 Teleport 或是 WebZip 一类的下载网站内容的软件了。WebDown 就是要实现这样的一个功能。程序运行时如图 7-14 所示。它所能实现的功能有：

- 给定起始下载的网页或网址，下载网站内容；
- 可以将下载信息保存，下次打开接着下载。

图 7-14　WebDown 的运行界面

首先加入下载的起始地址，如：http://www.sohu.com，接着点击“加入下载”按扭，这时程序会弹出一个保存的对话框，要求输入保存下载信息的文件名及选择保存路径，如选择的是 C:\My Documents\down.web（文件名是“*.web”），则下载的文件保存在 C:\My Documents\down 目录下（如果该目录不存在，程序会创建目录）。保存下载信息的文件为 C:\My Documents\down.web。

在本工程中，一共有一个窗体和一个类模块。

- 窗体 frmMain
- 类模块 clsNetGet

下载的原理是根据起始 HTML 页面的连接决定下一个下载的文件。如果链接是 HTML 文件，下载后分析提取链接信息加入到链接的队列中。如果链接是二进制文件如 JPG、EXE、ZIP、GIF 文件等，则下载就可以了。在下载之前，还需判断是否已经下载过了，有时一些页面中互相引用，如果不加以判断是否已经下载过，会造成无休止的下载。

7.4.2　WinInet HTTP API 实现文件下载的使用方法

在该实例中，主要应用 HTTP API 函数取得指定的 URL 文件。步骤一般分为以下几步：

（1）打开 Internet 会话

在进行 Internet 连接之前，不管是采用 FTP，还是使用其他协议如 HTTP、GOPHER 等，首先使用 API 函数 InternetOpen（）打开 Internet 会话。如果是直接连接，具体做法如下：

```
    hInternetSession=    InternetOpen(sUserAgent,    INTERNET_OPEN_TYPE_DIRECT,
vbNullString,
    vbNullString, 0)
```

（sUserAgent 可以是标示该应用程序的任何字符串）

如果是通过代理服务器连接：

```
    hInternetSession  =  InternetOpen(scUserAgent,  INTERNET_OPEN_TYPE_PROXY,
sProxy, vbNullString, 0)
```

（其中 sProxy 为存放代理服务器地址的字符串）

如果连接成功，将返回非 0 的句柄 hInternetSession，此返回值可以作为建立 HTTP 连接时的必需的参数之一。该函数的使用在程序中只使用一次，但是如果要改变连接使用的参数如改变代理服务器的使用，则需要使用 InternetCloseHandle 进行关闭（见关闭 Internet 连接），关闭后重新打开，使用不同的参数设置。

（2）打开 URL 请求

当使用 InternetOpen 函数进行 Interent 连接之后，通过 InternetOpenUrl 函数可以获得指定的 URL 资源。如：

```
    hUrlFile = InternetOpenUrl(hInternetSession, sUrl, vbNullString, 0,
    INTERNET_FLAG_EXISITING_CONNECT, 0)
```

其中 hInternetSession 为 InternetOpen 返回的有效句柄（handle），sUrl 为 URL 字符串，如：http://www.sohu.com/index.htm。如果函数成功，则返回一个句柄 hUrlFile，以下的函数

中要使用该句柄。在读取资源文件后，需要使用 InternetCloseHandle 进行关闭（见关闭 Internet 连接），关闭后重新打开，使用不同的参数设置。

（3）读取 URL 文件数据

在成功打开 URL 请求后，使用 InternetReadFile 函数获取数据。在工程中，我们使用了两个不同的参数类型函数，声明调用的 InternetReadFile，一个为 InternetReadHTML，用于读取基于文本的 URL 文件数据，另一个为 InternetReadImg，用于读取基于二进制的 URL 文件数据。之所以这样用，主要是由于在 VB 中使用 String 类型的变量通过 API 时取得缓冲区的二进制数据会有变化，这一点读者会有所体会的。下面先看看两个函数的声明。

函数 InternetReadHTML 的声明：

```
    Private Declare Function InternetReadHTML Lib "wininet.dll" Alias
"InternetReadFile" _
          (ByVal hFile As Long, ByVal lpBuffer As String, ByVal
dwNumberOfBytesToRead As Long, _
           lNumberOfBytesRead As Long) As Integer
```

函数InternetReadImg的声明：

```
    Private Declare Function InternetReadImg Lib "wininet.dll" Alias
"InternetReadFile" _
          (ByVal hFile As Long, lpBuffer As Any, ByVal dwNumberOfBytesToRead As
Long, _
           lNumberOfBytesRead As Long) As Integer
```

比较这两个函数，不同在于参数 lpBuffer 在 InternetReadHTML 中是 ByVal 传递的 String 类型，在 InternetReadImg 中是 ByRef 传递的 Any 类型，该参数是用于提供缓冲区以通过 API 函数接收 URL 数据。其他参数解释如下：

hFile 是 InternetOpenUrl 打开 URL 后的句柄；

dwNumberOfBytesToRead 是接收数据的缓冲区 lpBuffer 的大小；

lNumberOfBytesRead 是调用一次 InternetReadFile 函数读取的字节数；

函数调用成功，返回值为 True，否则为 False。

为了能让读者对该参数的结构了解得更清楚，给出该函数的 C 原型：

```
    BOOL InternetReadFile( IN HINTERNET hFile, IN LPVOID lpBuffer,
    IN DWORD dwNumberOfBytesToRead, OUT LPDWORD lpNumberOfBytesRead)
```

在使用结束后，应使用 InternetCloseHandle 关闭 URL 句柄。

（4）关闭 Internet 连接或 URL 句柄

在结束使用 InternetOpenUrl 后，应关闭其句柄，在程序结束或使用 InternetOpen 重新建立 Internet 连接时，应关闭连接句柄。这两种情况下都是使用 InternetCloseHandle。其声明为：

```
    Private Declare Function InternetCloseHandle Lib "wininet.dll" (ByVal hInet
As Long) As Integer
```

使用如下：

```
    InternetCloseHandle(hInternetSession)
```

7.4.3　代码分析

1. 窗体 frmMain

该窗体是整个工程唯一的窗体，在程序运行时如图 7-14 所示，窗体中的控件主要由三个按扭，用于“加入下载”、“停止下载”或“继续下载”、“打开下载”功能。有一个加入起始地址的文本框。两个 ListBox 控件（lstReferences 和 lstImages），这两个控件用于存储链接的文本和二进制文件的相对 URL 地址。此外还有一个 ListView 控件，用于显示已下载的文件。其他控件请读者自己浏览工程文件。

以下为该窗体的代码：

```
Option Explicit
Private Const scBlankStr = ""                '定义空字符串常量
Dim objNetGet       As New ClsNetGet     '定义类对象
Dim colReferences   As New Collection    '定义存储链接（href）的集合
Dim colImages       As New Collection    '定义存储链接（src）的集合
Dim vReference     As Variant
Dim sHost           As String            '请求的 Server 地址
Dim SaveFilePath    As String            '保存下载文件的文件夹
Dim SaveFileName   As String         '保存下载信息的文件
Dim sStartUrl       As String            '起始 url
Dim bCancel        As Boolean        '取消下载的标志
Dim ReadNum         As Long              '下载文件记数
```

上面的代码定义变量和类对象。其中 objNetGet 为类 ClsNetGet 的实例对象，用于下载指定的 URL，分析 HTML 文件提取链接等，colReferences 和 colImages 是用于存储分析 HTML 文本时提取的链接（URL），在每次分析前都要清空其内容。分析后根据链接的属性加入到相应的 ListBox 中。其他变量是与下载和保存有关的变量。

```
'该按扭数组事件用于开始新下载，继续和停止下载，打开下载
Private Sub cmdStart_Click(Index As Integer)
On Error GoTo err1:
If Index = 0 Then
    '选择保存下载文件的目录和保存下载信息的文件
    CDlg.Filter = "*.web"
    CDlg.FileName = "*.web"
    CDlg.ShowSave
    '取消则退出该过程
    If Right(CDlg.FileName, 4) <> ".web" Then GoTo err1
    '保存下载信息的文件
    SaveFileName = CDlg.FileName
```

```
    '从保存下载信息的文件中取得保存下载文件的目录
    SaveFilePath = Left(CDlg.FileName, Len(CDlg.FileName) - 4)
    If Dir(SaveFilePath, vbDirectory) = "" Then
        '如果目录不存在，创建目录
        MkDir SaveFilePath
    End If
    '清除文件链接的 ListBox
    lstReferences.Clear
    lstImages.Clear
    '清除已下载文件的 ListView
    LView.ListItems.Clear
    sStartUrl = txtUrl.Text
    '显示下载设置信息
    txtInfo.Text = "下载起始 URL：" & sStartUrl & vbCrLf
    txtInfo.Text = txtInfo.Text & "保存路径：   " & SaveFilePath & vbCrLf
    txtInfo.Text = txtInfo.Text & "信息文件：   " & SaveFileName & vbCrLf
    txtInfo.Text = txtInfo.Text & "下载文件类型：*.EXE *.HTM *.HTML *.ZIP *.GIF * JPG"
    '从起始网页开始下载文件
    bCancel = False
    If StartGet(sStartUrl) = True Then
        cmdStart(1).Caption = "停止下载"
        cmdStart(1).Enabled = True
        cmdStart(0).Enabled = False
        cmdStart(2).Enabled = False
        ReadNum = 0
        Do While bCancel = False
            DoEvents: DoEvents: DoEvents: DoEvents: DoEvents
            '根据 ListBox 中的链接继续下载
            If ContinueGet() = False Then
                '下载完毕，退出
                cmdStart(0).Enabled = True
                cmdStart(1).Enabled = False
                cmdStart(2).Enabled = True
                Exit Sub
            End If
        Loop
    Else
        MsgBox "地址出错，无法开始下载"
        cmdStart(0).Enabled = True
```

```
        cmdStart(1).Enabled = False
        cmdStart(2).Enabled = True
    End If
ElseIf Index = 1 Then
    If cmdStart(1).Caption = "停止下载" Then
        '设置取消下载的标志
        bCancel = True
        cmdStart(1).Enabled = False
        cmdStart(1).Caption = "继续下载"
        '写入日志文件
    Else
        '继续下载
        bCancel = False
        cmdStart(0).Enabled = False
        cmdStart(2).Enabled = False
        cmdStart(1).Caption = "停止下载"
        cmdStart(1).Enabled = True
        While bCancel = False
            DoEvents: DoEvents: DoEvents: DoEvents: DoEvents
            If ContinueGet() = False Then
                cmdStart(0).Enabled = True
                cmdStart(1).Enabled = False
                cmdStart(2).Enabled = True
                Exit Sub
            End If
        Wend
    End If
Else
    CDlg.Filter = "*.web"
    CDlg.FileName = "*.web"
    CDlg.ShowOpen
    '选择下载信息文件开始下载
    If Right(CDlg.FileName, 4) <> ".web" Then GoTo err1
    lstReferences.Clear
    lstImages.Clear
    LView.ListItems.Clear
    ReadNum = 0
    SaveFileName = CDlg.FileName
    If ReadDownInfo(CDlg.FileName) = True Then
        '读取保存下载信息的文件开始下载
```

```
            bCancel = False
            cmdStart(0).Enabled = False
            cmdStart(1).Caption = "停止下载"
            cmdStart(2).Enabled = False
            cmdStart(1).Enabled = True
            While bCancel = False
                DoEvents: DoEvents: DoEvents: DoEvents: DoEvents
                If ContinueGet() = False Then
                    cmdStart(0).Enabled = True
                    cmdStart(1).Enabled = False
                    cmdStart(2).Enabled = True
                    Exit Sub
                End If
            Wend
        Else
            '读取文件出错
            cmdStart(0).Enabled = True
            cmdStart(1).Enabled = False
            cmdStart(2).Enabled = True
            MsgBox "该文件不是保存下载信息的文件！"
        End If
    End If
    err1:
    End Sub
```

上面的代码是用以响应控件数组 cmdStart 的点击事件的。

CmdStart(0)为“开始下载”，首先在 txturl 中填入下载的起始地址，如 http://www.sohu.com/index.htm，点击该按扭，接着在弹出的保存文件的对话框中选择保存下载信息的文件名，从中也可以得到下载文件保存的目录，如选择 C://My Document/down.web，则保存下载文件的目录为 C://My Document/down，注意如果目录不存在时应创建目录，否则后面会出错。接着清空 lstReferences、lstImages 和 Lview 等控件以及进行必要的变量设置（如保存下载文件的目录，下载信息文件的路径等）。接下来使用函数 StartGet 下载起始网页，成功后再调用函数 ContinueGet 继续下载，如果出错或下载完毕都会退出该事件过程。要注意的是，在使用 While…Wend 循环调用 ContinueGet 函数时，最好加上几个 DoEvents，否则程序会在下载的过程中响应不了其他事件。

CmdStart(1)为“停止下载”或“继续下载”，视当前是否正在下载情况而定。如果是“停止下载”，则设置 bCancel 变量为 True，在函数 ContinueGet 中，如果判断到 bCancel=true，则退出下载。如果是“继续下载”，则调用函数 ContinueGet 继续下载。

CmdStart(2)为“打开下载”，先从弹出的对话框中选择保存下载信息的文件，接着使用 ReadDownInfo 函数读取以前下载停止时所保存的信息和设置，然后调用函数 ContinueGet 继续下载。

```
'窗体加载事件用于初始化
Private Sub Form_Load()
'初始化类对象
objNetGet.Init
'设置显示状态信息为 StatusBar 的第一个 Panel
objNetGet.SetStatusWindow = SBar1.Panels(1)
'lstReferences.Sorted = False
'lstImages.Sorted = False
End Sub

'该事件用于释放资源
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
objNetGet.Term
Set objNetGet = Nothing
Unload Me
End
End Sub
```

上面的代码在 frmMain **的** Form_load **事件时进行初始化，在** Form_QueryUnload **事件中释放资源。注意要在设计窗体控件时将** lstReferences **和** lstImages **的** Sorted **属性设置为** False **（在程序运行时这个属性是只读的），这样加入到这两个** ListBox **中的链接是放在最后的，下载时先从第一个开始下载。**

```
'该函数用于开始一个新下载起始页面
Public Function StartGet(ByVal strUrl As String) As Boolean
Dim iEndPos As Integer
strUrl = Replace(strUrl, "\", "/")
iEndPos = InStr(8, strUrl, "/")
'从下载的起始页面中取得主机名
If CBool(iEndPos) Then
   sHost = Mid$(strUrl, 1, iEndPos - 1)
Else
   sHost = Mid$(strUrl, 1, Len(strUrl))
End If
'取得相对的 url 资源目录名
iEndPos = InStrRev(strUrl, "/")
If CBool(iEndPos) Then
   sStartUrl = Mid$(strUrl, 1, iEndPos - 1)
Else
   sStartUrl = Mid$(strUrl, 1, Len(strUrl))
End If
'清空保存链接的集合
```

```
    Set colReferences = Nothing
    Set colImages = Nothing
    If Len(strUrl) Then
        '开始下载起始页面
        If objNetGet.ReadUrlHTML(strUrl, SaveFilePath & "\index.htm") = True Then
            LView.ListItems.Add , , "index.htm", "html", "html"
            '从下载页面中提取链接，加入到 lstReferences 和 lstImages 中
            If objNetGet.ParseHTML("HREF=", colReferences) Then AddReferences ""
            If objNetGet.ParseHTML(" SRC=", colImages) Then AddImages ""
        Else
            '请求 URL 出错
            StartGet = False
            Exit Function
        End If
    End If
    StartGet = True
    End Function
```

上面的代码用于下载起始页面并提取起始页面的链接加到相应的 ListBox 中。SStartUrl 是保存起始页面的目录，如果起始 URL 为 http://10.11.111.119/index.htm，则 sStartUrl 的值为 http://10.11.111.119，如果起始 URL 为 http://10.11.111.119/html/index.htm，则 sStartUrl 为 http://10.11.111.119/html，以后接着下载的网页地址都是相对于 sStartUrl 的 URL。

```
    '该函数用于下载 lstReferences 和 lstImages 的 URL
    Public Function ContinueGet() As Boolean
    Dim i As Integer
    Dim sReUrl As String
    bCancel = False
    ContinueGet = True
    '下载二进制文件，如图片文件
    While lstImages.ListCount > 0 And bCancel = False
        sReUrl = lstImages.List(0)
        '如果链接文件中已经包含了起始的 url 目录，取得相对 url
        If InStr(1, sReUrl, sStartUrl) = 1 Then
            sReUrl = Mid(sReUrl, Len(sStartUrl) + 1)
        End If
        '排除不在起始 url 目录的链接，判别是否文件已经下载
        If InStr(1, sReUrl, "http://") = 0 And FileIsGet(sReUrl) = False Then
            '检查是否存在存储文件的目录
            CheckDir "\" & sReUrl
            '下载二进制文件
            If objNetGet.ReadUrlImg(sStartUrl & "/" & sReUrl, SaveFilePath & "\"
```

```
& sReUrl) = True Then
            '下载成功，加入到 listview 中，下载文件记数+1
            LView.ListItems.Add , , sReUrl, "img", "img"
            ReadNum = ReadNum + 1
            SBar1.Panels(2).Text = "已读取文件 " & ReadNum & " 个"
        End If
    End If
    '清除链接
    lstImages.RemoveItem (0)
    DoEvents
  Wend
  If lstReferences.ListCount > 0 And bCancel = False Then
    sReUrl = lstReferences.List(0)
    '如果链接文件中已经包含了起始的 url 目录，取得相对 url
    If InStr(1, sReUrl, sStartUrl) = 1 Then
      sReUrl = Mid(sReUrl, Len(sStartUrl) + 1)
    End If
    '排除不在起始 url 目录的链接，判别是否文件已经下载
    If InStr(1, sReUrl, "http://") = 0 And FileIsGet(sReUrl) = False Then
      '检查是否存在存储文件的目录
      CheckDir "\" & sReUrl
      '下载文本的 html 文件
      If objNetGet.ReadUrlHTML(sStartUrl & "/" & sReUrl, SaveFilePath & "\"
& sReUrl) = True Then
        '下载成功，加入到 listview 中，下载文件记数+1
        LView.ListItems.Add , , sReUrl, "html", "html"
        ReadNum = ReadNum + 1
        SBar1.Panels(2).Text = "已读取文件  " & ReadNum & " 个"
        '清空存储链接的集合
        Set colReferences = Nothing
        Set colImages = Nothing
        '从下载的页面中提取链接，加入到两个 ListBox 中
        If objNetGet.ParseHTML("HREF=", colReferences) Then
          AddReferences sReUrl
        End If
        If objNetGet.ParseHTML(" SRC=", colImages) Then
          AddImages sReUrl
        End If
      End If
    End If
```

```
        DoEvents
        If lstReferences.ListCount > 0 Then
            '清除 LstReferences 中的链接
            lstReferences.RemoveItem (0)
        End If
        DoEvents
    Else
        '下载完成，返回 False
        ContinueGet = False
    End If
    If bCancel = True Or ContinueGet = False Then
        SaveDownInfo
        cmdStart(0).Enabled = True
        cmdStart(1).Enabled = True
        cmdStart(2).Enabled = True
    End If
    End Function
```

上面的代码用于继续下载已经取得的链接。从 HTML 页面中分析得到的链接存在两个 ListBox 中（lstReferences 和 lstImages），lstReferences 中存放 HTML 页面的链接，lstImages 中存放相应的二进制文件的链接，一般只有 HTML 页面能够提取新的链接，所以在上面的函数 ContinueGet 中首先下载 lstImages 中的链接，这时不会产生新的链接，将 lstImages 中的链接全部下载完成后，再下载 lstReference 的处在最前面的链接。由于下载 HTML 页面进行分析后可能会得到一些新的链接，因此一时不可能将所有的 lstReferences 总的链接全部下载完成，并且每下载一个 HTML 页面可能会有相应的二进制文件链接，因此每次调用该函数时，只下载一个 lstReferencs 中的 HTML 链接。如果全部的 HTML 下载完成，lstReferences 为空，表明全部下载结束了。这时 ContinueGet 返回 False（否则返回 True）。此外也可以在下载的过程中点击“停止下载”，将标志停止下载的变量 bCancel 设置为 False，也可以退出下载。

```
    '该过程将集合 colReferences 中加入相应的 listBox 中
    Private Sub AddReferences(sReUrl As String)
    Dim tStr As String
    Dim Pos1 As Integer
    '取得链接所在的 HTML 页面的 Url 目录
    tStr = ""
    Pos1 = InStrRev(sReUrl, "/")
    If Pos1 > 0 Then
        tStr = Left(sReUrl, Pos1)
    End If
    For Each vReference In colReferences
        If Len(vReference) > 4 And (UCase(Right(vReference, 4)) = ".EXE" Or
UCase(Right(vReference, 5))
```

```
    = ".ZIP" Or UCase(Right(vReference, 4)) = ".JPG" Or
    UCase(Right(vReference, 4)) = ".GIF") Then
            '二进制文件加入到 lstImages 中
            lstImages.AddItem tStr & vReference
        ElseIf Len(vReference) > 4 And (UCase(Right(vReference, 4)) = ".HTM" Or
    UCase(Right(vReference, 5)) = ".HTML") Then
            '文本文件加入到 lstReferences 中
            lstReferences.AddItem tStr & vReference, lstReferences.ListCount
        End If
    Next vReference
    End Sub

    '该过程用于将集合 colImages 加入到相应的 ListBox 中
    Private Sub AddImages(sReUrl As String)
    Dim tStr As String
    Dim Pos1 As Integer
    tStr = ""
    '取得链接所在的 HTML 页面的 Url 目录
    Pos1 = InStrRev(sReUrl, "/")
    If Pos1 > 0 Then
        tStr = Left(sReUrl, Pos1)
    End If
    For Each vReference In colImages
        If Len(vReference) > 4 And (UCase(Right(vReference, 4)) = ".EXE" Or
    UCase(Right(vReference, 5)) = ".ZIP" Or UCase(Right(vReference, 4)) = ".JPG"
Or
    UCase(Right(vReference, 4)) = ".GIF") Then
            '二进制文件加入到 lstImages 中
            lstImages.AddItem tStr & vReference
        ElseIf Len(vReference) > 4 And (UCase(Right(vReference, 4)) = ".HTM" Or
    UCase(Right(vReference, 5)) = ".HTML") Then
            '文本文件加入到 lstReferences 中
            lstReferences.AddItem tStr & vReference
        End If
    Next vReference
    End Sub
```

以上两个过程 AddReferences **和** AddImages **将集合** colReferences **和** colImages **的内容加入到** LstReferences **和** LstImages **中。在加入之前判断集合的文件类型。如果是** HTML **文件，加入到** lstReferences**，二进制文件类型如***.EXE、*.ZIP、*.JPG、*.GIF **等加入到** lstImages **中。集合** colReferences **和** colImages **的内容是从下载的** HTML **中分析得到的。后面在类** clsNetGet

中有介绍怎样提取 HTML 页面中的链接。

```
'该过程用于检查保存文件的目录是否存在，如不存在，创建该目录
Function CheckDir(sFName As String) As Boolean
On Error Resume Next
Dim tStr, tStr2 As String
Dim Pos1 As Integer
tStr = Replace(sFName, "/", "\")
tStr2 = tStr
Pos1 = InStr(1, tStr2, "\")
While Pos1 > 0
    tStr = Left(tStr2, Pos1 - 1)
    If Dir(SaveFilePath & tStr, vbDirectory) = "" Then
        '目录不存在，创建目录
        MkDir (SaveFilePath & tStr)
    End If
    Pos1 = InStr(Pos1 + 1, tStr2, "\")
Wend
End Function
```

上面的过程用于检查保存下载文件的目录是否存在，如果不存在，则创建相应的目录。链接是同一个网站内的网页一般是以相对目录的形式进行引用的，如下载网页 Index.hml 后，有如下链接：

```
<A herf="down/zj2.htm">下载网址</A>
```

接着请求的网页 zj2.htm，必须存放在相对 Index.html 文件所在的目录的子目录 down 之下，在下载后点击 Index.html 中的该链接才会正确地链接到相应的文件之下，因此在下载 zj2.htm 保存之前，必须存在目录 down，如果不存在，则需要先创建，否则在下载存盘时读写文件会出错。

此外，还必须注意存在这样的情况，可能相对的子目录有好几层。如上面的链接如果是这样的：

```
<A herf="htmlfile/down/zj2.htm">下载网址</A>
```

而还不存在 htmlfile 和 Down 这两个目录，因此先要创建两个目录，因此函数 CheckDir 用循环的形式判断要存储的文件的目录是否都存在，如果不存在则创建相应的目录。

```
'该函数用于检查要下载的文件是否存在
Private Function FileIsGet(ByVal sFile As String) As Boolean
On Error Resume Next
If Len(sFile) > 0 And Dir(SaveFilePath & "\" & sFile) <> "" Then
    '文件已下载，函数返回 true
    FileIsGet = True
Else
    '文件未下载，函数返回 false
    FileIsGet = False
```

```
End If
End Function
```

上面的函数用于检查文件是否已经被下载，在一个页面中可能会有多个相同的链接，或几个页面中有相同的文件链接。如果每次都不加以判断地重新下载，会造成效率低下，因此必须判断链接文件是否已经下载了。因为下载后根据链接存储在本地硬盘上的一个目录下的文件，如果链接相同，存储的路径也是相同的（除非重新下载时改变保存目录）。因此可以先判断硬盘上是否已经存在该文件，如果存在，则不用再次下载了。上面的函数判断指定的文件是否，若存在则返回 True，否则返回 False。

```
'该函数用于保存下载信息
Private Function SaveDownInfo() As Boolean
Dim i As Long
Dim iFnum As Integer
iFnum = FreeFile()
Open SaveFileName For Output As #iFnum
'写入标识文件的字符串
Print #iFnum, "WEBDOWN INFORMATION"
'写入存盘路径
Print #iFnum, "[SaveFilePath]"
Print #iFnum, SaveFilePath
'写入下载起始地址
Print #iFnum, "[StartUrl]"
Print #iFnum, sStartUrl
'写入未下载的 HTML 文件
Print #iFnum, "[HTML]"
For i = 0 To lstReferences.ListCount - 1
   Print #iFnum, lstReferences.List(i)
Next i
'写入未下载的二进制文件
Print #iFnum, "[IMAGE]"
For i = 0 To lstImages.ListCount - 1
   Print #iFnum, lstImages.List(i)
Next i
'写入已经下载的文件
Print #iFnum, "[DOWNED]"
For i = 1 To LView.ListItems.Count
   Print #iFnum, LView.ListItems(i).Text
Next i
Close iFnum
End Function
```

上面函数用于保存下载的信息到指定的文件中。保存的文件分为 6 个部分，首先是文件

的标识字符串“WEBDOWN INFORMATION”，然后是存盘文件路径、下载的起始URL、未下载的HTML文件、未下载的二进制文件以及已经下载的文件部分，每个部分前面都有相应的标识字符串，用以标识接下来的文本是属于哪个部分。如下面是一个完整的下载信息，保存文件的内容：

```
WEBDOWN INFORMATION
[SaveFilePath]
C:\My Documents\down1
[StartUrl]
http://www.zju.edu.cn/
[HTML]
http://www.zju.edu.cn/chinese/zupress/ywb/ywb4-0.htm
chinese/Chinese.htm
eexplain.htm
suggestion.htm
[IMAGE]
home.gif
[DOWNED]
index.htm
zhejiang_university.gif
standXB75.jpg
Etitle02.gif
zhukz.gif
eng04.htm
```

保存文件的函数是在点击“取消下载”按扭才执行的，因此如果在下载的过程中因意外情况退出程序，下载信息就没有保存，下次重新打开保存文件继续下载时就会产生错误。下面的函数ReadDownInfo是用于读取指定的下载信息文件，接着下载文件的。首先判断保存文件的标识字符串是否存在，如果不存在则不保存下载信息的文件。接着将存盘文件路径、下载的起始URL、未下载的HTML文件、未下载的二进制文件以及已经下载的文件读取出来，设置相应的变量和控件。

```
'该函数用于读取下载信息文件以恢复下载设置
Private Function ReadDownInfo(sFName As String) As Boolean
Dim i As Long
Dim iFnum As Integer
Dim LineStr As String
Dim ActionStr As String
iFnum = FreeFile()
Open sFName For Input As #iFnum
'判断是否下载的信息文件
Line Input #iFnum, LineStr
If LineStr <> "WEBDOWN INFORMATION" Then
```

```
        ReadDownInfo = False
        Close iFnum
        Exit Function
    End If
    Do While Not EOF(iFnum)
    Line Input #iFnum, LineStr
    If LineStr = "[SaveFilePath]" Or LineStr = "[StartUrl]" Or LineStr = "[HTML]"
Or _
        LineStr = "[IMAGE]" Or LineStr = "[DOWNED]" Then
        ActionStr = LineStr
    Else
        Select Case ActionStr
            Case "[SaveFilePath]"
                '保存下载文件路径
                SaveFilePath = LineStr
            Case "[StartUrl]"
                '起始 URL
                sStartUrl = LineStr
            Case "[HTML]"
                '未下载的 HTML 文件链接
                lstReferences.AddItem LineStr
            Case "[IMAGE]"
                '未下载的二进制文件链接
                lstImages.AddItem LineStr
            Case "[DOWNED]"
                '已下载的文件
                If LCase(Right(LineStr, 4)) = ".htm" Or LCase(Right(LineStr, 5))
= ".html" Then
                    LView.ListItems.Add , , LineStr, "html", "html"
                Else
                    LView.ListItems.Add , , LineStr, "img", "img"
                End If
        End Select
    End If
    Loop
    Close iFnum
    '显示下载信息
    ReadNum = LView.ListItems.Count
    txtInfo.Text = "下载起始 URL：" & sStartUrl & vbCrLf
    txtInfo.Text = txtInfo.Text & "保存路径：   " & SaveFilePath & vbCrLf
```

```
    txtInfo.Text = txtInfo.Text & "信息文件：   " & sFName & vbCrLf
    txtInfo.Text = txtInfo.Text & "下载文件类型：*.EXE *.HTM *.HTML *.ZIP *.GIF *
JPG"
    ReadDownInfo = True
    End Function
```

2. **类模块** clsNetGet

类模块 clsNetGet **是整个工程实例的核心，读者一定要掌握其中的原理和** WinInetHTTP **函数的使用方法。该类模块的功能主要有以下几个方面：**

- **从指定的** URL **下载** HTML **文件和二进制文件并存盘；**
- **根据指定的链接起始字符串分析** HTML **页面中的** URL **链接；**
- **显示下载信息。**

以下是 clsNetGet **的程序代码：**

```
    Option Explicit
    '打开并初始化 Internet 连接
    Private Declare Function InternetOpen Lib "wininet.dll" Alias "InternetOpenA"
_
          (ByVal lpszAgent As String, ByVal dwAccessType As Long, ByVal
lpszProxyName As String, _
          ByVal lpszProxyBypass As String, ByVal dwFlags As Long) As Long

    '关闭 Internet 连接或 URL 句柄
    Private Declare Function InternetCloseHandle Lib "wininet.dll" (ByVal hInet
As Long) As Integer

    '打开一个 URL，获得句柄
    Private Declare Function InternetOpenUrl Lib "wininet.dll" Alias
"InternetOpenUrlA" _
          (ByVal hInternetSession As Long, ByVal lpszUrl As String, ByVal
lpszHeaders As String, _
          ByVal dwHeadersLength As Long, ByVal dwFlags As Long, ByVal dwContext
As Long)
      As Long

    '从打开的 URL 中读取数据(用于读取文本文件)
    Private Declare Function InternetReadHTML Lib "wininet.dll" Alias
"InternetReadFile" _
          (ByVal hFile As Long, ByVal lpBuffer As String, ByVal
dwNumberOfBytesToRead As Long, _
```

```
        lNumberOfBytesRead As Long) As Integer

    '从打开的 URL 中读取数据(用于读取二进制文件)
    Private Declare Function InternetReadImg Lib "wininet.dll" Alias
"InternetReadFile" _
        (ByVal hFile As Long, lpBuffer As Any, ByVal dwNumberOfBytesToRead As
Long, _
        lNumberOfBytesRead As Long) As Integer

    Private Const INTERNET_OPEN_TYPE_PRECONFIG = 0   '使用缺省的配置
    Private Const INTERNET_FLAG_EXISITING_CONNECT = &H20000000
    Private Const INTERNET_FLAG_RELOAD = &H80000000  '不从 Cache 中下载
    Private Const scBlankStr = ""                    '空字符串常量
```

上面是对所用到的 HTTP 函数和常数的声明，值得注意的是：InternetReadHTML 与 InternetReadImg 调用的是同一个函数 InternetReadFile，只是函数中的参数声明类型稍有差别，在后面我们将会介绍这样声明的作用和原因。

```
    Private hInternetSession    As Long     ' Internet 会话句柄
    Private bInitialized        As Boolean  ' 标志是否已经初始化
    Private hUrlFile            As Long     ' 打开的 URL 句柄
    Private sContents           As String   ' HTML 页面文本内容
    Private sLastError        As String     ' 最近一次错误
    Private sStatus             As String   ' 存放状态字符串的变量
    Private objWindow       As Object   ' 显示状态字符串的窗口
    Private sUserAgent       As String      ' 在 HTTP 协议中的 UserAgent
```

上面是声明所用到的一些变量。

```
    '初始化过程，用于打开 Internet 会话及设置变量
    Public Sub Init(Optional vInUserAgent As Variant)
    On Error Resume Next
    If IsMissing(vInUserAgent) Then
       sUserAgent = App.EXEName
    Else
       sUserAgent = vInUserAgent
    End If
    Term
    hUrlFile = 0
    sContents = scBlankStr
    sLastError = scBlankStr
    sStatus = scBlankStr
    '打开 Internet 连接
    hInternetSession = InternetOpen(sUserAgent, INTERNET_OPEN_TYPE_PRECONFIG,
```

```
    vbNullString, vbNullString, 0)
    bInitialized = CBool(hInternetSession)
    End Sub
```

以上过程用于初始化类变量：设置 sUserAgent 的值，打开 Internet 会话函数并获得句柄。

```
    '从给定的 Url 取得 HTML 文件的内容。并保存在字符串 sContents 和指定的文件中
    Public Function ReadUrlHTML(ByVal sUrl As String, Optional vFileName As Variant)
As Boolean
    Dim sReadBuffer As String * 2048       '为 InternetReadFile 函数提供字符串作为
数据接收缓冲区
    Dim lNumberOfBytesRead  As Long     '调用 InternetReadFile 函数读取的字节数
    Dim lTotalBytesRead     As Long      '一共读取的字节数
    Dim bDoLoop             As Boolean  '调用 InternetReadFile 的返回值
    Dim i                  As Integer
    On Error GoTo errReadUrl
    '设置鼠标状态为等待
    Screen.MousePointer = vbHourglass
    SetStatus "下载开始: " & sUrl
    '打开 URL 下载
    hUrlFile = InternetOpenUrl(hInternetSession, sUrl, vbNullString, 0,
                   INTERNET_FLAG_EXISITING_CONNECT, 0)
    If CBool(hUrlFile) Then
       sContents = scBlankStr
       bDoLoop = True
       While bDoLoop
          sReadBuffer = scBlankStr
          '从打开的 InternetOpenUrl 句柄中读取文件数据
          bDoLoop = InternetReadHTML(hUrlFile, sReadBuffer, Len(sReadBuffer),
lNumberOfBytesRead)
          If Not CBool(bDoLoop) Then CheckError
          lTotalBytesRead = lTotalBytesRead + lNumberOfBytesRead
          SetStatus "读取" & CStr(lTotalBytesRead) & "字节：" & sUrl
          If CBool(lNumberOfBytesRead) Then
             '将读取的数据保存到变量 sContents 中，分析链接时要使用
             sContents = sContents & Left$(sReadBuffer, lNumberOfBytesRead)
          Else
             bDoLoop = False
          End If
       Wend
       '关闭 InternetOpenUrl 打开的句柄
       InternetCloseHandle (hUrlFile)
```

```
        '将读取的数据存到指定的文件中
        If Len(sContents) > 0 Then
            Dim iFileNum As Integer
            iFileNum = FreeFile
            Open CStr(vFileName) For Binary As iFileNum
            Put #iFileNum, , sContents
            Close (iFileNum)
            ReadUrlHTML = True
        Else
            ReadUrlHTML = False
        End If
    Else
        CheckError
    End If
    SetStatus "下载完毕：" & sUrl
    '设置鼠标为缺省状态
    Screen.MousePointer = vbDefault
    Exit Function
    errReadUrl:
    sLastError = Error$(Err)
    Screen.MousePointer = vbDefault
    End Function
```

上面的函数从给定的 URL 取得 HTML 文件的内容。并保存在字符串 sContents 和指定的文件中。向 URL 请求文件，首先使用 InternetOpenUrl 打开 URL 取得相应的句柄，如果成功则使用 InternetReadFile 函数（上面声明为 InternetReadHTML 的形式）读取 URL 的数据，我们再看 InternetReadHTML 的声明：

```
    Private Declare Function InternetReadHTML Lib "wininet.dll" Alias
"InternetReadFile" _
            (ByVal hFile As Long, ByVal lpBuffer As String,
                        ByVal dwNumberOfBytesToRead As Long, _
            lNumberOfBytesRead As Long) As Integer
```

其中 hFile 是使用 InternetOpenUrl 成功打开后返回的句柄，lpBuffer 是提供的接收数据的缓冲区，dwNumberOfBytesToRead 是缓冲区的大小，lNumberOfBytesRead 是实际读取的字节数。

上面使用该函数如下：

```
    bDoLoop = InternetReadHTML(hUrlFile, sReadBuffer, Len(sReadBuffer),
                            lNumberOfBytesRead)
```

其中 sReadBuffer 定义为指定长度的字符串变量：

```
    Dim sReadBuffer As String * 2048
```

Visual Basic 使用 Unicode 存储和操作字符串。Unicode 是一种用两个字节表示一个字符

的字符集。另外一些程序，如 Windows 95 或 98 API，使用 ANSI（American National Standards Institute）或 DBCS 存储和操作字符串。当从 Visual Basic 移出字符串时，会遇到 Unicode 和 ANSI／DBCS 之间的差别。

由于在 VB 中使用的是 Unicode，也就是长度为 1 的字符串变量实际占用的字节是 2 个，而在 Windows 98、Windows 95 等不是 Unicode 系统中，使用的 API 函数的字符串也不是 Unicode。因此在 VB 中通过 String 类型提供缓冲区给 API 函数使用时先转换成的 ANSI，然后调用 API，得到数据后再转换成为 VB 中使用的 Unicode 字符串。存盘的过程也是这样，存盘文件是将字符串转换成的 ANSI 的形式的。可以使用 LenB 函数证明这一点，Len（"ABC 中文"）的值为 5，而计算字符串的字节数的 LenB（"ABC 中文"）函数得到的结果却是 10，实际上在 ANSI 中，ASCII 的字符为单字节的，一个汉字占两个字节，所以在 ANSI 中占用 7 个字节。是着在 VB 中将这个字符串存到文件中，再用其他的 16 进制编辑软件如 UltraEdit 打开，发现只有 7 个字节。

正是由于 VB 中使用的 Unicode 字符串和字符串处理函数，使得从 API 缓冲区中提取数据变得困难，要碰到类似的处理，如果数据只是文本类型的，那么直接提供定长的 String 变量作为缓冲区，VB 能够自动的进行转换，而直接得到正确的数据。如果要得到的数据中有二进制的数据，那么得到的数据就会因为 VB 中的转换改变内容。

因此在使用 InternetReadFile 函数时根据数据类型使用不同的数据类型作为缓冲区接收数据。在上面的函数中使用了定长的 String 类型变量作为缓冲区，在下面读取二进制数据的函数中，则使用其他类型的变量作为缓冲区。

```
    '从给定的 Url 取得二进制文件的内容。并保存到指定的文件
    Public Function ReadUrlImg(ByVal sUrl As String, Optional vFileName As Variant)
As Boolean
    Dim sReadBuffer(2048)   As Byte        '为 InternetReadFile 函数提供字符串作为数
据接收缓冲区
    Dim lNumberOfBytesRead  As Long        '调用 InternetReadFile 函数读取的字节数
    Dim lTotalBytesRead     As Long        '一共读取的字节数
    Dim bDoLoop           As Boolean     '调用 InternetReadFile 的返回值
    Dim i                 As Integer
    On Error GoTo errReadUrl
    Screen.MousePointer = vbHourglass
    SetStatus "下载开始: " & sUrl
    If IsMissing(vFileName) Then
       ReadUrlImg = False
       Exit Function
    End If
    '打开 url 下载
    hUrlFile = InternetOpenUrl(hInternetSession, sUrl, vbNullString, 0,
    INTERNET_FLAG_EXISITING_CONNECT, 0)
    If CBool(hUrlFile) Then
       sContents = scBlankStr
```

```
        bDoLoop = True
        Dim iFileNum As Integer
        While bDoLoop
            '从打开的 InternetOpenUrl 句柄中读取文件数据
            bDoLoop    =    InternetReadImg(hUrlFile,    sReadBuffer(0),    2048,
lNumberOfBytesRead)
            If Not CBool(bDoLoop) Then CheckError
            lTotalBytesRead = lTotalBytesRead + lNumberOfBytesRead
            SetStatus "读取" & CStr(lTotalBytesRead) & "字节：" & sUrl
            If CBool(lNumberOfBytesRead) Then
                '读取的数据存到指定的文件中
                If iFileNum = 0 Then
                    iFileNum = FreeFile
                    Open CStr(vFileName) For Binary As iFileNum
                End If
                For i = 0 To lNumberOfBytesRead - 1
                    Put #iFileNum, , sReadBuffer(i)
                Next i
            Else
                bDoLoop = False
            End If
        Wend
        '关闭 InternetOpenUrl 打开的句柄
        InternetCloseHandle (hUrlFile)
        If iFileNum <> 0 Then
            Close (iFileNum)
            ReadUrlImg = True
        Else
            ReadUrlImg = False
        End If
    Else
        CheckError
    End If
    SetStatus "下载完毕：" & sUrl
    '设置鼠标为缺省状态
    Screen.MousePointer = vbDefault
    Exit Function
    errReadUrl:
    sLastError = Error$(Err)
    Screen.MousePointer = vbDefault
```

```
    Exit Function
    End Function
```

以上的函数是从给定的 URL 取得二进制文件的内容，并保存到指定的文件。在该函数中，读取打开的 URL 使用的函数是 InternetReadFile，声明为 InternetReadImg 的形式：

```
    Private Declare Function InternetReadImg Lib "wininet.dll" Alias
"InternetReadFile" _
                (ByVal hFile As Long, lpBuffer As Any,
    ByVal dwNumberOfBytesToRead As Long, _
                lNumberOfBytesRead As Long) As Integer
```

其中参数 lpBuffer 与函数 InternetReadHTML 中的不同，这里声明为 Any 类型，实际函数的使用如下：

```
    bDoLoop = InternetReadImg(hUrlFile, sReadBuffer(0), 2048, lNumberOfBytesRead)
```

其中 sReadBuffer 的声明为 Byte 类型的形式：

```
    Dim sReadBuffer(2048)  As Byte
```

将字节数组 sReadBuffer 作为缓冲区传递给读取数据的 API 函数 InternetReadFile，用以接收二进制数据。事实上，用于读取文本数据也是可以的，但由于在程序中要分析得到的文本字符串，所以声明成为两个不同的函数使用起来更加方便。

只要将 sReadBuffer 数组的第一个字节传递过去就可以了，API 得到的是该数组的起始地址，能够正确地将数据写到缓冲区中。此外需要说明的是将 InternetReadImg 中的参数 lpBuffer 声明为 Any 类型不是函数能够接受 Variant 类型的数据，API 函数参数类型的使用都是非常严格的（与 C 标准一样），而 VB 中的 Variant 类型的结构是非常复杂的，这样做的目的只是能在 API 调用时，VB 不检查 sReadBuffer(0)的类型，而弹出 Type Mistach 之类的对话框。这一点读者可以自己试试。

```
    '分析 HTML 文本，从中提取有用的文件链接
    Public Function ParseHTML(ByVal sToken As String, colItems As Collection) As
Boolean
    Dim lPosInStr As Long
    Dim lEndPosInStr1, lEndPosInStr2 As Long
    Dim lStartPos As Long
    Dim sAddItem As String
    On Error Resume Next
    '设置鼠标状态为等待
    Screen.MousePointer = vbHourglass
    SetStatus "分析 HTML 页面开始..."
    sContents = Replace(sContents, vbCrLf, "")
    If Len(sContents) > Len(sToken) Then
        lStartPos = 1
        '查找字符串中的链接起始标志字符串 sToken
        lPosInStr = InStr(lStartPos, sContents, sToken, vbTextCompare)
        '如果起始标志字符串找到
```

```
    While lPosInStr > 0
        DoEvents: DoEvents
        lPosInStr = lPosInStr + Len(sToken)
        '查找链接的结束位置
        If Mid(sContents, lPosInStr, 1) = """" Then
            lPosInStr = lPosInStr + 1
            lEndPosInStr1 = InStr(lPosInStr, sContents, """", vbTextCompare)
        ElseIf lEndPosInStr1 = 0 Then
            lEndPosInStr1 = InStr(lPosInStr, sContents, " ", vbTextCompare)
        Else
            lEndPosInStr1 = InStr(lPosInStr, sContents, ">", vbTextCompare)
        End If
        '得到链接字符串
        sAddItem = Mid(sContents, lPosInStr, lEndPosInStr1 - lPosInStr)
        '检查链接字符串
        lEndPosInStr1 = InStr(1, sAddItem, " ", vbTextCompare)
        lEndPosInStr2 = InStr(1, sAddItem, ">", vbTextCompare)
        If lEndPosInStr1 > 0 Or lEndPosInStr2 > 0 Then
            If lEndPosInStr1 > lEndPosInStr2 Then
                sAddItem = Mid(sAddItem, 1, lEndPosInStr1 - 1)
            Else
                sAddItem = Mid(sAddItem, 1, lEndPosInStr2 - 1)
            End If
        End If
        '将链接加入到集合中
        If Len(sAddItem) Then colItems.Add sAddItem, sAddItem
        lStartPos = lPosInStr + Len(sAddItem)
        lPosInStr = InStr(lStartPos, sContents, sToken, vbTextCompare)
    Wend
    ParseHTML = True
End If
SetStatus "分析 HTML 页面结束"
'恢复鼠标状态为缺省
Screen.MousePointer = vbDefault
End Function
```

上面的函数用于分析 HTML **文本，从中提取有用的文件链接。该函数在** InternetReadHTML **成功进行，用于对下载的** HTML **页面进行语法分析。成功调用** InternetReadHTML **后，**HTML **文件保存在相应的硬盘文件和字符串** sContents **中。首先指出要提取链接的起始字符串和保存链接的集合，在** frmMain **中使用如下：**

```
If objNetGet.ParseHTML("HREF=", colReferences) Then
```

```
            AddReferences sReUrl
        End If
        If objNetGet.ParseHTML(" SRC=", colImages) Then
            AddImages sReUrl
        End If
```

对 HTML 页面熟悉的读者肯定知道这其中的原因，HTML 中大多数链接都是用以下的形式表示的：

```
<A HREF=……>…..</A>
```

或者

```
<IMG SRC=……>
```

在“ HREF= ”和“ SRC= ”后面的内容是目标 URL。

指定了提取链接的起始字符串和保存链接的集合，首先将 HTML 字符串中的 Crlf 字符清除，因为对于 HTML 来说 Crlf 是不存在的。如：

```
<A HREF=……>…..</A>
```

与

```
<A
HREF=……>…..</A>
```

表示的内容是一样的。

此外在找到链接的起始字符串之后，提取其中的链接也需要进行一些处理。考虑到以下的几种情况：

```
<IMG SRC="link.gif">
<IMG SRC=link.gif>
<IMG SRC=link.gif width=100>
```

如果链接字符串在引号之间，提取字符串可以根据引号的位置，但同时也有无引号的情况，这时可以以空格或“ > ”作为链接字符串结尾的标志。

```
'根据错误设置描述错误的变量
Sub CheckError()
Dim lLastErrorNo As Long
lLastErrorNo = Err.LastDllError
If lLastErrorNo > 0 Then sLastError = TranslateErrorCode(lLastErrorNo)
End Sub
```

以上函数用于设置描述错误的字符串变量 sLastError

```
'解释错误码
Function TranslateErrorCode(ByVal lErrorCode As Long) As String
Select Case lErrorCode
   Case 12001: TranslateErrorCode = "No more handles could be generated at
this time"
   Case 12002: TranslateErrorCode = "The request has timed out."
   Case 12003: TranslateErrorCode = "An extended error was returned from the
server."
```

```
        Case 12004: TranslateErrorCode = "An internal error has occurred."
        Case 12005: TranslateErrorCode = "The URL is invalid."
        Case 12006: TranslateErrorCode = "The URL scheme could not be recognized,
or is not supported."
        Case 12007: TranslateErrorCode = "The server name could not be resolved."
        Case 12008: TranslateErrorCode = "The requested protocol could not be
located."
        Case 12009: TranslateErrorCode = "A request to InternetQueryOption or
InternetSetOption specified an
    invalid option value."
        Case 12010: TranslateErrorCode = "The length of an option supplied to
InternetQueryOption or
    InternetSetOption is incorrect for the type of option specified."
        Case 12011: TranslateErrorCode = "The request option can not be set, only
queried. "
        Case 12012: TranslateErrorCode = "The Win32 Internet support is being shutdown
or unloaded."
        Case 12013: TranslateErrorCode = "The request to connect and login to an
FTP server could not be
    completed because the supplied user name is incorrect."
        Case 12014: TranslateErrorCode = "The request to connect and login to an
FTP server could not be
    completed because the supplied password is incorrect. "
        Case 12015: TranslateErrorCode = "The request to connect to and login to
an FTP server failed."
        Case 12016: TranslateErrorCode = "The requested operation is invalid. "
        Case 12017: TranslateErrorCode = "The operation was canceled, usually because
the handle on which
    the request was operating was closed before the operation completed."
        Case 12018: TranslateErrorCode = "The type of handle supplied is incorrect
for this operation."
        Case 12019: TranslateErrorCode = "The requested operation can not be carried
out because the handle
    supplied is not in the correct state."
        Case 12020: TranslateErrorCode = "The request can not be made via a proxy."
        Case 12021: TranslateErrorCode = "A required registry value could not be
located. "
        Case 12022: TranslateErrorCode = "A required registry value was located
but is an incorrect type or has
    an invalid value."
```

```
        Case 12023: TranslateErrorCode = "Direct network access cannot be made at
this time. "
        Case 12024: TranslateErrorCode = "An asynchronous request could not be made
because a zero context
    value was supplied."
        Case 12025: TranslateErrorCode = "An asynchronous request could not be made
because a callback
    function has not been set."
        Case 12026: TranslateErrorCode = "The required operation could not be
completed because one or more
    requests are pending."
        Case 12027: TranslateErrorCode = "The format of the request is invalid."
        Case 12028: TranslateErrorCode = "The requested item could not be located."
        Case 12029: TranslateErrorCode = "The attempt to connect to the server
failed."
        Case 12030: TranslateErrorCode = "The connection with the server has been
terminated."
        Case 12031: TranslateErrorCode = "The connection with the server has been
reset."
        Case 12036: TranslateErrorCode = "The request failed because the handle
already exists."
        Case Else: TranslateErrorCode = "Error details not available."
    End Select
    End Function
```

以上函数根据错误码的值设置相应的描述错误的字符串

```
    '设置用于显示描述信息的窗口
    Property Let SetStatusWindow(objStatusWindow As Object)
    On Error Resume Next
    Set objWindow = objStatusWindow
    End Property

    '根据下载的状态设置窗口的文字
    Private Sub SetStatus(sInStatus As String)
    On Error Resume Next
    objWindow = sInStatus
    DoEvents
    End Sub
```

上面的代码用于设置显示提示信息的窗口和显示信息。

```
    '结束程序所做的一些处理，释放资源
    Public Sub Term()
```

```
On Error Resume Next
If InternetCloseHandle(hInternetSession) Then
    bInitialized = False
    SetStatus ("Ready")
Else
    CheckError
End If
End Sub
'获得最近的一次错误信息
Property Get GetLastError() As String
GetLastError = sLastError
End Property

'设置程序名（在 InternetOpen 函数中使用）
Property Let SetUserAgent(sInUserAgent As String)
If Len(sInUserAgent) > 0 Then sUserAgent = sInUserAgent
End Property

'类对象使用结束
Private Sub Class_Terminate()
Term
End Sub
```

以上的代码都很好理解，这里就不再解释了。

7.5　HTTP API 高级开发

从上面介绍的 HTTP 的协议和实现可以看出，开发一个 Internet 需要了解许多基于协议和 Winsock 通信的知识。虽然 VB 已经将 Windows Socket 封装 Winsock 控件的形式方便程序的开发，而不需要进一步了解 socket 的底层工作原理，使得开始 Internet 应用程序变得相对容易，但是还是得对有关的 Internet 协议如 HTTP、FTP 等了解比较清楚。

值得庆幸的是，Windows 提供了一系列的 Internet 开发 API 函数（如 WinInet），这些函数隐藏了底层通信细节，方便 Internet 应用程序的开发。

WinInet 是 Win32 Internet functions 的缩写，这些函数是封装在 WININET.DLL 动态连接库中的。

WinInet 中有以下几类函数：处理 Internet URL 的函数、FTP 函数、HTTP 函数和 Gopher 函数。本章要重点介绍 HTTP 函数的使用，另外处理 Internet URL 的函数简单介绍，FTP 函数的使用在“FTP 高级编程”一章有应用实例，本章就不再重复了。

7.5.1 实例介绍

这一节介绍的实例 4 所能实现的功能与实例 1 的功能基本上是相似的。只是在这一节中利用 WinInet HTTP API 函数来实现断点续传文件的下载。

通过该实例，即使对 HTTP 协议的基本内容不太了解的读者也能开发基于 HTTP 协议的程序。该实例实现的功能有：

- 实现多个文件同时下载。
- 支持断点续传。
- 支持身份验证。

该工程由两个窗体、两个模块和一个类模块组成：

- 窗体 frmAdd。
- 窗体 frmDown。
- 模块 modWinInet。
- 模块 modDown。
- 类模块 clsDown。

本实例中实现文件下载的原理跟实例 1 的差不多，只是向服务器请求不是通过进行 Winsock 控件编程，处理发送请求和接收响应的数据进行处理存盘，而是通过 WinInet 函数相关的函数进行编程，使用这些函数连接服务器，发送请求和获得数据。

7.5.2 WinInet HTTP API 实现断点续传

使用 WinInet HTTP API 进行断点续传的步骤为：

（1）打开 Internet 会话

在进行 Internet 连接之前，不管是采用 FTP，还是使用其他协议如 HTTP、GOPHER 等，首先使用 API 函数 InternetOpen（）打开 Internet 会话。程序中的使用如下：

```
    hInternetSession = InternetOpen(scUserAgent, INTERNET_OPEN_TYPE_PRECONFIG,
vbNullString, vbNullString, 0)
    If CBool(hInternetSession) Then
        SetStatus "InternetOpen 打开成功 !! "
    Else
        SetStatus "InternetOpen 失败 ! "
    End If
```

因为函数执行成功之后的返回值为一个非 0 的句柄值，否则为 0，所以可以依此来判断是否成功打开了 Internet 会话句柄。

（2）进行 Internet 连接

在实例 3 中，直接使用 InternetOpenUrl 打开指定的 URL 再读取文件数据，这里使用另外的一种更加灵活的方式打开 Internet 连接句柄，程序中的使用如下：

```
    hInternetConnect = InternetConnect(hInternetSession, mHost, mPort, _
```

```
                vbNullString, vbNullString, INTERNET_SERVICE_HTTP, 0, 0)
```

其中 hInternetSession 是 InternetOpen 打开的句柄，mHost 是连接的主机地址，mPort 是端口号。InternetConnect 的 API 声明如下：

```
    Public Declare Function InternetConnect Lib "wininet.dll" Alias
"InternetConnectA" _
    (ByVal hInternetSession As Long, ByVal sServerName As String, ByVal nServerPort
As Integer, _
    ByVal sUsername As String, ByVal sPassword As String, ByVal lService As Long,
_
    ByVal lFlags As Long, ByVal lContext As Long) As Long
```

其余的参数，如 sUsername 和 sPassword 是用于身份验证的账号和密码，lService 是表示连接的是什么类型的服务，一共有以下三种可以选择：

```
    INTERNET_SERVICE_FTP        FTP 服务
    INTERNET_SERVICE_GOPHER     Gopher 服务
    INTERNET_SERVICE_HTTP       HTTP 服务
```

很显然，本实例介绍的是 HTTP 下载，用的就是 INTERNET_SERVICE_HTTP 了，该值为常数，不过在 VB 中是没有这个常数变量的，需要自己声明：

```
    Public Const INTERNET_SERVICE_FTP = 1
    Public Const INTERNET_SERVICE_GOPHER = 2
    Public Const INTERNET_SERVICE_HTTP = 3
```

为了让读者增加这方面的了解，在此同时声明另外两个服务的常数变量。

此外 LFlags 与 lContext 在本实例中只要设置为 0 就可以了。

（3）打开并发送请求

在利用 InternetConnect 成功打开 Internet 连接之后，与实例 3 的 InternetOpenUrl 不同的是，不能立即的读取服务器返回的数据。事实上，使用 InternetOpenUrl 后，函数执行两个动作，连接服务器并发送请求，接着就可以使用其他函数读取服务器返回的数据。而 InternetConnect 只是连接服务器，如果连接成功需要使用其他函数来发出请求。程序中使用如下：首先使用 hHttpOpenRequest 打开请求句柄：

```
    hHttpOpenRequest = HttpOpenRequest(hInternetConnect, "GET", mRelativeUrl,
"HTTP/1.0",
    vbNullString, 0, INTERNET_FLAG_RELOAD, 0)
```

hHttpOpenRequest 的函数声明如下：

```
    Public Declare Function HttpOpenRequest Lib "wininet.dll" Alias
"HttpOpenRequestA" _
    (ByVal hHttpSession As Long, ByVal sVerb As String, ByVal sObjectName As String,
    ByVal sVersion As String, ByVal sReferer As String, ByVal something As Long,
    ByVal lFlags As Long, ByVal lContext As Long) As Long
```

其中 hHttpSession 为 InternetConnect 打开的句柄，sVerb 是表明使用请求方法的字符串（“GET”、“POST”等），sObjectName 是请求的相对地址。SVersion 是使用的 HTTP 的

版本号，sReferer 是包含请求连接的 URL（指定请求的 Reference 标题字段的值），可以设置为 vbNullString，lFlags 设置为以下常数值时，代表不管在 Cache 中是否存在，都重新从服务器下载：

```
    Public Const INTERNET_FLAG_RELOAD = &H80000000
```

something 和 lContext 可以设置为 0。

在成功打开请求句柄之后，接着可以利用函数 HttpAddRequestHeaders，加入请求标题字段，实例中的使用如下：

```
    Getstr = "User-Agent: DownJet1.0"
    HttpAddRequestHeaders     hHttpOpenRequest,     Getstr,     Len(Getstr),
HTTP_ADDREQ_FLAG_ADD
```

函数 HttpAddRequestHeaders 的 API 声明如下：

```
    Public Declare Function HttpAddRequestHeaders Lib "wininet.dll" Alias
"HttpAddRequestHeadersA" _
    (ByVal hHttpRequest As Long, ByVal sHeaders As String, ByVal lHeadersLength
As Long, _
    ByVal lModifiers As Long) As Integer
```

其中 hHttpRequest 是 HttpOpenRequest 打开的句柄，sHeaders 是增加的标题字段，lHeadersLength 是增加的标题字段的长度，lModifiers 可以设置的的值如下：

```
    '请求的标题字段不存在时才增加，否则出错
    Public Const HTTP_ADDREQ_FLAG_ADD_IF_NEW = &H10000000
    '请求的标题字段不存在时增加，否则更新
    Public Const HTTP_ADDREQ_FLAG_ADD = &H20000000
    '如果标题字段的值为空，则删除，不为空则代替
    Public Const HTTP_ADDREQ_FLAG_REPLACE = &H80000000
```

如果要实现断点续传，请求服务器返回部分数据，则要增加标题字段 Ranges：

```
    Getstr = "Range: bytes=" & ReceiveBytes & "-"
    HttpAddRequestHeaders     hHttpOpenRequest,     Getstr,     Len(Getstr),
HTTP_ADDREQ_FLAG_ADD
```

最后，使用函数 HttpSendRequest 发送请求：

```
    iRetVal = HttpSendRequest(hHttpOpenRequest, vbNullString, 0, 0, 0)
```

（4）读取服务器返回的响应标题字段内容

发送请求给服务器之后，服务器响应客户端程序的请求，API 函数 HttpQueryInfo 可用于读取响应的标题字段信息，同时，也可以读取发送的请求的标题字段内容。该函数的声明如下：

```
    Public Declare Function HttpQueryInfo Lib "wininet.dll" Alias "HttpQueryInfoA"
_
    (ByVal hHttpRequest As Long, ByVal lInfoLevel As Long, ByRef sBuffer As Any,
_
    ByRef lBufferLength As Long, ByRef lIndex As Long) As Integer
```

其中 hHttpReqest 是 HttpOpenRequest 打开的句柄，lInfoLevel 指定要取得的字段内容，sBuffer 为程序提供给 API 接收内容的缓冲区，lBufferLength 为缓冲区的长度，lIndex 可设置为 0。如表 7-4 所示是 lInfoLevel 的取值及意思。

表 7-4　　lInfoLevel 设置的值

常数变量	值	字段或意思
HTTP_QUERY_CONTENT_TYPE	= 1	Content-Type
HTTP_QUERY_CONTENT_LENGTH	= 5	Content-Length
HTTP_QUERY_EXPIRES	= 10	Expires
HTTP_QUERY_LAST_MODIFIED	= 11	Last-Modified
HTTP_QUERY_PRAGMA	= 17	Pragma
HTTP_QUERY_VERSION	= 18	Http的版本号
HTTP_QUERY_STATUS_CODE	= 19	响应码
HTTP_QUERY_STATUS_TEXT	= 20	响应文本行
HTTP_QUERY_RAW_HEADERS	= 21	所有响应标题
HTTP_QUERY_RAW_HEADERS_CRLF	= 22	所有响应标题（带Crlf）
HTTP_QUERY_FORWARDED	= 30	Forwarded
HTTP_QUERY_SERVER	= 37	Server
HTTP_QUERY_USER_AGENT	= 39	Agent
HTTP_QUERY_SET_COOKIE	= 43	Set-Cookie
HTTP_QUERY_REQUEST_METHOD	= 45	请求方法

如果要取得再请求之后服务器返回的响应码，可以使用如下：

```
    Dim sBuffer        As String * 1024
    Dim lBufferLength   As Long
    lBufferLength = Len(sBuffer)
    HttpQueryInfo(hHttpRequest,    HTTP_QUERY_STATUS_CODE,    ByVal    sBuffer,
lBufferLength, 0)
    sReContent = Mid(sBuffer, 1, lBufferLength)
```

在 HTTP 请求下载文件中，知道响应码是非常重要的，根据响应码的值可以判断当前的请求是否正确。

（5）读取服务器返回的数据

读取文件的函数使用 InternetReadFile，在上一个实例中已经介绍过使用该函数读取二进制数据与读取文本数据时的一些不同，在这里我们使用读取二进制文件数据的方法，使用声明的 InternetReadImg。该声明如下：

```
    Private   Declare   Function   InternetReadImg   Lib   "wininet.dll"   Alias
"InternetReadFile" _
           (ByVal hFile As Long, lpBuffer As Any, ByVal dwNumberOfBytesToRead As
Long, _
            lNumberOfBytesRead As Long) As Integer
```

其使用方法在这里就不再重复了，请读者们参看实例 3 的相关内容。

（6）关闭打开的 Internet 句柄

Internet 函数请求文件完毕，关闭所有打开的句柄，释放资源。不管是使用 InternetOpen

函数打开或是使用 InternetConnect 打开的句柄，使用的是同一个函数关闭。

```
InternetCloseHandle(handle)
```

7.5.3 关键代码分析

由于本实例中的一些功能和代码与实例 1 中有着相似之处，因此在这里就不再解释全部代码了，而仅仅是对一些做了改动的关键代码进行介绍和解释。其他代码读者可以先掌握实例 1 的内容，再看本实例。

1. 类模块 clsDown

该模块用于处理下载文件和存盘的，是整个程序的核心部分。与实例 1 的差别主要在此，在该模块是利用 WinInet HTTP API 函数实现的文件下载的，而实例 1 中使用了 Winsock 控件的通信。

```
Option Explicit
Public bBusy As Boolean          '表示正在下载一个任务
Public DownUrl As String             '要下载的 url 地址
Public WhichDown As Integer      '下载任务的索引
Public StartTime As Date             '下载的开始连接时间
Public ReceiveBytes As Long      '已下载的文件数据字节数
Public mFlen As String               '下载文件长度
Public bCancel As Boolean            '用户是否取消下载
Public mProxy As String          '代理服务器地址和端口
Public mProxyPort As Integer
Public mAuthId As String             '认证账号及密码
Public mAuthPass As String
Public mFile As String               '保存的文件路径
Private mHost As Strin               '连接的主机名和端口
Private mPort As Integer
Private mRelativeUrl As String       '下载的相对 URL
Private sStatusCode As String        '服务器响应码
Private hInternetSession As Long         'InternetOpen 打开的句柄
Private hInternetConnect As Long         'InternetConnect 打开的句柄
Private hHttpOpenRequest As Long     'HttpOpenRequest 打开的句柄
```

以上代码声明了一些使用到的变量和类接口变量，需要注意的是，在本实例中为了简单起见，并没有实现通过代理服务器下载，但考虑到与实例 1 的兼容，这里还是声明有关变量，以方便读者自己进行扩展。

```
'该过程用于取得响应得标题字段信息
Private Function GetQueryInfo(ByVal hHttpRequest As Long, sReContent As String,
ByVal iInfoLevel As Long) As Boolean
Dim sBuffer        As String * 1024
```

```
    Dim lBufferLength   As Long
    lBufferLength = Len(sBuffer)
    GetQueryInfo = CBool(HttpQueryInfo(hHttpRequest, iInfoLevel, ByVal sBuffer,
lBufferLength, 0))
    sReContent = Mid(sBuffer, 1, lBufferLength)
    End Function
```

上面的函数用于取得指定的 HTTP **服务器相应的标题字段，函数成功返回** True**，否则返回** False**。需要注意的是要从定长的字符串** sBuffer **提取一定长度的内容，否则得到的字符串长度都是** 1024**。**

```
    '分析下载的 URL
    Public Function AnalyzeUrl() As Boolean
    Dim pos1, pos2 As Integer
    Dim mUrl As String
    mUrl = DownUrl
    If InStr(1, mUrl, "http://") > 0 Then
        '得到端口号
        mPort = INTERNET_DEFAULT_HTTP_PORT
    Else
        AnalyzeUrl = False
        Exit Function
    End If
    pos1 = InStr(1, mUrl, "http://")
    pos2 = InStr(8, mUrl, "/")
    If pos2 = 0 Then
        AnalyzeUrl = False
        Exit Function
    Else
        '得到主机地址
        mHost = Mid(mUrl, 8, pos2 - 8)
        pos1 = InStr(1, mHost, ":")
        If pos1 > 0 Then
            mPort = Mid(mHost, pos1 + 1)
            mHost = Mid(mHost, 1, pos1 - 1)
        End If
        '得到相对路径
        mRelativeUrl = Mid(mUrl, pos2)
    End If
    pos2 = InStrRev(mUrl, "/")
    If pos2 > 8 Then
        '得到文件名
```

```
    mFile = Mid(mUrl, pos2 + 1)
Else
    AnalyzeUrl = False
    Exit Function
End If
AnalyzeUrl = True
End Function
```

上面代码分析下载的 URL 提取服务器地址、端口号、相对 URL 等有用的信息，几乎与实例 1 的函数一样，只是增加了提取端口号的功能，如下载的 URL 为：http://10.11.111.119:81/my.mp3，则可以将端口号 81 提取出来，连接服务器时依据此端口设定连接。

```
'调用该函数下载文件
Public Function StartDown() As Boolean
bBusy = True
SetStatus "开始下载"
'打开 Internet 会话句柄
hInternetSession = InternetOpen(scUserAgent, INTERNET_OPEN_TYPE_PRECONFIG,
vbNullString, vbNullString, 0)
If CBool(hInternetSession) Then
    SetStatus "InternetOpen 打开成功 !! "
Else
    SetStatus "InternetOpen 失败!"
End If
StartDown = StartDownFile()
'无论下载成功，都关闭所有的 Internet 函数打开的句柄
InternetCloseHandle hInternetSession
InternetCloseHandle hInternetConnect
InternetCloseHandle hHttpOpenRequest
bBusy = False
End Function
```

以上代码是用于模块外程序调用的下载接口。

```
'直接连接 Url 指定的服务器下载
Public Function StartDownFile() As Boolean
Dim iRetVal    As Integer
Dim vDllVersion As tWinInetDLLVersion
Dim sStatus    As String
SetStatus "正在打开 Internet 连接"
hInternetConnect = InternetConnect(hInternetSession, mHost, mPort, _
                        vbNullString, vbNullString, INTERNET_SERVICE_HTTP, 0, 0)
If hInternetConnect > 0 Then
```

```
        SetStatus "HttpOpenRequest"
        '打开请求
        hHttpOpenRequest = HttpOpenRequest(hInternetConnect, "GET", mRelativeUrl,
"HTTP/1.0",
    vbNullString, 0, INTERNET_FLAG_RELOAD, 0)
        If CBool(hHttpOpenRequest) Then
            '加入请求的标题字段
            Dim Getstr As String
            Getstr = "Accept: */*"
            HttpAddRequestHeaders   hHttpOpenRequest,   Getstr,   Len(Getstr),
HTTP_ADDREQ_FLAG_ADD
            Getstr = "User-Agent: DownJet1.0"
            HttpAddRequestHeaders   hHttpOpenRequest,   Getstr,   Len(Getstr),
HTTP_ADDREQ_FLAG_ADD
            Getstr = "Host: " & mHost
            HttpAddRequestHeaders   hHttpOpenRequest,   Getstr,   Len(Getstr),
HTTP_ADDREQ_FLAG_ADD
            If ReceiveBytes > 0 Then
                '如果以前已经下载了一部分数据，发送断点续传请求
                Getstr = "Range: bytes=" & ReceiveBytes & "-"
                HttpAddRequestHeaders hHttpOpenRequest, Getstr, Len(Getstr),
    HTTP_ADDREQ_FLAG_ADD
            End If
            If mAuthId <> "" Then
                Getstr = "Authorization: Basic " & EncodeStr(mAuthId & ":" & mAuthPass)
                HttpAddRequestHeaders hHttpOpenRequest, Getstr, Len(Getstr),
    HTTP_ADDREQ_FLAG_ADD
            End If
            Getstr = "Connection: close"
            HttpAddRequestHeaders   hHttpOpenRequest,   Getstr,   Len(Getstr),
HTTP_ADDREQ_FLAG_ADD
            SetStatus "发送请求"
            '向服务器发送下载文件请求
            iRetVal = HttpSendRequest(hHttpOpenRequest, vbNullString, 0, 0, 0)
            If iRetVal Then
                SetStatus "HttpQueryInfo"
                '取得服务器返回的数据长度
                GetQueryInfo hHttpOpenRequest, mFlen, HTTP_QUERY_CONTENT_LENGTH
                '取得服务器返回的响应码
                GetQueryInfo hHttpOpenRequest, sStatusCode, HTTP_QUERY_STATUS_CODE
```

```
            sStatus = "请求完毕"
        Else
            sStatus = "HttpSendRequest 失败"
        End If
    Else
        sStatus = "HttpOpenRequest 失败"
    End If
Else
    sStatus = "InternetConnect 失败"
End If
SetStatus sStatus
If sStatusCode = "200" Or sStatusCode = "206" Then
    '根据响应码判断请求结果
    SaveData
Else
    SetStatus sStatusCode & "请求失败"
End If
StartDownFile = True
End Function
```

以上代码是下载函数，首先打开 Internet 连接，接着打开请求，加入所需的请求标题字段，然后发送请求等待服务器的响应，读取服务器的响应后，根据响应的内容（一般是根据响应码）判断是否请求成功，然后决定是否读取数据并存盘。

```
'该函数用于读取和保存请求文件的数据
Public Function SaveData() As Boolean
Dim sReadBuffer(2048)   As Byte     '为 InternetReadFile 函数提供字符串作为数据接收缓冲区
Dim lNumberOfBytesRead  As Long     '调用 InternetReadFile 函数读取的字节数
Dim lTotalBytesRead     As Long         '一共读取的字节数
Dim bDoLoop             As Boolean      '调用 InternetReadFile 的返回值
Dim i                As Integer
On Error GoTo errReadUrl
SetStatus "正在读取数据 "
If Len(mFile) = 0 Then
    SaveData = False
    Exit Function
End If
'判断打开的 HttpOpenRequest 句柄是否有效
If CBool(hHttpOpenRequest) Then
        bDoLoop = True
    Dim iFileNum As Integer
```

```
        While bDoLoop And bCancel = False
            DoEvents: DoEvents: DoEvents
            '读取缓冲区的数据
            bDoLoop = InternetReadImg(hHttpOpenRequest, sReadBuffer(0), 2048,
lNumberOfBytesRead)
            If Not CBool(bDoLoop) Then Exit Function
            lTotalBytesRead = lTotalBytesRead + lNumberOfBytesRead
            If CBool(lNumberOfBytesRead) Then
                If iFileNum = 0 Then
                    SetStatus "Reading Url:开始下载数据"
                    iFileNum = FreeFile
                    Open CStr(mFile) For Binary As iFileNum
                    Seek #iFileNum, ReceiveBytes + 1
                End If
                '保存数据
                For i = 0 To lNumberOfBytesRead - 1
                    Put #iFileNum, , sReadBuffer(i)
                Next i
                SetStatus "接收字节", lNumberOfBytesRead
                ReceiveBytes = ReceiveBytes + lNumberOfBytesRead
            Else
                bDoLoop = False
            End If
        Wend
        InternetCloseHandle (hHttpOpenRequest)
        If iFileNum <> 0 Then
            Close (iFileNum)
            SaveData = True
        Else
            SaveData = False
        End If
    Else
        SetStatus "下载过程中出错了"
        Exit Function
    End If
    If bCancel = False Then
        SetStatus "数据读取完毕"
    Else
        SetStatus "取消下载"
    End If
```

```
    Exit Function
    errReadUrl:
        SetStatus "下载过程中出错了"
    End Function
```

上面的函数用于请求下载成功读取服务器返回的数据。但注意要使用读取二进制数据的方法。

```
    '用于根据状态字符串进行相应的操作
    Sub SetStatus(sStatus As String, Optional lReadNum As Long)
    If lReadNum = 0 Then
        '如果没有读取字节数的参数，显示信息
        frmDown.txtInfo.Text = frmDown.txtInfo.Text & sStatus & DownUrl & vbCrLf
    Else
        '记录下载文件的信息
        mDownInfo(WhichDown).mGetSize = ReceiveBytes + lReadNum
        If mDownInfo(WhichDown).mSize = 0 Then
            mDownInfo(WhichDown).mSize = mFlen
            frmDown.LView.ListItems(WhichDown).SubItems(1) = mFlen
        End If
        frmDown.LView.ListItems(WhichDown).SubItems(2) = ReceiveBytes + lReadNum
        If WhichDown = frmDown.LView.SelectedItem.Index Then
            '如果当前下载的是在 LView 选中的任务，画下载情况图
            frmDown.DrawDownPic WhichDown, lReadNum, CLng(mFlen), ReceiveBytes
        End If
    End If
    '设置 Lview 中的 SmallIcon
    If InStr(1, sStatus, "数据读取完毕") > 0 Then
           frmDown.LView.SelectedItem.SmallIcon = "ok"
    ElseIf InStr(1, sStatus, "取消下载") > 0 Or InStr(1, sStatus, "下载过程中出错了") > 0 Or
    InStr(1, sStatus, "请求失败") > 0 Then
           frmDown.LView.SelectedItem.SmallIcon = "stop"
    ElseIf InStr(1, sStatus, "开始下载") > 0 Then
           frmDown.LView.SelectedItem.SmallIcon = "start"
    End If
    DoEvents: DoEvents
    End Sub
```

上面代码用于设置下载的状态，如提示信息等。

2. 窗体 frmDown

```
    Option Explicit
```

```
Dim Xitem As ListItem
Dim mDownInfoSave As DownInfoSave

'声明正在下载的任务的类变量
Dim DownJet(MaxDown) As clsDown

Private Sub add_Click()
frmAdd.Show vbModal
End Sub

Private Sub Form_Load()
Dim i As Integer
CDlg.FileName = App.Path & "\TEMP"
'创建两个类对象，每个对象负责一个下载任务
For i = 1 To MaxDown
   Set DownJet(i) = New clsDown
Next i
'初始化 ListView 控件
LView.ColumnHeaders.Clear
LView.ColumnHeaders.Add,,"URL 地址",LView.Width - 240 * Screen.TwipsPerPixelX
LView.ColumnHeaders.Add , , "大小", 80 * Screen.TwipsPerPixelX
LView.ColumnHeaders.Add , , "已下载大小", 80 * Screen.TwipsPerPixelX
LView.ColumnHeaders.Add , , "时间", 80 * Screen.TwipsPerPixelX
'从数据文件中读取下载任务的信息，加入到 ListView 中
Dim Fnum As Integer
Dim mFname As String
'保存下载任务信息的文件
mFname = App.Path & "\Downjet.djt"
Fnum = FreeFile
Open mFname For Random As #Fnum Len = Len(mDownInfoSave)
i = 1
While Not EOF(Fnum)
   ReDim Preserve mDownInfo(i)
   Get #Fnum, i, mDownInfoSave
   If Not EOF(Fnum) Then
      mDownInfo(i).mFile = Trim(mDownInfoSave.mFile)
      mDownInfo(i).mGetSize = mDownInfoSave.mGetSize
      mDownInfo(i).mIndex = mDownInfoSave.mIndex
      mDownInfo(i).mProxy = Trim(mDownInfoSave.mProxy)
      mDownInfo(i).mAuthId = Trim(mDownInfoSave.mAuthId)
```

```
            mDownInfo(i).mAuthPass = Trim(mDownInfoSave.mAuthPass)
            mDownInfo(i).mProxyPort = mDownInfoSave.mProxyPort
            mDownInfo(i).mSize = mDownInfoSave.mSize
            mDownInfo(i).mUrl = Trim(mDownInfoSave.mUrl)
            mDownInfo(i).mUseProxy = mDownInfoSave.mUseProxy
            If mDownInfo(i).mGetSize <> mDownInfo(i).mSize Then
               If Dir(mDownInfo(i).mFile) <> "" And mDownInfo(i).mFile <> "" Then
                  mDownInfo(i).mGetSize = FileLen(mDownInfo(i).mFile)
               Else
                  mDownInfo(i).mGetSize = 0
               End If
            End If
            AddUrl mDownInfo(i).mUrl, mDownInfo(i).mSize, mDownInfo(i).mGetSize,
mDownInfo(i).mFile
            i = i + 1
         End If
      Wend
      Close Fnum
      End Sub

      Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
      Dim i As Integer, j As Integer
      Dim Fnum As Integer
      Dim mFname As String
      '往文件 DownJet.djt 中保存在 ListView 中的下载任务
      mFname = App.Path & "\Downjet.djt"
      If Dir(mFname) <> "" Then
         Kill mFname
      End If
      Fnum = FreeFile
      j = 1
      Open mFname For Random As #Fnum Len = Len(mDownInfoSave)
      For i = 1 To UBound(mDownInfo) - 1
         If mDownInfo(i).mUrl <> "" And LView.ListItems(i).SmallIcon <> "delete"
Then
            mDownInfoSave.mFile = mDownInfo(i).mFile
            mDownInfoSave.mGetSize = mDownInfo(i).mGetSize
            mDownInfoSave.mIndex = mDownInfo(i).mIndex
            mDownInfoSave.mProxy = mDownInfo(i).mProxy
            mDownInfoSave.mAuthId = mDownInfo(i).mAuthId
```

```
            mDownInfoSave.mAuthPass = mDownInfo(i).mAuthPass
            mDownInfoSave.mProxyPort = mDownInfo(i).mProxyPort
            mDownInfoSave.mSize = mDownInfo(i).mSize
            mDownInfoSave.mUrl = mDownInfo(i).mUrl
            mDownInfoSave.mUseProxy = mDownInfo(i).mUseProxy
            Put #Fnum, j, mDownInfoSave
            j = j + 1
        End If
    Next i
    Close Fnum
    End
    End Sub

    Private Sub LView_DblClick()
    '如果选中的任务处于下载状态则停止，否则开始下载
    Dim i As Integer
    Dim mSel2 As Integer
    mSel2 = LView.SelectedItem.Index
    '如果选中的已经下载完毕，显示下载完毕提示
    If     mDownInfo(mSel2).mGetSize     =     mDownInfo(mSel2).mSize     And
mDownInfo(mSel2).mSize > 0 Then
        lblTishi.Caption = mDownInfo(mSel2).mFile & "已经下载完毕!!!"
        txtInfo.Text = txtInfo.Text & lblTishi.Caption & vbCrLf
        Exit Sub
    End If
    '如果正在下载，提示并退出该过程
    If LView.SelectedItem.SmallIcon = "start" Then
        lblTishi.Caption = "取消下载" & mDownInfo(mSel2).mUrl
        For i = 1 To MaxDown
            If DownJet(i).WhichDown = mSel2 Then
                DownJet(i).bCancel = True
                Exit For
            End If
        Next i
        Exit Sub
    End If

    '检查是否有空闲的下载线程
    For i = 1 To MaxDown
        If DownJet(i).bBusy = False Then
```

```
            Set DownJet(i) = Nothing
            Set DownJet(i) = New clsDown
            mSel2 = LView.SelectedItem.Index
            DownJet(i).DownUrl = LView.SelectedItem.Text
            '分析下载的 Url 是否合法
            If DownJet(i).AnalyzeUrl = False Then
                Exit Sub
            End If
            '如果第一次下载，选择路径保存下载文件
            If mDownInfo(mSel2).mFile = "" Then
                '如果取消保存则退出该过程，取消下载
                On Error GoTo err1
                CDlg.CancelError = True
                CDlg.Flags = cdlOFNOverwritePrompt
                CDlg.FileName = DownJet(i).mFile
                CDlg.ShowSave
                DownJet(i).mFile = CDlg.FileName
                mDownInfo(mSel2).mFile = DownJet(i).mFile
            Else
                DownJet(i).mFile = mDownInfo(mSel2).mFile
            End If
            DownJet(i).bBusy = True
            '下载任务索引
            DownJet(i).WhichDown = mSel2
            '下载的文件总长度
            DownJet(i).mFlen = mDownInfo(mSel2).mSize
            '已经下载的文件的大小
            DownJet(i).ReceiveBytes = mDownInfo(mSel2).mGetSize
            '****设置代理服务器选项
            DownJet(i).mProxy = mDownInfo(mSel2).mProxy
            DownJet(i).mProxyPort = mDownInfo(mSel2).mProxyPort
            DownJet(i).mAuthId = mDownInfo(mSel2).mAuthId
            DownJet(i).mAuthPass = mDownInfo(mSel2).mAuthPass
            '****
            '开始下载
            DownJet(i).StartDown
            Exit For
        End If
    Next i
    err1:
```

```
    End Sub

    '向 ListView 添加 Item 的过程
    Public Sub AddUrl(myUrl As String, Optional ByVal mSize As String, Optional
ByVal mGetSize As String, Optional ByVal mFile As String)
    If myUrl <> "" Then
        If Val(mGetSize) + 1 < Val(mSize) Or Val(mSize) = 0 Then
            Set Xitem = LView.ListItems.Add(, "", myUrl, "stop", "stop")
        Else
            Set Xitem = LView.ListItems.Add(, "", myUrl, "ok", "ok")
        End If
        Xitem.Tag = mFile
        Xitem.ListSubItems.Add , "size", mSize
        Xitem.ListSubItems.Add , "getsize", mGetSize
    End If
    End Sub

    Private Sub LView_MouseUp(Button As Integer, Shift As Integer, x As Single,
y As Single)
    Static mIndex As Integer
    Dim SelectDown As Integer
    SelectDown = LView.SelectedItem.Index
    If SelectDown < 1 Then Exit Sub
    '在 txtfile 中显示选中的下载任务信息
    txtFile.Text = "URL:        " & mDownInfo(SelectDown).mUrl & vbCrLf
    txtFile.Text = txtFile.Text & "文件名:     " & mDownInfo(SelectDown).mFile &
vbCrLf
    txtFile.Text = txtFile.Text & "大小:       " & mDownInfo(SelectDown).mSize &
"字节" & vbCrLf
    txtFile.Text = txtFile.Text & "已下载大小:  " & mDownInfo(SelectDown).mGetSize
& "字节" & vbCrLf
    txtFile.Text = txtFile.Text & "代理服务器:  " & mDownInfo(SelectDown).mProxy
& vbCrLf
    '在 picturebox 控件中显示当前选中的下载任务的 block 信息
    Pic(0).Cls
    DrawDownPic          0,          0,          mDownInfo(SelectDown).mSize,
mDownInfo(SelectDown).mGetSize
    '如果按了鼠标右键弹出删除菜单
    If Button = 2 Then
        PopupMenu menusetup
```

```
    End If
    End Sub

    '接收拖放的信息，如果是可下载的 Url，加到 ListView 中
    Private Sub LView_OLEDragDrop(Data As MSComctlLib.DataObject, Effect As Long,
Button As Integer, Shift As Integer, x As Single, y As Single)
    If Data.GetFormat(vbCFText) Then
       If vbOK = MsgBox("确定要下载" & Data.GetData(vbCFText), vbOKCancel) Then
          AddNewUrl Data.GetData(vbCFText)
       End If
    End If
    End Sub

    Private Sub menuadd_Click()
    '加入下载任务
    frmAdd.Show vbModal
    End Sub

    Private Sub menuDel_Click()
    '删除下载任务
    LView.ListItems(LView.SelectedItem.Index).SmallIcon = "delete"
    End Sub

    Private Sub menuquit_Click()
    '退出程序
    Unload Me
    End Sub

    '根据接收到的文件长度，已经下载长度的信息在 Pic 画 Block 图
    'mflen：文件长度
    'mNum：接收到的字节数
    'ReceiveBytes：已经接收到的字节数
    Public Sub DrawDownPic(Index As Integer, mNum As Long, Optional mFlen As Long,
Optional ReceiveBytes As Long)
    Dim Getnum As Long
    Getnum = ReceiveBytes
    Dim TGetNum As Long
    Dim i, j As Long
    Dim kk1 As Long, kk2 As Long
    If mNum = 0 Then
```

```
    Getnum = 0
End If
If Getnum = 0 Then
    Pic(0).FillColor = vbWhite
    kk1 = mFlen / 4096
    j = 0
    For i = 1 To mFlen / 4096
        Pic(0).Circle ((i - j * 50) * 120 + 0, j * 120 + 100), 50, vbBlack
        j = Fix(i / 50)
    Next i
    Pic(0).FillColor = &HFF0000
End If
TGetNum = Getnum
If Getnum = 0 And ReceiveBytes > 0 Then
    '加上以前已经接收到的
    Getnum = ReceiveBytes
End If
Getnum = Getnum + mNum
kk1 = Fix(TGetNum / 4096)
kk2 = Fix(Getnum / 4096)
j = Fix(kk1 / 50) + 1
For i = kk1 To kk2
    Pic(0).Circle ((i - j * 50) * 120 + 0, j * 120 + 100), 50, vbRed
    j = Fix(i / 50)
Next i
End Sub

'加入新的下载任务
Public Function AddNewUrl(myUrl As String)
    Dim i As Integer
    For i = 1 To LView.ListItems.Count
        If LView.ListItems(i).Text = myUrl Then
            MsgBox "该 URL 已经在下载队列中了！"
            Exit Function
        End If
    Next i
    AddUrl myUrl
    Dim kk As Integer
    kk = UBound(mDownInfo)
    ReDim Preserve mDownInfo(kk + 1)
```

```
    mDownInfo(kk).mUrl = myUrl
End Function
```

3. **窗体** frmAdd

该窗体是用于加入新的下载任务。在本实例中，由于不必实现通过代理服务器下载而没有加入代理服务器选项。整个窗体运行时界面如图 7-15 **所示。**

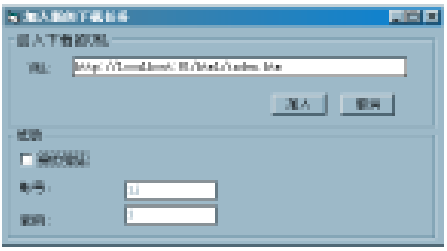


图 7-15　**加入新的下载任务**

代码如下：

```
Option Explicit

Private Sub chkAuth_Click()
If chkAuth.Value = 1 Then
    txtID.Enabled = True
    txtPass.Enabled = True
Else
    txtID.Enabled = False
    txtPass.Enabled = False
End If
End Sub

Private Sub cmdAdd_Click(Index As Integer)
If Index = 0 Then
    Dim i As Integer
    For i = 1 To frmDown.LView.ListItems.Count
        If frmDown.LView.ListItems(i).Text = txtUrl.Text Then
            MsgBox "该 URL 已经在下载队列中了！"
            txtUrl.Text = ""
            Exit Sub
        End If
    Next i
    frmDown.AddUrl txtUrl
    Dim kk As Integer
    kk = UBound(mDownInfo)
    ReDim Preserve mDownInfo(kk + 1)
```

```
        mDownInfo(kk).mUrl = txtUrl.Text

            If chkAuth.Value = 1 Then
                mDownInfo(kk).mAuthId = txtID.Text
                mDownInfo(kk).mAuthPass = txtPass.Text
            End If

    End If
    Unload Me
    End Sub

    Private Sub Form_Activate()
    Dim mStr1 As String
    mStr1 = Clipboard.GetText
    If InStr(1, mStr1, "http://") = 1 And Len(mStr1) < 180 Then
        txtUrl.Text = mStr1
    End If
    End Sub
```

4. **模块** ModDown

```
    Option Explicit
    Public Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal
hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal
cy As Long, ByVal wFlags As Long) As Long

    '定义保存下载任务信息的类型
    Public Type DownInfo
        mIndex As Integer        '位置索引
        mUrl As String           '地址
        mFile As String          '文件名
        mSize As Long            '文件的大小
        mGetSize As Long         '已获得的字节数
        mUseProxy As Boolean     '是否使用 proxy
        mProxy As String         '使用的代理服务器地址
        mProxyPort As Integer    '代理服务器端口
        mAuthId As String        '代理服务器的用户名
        mAuthPass As String      '代理服务器的密码
    End Type

    '用于把下载任务信息读写到随机文件的类型
```

```
Public Type DownInfoSave
    mIndex As Integer        '位置索引
    mUrl As String * 180     '地址
    mFile As String * 50     '文件名
    mSize As Long            '文件的大小
    mGetSize As Long         '已获得的字节数
    mUseProxy As Boolean     '是否使用 proxy
    mProxy As String * 50    '使用的代理服务器地址
    mProxyPort As Integer    '代理服务器端口
    mAuthId As String * 20       '代理服务器的用户名
    mAuthPass As String * 20     '代理服务器的密码
End Type

'最多同时下载的线程
Public Const MaxDown = 3

'记录每个下载任务信息的变量数组
Public mDownInfo() As DownInfo
Public hInternetSession As Long

'base64 编码函数：Base64Encode
'Instr1     编码前字符串
'Outstr1    编码后字符串
Public Function Base64Encode(InStr1 As String, OutStr1 As String)
Dim mInByte(3) As Byte, mOutByte(4) As Byte
Dim myByte As Byte
Dim i As Integer, LenArray As Integer, j As Integer
Dim myBArray() As Byte
myBArray() = StrConv(InStr1, vbFromUnicode)
LenArray = UBound(myBArray) + 1
For i = 0 To LenArray Step 3
    If LenArray - i = 0 Then
        Exit For
    End If
    If LenArray - i = 2 Then
        mInByte(0) = myBArray(i)
        mInByte(1) = myBArray(i + 1)
        Base64EncodeByte mInByte, mOutByte, 2
    ElseIf LenArray - i = 1 Then
        mInByte(0) = myBArray(i)
```

```
            Base64EncodeByte mInByte, mOutByte, 1
        Else
            mInByte(0) = myBArray(i)
            mInByte(1) = myBArray(i + 1)
            mInByte(2) = myBArray(i + 2)
            Base64EncodeByte mInByte, mOutByte, 3
        End If
        For j = 0 To 3
            OutStr1 = OutStr1 & Chr(mOutByte(j))
        Next j
    Next i
    End Function

    Private Sub Base64EncodeByte(mInByte() As Byte, mOutByte() As Byte, Num As
Integer)
    Dim tByte As Byte
    Dim i As Integer
    If Num = 1 Then
        mInByte(1) = 0
        mInByte(2) = 0
    ElseIf Num = 2 Then
        mInByte(2) = 0
    End If
    tByte = mInByte(0) And &HFC
    mOutByte(0) = tByte / 4
    tByte = ((mInByte(0) And &H3) * 16) + (mInByte(1) And &HF0) / 16
    mOutByte(1) = tByte
    tByte = ((mInByte(1) And &HF) * 4) + ((mInByte(2) And &HC0) / 64)
    mOutByte(2) = tByte
    tByte = (mInByte(2) And &H3F)
    mOutByte(3) = tByte
    For i = 0 To 3
        If mOutByte(i) >= 0 And mOutByte(i) <= 25 Then
            mOutByte(i) = mOutByte(i) + Asc("A")
        ElseIf mOutByte(i) >= 26 And mOutByte(i) <= 51 Then
            mOutByte(i) = mOutByte(i) - 26 + Asc("a")
        ElseIf mOutByte(i) >= 52 And mOutByte(i) <= 61 Then
            mOutByte(i) = mOutByte(i) - 52 + Asc("0")
        ElseIf mOutByte(i) = 62 Then
            mOutByte(i) = Asc("+")
```

```
        Else
            mOutByte(i) = Asc("/")
        End If
    Next i
    If Num = 1 Then
        mOutByte(2) = Asc("=")
        mOutByte(3) = Asc("=")
    ElseIf Num = 2 Then
        mOutByte(3) = Asc("=")
    End If
    End Sub
    '编码函数：EncodeStr
    'Str1   编码前字符串
    '返回值 编码后字符串
    Public Function EncodeStr(Str1 As String) As String
        Dim OutStr1 As String
        Call Base64Encode(Str1, OutStr1)
        EncodeStr = OutStr1
    End Function
```

5. **模块** ModWinInet

该模块用于声明使用到的 WinInet **函数及相应的常数变量。**

```
    Option Explicit

    '打开 Internet 连接，得到其他 WinInet 函数所用的句柄
    Public Declare Function InternetOpen Lib "wininet.dll" Alias "InternetOpenA"
_
    (ByVal sAgent As String, ByVal lAccessType As Long, ByVal sProxyName As String,
_
    ByVal sProxyBypass As String, ByVal lFlags As Long) As Long
    '字符串常量
    Public Const scUserAgent = "DownJetApi"

    '使用注册表中的设置
    Public Const INTERNET_OPEN_TYPE_PRECONFIG = 0

    '打开 Http 连接会话
    Public Declare Function InternetConnect Lib "wininet.dll" Alias
"InternetConnectA" _
    (ByVal hInternetSession As Long, ByVal sServerName As String, ByVal nServerPort
```

```
As Integer, _
    ByVal sUsername As String, ByVal sPassword As String, ByVal lService As Long,
_
    ByVal lFlags As Long, ByVal lContext As Long) As Long

    'TCP/IP 的端口
    Public Const INTERNET_DEFAULT_FTP_PORT = 21
    Public Const INTERNET_DEFAULT_GOPHER_PORT = 70
    Public Const INTERNET_DEFAULT_HTTP_PORT = 80
    Public Const INTERNET_DEFAULT_HTTPS_PORT = 443
    Public Const INTERNET_DEFAULT_SOCKS_PORT = 1080

    '请求的服务
    Public Const INTERNET_SERVICE_FTP = 1
    Public Const INTERNET_SERVICE_GOPHER = 2
    Public Const INTERNET_SERVICE_HTTP = 3

    '打开 Http 请求
    Public Declare Function HttpOpenRequest Lib "wininet.dll" Alias
"HttpOpenRequestA" _
    (ByVal hHttpSession As Long, ByVal sVerb As String, ByVal sObjectName As String,
ByVal sVersion As String, _
    ByVal sReferer As String, ByVal something As Long, ByVal lFlags As Long, ByVal
lContext As Long) As Long

    '不从 Cache 下载数据
    Public Const INTERNET_FLAG_RELOAD = &H80000000

    '向 HTTP 服务器发送请求
    Public Declare Function HttpSendRequest Lib "wininet.dll" Alias
"HttpSendRequestA" (ByVal _
    hHttpRequest As Long, ByVal sHeaders As String, ByVal lHeadersLength As Long,
sOptional As _
    Any, ByVal lOptionalLength As Long) As Integer

    '取得请求或响应的标题字段
    Public Declare Function HttpQueryInfo Lib "wininet.dll" Alias "HttpQueryInfoA"
_
    (ByVal hHttpRequest As Long, ByVal lInfoLevel As Long, ByRef sBuffer As Any,
_
```

```
    ByRef lBufferLength As Long, ByRef lIndex As Long) As Integer

    ' HttpQueryInfo 的 lInfoLevel 的值
    Public Const HTTP_QUERY_CONTENT_TYPE = 1
    Public Const HTTP_QUERY_CONTENT_LENGTH = 5
    Public Const HTTP_QUERY_EXPIRES = 10
    Public Const HTTP_QUERY_LAST_MODIFIED = 11
    Public Const HTTP_QUERY_PRAGMA = 17
    Public Const HTTP_QUERY_VERSION = 18
    Public Const HTTP_QUERY_STATUS_CODE = 19
    Public Const HTTP_QUERY_STATUS_TEXT = 20
    Public Const HTTP_QUERY_RAW_HEADERS = 21
    Public Const HTTP_QUERY_RAW_HEADERS_CRLF = 22
    Public Const HTTP_QUERY_FORWARDED = 30
    Public Const HTTP_QUERY_SERVER = 37
    Public Const HTTP_QUERY_USER_AGENT = 39
    Public Const HTTP_QUERY_SET_COOKIE = 43
    Public Const HTTP_QUERY_REQUEST_METHOD = 45

    '取得请求标题字段的标志
    Public Const HTTP_QUERY_FLAG_REQUEST_HEADERS = &H80000000

    '从 HttpOpenRequest 函数打开的 URL 请求中读取数据
    Public Declare Function InternetReadFile Lib "wininet.dll" _
    (ByVal hFile As Long, ByVal sBuffer As String, ByVal lNumBytesToRead As Long,
_
    lNumberOfBytesRead As Long) As Integer

    '从 HttpOpenRequest 函数打开的 URL 请求中读取数据(用于读取二进制文件)
    Public   Declare   Function   InternetReadImg   Lib   "wininet.dll"   Alias
"InternetReadFile" _
          (ByVal hFile As Long, lpBuffer As Any, ByVal dwNumberOfBytesToRead As
Long, _
           lNumberOfBytesRead As Long) As Integer

    '关闭 WinInet 函数打开的句柄
    Public Declare Function InternetCloseHandle Lib "wininet.dll" _
    (ByVal hInet As Long) As Integer

    '用于获得或设置 Internet 选项
```

```
    Public Declare Function InternetQueryOption Lib "wininet.dll" Alias
"InternetQueryOptionA" _
    (ByVal hInternet As Long, ByVal lOption As Long, ByRef sBuffer As Any, ByRef
lBufferLength As Long) As Integer
    '函数 InternetQueryOption 的 lOption 参数值，用于返回 WinInet 函数的版本信息
    Public Const INTERNET_OPTION_VERSION = 40
    '函数 InternetQueryOption 的 sBuffer 参数值，包含版本信息的类型
    Public Type tWinInetDLLVersion
       lMajorVersion As Long
       lMinorVersion As Long
    End Type

    '增加请求的标题字段
    Public Declare Function HttpAddRequestHeaders Lib "wininet.dll" Alias
"HttpAddRequestHeadersA" _
    (ByVal hHttpRequest As Long, ByVal sHeaders As String, ByVal lHeadersLength
As Long, _
    ByVal lModifiers As Long) As Integer
    '函数 HttpAddRequestHeaders 的 lModifiers 参数值
    '请求的标题字段不存在时才增加，否则出错
    Public Const HTTP_ADDREQ_FLAG_ADD_IF_NEW = &H10000000

    '请求的标题字段不存在时增加，否则更新
    Public Const HTTP_ADDREQ_FLAG_ADD = &H20000000
    '如果标题字段的值为空，则删除，不为空则代替
    Public Const HTTP_ADDREQ_FLAG_REPLACE = &H80000000
```

7.6　HTTP 代理服务器高级开发

代理服务器是目前应用非常广泛的一种服务器，在很多中小型的企业院校，由于到外界只有单一出口，所以内部机器要想连接到外部的站点，就必须通过代理服务器。其中浙江大学的应用就是一个例子。当通过 IE 连接外部站点的时候，需要输入代理服务器的地址，用户名和密码。

使用代理来间接访问网络资源，一般基于以下几个原因：

- **在局域网情况下，可以使用的外部 IP 地址资源一般是比较有限的，所以要通过网络代理服务，让内部 IP 主机也可以访问 Internet 资源。**
- **由于安全方面的考虑，需要防火墙来控制内部网络和 Internet 的信息传输。网络代理这个时候就可以作为通过防火墙的门户。**
- **由于没有和需要访问的目标主机直接打交道，因此客户端可以隐藏自身的 IP 地址。**

目标主机只能知道代理服务器的 IP 地址。

- 使用网络代理（主要是 HTTP 网络代理）,可以利用它的 cache 功能减少网络流量，提高浏览效率。
- 使用代理，可以绕过一些网络上的访问限制，获取直接访问不能获得的信息。

本节主要介绍 HTTP 代理服务器的开发，通过代理服务器的开发，可以更加容易理解客户端代理程序以及代理服务器的原理。

7.6.1 建立工程项目

该代理服务器很简单，只需要简单的基本控件，重要是增加两个 Winsock 控件，设计界面如图 7-16 所示。

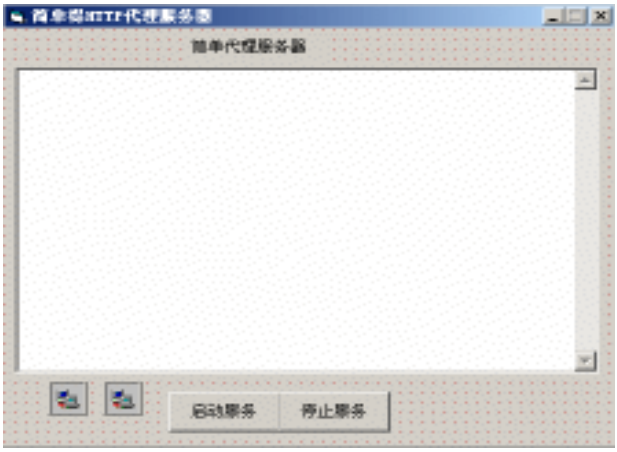


图 7-16 代理服务器设计界面

该程序只有一个窗体，在实际开发的时候，可以考虑最小化的时候，放在任务栏的托盘处。窗体名称为“ myproxy ”。

代理服务器工作界面如图 7-17 所示。

图 7-17 代理服务器工作界面

程序启动以后，必须单击“启动服务”按钮才可以完成代理工作。要使用代理服务器，必须通过 IE 设置代理服务器。设置如下，如果是普通的局域网用户，设置代理服务器按照如下步骤，首先打开 IE 浏览器，进入菜单【工具】->【Internet 选项】，如图 7-18 所示。

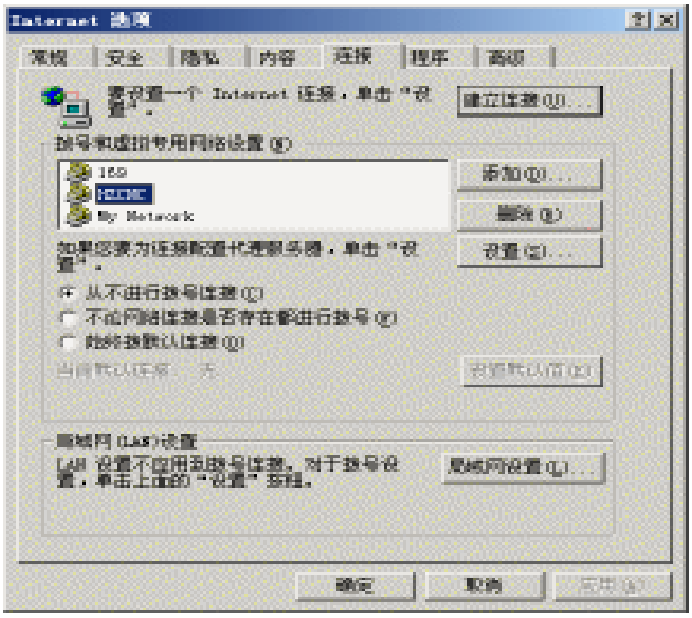


图 7-18　IE 的选项设置

单击“局域网设置”按钮，进入代理设置界面，如图 7-19：

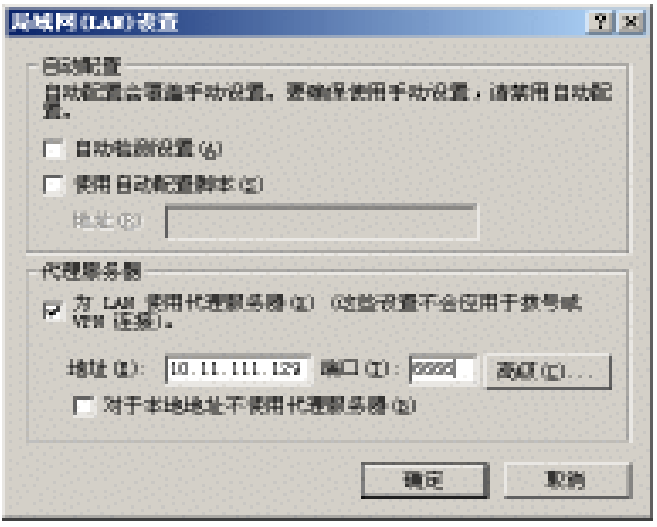


图 7-19　代理服务器设置

由于笔者使用的是网通宽带，因此要在自己机器上测试该程序，需要通过网通的选项进行设置，网通的代理设置，在图 7-19 中有一项是“HZCNC”，鼠标选中，然后单击“设置”按钮，弹出设置界面如图 7-20 所示。

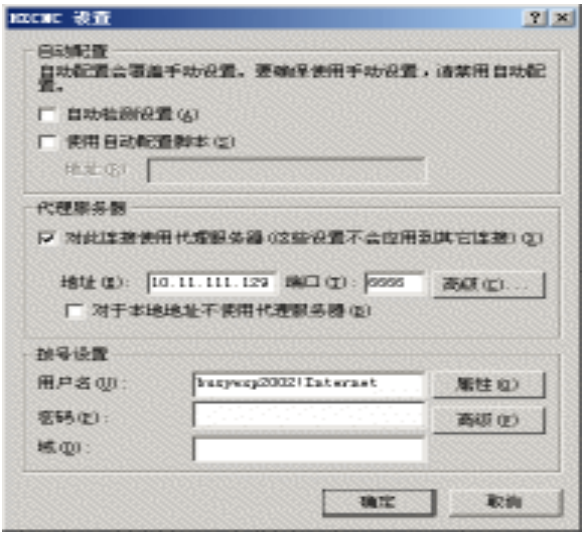


图 7-20　HZCNC 代理服务器设置

代理设置好以后，通过代理连接到远程站点，就可以很方便了，现在在 IE **里面输入** www.sohu.com **的站点，就通过刚才我们开发的代理服务器了。**

代理服务器首先弹出一个对话框，表示有客户连接，如图 7-21 **所示。**

图 7-21　代理服务器接收连接

连接成功以后，变可以从远程站点下载网页了，由于这是一个初级的代理服务器，所以只能下载第一个页面。

信息提示界面如图 7-22 **所示。**

图 7-22　代理服务器的信息提示

7.6.2　代码分析

窗体 myproxy，**其代码如下：**

```
Option Explicit

Public blnManagingData As Boolean
Public blnNewConnection As Boolean
Public socketNum As Integer

Public Sub addLog(strEvent, intEventType)
    '象文本框加入信息
    With myproxy.txtLog
    .Text = .Text & Date$ & " " & Time$ & " " & strEvent & vbCrLf
    .SelStart = Len(.Text)
    End With
End Sub
Private Sub Command1_Click()
    On Error Resume Next
    insocket(0).LocalPort = 6666
    insocket(0).Listen
    addLog "开始启动服务，端口是 6666。", 0
End Sub

Private Sub Command2_Click()
    On Error Resume Next
    insocket(0).Close
    outsocket(0).Close
    addLog "停止服务。", 0
End Sub

Private Sub Form_Load()
    addLog "程序启动，欢迎使用本代理", 0
End Sub

Private Sub insocket_ConnectionRequest(Index As Integer, ByVal requestID As
Long)
    '如果收到连接请求，接受
    'insocket(0).Close
    blnNewConnection = True
    socketNum = socketNum + 1
```

```
    Load insocket(socketNum)
    Load outsocket(socketNum)

    insocket(socketNum).Accept requestID
    addLog "接收连接来自于 " & insocket(0).RemoteHostIP, 0
End Sub

Private Sub insocket_DataArrival(Index As Integer, ByVal bytesTotal As Long)

    '改子程序等待浏览器发送 HTTP 请求头
    '当所有必须的信息获得以后，连接到真正的目的服务器，然后传送
    '请求头信息
    '错误处理
    On Error Resume Next

    '变量声名
    Static strInBuffer As String              '接受缓冲区
    Static blnHeaderRead As Boolean           '是否读 HTTP 头

    Dim strDataReceived As String             '已经获得的数据
    Dim strDestinationHost As String          '目标主机
    Dim strDestinationPort As String          '目标端口
    Dim intPos As Integer, intPos2 As Integer   '字符串位置

    '通知其他程序，数据已经被处理
    blnManagingData = True

    '如果是新连接，重新设置缓冲区和
    If blnNewConnection Then
        strInBuffer = ""
        strDestinationHost = ""
        strDestinationPort = ""
        blnHeaderRead = False
        blnNewConnection = False
    End If

    '获取数据
    insocket(Index).GetData strDataReceived
```

```
        '如果 HTTP 头完成，然后进行外部连接
        '然后退出
        If blnHeaderRead Then
            outsocket(Index).SendData strDataReceived
            Exit Sub
        End If

        '把数据放入缓冲区
        strInBuffer = strInBuffer & strDataReceived

        '从请求头信息中获取远处计算机的主机地址
        intPos = InStr(strInBuffer, "Host: ")
        If intPos > 0 Then

            intPos = intPos + Len("Host: ")
            intPos2 = InStr(intPos + 1, strInBuffer, vbCrLf)
            If intPos2 > 0 Then
                '如果查到主机地址，然后获得端口号
                '默认的端口是 80
                strDestinationHost = Mid$(strInBuffer, intPos, intPos2 - intPos)

                intPos = InStr(strDestinationHost, ":")
                If intPos > 0 Then
                    strDestinationPort      =      Int(Right$(strDestinationHost,
Len(strDestinationHost) - intPos + 1))
                    strDestinationHost = Left$(strDestinationHost, intPos - 1)
                Else
                    strDestinationPort = 80
                End If

                addLog "连接到" & strDestinationHost & ":" & strDestinationPort, 0
                '打开外部连接
    '            MsgBox "连接到:" & strDestinationHost & "   站点"
    '            MsgBox "连接到:" & strDestinationHost & "   站点"
                outsocket(Index).Connect strDestinationHost, strDestinationPort
                '等待连接成功
                While outsocket(Index).State <> sckConnected
                    DoEvents
                Wend
```

```
            '发送目前缓冲区的信息
            outsocket(Index).SendData strInBuffer

            '表示头信息已经被阅读
            blnHeaderRead = True

        End If

    End If
    '通知其他程序表示已经完成
    blnManagingData = False
End Sub
Private Sub outsocket_Close(Index As Integer)
On Error Resume Next
    addLog "外部连接关闭", 0
    While blnManagingData
        DoEvents
    Wend
    DoEvents
    'insocket(Index).Close
    Debug.Print Index & "关闭"
End Sub
Private Sub outsocket_DataArrival(Index As Integer, ByVal bytesTotal As Long)
    '接收外部数据
    '然后把数据传送给请求的客户
    On Error Resume Next
    Dim strDataReceived As String
    outsocket(Index).GetData strDataReceived
    insocket(Index).SendData strDataReceived
End Sub
'出现错误，关闭连接
Private Sub outsocket_Error(Index As Integer, ByVal Number As Integer,
Description As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile
As String, ByVal HelpContext As Long, CancelDisplay As Boolean)
On Error Resume Next
    addLog "外部连接关闭", 0
    DoEvents
'    insocket(Index).Close
    'insocket(Index).Listen
End Sub
```

第 8 章　FTP 协议及高级编程

8.1　FTP 简介

FTP（File Transfer Protocol）协议主要用来在网络上进行文件传输。在 1994 年以前，也就是 WWW（World Wide Web）协议占领 Internet 的时候，FTP 协议是除了电子邮件以外应用最广泛的 Internet 协议，通常我们把采用这种协议的应用程序称为 FTP。FTP 经过不断的改进和发展，已成为 Intenet 上普遍应用的重要信息服务工具之一。尽管 FTP 的最初设计是从传输网络文件的角度出发的，但是至今它已用于从 Internet 上获取远程主机的各类文件信息、数据，包括公用程序、程序代码及其说明文件、研究报告、技术情报、科技论文、数据和图表等。

从根本上说，FTP 协议就是在网络中各种不同的计算机之间按照 TCP/IP 协议来传输文件。FTP 协议采用的也是现在流行的 C/S（客户/服务器）模式，由 FTP 客户端程序和 FTP 服务器端程序组成。通常服务器端是远程站点，用户可以通过 Internet 网络连接到远程的 FTP 服务器站点。当然并不是所有的站点用户都有权限访问，有些服务器需要用户有一定的权限。FTP 站点可以是公共的，任何能连接上 Internet 的人都能够访问这种站点；也可以是私有的，这个时候就必须具备一定的权限，比如有相应的用户名和密码；或者是两种性质都有，这时，用户能访问和操作有权限的那部分内容。FTP 服务器端可以进行各种设置，比如有些目录读写删除属性等等。现在许多公共的 FTP 站点都会有几种权限的目录。一种是只允许用户下载，除此之外，用户不能进行其他的操作，如“ /public ”等；而有些是允许用户上传或者删除的，如“ /incoming ”等。对于一些公共的 FTP 站点，任何人都可以下载文件，而不管他在 FTP 服务器端是否有帐号，因此又可以称这种站点为“ 匿名（Anonymous）FTP ”。当连接到一个匿名服务器的时候，通常指定“ Anonymous ”作为用户名，并且以“ Guest ”或者电子邮件作为密码。

目前 FTP 服务器端的程序有很多，其中使用比较频繁的如 Serv-U，它功能强大、设置简单、性能稳定，并且能够进行多重用户设置。Windows 2000 server 与 Windows NT server 操作系统自带一个 FTP 服务端程序。客户端程序也有许多，如 CuteFTP、LeapFTP 都是现在非常流行的 FTP 客户端程序，Windows 操作系统里面也自带一个功能非常简单的 FTP 客户端程序。读者在调试自己程序的时候可以使用系统自带的 FTP 服务器程序，而且可以使用 FTP 客户端程序来熟悉 FTP 指令。在 IE 和 Netscape 等 WWW 浏览器中，也具有 FTP 客户端程序的功能，因此最开始学习的时候可以通过这两个浏览器来测试自己的 FTP 服务器是否安装正确。

在利用 FTP 协议开发应用程序的时候，服务器的功能是主要的，只有当 FTP 服务器支持各种协议和指令时，我们才可以开发客户端程序，用以和服务器建立连接。以下为 FTP 服务

器所提供的功能：

- 对远程客户机发出的“连接”与“关闭”请求作出响应，建立与解除客户机与服务器之间的连接。
- 支持开放性公共访问，如匿名访问功能。
- 支持对 FTP 文件库的管理及应用。
- 支持以不同方式（如 FTP 命令方式、E-mail 方式）和对不同类型文件（如 ASCII 文件、图形和图像文件、音响文件、其他二进制文件、各种压缩形式文件）的远程传输。
- 能够提供用户“帮助”（Help）信息。
- 支持网络文件打印。
- 进行出错处理（信息显示，系统保护），以及维护系统安全性。

FTP 客户端程序主要有以下功能，对大多数人而言，开发客户端程序是更切合实际的。它的主要功能包括：

- 向远程 FTP 服务器提出服务请求。
- 支持 FTP 用户界面的操作管理。
- 支持 FTP 对各种文件的传输、拷贝、打印和管理。

当用户使用 FTP 客户端程序访问远程服务器的时候，首先要在本地计算机启动 FTP 的客户端程序，然后向远程服务器发出请求。一旦远程计算机响应并实现连接，就在两台计算机之间建立起一条临时通路，用以执行会话命令和传输文件。在用户完成文件传送操作后，对服务器发出解除连接的请求，结束整个 FTP 会话过程。FTP 的用户和服务器依靠 FTP 协议进行的全部会话（通信活动），都是建立在 TCP/IP 协议基础上的。网络传输协议 TCP/IP 作为一个统一的工业标准，支持网络环境下不同机器平台和不同操作系统之间的文件传输和其他应用中的信息交换。对于不采用 TCP/IP 的系统，可以通过协议转换软件解决信息传输问题。因此，在 Internet 上使用 FTP 传输文件是不受机器类型的限制的。

8.2 安装设置 FTP 服务器

这一节主要是为那些刚刚学习了 VB 基础知识后，对网络还不是很熟悉，但很想进行网络高级开发的读者而写的。因为在程序的开发调试的时候，总是需要服务器支持的，但如果每次都要连接到外面的站点，就可能因为速度比较慢而影响开发进程，因此建议在开发和调试 FTP 客户端程序的时候在自己的机器上安装 FTP 服务器。本节主要介绍如何使用 Windows 2000 中自带的服务器，其他的服务器设置原理也类似，必要的时候查看帮助。具体设置步骤如下：

（1）进入开始菜单→程序→管理工具→Internet 服务管理器，界面如图 8-1 所示。

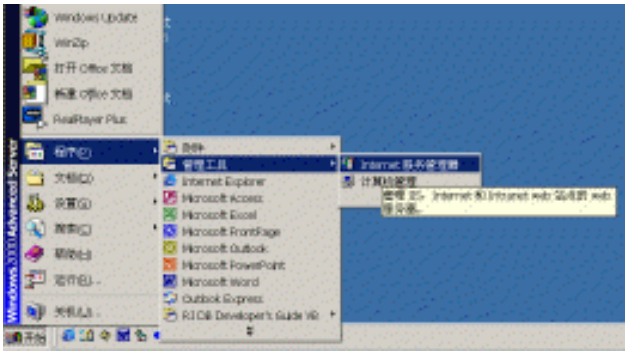


图 8-1　进入 Internet 服务管理器

（2）接下来是 Internet 信息服务界面，如图 8-2 所示。

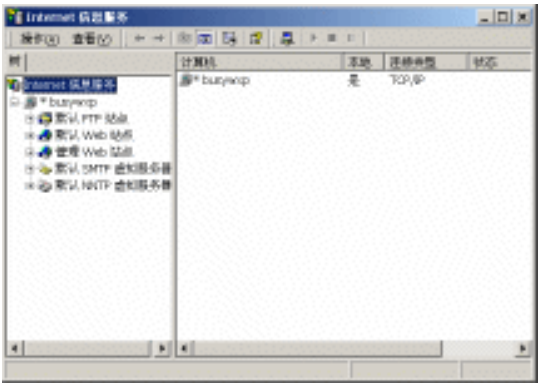


图 8-2　Internet 信息服务界面

在上图中，读者可以看到，Windows 2000 自带了许多服务器，如 FTP、WWW、SMTP 等。但这里，只需要 FTP 服务器。当系统启动的时候，这些服务都是自动打开的。在 Windows NT 系统中，同样也有 FTP 服务器，具体设置过程类似。

（3）在“默认 FTP 站点”上单击鼠标右键，单击后弹出一个显示菜单，如图 8-3 所示。

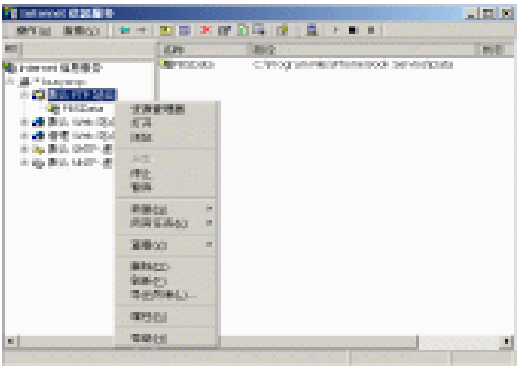


图 8-3　单击鼠标右键后的界面

这里我们只需要知道几个菜单项的功能就可以了，如要想让其它用户能够访问你的站点，就必须启动该服务，单击“启动”项就可以了。如果想停止服务，单击“停止”。服务器现在的状态是启动状态，也就是说服务已经开始，可以接受客户访问。

弄清楚启动和停止服务后，接下来就可以设置一些参数了，单击“属性”项进入设置界面。

（4）“FTP 站点”属性页设置，如图 8-4 所示。

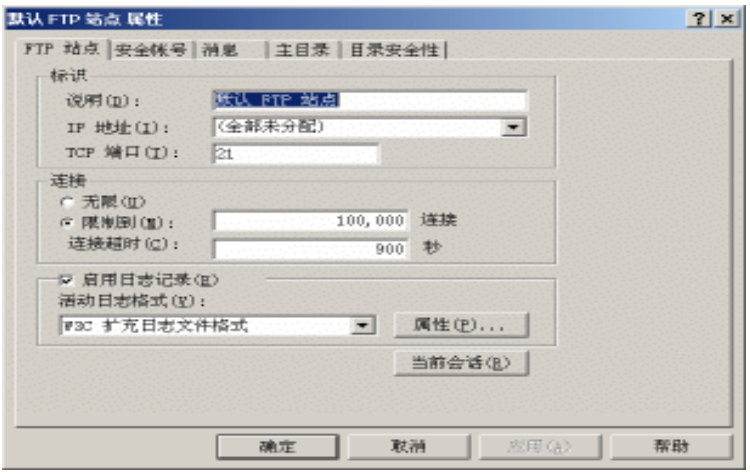


图 8-4 “FTP 站点”属性页设置

在“FTP 站点”这一个属性页里面，主要是设置服务器地址端口号，通常采用默认的端口和地址。默认的端口就是 21，默认地址是计算机的 IP 地址或者机器名。如果需要安装其他的服务器，可能需要修改端口，因为该服务器已经占用了端口 21 或者停止该服务器的服务，然后启动自己安装的其他 FTP 服务器。

（5）“安全帐号”属性页设置，如图 8-5 所示。

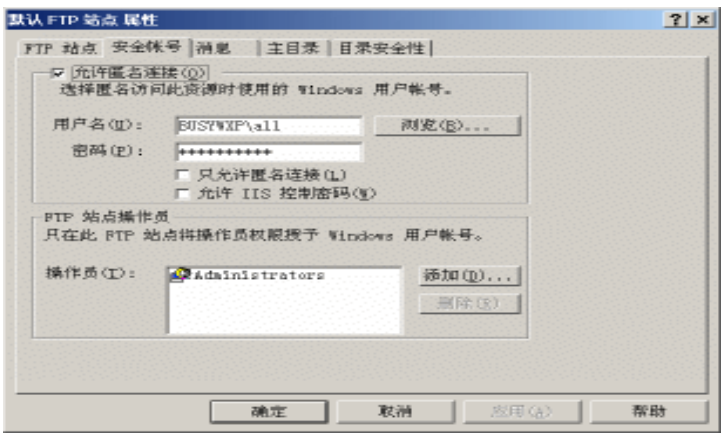


图 8-5 “安全帐号”属性页设置

在该属性页中主要是设置用户访问的帐号和密码，Windows 2000 里面的安全帐号采用的是系统登录帐号，管理员可以自己设置多个帐号和密码。在这里设置了一个用户名为“all”的用户，因此可以通过输入该帐号和密码来访问该 FTP 服务器了。

（6）“消息”属性页设置，如图 8-6 所示。

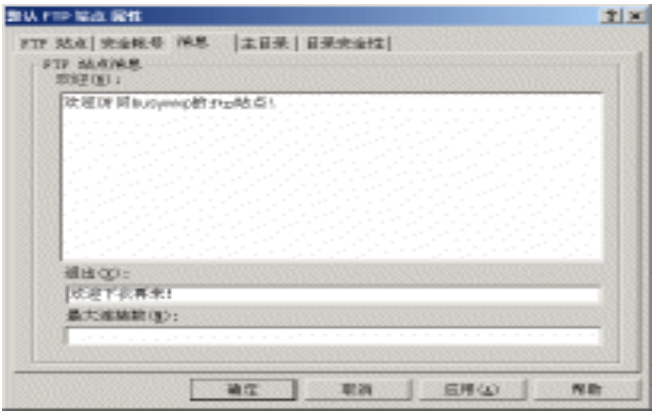


图 8-6　“消息”属性页设置

该属性页主要是设置一些用户登录系统的欢迎信息和离开时的显示信息。

（7）“主目录”属性页设置，如图 8-7 所示。

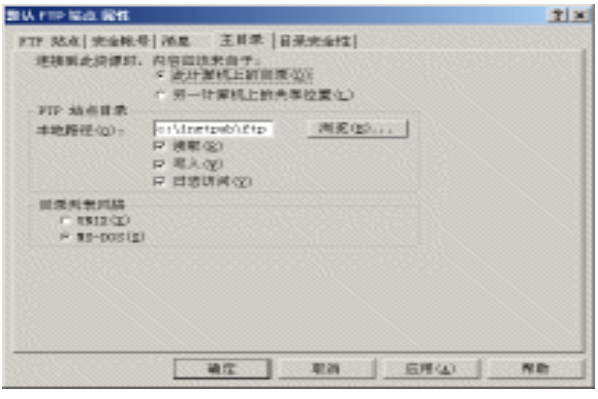


图 8-7　“主目录”属性页设置

该属性页主要是设置 FTP 服务器的路径，也就是用户连接上服务器以后，能够显示给用户的内容，通常在主目录下的文件和子目录都能被用户看见，同时还可以设置权限，如读取、写入等。

（8）“目录安全性”属性页设置，如图 8-8 所示。

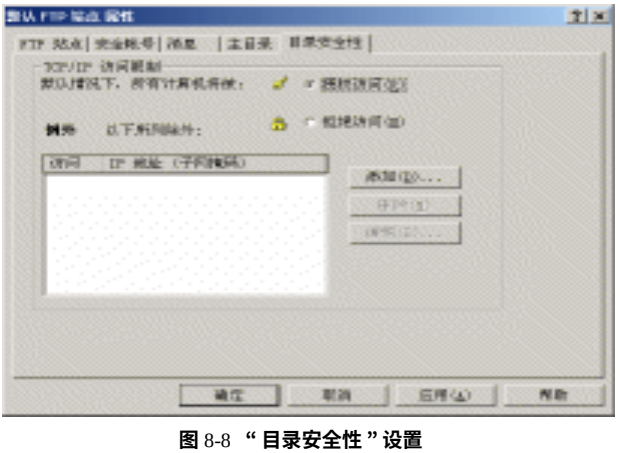


图 8-8　“目录安全性”设置

该属性页主要设置计算机的访问权限，可以拒绝某些计算机的访问，默认情况下，所有的联网计算机都是可以接受的访问的。

这样，一个简单的服务器就设置完了，对于网络初级用户，通常可以使用默认设置。

（9）检测 FTP 服务器。

下面通过 IE 浏览器来检测 FTP 服务器地址是否安装成功。在上面的设置中，使用的是默认的 FTP 文件路径即“ C:\Inetpub\ftproot ”，如图 8-9 所示的是该目录下的文件。

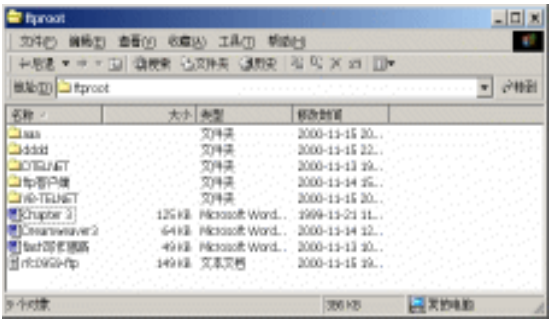


图 8-9　FTP 服务器文件目录下的文件

下面通过连接 FTP 服务器来查看是否为上面的那些文件。图 8-10 所示为通过 IE 连接服务器后的结果，首先是要求验证用户名和密码。可以输入在地址栏里面输入地址“ ftp://机器名（仅对局域网而言）”或者输入“ ftp：//IP 地址（域名）”，就可以访问服务器了。

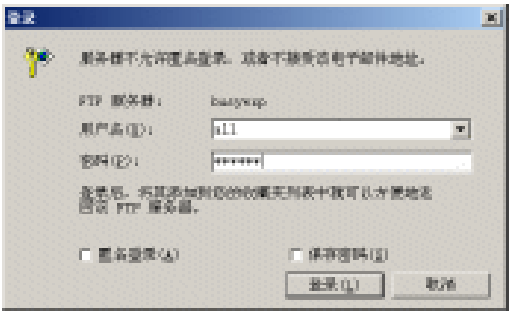


图 8-10　登录验证窗口

在图 8-10 中，输入帐号“ all ”和对应的密码，则可以进入 FTP 服务器目录，如图 8-11 所示。

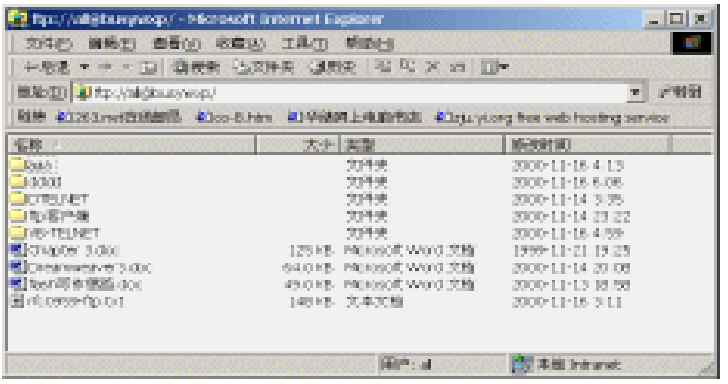


图 8-11　登录成功后显示的文件列表

可以看到，通过登录 FTP 服务器访问的文件列表和图 8-9 所示的目录是相同的，因此现在这个 FTP 服务器安装是成功的。在以后的编程中就可以通过这个 FTP 服务器来调试程序。

8.3　使用 Windows 内置 FTP 程序

在读者正式进行 FTP 客户端程序开发以前，可以先尝试使用该程序来体会一下 FTP 的工作过程。FTP 协议定义了客户发送给服务器的一系列的命令以及客户与服务器之间如何传输数据。当客户端发出一个命令后，服务器端总是会作出一定的响应。FTP 应用程序通常是一个基于字符的终端类型的应用程序。如果在 Windows 里面安装了 TCP/IP 协议，就会在 windows 的目录下面有一个 FTP.EXE 文件（在 Windows 2000 里面该文件位于目录 C:\WINNT\system32 下面）。这就是 Windows 内置的 FTP 客户端应用程序。不过要注意的是，在这个 FTP 程序里面使用的命令并不是 FTP 协议里面的命令，而是作出了一些改变，让用户使用比较熟悉的命令代替协议里的命令。在后面的实际开发中会看到这个命令和协议里的命令是有区别的。图 8-12 为该程序运行后的界面，类似于 DOS 的文本模式应用程序一样。

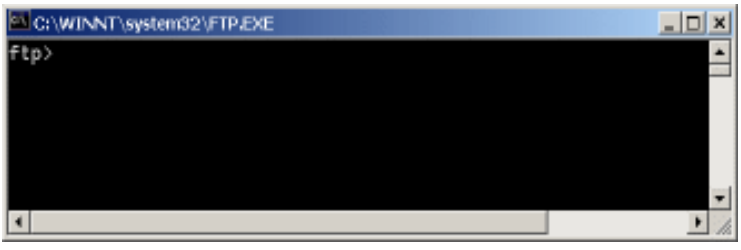


图 8-12　FTP 程序运行界面

1. 使用“help”获得命令列表

前面说了，这里使用的命令同标准的命令是不一样的，因此如何获得这些命令呢？可以通过“help”命令来获得命令列表，如图 8-13 所示。

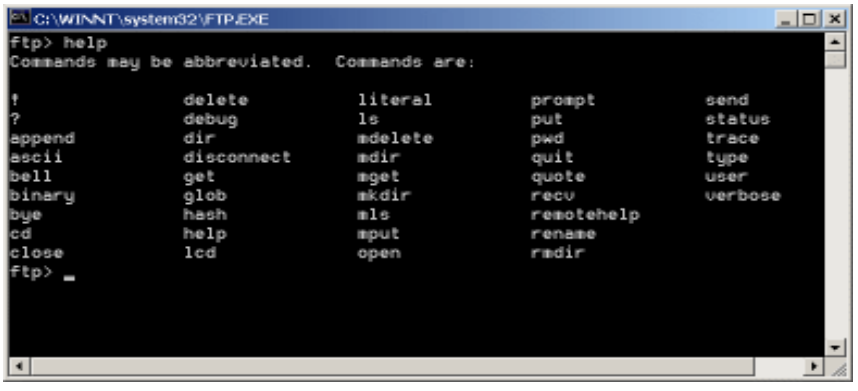


图 8-13　FTP 程序命令列表

以上就是该 FTP 程序所支持的部分命令，下面就介绍一些主要命令的使用方法。

2. 使用“open”命令连接远程服务器

当需要连接到一个站点的时候，只需要在命令行输入“open ftp 站点地址”，然后回车，程序就会试图去连接这个站点。如果连接成功，就会返回一条 220 的响应码，然后就是一条关于该服务器的版本信息。如图 8-14 所示的为成功连接到一个站点后显示的界面。

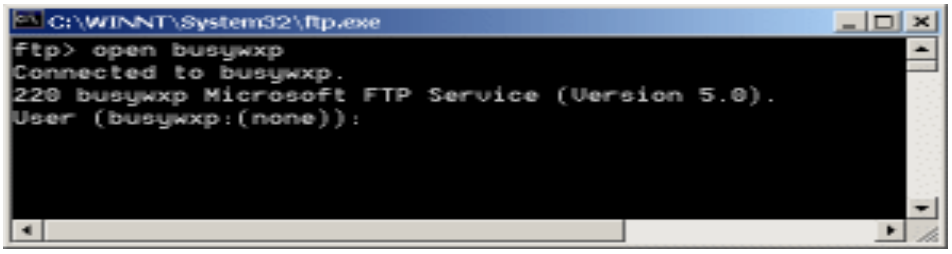


图 8-14　成功连接到站点后的界面

连接到站点以后，就会接着验证帐号和密码，如图 8-15 所示为验证通过后的界面，登录成功后显示欢迎信息。

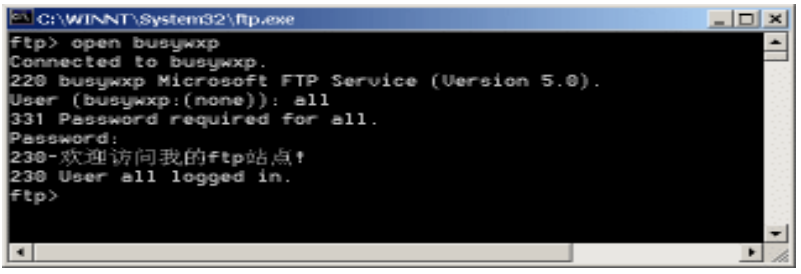


图 8-15　验证通过后的界面

3. 使用“dir”命令列出目录

实际上在 FTP 协议中并没有 dir 这个命令。这里主要是沿用了操作系统中的命令。如图 8-16 所示是使用“dir”后的结果，列出了服务器指定目录中的所有子目录和文件以及其他的一些信息，如大小、日期等。

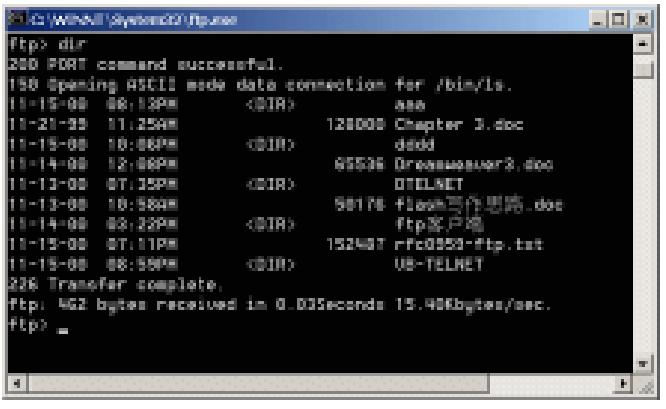


图 8-16　显示的文件列表

4. 使用“cd”命令改变目录

使用“cd”命令改变目录，如图 8-17 所示。

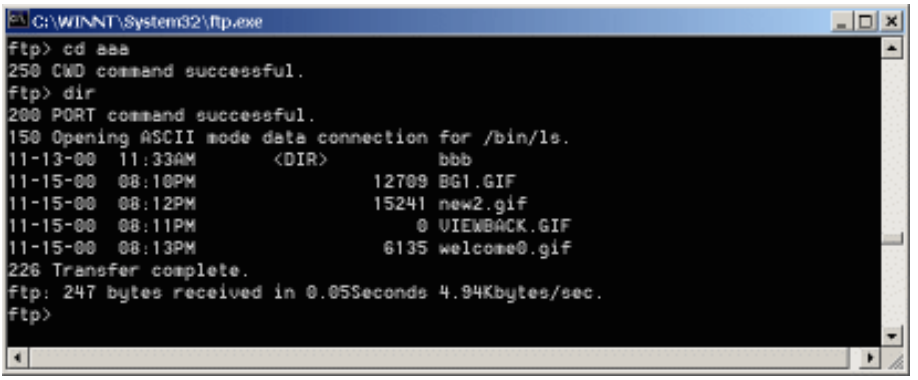


图 8-17　改变目录后的文件列表

图 8-17 是使用了命令“ cd aaa ”而进入目录“ aaa ”，然后使用“ dir ”命令打印出文件列表。同样可以使用“ cd / ”命令回到根目录。

还有其他的许多命令，比如上载文件、下载文件等。读者朋友可以自己去试试，因为本书的主要目的是进行网络的高级开发，而不是软件的使用，因而只作了部分介绍。

8.4　深入 FTP 协议

自从 FTP 诞生以来，就成为 Internet 的主要传输机制，但是作为网络传输，要在不同的计算机、操作系统、网络之间进行，总是需要相应的协议来保证，否则文件的传输就不能按照预期的方式进行。但是正是因为使用 FTP 的人非常多，从而就会有许多人会对原有的 FTP 协议进行改进，也就有了巨大数量的对 FTP 协议的补充和改进，但是这也就给程序的开发造成了一定的困难。在自己开发 FTP 程序的时候，不要加入一些改进的东西，这样可能就会造成不到的结果。如果你的软件只是针对一部分用户，那可以考虑加入一些功能，这样可能客户端程序和服务器端程序都要作些修改。如果是对外开放的 FTP 服务器站点或者是通用的 FTP 客户端程序，就应该谨慎一点。在本节介绍的 FTP 协议都是以 RFC 959 为基础的，它是标准 FTP 协议的定义。

3.4.1　FTP 命令大全

FTP 协议采用一系列简单的协议来完成文件传输的各种任务，在发送命令的时候，总是在命令的最后加上一个回车换行符，在 VB 中可以用“ vbcrlf ”来实现。以下的命令是从 Postel 和 Reynolds 所著的 RFC 929 修改而来，如果读者想查看详细内容，可以参见英文版 RFC 929。

1.　ABOT（Abort，终止）命令：

① 说明：告诉服务器终止上一次 FTP 服务命令及所有相关的数据传输。

② 用法：ABOR〔 CRLF 〕

③ 参数：无。

④ 例子：

```
SendData "ABOR" & vbCrlf
```

⑤ 注释：终止命令可以请求“ 特殊操作 ”以强行引起服务器的重视（详情请参看 RFC

959）。如果上一次命令已经完成（包括数据传输），就不会导致任何操作。服务器不会关闭控制连接，但必须关闭数据连接。

服务器在接收到此命令时可能处于两种状态下：（1）FTP 服务命令已经完成，（2）FTP 服务命令尚在处理中。

在第一种状态下，服务器关闭数据连接（如果它是打开的）并响应以 226 应答，表示已成功执行了终止命令。

在第二种状态下，服务器终止正在处理中的 FTP 服务并关闭数据连接，返回 426 应答，表示该服务请求被异常终止。然后服务器发送 226 应答，表示成功执行了终止命令。

⑥ 返回值（粗体表示成功）：

- 225 数据连接打开，没有正在进行的传输
- 226 关闭数据连接，请求的文件操作成功。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 226 连接关闭，传输终止。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。

2. ACCT（Account，帐号）命令：

① 说明：指定用户的帐号信息。这条命令只能在发送 PASS 命令并接收到 332 代码之后发送。

② 用法：ACCT<Account><CRLF>

③ 参数：Account 是用户的帐号，访问某些服务时可能另外需要它。

④ 例子：

```
SendData "ACCTN322s" & vbCrlf
```

⑤ 注释：当登录需要帐号信息时，一条成功的 PASS 命令的响应是应答代码 332。反之，如果登录不需要帐号信息，成功的 PASS 命令的应答是 230；如果在对话中后来发出的命令需要帐号信息，服务器会返回 332 或 532 应答，这取决于它在接收 ACCT 命令期间是保存还是丢弃此命令。

⑥ 返回值（粗体表示成功）：

- 202 命令还没有被实现，在此站点上是多余的。
- 230 用户已登录，请继续。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。
- 503 命令的顺序不对。
- 530 无法登录。

3. ALLO（Allocate，分配）命令：

① 说明：发送文件前在服务器上分配 X 个字节

② 用法：ALLO<NumberBytes[<MaxSize>]<CRLF>

③ 参数：NumberBytes 是一个整数，代表为该文件保留的内存字节数（使用逻辑字节大小计算）。MaxSize 是在使用记录或页数据结构时可选的最大记录或页大小。

④ 例子：

```
SendData "ALLO 3000 128" & vbCrlf
```

⑤ 注释：一些要保留足够内存以容纳将要传输的新文件的服务，会请求这条命令。对于用记录或页结构发送的文件来说，最大记录或页大小（以逻辑字节计）也可能是必需的；它以这条命令的第二个参数字段中的十进制整数表示。这第二个参数是可选的，但在出现时应该与第一个参数的 3 个 ASCII 字符<SP>P<SP>分隔开。这条命令后跟一条 STORe 或 APPEnd 命令。那些不要求事先声明最大文件大小的服务器应该把 ALLO 命令视为 NOOP（不操作），而那些只关心最大记录或页大小的服务器应该受第一个参数的值，然后忽略它。

⑥ 返回值（粗体表示成功）：

- 200　命令成功。
- 202　命令还没有被实现，在此站点上是多余的。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将被关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 504　命令中还没有实现该参数。
- 530　无法登录。

4. APPE(Append with create,附加和创建)命令：

① 说明：让服务器准备接收一个文件并指示它把这些数据附加到指定的文件名，如果指定的文件尚未存在，就创建它。

② 用法：APPE<FileName><CRLE>

③ 参数：FileName 是服务器站点上一个完全合格的路径和文件名。

④ 例子：

```
SendData "APPE" & szFileName & vbCrlf
```

⑤ 返回值（粗体表示成功）。

- 110　重新启动标记应答。在此情况下，文本是精确的，而且不会留给特殊的实现处理；它必须读取：

  MARK yyyy=mmmm

  其中 yyyy 是用户进程的数据流标记，mmmm 是服务器上的相应标记（请注意标记和“ = ”之间的空格）。
- 125　数据连接已打开，传输启动。
- 150　文件状态没问题，准备打开数据进行连接。
- 226　关闭数据连接，请求的文件操作已成功。

- 250 请求的文件操作没问题，已完成。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 425 无法打开数据连接。
- 426 连接关闭；传输中止。
- 450 请求的文件操作无法执行，文件不可用（例如文件正忙）。
- 451 请求的操作被中止，处理中发生本地错误。
- 452 请求的操作无法执行，系统的存储空间不足。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。
- 530 无法登录。
- 532 文件的存储需要帐号。
- 550 请求的操作无法执行，文件不可用（例如，找不到文件，无访问权）。
- 551 请求的操作被中止，未知的页类型。
- 552 请求的文件操作被中止，超过了分配的存储单元（对当前目录或数据集而言）。
- 553 请求的操作无法执行，不允许的文件名。

5. CDUP(Change to Parent Directory，变为父目录) 命令：

① 说明：把当前目录改为远程文件系统的根目录，无需改变登录、帐号信息或传输参数。

② 用法：CDUP<CRLF>

③ 参数：无。

④ 例子：

```
SendData "CDUP" & vbCrlf
```

⑤ 注释：CDUP 目录可改为父目录。MS-DOS 中的等效命令是 cd\。创建这条命令是为了适应 FTP 的不同操作系统。

⑥ 返回值（粗体表示成功）：

- 250 请求的文件操作正常进行，已完成。
- 421 服务不可用，关闭控制连接。
  如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。
- 530 无法登录。
- 550 请求的操作无法执行，文件不可用（例如找不到文件、无访问权等）。

6. CWD(Change Working Directory，改变工作目录) 命令：

① 说明：把当前目录改为远程文件系统的指定路径，而无需改变登录、帐号信息或传输

参数。

② 用法：CWD<Path><CRLF>

③ 参数：Path 是远程系统上的一个工作目录。

④ 例子：

```
SendData "CWD/pub/cgvb/uploads" & vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 250　请求的文件操作正常进行，已完成。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 502　命令还没有被实现。
- 530　无法登录。
- 550　请求的操作无法执行，文件不可用（例如找不到文件，无访问权）。

7. DELE（Delete，删除）命令：

① 说明：删除服务器站点上在路径名中指定的文件。

② 用法：DELE<FileName><CRLF>

③ 参数：FileName 是服务器站点上一个完全合格的路径和文件名。

④ 例子：

```
SendData "DELE temp.fil" & vbCrlf
```

⑤ 注释：如果期望有额外的保护级别（例如选项"确实要删除此文件吗？"），这应该由客户软件提供。

⑥ 返回值（粗体表示成功）：

- 250　请求的文件操作正常进行，已完成。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 450　请求的文件操作无法执行，文件不可用（例如文件正忙）。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 502　命令还没有被实现。
- 530　无法登录。
- 550　请求的操作无法执行，文件不可用（例如，找不到文件、无访问权等）。

8. HELP（Help，帮助）命令：

① 说明：让服务器通过到客户的控制连接发送有关其实现状态的帮助信息。

② 用法：HELP[<Topic>]<CRLF>

③ 参数：Topic 是一个可选的命令，或是请求哪条命令有关文本的其他参数。

④ 例子：

```
SendData "HELP" & vbCrlf
```

⑤ 注释：HELP 可以带一个参数（例如任何命令的名称），以在响应中返回更具体的信息。应答为类型 211 或 214。建议在输入 USER 命令前允许使用 HELP 命令。服务器可以使用这个应答来指定站点相关的参数，例如在对 HELP SITE 的响应中。

⑥ 返回值（粗体表示成功）：

- 211 系统状态，或系统的帮助应答。
- 214 帮助消息。
- 描述如何使用服务器或某条不常用的具体命令的方法。这个应答只对用户有用，因为帮助消息没有标准的格式。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向有命令做出这个应答。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。

9. LIST（List，列表）命令：

说明：让服务器给客户发送一份列表。

用法：LIST[<PathName>]<CRLF>

参数：PathName 是服务器系统上的一个有效路径和文件规范。

例子：

```
SendData "LIST pub/*.*" & vbCrlf
```

注释：如果路径名指定的是一个目录或其他文件组，服务器传送一份位于指定目录中的文件的列表。如果路径名指定的是一个文件，那么服务应该发送此文件的当前信息。空变元则暗指用户的当前工作目录或默认目标。数据将在类型 ASCII 或类型 EBCDIC 中通过数据连接传送（用户必须确保 ASCII 或 EBCDIC 的类型适当）。

因为系统与系统之间的文件信息可能有很大差别，所以这项信息要在程序中自动使用可能很困难，但对用户来说会很有用。

返回值如下（粗体表示成功）：

- 125 数据连接已打开，传输启动。
- 150 文件状态没问题，准备打开数据连接。
- 226 关闭数据连接，请求的文件操作已成功。
- 250 请求的文件操作正常进行，已完成。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 425 无法打开数据连接。
- 426 连接关闭，传输中止。
- 450 请求的文件操作无法执行，文件不可用（例如文件正忙）。
- 451 请求的操作被中止，处理中发生本地错误。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。

- 501　参数或变元中有语法错误。
- 502　命令还没有被实现。
- 530　无法登录。

10. MKD（Make Directory ，创建目录）命令：

① 说明：创建一个在路径名中指定的目录（如果是绝对路径名）或当前工作目录的子目录（如果是相对路径名）。

② 用法：MKD<Path><CRLF>

③参数：Path 是服务器端上的一个有效路径。

④ 例子：

```
SendData "MKD /users/johnsmith" & vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 257　“ PathName ”已创建。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 502　命令还没有被实现。
- 530　无法登录。
- 550　请求的操作无法执行，文件不可用（例如，找不到文件、无访问权等）。

11. MODE（Transfer Mode，传输模式）命令：

说明：指定传输模式。

用法：STRU<Mode><CRLF>

参数：Mode 是如下 ASCII 值的其中之一：

- S—Stream（流，默认值）
- B—Block（块）
- C—Compressed（经过压缩）

例子：

```
SendData "STRU B" & vbCrlf
```

返回值如下（粗体表示成功）：

- 200　命令成功。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。这其中包括令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 504　命令中还没有实现该参数。
- 530　无法登录。

12. NLST（Name List，名称列表）命令：

① 说明：让服务器给客户发送一份目录列表。

② 用法：NLST[<PathName>]<CRLF>

③ 参数：PathName 是服务器系统上的一个有效路径和文件规范。

④ 例子：

```
SendData "NLST /pub/cgvB" & vbCrlf
```

⑤ 注释：路径名应该指定一个目录或其他由系统指定的文件组描述符，空变元则暗指当前目录。服务器将返回一个文件名称的流，除此之外没有其他信息。数据将以 ASCII 或 EBCDIC 类型通过数据连接传送，其中的有效路径名字符串由<CRLF>或<NL>分隔（用户必须确保类型正确）。

NLST 希望返回的信息可被程序用来进一步地自动处理这些文件。例如，在一个“断点续传”功能的实现中。

⑥ 返回值（粗体表示成功）：

- 125 数据连接已打开，传输启动。
- 150 文件状态正常，准备打开数据连接。
- 226 关闭数据连接，请求的文件操作已成功。
- 250 请求的文件操作正常进行，已完成。
- 421 服务不可用，关闭控制连接。

如果某项服务获知自己即将关闭，会向所有命令做出这个应答。

- 425 无法打开数据连接。
- 426 连接关闭，传输中止。
- 450 请求的文件操作无法执行，文件不可用（例如文件正忙）。
- 451 请求的操作被中止，处理中发生本地错误。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。
- 530 无法登录。

13. NOOP（NOOP，无操作）命令：

① 说明：这是一条不进行操作的命令，即它什么都不做。

② 用法：NOOP<CRLF>

③ 参数：无。

④ 例子：

```
SendData "NOOP" & vbCrlf
```

⑤ 注释：NOOP 不会影响任何参数或以前输入的命令。除了让服务器发送一条 OK 应答外，它不指定任何操作。

⑥ 返回值（粗体表示成功）：

- 200 命令成功。
- 421 服务不可用，关闭控制连接。

- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。

14. PASS（Password，密码）命令：

① 说明：向远程系统发送用户的密码，该命令在 USER 命令后使用。

② 用法：PASS<Password><CRLF>

③ 参数：Password 是由 USER 命令指定的已注册用户密码。

④ 例子：

```
SendData "PASS mypassworD" & vbCRLF
```

⑤ 注释：在连接到一台 FTP 服务器的端口 21 并接收到一个由代码 220 打头的行，表示服务器已准备好你向它发 USER 和 PASS 命令，以登录进此 FTP 服务器之后，紧跟着发送 USER 命令。

PASS 命令应该紧跟着 USER 命令。

如果你在此 FTP 服务器上有帐号，就可以指定自己的用户名和密码。如果想匿名登录，可以指定用户名为“ Anonymous ”，而用自己的电子邮件地址当密码。

⑥ 返回值（粗体表示成功）：

- 202　命令还没有被实现，在此站点上是多余的。
- 230　用户已登录，请继续。
- 332　登录需要帐号（请参看 ACCT 命令）。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 530　无法登录。

15. PASV（Passive，被动的）命令：

① 说明：告诉服务器在一个非标准端口上收听数据连接。

② 用法：PASV<CRLF>

③ 参数：这条命令请求 Server-DTP（服务器数据传送规约）“ 收听 ”某个数据端口（该端口不是它的默认数据端口），并等待一个连接而不是在收到传输命令时初始化一个连接。对这条命令的响应包括主机地址和此服务器正在收听的端口地址。

④ 例子：

```
SendData "PASV" & vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 227　输入被动模式（h1,h2,h3,h4,p1,p2）。
- 返回值包括一个在数据连接中使用的数据端口的 HOST-PORT 规约。此参数是一个 32 位 Internet 主机地地址和一个 16 位 TCP 端口地的拼接。这个地址信息被拆分为 8 位的字段，并且每个字段的值是作为一个十进制数传输的（在字符串意义上）。H1 是 Internet 主机地址的高位字节，p1 是端口地址的高位字节。
- 421　服务不可用，关闭控制连接。

- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 503 无法登录。

16. PORT（Data Port，数据端口）命令：

① 说明：为数据连接指定一个 IP 地址和本地端口。

② 用法：PORT<Socket><CRLF>

③ 参数：Socket 是数据连接中要使用的数据端口的一个 HOST-PORT 规约。用户和服务器都有默认的数据端口，在正常情况下不需要这条命令和对它的应答。如果使用了这条命令，则参数是一个 32 个 Internet 主机地址和一个 16 位 TCP 端口地的拼接。这个地址信息被拆分为 8 位的字段，并且每个字段的值是作为一个十制数传输的（在字符串意义上）。各个字段用逗号分隔开。一条端口命令可以是这样：

PORT h1,h2,h3,p1,p2

其中 h1 是 Internet 主机地址的高位字节，p1 是本地端口的高位字节。

④ 例子：

```
SendData "PORT 199,199,199,0,33,1" & vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 200 命令成功。
- 421 服务不可用，关闭控制连接。

如果某项服务获知自己即将关闭，会向所有命令做出这个应答。

- 500 语法错误，无法识别命令。

这其中包括命令行过长之类的错误。

- 501 参数或变元中有语法错误。
- 530 无法登录。

17. PWD(Print Working Directory，打印工作目录) 命令：

① 说明：在应答中返回当前工作目录的名称。

② 用法：PWD<CRLF>

③ 参数：无

④ 例子：

```
SendData "PWD" & vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 257 “Path Name”已创建
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500 语法错误，无法识命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。
- 550 请求的操作无法执行，文件不可用（例如，找不到文件，无访问权）。

18. QUIT（Logout，注销）命令：

① 说明：终止连接。

② 用法：QUIT<CRLF>

③ 参数：无。

④ 例子：

```
SendData "QUTI" & vbCrlf
```

⑤ 注释：QUIT 将终止此 USER，如果没有正在进行的文件传送，服务器将关闭控制连接。如果文件传送正在进行，该连接仍然打开直至结果呼响应，然后服务器将其关闭。如果用户进程正在为几个 USER 传送文件，不想关闭后再逐个重新打开连接，那么应该使用 REIN 命令代替 QUIT。

控制连接的意外关闭会导致服导器采取有效的中止（Abort）和注销（Quit）操作。

⑥ 返回值（粗体表示成功）：

- 221　服务正在关闭控制连接。
- 500　语法错误，无法识别命令。

19. REIN(Reinitialize，重新初始化) 命令：

① 说明：终止一个用户。

② 用法：REIN<CRLF>

③ 参数：无。

④ 例子：

```
SendData "REIN" & vbCrlf
```

⑤ 注释：除了允许完成正在进行的传送之外，REIN 将刷新所有的 I/O 和帐号信息。所有参数都被复位到默认设置，控制连接仍然打开。这和用户在刚刚打开控制连接时的状态是一样的。如果后来想要登录，在发送 USER 命令之前送一条 REIN 命令。

⑥ 返回值（粗体表示成功）：

- 120　服务将在 nnn 分钟后准备好。
- 220　服务已为新用户准备好。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 502　命令还没有被实现。

20. REST（Restart，重新启动）命令：

① 说明：标识出文件内的数据点，将从这个点开始继续传送文件。

② 用法：REST<Marker><CRLF>

③ 参数：Marker 代表文件传送将要由此重新启动的服务器标记。

④ 例子：

```
SendData "REST 244" & vbCrlf
```

⑤ 注释：REST 不会引起文件传送，而只是跳到文件中指定的数据检验点上。RSET 后紧跟着适当的 FTP 服务命令，该命令才会引起文件的继续传送。

⑥ 返回值（粗体表示成功）：

- 无响应即表示成功。
- 350 请求的文件操作在等待更进一步的信息。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。
- 530 无法登录。

21. RETR（Retrieve，检索）命令：

① 说明：这条命令让服务器给客户传送一份在路径名中指定的文件的副本。这不会影响该文件在服务器站点上的状态和内容。

② 用法：RETR<FileName><CRLF>

③ 参数：FileName 是服务器站点上一个完全合格的路径和文件名。

④ 例子：

```
SendData "RETR /pub/cgvb/misc/somefile.zip" & vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 110 重新启动标记应答。

在此情况下，文体是精确的，而且不会留给特定的实现处理；它必须读取：

MARK yyyy=mmmm

其中 yyyy 是用户进程的数据流标记，mmmm 是服务器上的相应标记（请注意标记和“ = ”之间的空格）。

- 125 数据连接已打开，传输启动。
- 150 文件状态没问题，准备打开数据连接。
- 226 关闭数据连接，请求的文件操作已成功。
- 250 请求的文件操作正常进行，已完成。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 425 无法打开数据连接。
- 426 连接关闭，传输中止。
- 450 请求的文件操作无法执行，文件不可用（例如，文件正忙）。
- 451 请求的操作被中止，处理中发生本地错误。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 504 命令中还没有实现该参数。
- 530 无法登录。
- 550 请求的操作无法执行，文件不可用（例如找不到文件、无访问权等）。

22. RMD（Remove Directory，删除目录）命令：

① 说明：删除一个在路径名中指定的目录（如果是绝对路径名）或当前工作目录的子目录（如果是相对路径名）。

② 用法：RMD<Path><CRLF>

③ 参数：Path 是服务器端上一个完全合格的路径。

④ 例子：

```
SeddData "RMD /users/johnsmith" & vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 250　请求的文件操作没问题，已完成。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。
- 501　参数或变元中有语法错误。
- 502　命令还没有被实现。
- 530　无法登录。
- 550　请求的操作无法执行；文件不可用（例如，找不到文件，无访问权）。

23. RNFR（Rename From，把…重命名）命令：

① 说明：文件重命名进程的前一半。指定要重命名的文件的旧路径和文件名。

② 用法：RNFR<FileName><CRLF>

③ 参数：FileName 是服务器站点上一个完全有效的路径和文件名。

④ 例子：

```
SendData "RNFR source.zip" & vbCrlf
```

RNFR 后面必须紧跟着一条指定新路径和文件名的"Rename to"命令（RNTO）。

⑤ 返回值（粗体表示成功）：

- 无响应即表示成功。
- 350　请求的文件操作在等待更进一步的信息。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 450　请求的文件操作无法执行，文件不可用（例如文件正忙）。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 502　命令还没有被实现。
- 530　无法登录。
- 550　请求的操作无法执行，文件不可用（例如，找不到文件、无访问权等）。

24. RNTO（Rename TO，重命名为）命令：

① 说明：文件重命名进程的后一半。指定要重命名的文件的新路径和文件名。

② 用法：RNTO<FileName><CRLF>

③ 参数：FileName 是服务器站点上的一个有效文件名。

④ 例子：

```
SendData "RNTO destination.ziP" & vbCrlf
```

⑤ 注释：RNTO 前面必须是一条"Rename from"命令（RNFR）。RNER 和 RNTO 合在一起才能重命名服务器上的一个文件。

⑥ 返回值（粗体表示成功）：

- 250 请求的文件操作正常进行，已完成。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。
- 503 命令的顺序不对。
- 530 无法登录。
- 532 文件的存储需要帐号。
- 553 请求的操作无法执行，不允许的文件名。

25. SITE（Site Parameters，站点参数）命令：

① 说明：服务器使用 SITE 提供了某些服务特性，这些服务特性在系统中对文件传达而言是必要的，但又不是足够通用的可以包括到协议中作为命令的那些服务特性。这些服务的本质和他们的语法规范可以陈述在对 HELPSITE 命令的应答中。

② 用法：SITE<String><CRLF>

③ 参数：String 可以是任何串参数。

④ 例子：

```
SendData "SITE chmod 646 mylifle.ziP" & vbCrlf
```

⑤ 注释：举例来说，使用如下语法，你就可以使用 SITE 来改变一个文件的权限属性。

SITE CHMOD<Attribute><Filename>

⑥ 返回值（粗体表示成功）：

- 200 命令成功。
- 202 命令还没有被实现，在此站点上是多余的。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变无中有语法错误。
- 530 无法登录。

26. SMNT（Structure Mount，结构装配）命令：

① 说明：允许用户装配另一个文件系统的数据结构而无需改变登录、帐号信息或传输参数。

② 用法：SMNT<Path><CRLF>

③ 参数：Path 是另一个文件数据系统的路径。

④ 例子：

```
SendData "SMNT/users/johnsmith"&vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 202　命令还没有被实现，在此站点上是多余的。
- 250　请求的文件操作没问题，已完成。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 502　命令还没有被实现。
- 530　无法登录。
- 550　请求的操作无法执行，文件不可用（例如，找不到文件、无访问权等）。

27. STAT（Status，状态）命令：

① 说明：使一个状态响应以应答的形式通过控制连接发送出去

② 用法：STAT[<PathName>]<CRLF>

③ 参数：PathName 是服务器上的一个有效路径。

④ 例子：

```
SendData "STAT /users/johnsmith/mylilE" & vbCrlf
```

⑤ 注释：STAT 可以在文件传输（连同 Telnet IP 和 Synch 信号一起——请参看 RFC 959 的 FTP Commands 部分）期间发送，在此情况下服务器将响应操作中的进程状态，或者在两次文件传输之间发送。在后一种情况下，这条命令可以带一参数字段。如果指定了一个完整的路径名，STAT 将类似于 LIST 命令，除了它的数据是通过控制连接传送的之外。如果指定的是一个局部路径，服务器将在响应中给出一份与指定项相关联的文件名或属性列表。如果不指定参数，服务器将返回有关此服务器 FTP 进程的常规状态信息，这其中包括当前所有传输参数的值和连接状态。

⑥ 返回值（粗体表示成功）：

- 211　系统状态，或系统的帮助应答。
- 212　目录状态。
- 213　文件状态。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 450　请求的文件操作无法执行，文件不可用（例如文件正忙）。
- 500　语法错误，无法识命令。这其中包括包令行过长类的错误。
- 501　参数或变元中有语法错误。
- 502　命令还没有被实现。
- 530　无法登录。

28. STOR(Store，保存) 命令 ：

① 说明：让服务器接收一个来自数据连接的文件。

② 用法：STOR<FileName><CRLF>

③ 参数：FileName 是服务器站点上一个完全合格的路径和文件名。

④ 例子：

```
SendData "STOR newfile.zip" & vbCrlf
```

⑤ 注释：Store 命令让服务器接收通过数据连接传输而来的数据，并把数据存为务器站点上的文件。如果在路径名中指定的文件已经在服务站点上存在，则此文件的内容将被传输过来的数据所替代。如果在路径名中指定的文件尚未存在，将创建一个新文件。

⑥ 返回值（值体表示成功）：

- 110 重新启动标记应答

  在此情况下，文本是精确的，而且不会留给特殊的实现处理，它必须读取：MARK yyyy=mmmm

  其中 yyyy 是用户进程的数据流标记，mmmm 是服务器上的相应相标记（请注意标记和“=”之间的空格）。
- 125 数据连接已打开，传输启动。
- 150 文件状态一切正常，准备打开数据连接。
- 226 关闭数据连接，请求的文件操作已成功。
- 250 请求的文件操作正常进行，已完成。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 425 无法打开数据连接。
- 426 连接关闭，传输中止。
- 450 请求的文件操作无法执行，文件不可用（例如文件正忙）
- 451 请求的操作被中止，处理中发生本地错误。
- 452 请求的操作无法执行，系统的存储空间不足。
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 504 命令中还没有实现该参数。
- 530 无法登录。
- 532 文件的存储需要帐号。
- 550 请求的操作无法执行，文件不可用（例如，找不到文件、无访问权等）。
- 551 请求的操作被中止，未知的页类型。
- 552 请求的文件操作被子中止，超过了分配的存储单元（对当前目录或数据集而言）。
- 553 请求的操作无法执行，不允许的文件名。

29. STOU（Store Unique，存为唯一）命令：

① 说明：让服务器准备接收一个文件，并指示服务器把这个文件用唯一的名称保存到目的目录中。

② 用法：STOU<CRLF>

③ 参数：无。

④ 例子：

```
SendData "STOU"&vbCrlf
```

⑤ 注释：Store Unique 命令和 STOR 所做的工作是基本一样的，只是前者在当前目录下生成的文件是以一个该目录中唯一的名称来创建的，250 Transfer Started 响应中必须包括这个生成的名称。

⑥ 返回值（粗体表示成功）：

- 110　重新启动标记应答

在此情况下，文本是精确的，而且不会留给特殊的实现处理，它必须读取：

MARKyyyy=mmmm

其中 yyyy 是用户进程的数据流标记，mmmm 是服务器上的相应标记（请注意标记和“ = ”之间的空格）。

- 125　数据连接已打开，传输启动。
- 150　文件状态正常，准备打开数据连接。
- 226　关闭数据连接，请求的文件操作已成功。
- 250　请求的文件操作没问题，已完成（该行中包括一个唯一的文件名）。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 425　无法打开数据连接。
- 426　连接关闭，传输中止。
- 450　请求的文件操作无法执行，文件不可用（例如文件正忙）。
- 451　请求的操作被中止，处理中发生本地错误。
- 452　请求的操作无法执行，系统的存储空间不足。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 504　命令中还没有实现该参数。
- 530　无法登录。
- 532　文件的存储需要帐号。
- 550　请求的操作无法执行，文件不可用（例如找不到文件，无访问权）。
- 551　请求的操作被中止，未知的页类型。
- 552　请求的文件操作被中止，超过了分配的存储单元（对当前目录或数据集而言）。
- 553　请求的操作无法执行，存在不允许的文件名。

30. STRU（File Structure，文件结构）命令：

① 说明：指定传达数据的结构类型。

② 用法：STRU<Structure Type><CRLF>

③ 参数：StructureType 是如下 ASCII 字符的其中之一：

- F——文件结构（默认值）

- R——记录结构
- P——页结构

有关使用非文件结构的信息请参看 RFC 959。

④ 例子：

```
SendData "STRUR" & vbCrlf
```

⑤ 返回值（粗体表示成功）：

- 200 命令成功。
- 421 服务不可用，关闭控制连接。

如果某项服务获知自己即将关闭，会向所有命令做出这个应答

- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 504 命令中还没有实现该参数。
- 530 无法登录。

31. SYST（System，系统）命令：

① 说明：SYST 用于查明服务器上操作系统的类型。

② 用法 SYST<CRLF>

③ 参数：无。

④ 例子：

```
SendData "SYST" & vbCrlf
```

⑤ 注释：按照推测，应答中的第一个单词是在 Assigned Numbers 文档[4]的当前版本中列出的某一个系统名称。但是，并不一定都能成功，新的系统每时每刻都在涌现。而公开的文档也许能跟上趟，也许不能。

⑥ 返回值（粗体表示成功）：

- 215 NAME 系统类型。NAME 是来自 Assigned Numbers 文档中的一个正式的系统名称。
- 421 服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答
- 500 语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501 参数或变元中有语法错误。
- 502 命令还没有被实现。

32. TYPE（Representation Type，表达类型）命令：

① 说明：确定数据的传输方式。

② 用法：TYPE<Data Type Code><CRLF>

③ 参数：Data type Code 是一个（或几个）ASCII 字符，它标识出一种数据表达类型。

④ 例子：

```
'—设置二进制模式
SendData "type I" & vbCrlf
```

⑤ 注释：最典型的情况就是使用 TYPE 命令在 ASCII 或二进制模式（Image，图像）之

间切换。发送和接收任何类型的文件时都可以使用图像类型。ASCII 类型只在传送文本时使用。

有几种类型要用到第二个参数。第一个参数由一个单独的 ASCII 字符表示，ASCII 和 EBCDIC 的第二个 Format 参数也是如此；本地字节的第二个参数则是一个表示 Byte 大小的十进制整数。两个参数之间用<SP>（Space，即空格，ASCII 代码为 32）分隔开。

给类型分配如下代码：

```
A——ASCII*
E——EBCDIC*
I——Image（图像，传送二进制文件时用它）
L<byte size>——Local byte  Byte size（本地字节，字节大小）
-------------------------------------
*——A 和 E 类型的第二个 Parameter 要用到如下三个的其中一个
N——Nonprint（非打印）
T——Telnet format effector（Telnet 格式控制符）
C——Carriage Control（ASA）托架控制（美国国家标准）
```

默认的表达类型是 ASCII Nonprint。如果 Format 参数发生改变，后来只是第一个参数发生了改变，Format 将随即返回到默认的 Nonprint。

⑥ 返回值（粗体表示成功）：

- 200　命令成功。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 504　命令中还没有实现该参数。
- 530　无法登录。

33. USER（User Name，用户名称）命令：

① 说明：指定远程系统上的用户名称。和 PASS 命令一起使用来登录。

② 用法：USER<User><CRLF>

③ 参数：UserName 是 FTP 系统上已经注册的用户名称。

④ 例子：

```
SendData "USER johns" & vbCRLF
```

⑤ 注释：在连接到一台 FTP 服务器的端口并接收到一个由代码 220 开头的行，表示服务器已经准备好用户向它发送 USER 和 PASS 命令以登录进此 FTP 服务器之后，紧跟着发送 USER 命令。

PASS 命令应该紧跟着 USER 命令。

如果在此 FTP 服务器上有帐号，就可以指定自己的用户名和密码。如果想匿名登录，可以指定用户名为“ Anonymous ”，而用自己的电子邮件地址当密码。

在会话期间，可以随时发送 USER 和 PASS 命令，把控制权交给一个新用户。

⑥ 返回值（粗体表示成功）

- 230　用户已经登录。
- 331　用户名正常，需要密码。
- 332　登录需要帐号。
- 421　服务不可用，关闭控制连接。
- 如果某项服务获知自己即将关闭，会向所有命令做出这个应答。
- 500　语法错误，无法识别命令。这其中包括命令行过长之类的错误。
- 501　参数或变元中有语法错误。
- 530　无法登录。

以上为大部分的 FTP 命令，如果想进行 FTP 高级编程，就必须掌握这些具体的 FTP 命令。

8.4.2　FTP 工作模式

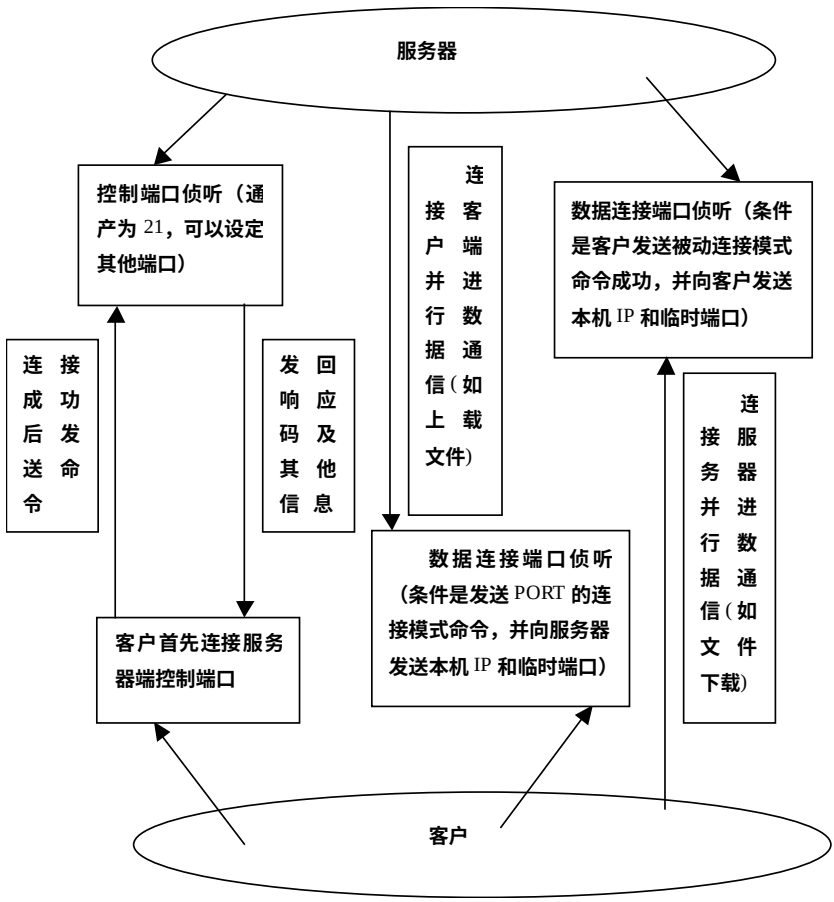


图 8-18　FTP 工作模式

FTP 的工作模式同其他的 C/S 模式的网络通信协议有一个很大的区别。通常在进行 HTTP 通信或者 Telnet 通信的时候，只需要一个端口进行通信。即客户端只需要连接一个端口进行数据传输。如 Telnet 的默认端口是 23，用户从头到尾都只需要和这个端口打交道。

而 FTP 通信除了有一个默认的端口 21 以外，还需要其他的端口，通常是两个端口同时进行数据传输的。一个是默认的端口，而另外一个是按照一定原则由服务器或者客户端产生的非标准端口。其中默认端口（通常是 21）主要是进行控制连接，顾名思义，控制连接主要是进行命令协议的以及服务器端的响应码的传输；另外一个非标准端口主要是进行数据的传递，比如上载文件、下载文件、打印目录信息等。至于非标准端口的产生要根据用户选择的连接模式而定，如果客户选择的是 PORT 模式，则需要客户端提供给服务器一个 IP 地址和一个非标准端口；而如果用户采用被动模式，则服务器端需要提供给客户端一个 IP 地址和一个非标准的端口。在进行文件传输的时候，通常每传送完毕一个文件后，又会重新建立连接模式并重新产生一个临时端口，读者可以在后面的编程中体会到这一点。具体的工作模式如图 8-18 所示。

8.5　Internet Transfer 控件实现 FTP 程序

VB 自带了一个控件，就是 Internet Transfer 控件，该控件主要支持 HTTP 协议和 FTP 协议，它把这些协议集成在控件里面，因此对于网络初级用户或者是那些想很快体会 FTP 编程的读者来说，是最适合的。读者只需要掌握它的一些属性和方法，就可以很方便地连接到其他的 FTP 站点上面。

关于该控件的具体属性、方法、事件在前面的章节有介绍。在本节里面，主要以例题的形式来介绍如何使用该控件连接到服务器上。这里要强调一下，使用这个控件的灵活性并不是很大，所以如果要编写一个完整的客户端程序，不推荐使用该控件。

而对于集成于其他应用程序内部的 FTP 程序可以使用该控件，比如公司内一个终端计算机每天需要把使用者的使用情况日志上报给公司的总部，就可以在程序内部使用该控件实现文件的上载，实现起来非常简单，开发者并不需要知道 FTP 工作的具体细节。注意这个程序的功能不是很完善，只是向读者演示如何用该控件来开发 FTP 客户端应用程序，读者可以自己完善或者增加新的功能。本程序所演示的功能有：通过帐号和密码访问远程服务器、列出目录、上载文件、下载文件、更改服务器文件名、获得文件大小、删除服务器文件、新建服务器目录、删除服务器目录。

8.5.1　建立工程项目

由于本书面向的是 VB 的高级读者，因此建立项目的详情就不再多说。最关键的就是在项目里增加控件“Microsoft Internet Transfer Controls 6.0”，读者要注意的是，最好使用 VB 6.0 以上版本开发。因为 VB 6.0 自带的是该控件的 6.0 版本，而版本 5.0 的控件中有许多的问题。

当然 6.0 也有些问题，读者可能会在使用的时候有所体会。不过因为是免费的，因此在特定的情况下可以使用该控件来开发项目。同时需要增加两个 listview 控件，用来显示本地和远程的目录列表。（程序的具体源代码参见本书所附光盘，里面有详细的解释）具体的工作界面如图 8-19 所示。

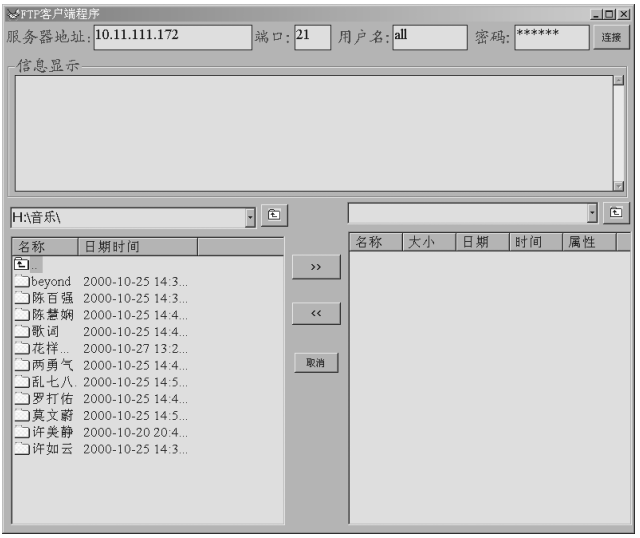


图 8-19　**工作界面**

8.5.2　关键代码分析

下面主要是针对和 FTP **有关的知识做讲解。**

1. **初始化** Internet Transfer **控件**

```
Private Sub InitInet()

With Inet1
                '设定远程服务器的地址
    If Left(Trim(txtURl.Text), 6) <> "ftp://" Then
.URL = "ftp://" & Trim(txtURl.Text)
    End If
'设定远程服务器的端口
    If txtPort.Text <> "" Then
        .RemotePort = CInt(Trim(txtPort.Text))
    Else
        .RemotePort = 21
    End If

    .UserName = Trim(txtUID.Text)
    .Password = Trim(txtPWD.Text)
End With
'设定开始目录为空，即表示是根目录
CurrentServerDir = ""

'如果上次的任务还没有结束，则程序不能立刻停下来，这也是使用该控件的一个不利之处
If Inet1.StillExecuting Then
```

```
    MsgBox "系统正忙，无法重新连接!"
                Exit sub
    End If

    '列出服务器的根目录，或者可以使用 ListServer("/")
    ListServer ("*")

    End Sub
```

上面的那段代码主要是当用户单击“连接”按钮时执行的，也就是重新初始化 Inet 控件，并建立新的连接。在连接服务器之前首先要设定服务器的地址，这是必须的，否则无法建立连接。接下来必须设定远程服务器的端口，这个值可以不设置，因为对于大部分的 FTP 站点都是使用默认的端口“21”，只有当端口不是默认端口的时候，才需要设置这个值，否则也无法连接服务器。然后是设置用户名和密码，同样也是调用“UserName”属性和“Password”属性。这里需要注意的是，有些情况并不需要这样设置，可以直接调用一些方法来连接服务器，也就是在发出命令的同时设置服务器地址和端口。比如下面的代码：

```
    Inet1.Execute "ftp://busyant:21", "SEND " & Trim(SourceFile) & " " & Trim(DestinatonFile)
```

因此在程序内部要想上载文件或者下载文件就可以直接通过这种方法来实现，其中“ftp://busyant”是站点的地址，而“21”是端口号。但这种方法只适合于匿名访问，如果需要帐号和密码验证，则下面的代码不能实现：

```
    .UserName = Trim(txtUID.Text)
    .Password = Trim(txtPWD.Text)
    .Execute "ftp://busyant:21", "SEND " & Trim(SourceFile) & " " & Trim(DestinatonFile)
```

上面的那段代码并不能正确地执行，所以为了能正确地连接服务器，最好使用程序里面的介绍代码。

注意当该控件正在忙的时候，不能响应其他的命令，因此当你调用它的“Cancel”方法时并不能让它立即停止下来，这也是使用该控件非常不方便的一个地方，读者在开发的时候会体会到，可以通过“StillExecuting”来判断该控件是否正在忙。

2. 列出服务器的目录

代码如下：

```
    '该函数是列出指定路径下的文件信息
    Private Sub ListServer(DirStr As String)

    '首先判断是否该控件正在忙
    If Not Inet1.StillExecuting Then
                '设置操作类型，1 为列出目录
                OperationStyle = 1
                '通过“LS”命令列出目录
```

```
                Inet1.Execute , "LS " & DirStr
    End If

    End Sub
```

在上面的代码中，是通过发送“LS”命令来向服务器请求文件列表的，后面的参数是指定的文件路径。如果要列出根目录下的文件，则 DirStr 可以是“*”或者“/”，如果要列出子目录“aaa”中的文件，则 DirStr 的值应该写成“aaa/”。“LS”命令类似于“DIR”命令，“DIR”命令后面也是接指定的目录路径，如果缺省，则返回当前目录中的文件信息。同样能完成改变目录功能的命令还有“CD”、“CDUP”。使用“CD”命令，后面需要跟参数，即具体要进入到哪个目录，然后调用“DIR”显示文件列表；“CDUP”命令是回到上一级目录，然后通过“DIR”显示文件列表。

注意在 Execute 后面的参数中，逗号前面的参数是远程服务器的地址，因为该地址已经在“URL”属性中设置过，所以可以省略。如果前面已经设置过，现在仍然设置了该参数，则会忽略“URL”中的属性设置，而使用最新的服务器地址和端口，同时密码和用户名设置也不起作用了，读者在写程序的时候要注意。同时通过 Execute 方法来执行各种命令，命令和后面的各个参数之间都是用空格隔开的，所以为了能够正确地执行程序，命令后面的参数应该不能包含空格，比如在上面的代码中，如果 DirStr 中包含空格，将会出现意外错误。

```
    '该函数的功能是获得服务器端当前目录上一级的文件目录
    Private Function UpServerDir() As String
    Dim tempPos1 As Integer
    On Error Resume Next
    If CurrentServerDir <> "" Then
    .    tempPos1 = InStrRev(CurrentServerDir, "/", Len(CurrentServerDir) - 1,
vbTextCompare)
        CurrentServerDir = Mid(CurrentServerDir, 1, tempPos1)
        UpServerDir = CurrentServerDir
    End If
    End Function
```

函数 UpServerDir 的功能是获得服务器的上一级目录，首先判断是不是根目录，如果是则返回空字符串，否则分析当前的路径。因为当前路径总是以“/”结束的（根目录除外，在本程序中，根目录是用空字符串来表示的），所以如果当前路径是“aaa/bbb/”，只需要找到“bbb”前面的‘/’的位置，然后去掉后面的字符串就能得到上一级的路径。然后调用 ListServer 方法来显示目录，如下：

```
    Private Sub cmdUpSDir_Click()

    If CurrentServerDir <> "" Then
        If Inet1.StillExecuting Then
            MsgBox "目前程序正在执行!"
        Else
```

```
        ListServer UpServerDir
    End If
Else
    MsgBox "目前已经是根目录了!"
End If
End Sub
```

上面的代码是“cmdUpSDir”按钮单击事件，通过“ListServer UpServerDir”来获得上一级目录中的文件信息。同样为了进入下一级目录，只需要双击控件“ListServerDir”中显示的目录，通过该事件来获得下一级目录路径，然后调用“ListServer”来显示文件信息。具体代码如下：

```
'ListServerDir 控件的双击事件
Private Sub ListServerDir_DblClick()

Dim Item As ListItem
If Inet1.StillExecuting Then
MsgBox "程序还在执行!"
            Inet1.Cancel
Else
            If ListServerDir.HitTest(xpos1, ypos1) Is Nothing Then
            Exit Sub
            Else
            Set Item = ListServerDir.HitTest(xpos1, ypos1)
            End If
'通过判断双击项的最后时候含有“/”来判断是文件还是文件
            If Right(CStr(Item), 1) = "/" Then
            CurrentServerDir = CurrentServerDir & Item
            ListServer (CurrentServerDir)
            Else
            Exit Sub
    End If
End If
End Sub
```

上面的代码主要为了获得要进入下一级目录的路径，通过双击 ListServerDir 控件来获得要进入的下一级目录，如果双击的是目录，则设置路径为“CurrentServerDir & Item”，其中“Item”是双击的项，如果是文件，则不做处理。判断目录还是文件是通过判断“Item”最后是否含有“/”来决定的，笔者在开发程序的时候，用 ListServerDir 控件显示文件和目录，如果是目录，则在后面加上“/”，以示区别。获得路径后，通过调用“ListServer (CurrentServerDir)”来显示目录和文件。

3. 上载文件

代码如下：

```
'该过程是向服务器上载一个文件
Private Sub UpFile(SourceFile As String, DestinatonFile As String)

OperationStyle = 8
Inet1.Execute , "SEND " & Trim(SourceFile) & " " & Trim(DestinatonFile)
End Sub
'该按钮事件调用 UpFile 过程
Private Sub cmdUpLoad_Click()
On Error Resume Next
   UpFile CurrentDir & ListClientDir.SelectedItem, CurrentServerDir &
ListClientDir.SelectedItem
End Sub
```

上面“UpFile”过程代码的功能是向服务器上载一个文件，使用的命令是“SEND”，后面的两个参数分别是源文件和目标文件，写的时候要注意，它们之间分别用空格分开。其中“SourceFile”表示源文件，在上载过程中，即本地文件，应该包括完整的路径；“DestinatonFile”表示目标文件，在这个过程中，表示的是要上载到服务器的路径和文件名，文件名不能省略，可以改文件名，如果只有路径而没有文件名，上载将会失败。具有同样功能的命令还有“PUT”，程序中并没有举例，读者可以自己测试一下。

“cmdUpLoad”按钮单击事件调用“UpFile”过程来上载文件，其中“CurrentDir”表示的是本地文件的路径，“ListClientDir.SelectedItem”为当前路径下选中的文件，“CurrentServerDir”表示的是服务器端当前路径，“ListClientDir.SelectedItem”表示上载到服务器后的文件名和原来的文件名一样，没有改动，当然也可以使用其他的文件名。

注意并不是所有的服务器都能上载文件，要看用户是否具有这个权限。

4. 下载文件

代码如下：

```
'该过程是向服务器发出请求要求下载文件
Private Sub DownFile(SourceFile As String, DestinatonFile As String)
   OperationStyle = 7
   Inet1.Execute , "GET " & Trim(SourceFile) & " " & Trim(DestinatonFile)
End Sub
' cmdDownLoad 按钮的单击事件
Private Sub cmdDownLoad_Click()
On Error Resume Next
   DownFile CurrentServerDir & ListServerDir.SelectedItem, CurrentDir &
ListServerDir.SelectedItem
End Sub
```

上面的一段代码主要是用于下载文件的，“DownFile”的过程是通过 INET 控件执行

“GET”命令来获得服务器上的文件的，同样要注意格式，其中“SourceFile”表示的是服务器端的文件路径，“DestinatonFile”表示的是目标文件路径，在函数中为本地文件。同时能完成这个功能的命令还有“RECV”，读者可以自己测试一下。

下面的 cmdDownLoad 按钮事件是调用“DownFile”过程，“CurrentServerDir”是当前服务器路径，“ListServerDir.SelectedItem”表示要下载的文件名，“CurrentDir”表示的是本地的文件路径，“ListServerDir.SelectedItem”为下载后的文件名，表示同服务器上的名字一样，当然也可以改成其他的文件名。

5. 删除文件

代码如下：

```
'该过程是删除服务器端文件的命令
Private Sub DeleteServerFile(FilePath As String)

OperationStyle = 4
Inet1.Execute , "delete " & FilePath

End Sub

'菜单事件，调用 DeleteServerFile 过程
Private Sub sdeletedir_Click()
If Inet1.StillExecuting Then
   MsgBox "程序还在连接!"
Else
   DeleteServerDir (CurrentServerDir & itemA)
End If
End Sub
```

在“DeleteServerFile”过程中主要的功能是向服务器发送“DELETE”命令删除一个文件，FilePath 即为要删除的文件，注意命令和文件路径之间也是用空格隔开的，设定操作类型为 4，以区别其他的操作。

在菜单事件里面，主要是调用“DeleteServerFile”过程。

注意并不是所有的服务器都能删除文件，要看用户是否具有这个权限。

6. 更改文件名

代码如下：

```
'该过程的功能是更改文件名
Private Sub RnameServerFile(FilePath As String, OldFileName As String,
NewFileName As String)

OperationStyle = 5
Inet1.Execute , "rename " & FilePath & OldFileName & " " & FilePath & NewFileName
```

```
End Sub

'菜单事件，调用 RnameServerFile 过程
Private Sub srename_Click()

If Inet1.StillExecuting Then
    MsgBox "程序还在连接!"
Else
    OldFileName = itemA
    NewFileName = InputBox("原来的文件名为" & itemA, "请输入新的文件名")
    If NewFileName = itemA Then
        MsgBox "新旧文件名相同!"
    ElseIf Trim(NewFileName) <> "" Then
        RnameServerFile CurrentServerDir, OldFileName, NewFileName
    End If
End If
End Sub
```

过程“RnameServerFile”是用于向服务器发送“RENAME”命令，要求更改一个文件的名字的。其中参数“FilePath”是服务器端文件路径，“OldFileName”为原来的文件名，“NewFileName”为新的文件名。

菜单事件的功能是调用“RnameServerFile”过程，首先让用户输入新的文件名，然后两个文件名不相同则提交，否则提示名字相同，停止提交。

注意，并不是所有的服务器都能更改文件名，要看用户是否具有这个权限。

7. 获得文件大小

代码如下：

```
'该过程是获得文件大小
Private Sub GetFileSize()
Dim i As Integer
OperationStyle = 2
If ListIndex < ListServerDir.ListItems.Count + 1 Then
    Set itemA = ListServerDir.ListItems.Item(ListIndex)
    Inet1.Execute , "size " & CurrentServerDir & ListServerDir.ListItems(ListIndex)
End If

End Sub
```

上面代码的功能是获得文件或文件夹的大小。首先设操作类型为 2，然后显示 ListServerDir 控件中各个文件的大小，变量“ListIndex”表示的是控件中的第几项，最后通过发送“SIZE”命令来获得文件大小。“CurrentServerDir”表示的是服务器当前路径，

“ListServerDir.ListItems(ListIndex)”表示的是要获得大小的文件或者文件夹。不过通过该控件获得大量文件的大小是不明智的，因为这样做效率很低，并不能在获得文件名的同时获得文件大小，而必须发送“SIZE”命令才可以。对于大量文件这样做速度很慢，而且很容易引起连接中断。

8. 创建目录

代码如下：

```
'该过程是向服务器发出建立目录的命令
Private Sub CreateServerDir(NewDir As String)
OperationStyle = 6
Inet1.Execute , "mkdir " & CurrentServerDir & NewDir
End Sub
'菜单事件，调用上面的建立目录过程
Private Sub screatedir_Click()
If Inet1.StillExecuting Then
   MsgBox "程序正在连接!"
Else
   NewDir = InputBox("请输入要创建的目录", "创建目录", "new")
   If Trim(NewDir) <> "" Then
      CreateServerDir (NewDir)
   End If
End If
End Sub
```

在上面的“CreateServerDir”代码中，首先设置了操作类型为 6，表示的是要创建目录。实现这个功能是通过命令“MKDIR”来完成的，后面的参数为完整路径，其中“CurrentServerDir”是当前路径，而“NewDir”是要建立的新的目录，注意命令“MKDIR”同后面的参数之间是用空格隔开的，而且新建的目录名中不能含空格，否则创建目录会失败。

在菜单单击事件中，主要是调用“CreateServerDir”过程，让用户输入新的目录，然后向服务器提交。

注意并不是所有的服务器都能允许创建目录的，要看用户是否具有这个权限。

9. 删除目录

代码如下：

```
'该过程是删除服务器端的目录
Private Sub DeleteServerDir(DirPath As String)

OperationStyle = 3
Inet1.Execute , "RMDIR " & DirPath

End Sub
```

```
'菜单事件，调用 DeleteServerDir 过程
Private Sub sdeletedir_Click()

If Inet1.StillExecuting Then
    MsgBox "程序正在忙!"
Else
    DeleteServerDir (CurrentServerDir & itemA)
End If
End Sub
```

过程“DeleteServerDir”的功能是通过向服务器发送“RMDIR”命令来删除一个特定的目录，变量“DirPath”是目录路径，和命令之间用空格隔开。

菜单事件的功能是调用“DeleteServerDir”过程，其中“CurrentServerDir”为服务器当前路径，而“ItemA”是要删除的目录。

注意并不是所有的服务器都能允许删除目录的，要看用户是否具有这个权限。

10. 获得服务器端数据

代码如下：

```
'该代码为 Inet 控件的 StateChanged 事件处理程序
Private Sub Inet1_StateChanged(ByVal State As Integer)
Dim tempArray As Variant
Dim i As Integer
Dim FileSize As Variant
Dim itmX As ListItem
On Error Resume Next
Select Case State
    Case 0 'icNone
        Text1.Text = Text1.Text & vbCrLf & "无状态可报告"
    Case 1 'icHostResolvingHost
        Text1.Text = Text1.Text & vbCrLf & "正在查询所指定的主机的 IP 地址"
    Case 2 'icHostResolved
        Text1.Text = Text1.Text & vbCrLf & "已成功地找到所指定的主机的 IP 地址"
    Case 3 'icConnecting
        Text1.Text = Text1.Text & vbCrLf & "正在与主机连接"
    Case 4 'icConnected
        Text1.Text = Text1.Text & vbCrLf & "已与主机连接成功"
    Case 5 'icRequesting
        Text1.Text = Text1.Text & vbCrLf & "正在向主机发送请求"
    Case 6 'icRequestSent
        Text1.Text = Text1.Text & vbCrLf & "发送请求已成功"
```

```
        Case 7 'icReceivingResponse
            Text1.Text = Text1.Text & vbCrLf & "正在接收主机的响应"
        Case 8 'icResponseReceived
            Text1.Text = Text1.Text & vbCrLf & "已成功地接收到主机的响应"
        Case 9 'icDisconnecting
            Text1.Text = Text1.Text & vbCrLf & "正在解除与主机的连接"
        Case 10 'icDisconnected
            Text1.Text = Text1.Text & vbCrLf & "已成功地与主机解除了连接"
        Case 11 'icError
            Text1.Text = Text1.Text & vbCrLf & "与主机通信时出现了错误"
            Text1.Text = Text1.Text & vbCrLf & "错误" & Inet1.ResponseCode & ":"
& Inet1.ResponseInfo
        Case 12 'icResponseCompleted
            Text1.Text = Text1.Text & vbCrLf & "该请求已经完成，并且所有数据均已接收到
"
            Select Case OperationStyle
                Case 1 '列出目录和文件
                    ListServerDir.ListItems.Clear
                    InetData = Inet1.GetChunk(1024, 0)
                    If Trim(InetData) <> 0 Then
                        tempArray = Split(InetData, vbCrLf, , vbTextCompare)
                        Combo2.Text = "Root/" & CurrentServerDir
                        i = 0
                        Do While i < UBound(tempArray)
                            If tempArray(i) <> "" Then
                                DealListServerDir (tempArray(i))
                            End If
                            i = i + 1
                        Loop
                        ListIndex = 1
                    End If
                    'GetFileSize
                Case 2 '获得每个文件的大小
                    FileSize = Inet1.GetChunk(1024, 0)
                    itemA.SubItems(1) = CStr(FileSize)
                    ListIndex = ListIndex + 1
                    GetFileSize
                Case 3 '删除目录
                    Text1.Text = Text1.Text & vbCrLf & itemA & "目录已经被删除!"
```

```
                ListServerDir.ListItems.Remove
(ListServerDir.SelectedItem.Index)
            Case 4 '删除文件
                Text1.Text = Text1.Text & vbCrLf & itemA & "文件已经被删除!"
                ListServerDir.ListItems.Remove
(ListServerDir.SelectedItem.Index)
            Case 5 '更改文件名
                Text1.Text = Text1.Text & vbCrLf & itemA & "文件被更名为" &
NewFileName
                ListServerDir.SelectedItem.Text = NewFileName
            Case 6 '创建目录
                Text1.Text = Text1.Text & vbCrLf & NewDir & "目录已经被创建成功!"
                Set itmX = ListServerDir.ListItems.Add(, , NewDir & "/")
                itmX.SmallIcon = 1
                itmX.Icon = 1
            Case 7 '下载文件
                Text1.Text = Text1.Text & vbCrLf & ListServerDir.SelectedItem
& "文件下载成功!"
                Set     itmX     =     ListClientDir.ListItems.Add(,     ,
ListServerDir.SelectedItem)
                itmX.Icon = 2
                itmX.SmallIcon = 2
            Case 8 '上载文件
                Text1.Text = Text1.Text & vbCrLf & ListClientDir.SelectedItem
& "文件上载成功!"
                Set     itmX     =     ListServerDir.ListItems.Add(,     ,
ListClientDir.SelectedItem)
                itmX.Icon = 2
                itmX.SmallIcon = 2

            Case Else
        End Select
    End Select
       Text1.SelLength = Len(Text1.Text)
    End Sub
```

前面进行的一系列操作都是客户端向服务器发出请求，但是请求是否成功，就需要查看服务器返回给客户的信息。Inet 控件有一个唯一的事件“ StateChanged ”，通过分析这个事件传来的参数，就可以分析客户发出命令后的状态。传来的参数“ State ”是一个整数，其取值为 0～12，共有 13 个状态，客户可以分析这些返回值从而判断程序目前的状态。在这 13 个

状态中，大部分是不需要处理的，只需要读出来并显示就可以了，比如调用控件的“Cancel”方法，就会返回“State”为 9、10，表示与主机解除连接。但是有两个状态是需要处理分析的，即“State”为 11、12。

“State”为 11 表示与主机连接错误，这个时候可以通过 Inet 控件的两个属性“ResponseCode”和“ResponseInfo”来分析具体的错误是什么。例如用户如果试图删除一个文件但是不具备权限，就返回一个错误代码和错误信息，如图 8-20 所示。

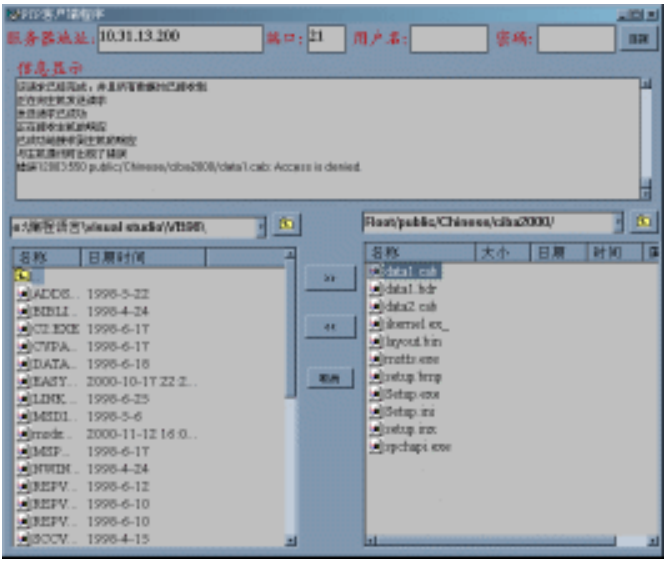


图 8-20　返回错误信息

从图中可以看出用户试图删除“public/Chinese/ciba2000/data1.cab”文件，就会返回如下错误信息：

```
与主机通信时出现错误！
错误 12003:550 public/Chinese/ciba2000/data1.cab: Access is denied.
```

上面的“12003”为错误代码，具体的错误信息就是错误代码后面的那句话，表示请求失败，要求删除目录不成功，即“Access is denied.”表示访问被拒绝。

“State”为 12 表示请求成功，客户发出的所有命令，如果成功，就会返回这个值，因此，在开发程序的时候，所有的操作都必须在返回了这个信息之后再处理。对于每个命令，服务器返回的数据是不相同的。在前面的理论学习中，大家已经知道，FTP 的连接不同于其他的连接，而是同时建立起两个连接的，一个是控制连接，另外一个数据连接。上面的事件里面返回的各种状态都是由控制连接返回的，因此，必须要读取数据连接上的数据才能够完整地完成一个命令。比如，当客户发出列出目录的命令，如果成功，则会返回“State”为 12。这个时候，服务器便会把目录和文件的具体列表从另外一个端口发送到客户端。

在前面发送命令的时候，对于每一个操作都定义了一个“OperationStyle”，就是便于现在分析。下面逐一分析每一个命令的返回值。

（1）OperationStyle=1，表示是发送“LS”命令成功

```
Case 1 '列出目录和文件
ListServerDir.ListItems.Clear
InetData = Inet1.GetChunk(1024, 0)
If Trim(InetData) <> 0 Then
tempArray = Split(InetData, vbCrLf, , vbTextCompare)
i = 0
Do While i < UBound(tempArray)
If tempArray(i) <> "" Then
DealListServerDir (tempArray(i))
End If
i = i + 1
Loop
ListIndex = 1
End If
'GetFileSize
```

上面的代码就是当用户发出“LS”命令后从另外一个端口读取数据。要读取服务器返回的数据，就必须使用到 Inet 控件的一个方法“GetChunk”，其具体的使用方法，可以参见前面的控件说明，其中有详细的解释，或者可以参看 MSDN。

由于数据是以字符串返回的，语句“InetData = Inet1.GetChunk(1024, 0)”就是将数据在接收到之后再存放在变量“InetData”里面，服务器端返回的目录数据之间是通过回车符“VBCRLF”隔开的，所以可以通过“Split”函数获得每个目录或者文件并放在一个数组中，然后通过循环读取每一项。如果不为空，则加入到“ListServerDir”控件中。“ListServerDir”函数处理的就是这一过程，该函数的代码如下：

```
Private Sub DealListServerDir(tempStr As String)
If Right(Trim(tempStr), 1) <> "/" Then
    '表示接收到的是文件
    AddFileToListServerDir (tempStr)
Else
    '表示接收到的是目录
    AddDirToListServerDir (tempStr)
End If
End Sub
```

“DealListServerDir”函数是将目录和文件加入到“ListServerDir”控件中，那么如何判断是目录还是文件，这就要根据数据的格式了。通常，如果返回的是目录，则会在最后加上“/”符号，而文件就没有这个符号，因此可以通过这个原则来区分是文件还是目录。

（2）OperationStyle=2，表示是发送“SIZE“成功

```
Case 2 '获得每个文件的大小
FileSize = Inet1.GetChunk(1024, 0)
```

```
    itemA.SubItems(1) = CStr(FileSize)
    ListIndex = ListIndex + 1
    GetFileSize
```

上面的代码是处理获得的文件大小的，它也是调用“ Inet1.GetChunk(1024, 0) ”来获得数据的。不过获得文件大小只能每次处理一个文件，所以效率很低，因此建议在列出目录的时候不要循环列出文件大小，而在要下载的时候可以发送“ SIZE ”来获得文件大小。

对于其他操作的处理，因为不需要获得额外的数据，这里就不一一解说了，关键是要学会如何使用“ Inet1.GetChunk ”方法。

11. 总结

关于如何通过 Internet Transfer 控件来开发简单 FTP 客户端程序，就先介绍到这里，读者可以结合光盘里面的程序和书中的讲解来学习。这一节主要是针对 VB 的初中级的读者而写的，在后面的两章里无论是网络方面和 VB 编程技巧方面都是比较高级的。

8.6　Winsock 开发高级 FTP 客户端程序

前面的那个例题，只是简单地利用一个控件来实现 FTP 的各种功能，其实它也利用了 Winsock 来编程，不过它只是集成一些功能，让用户使用起来比较方便，但是可以看到这很不灵活而且功能有限。在下面的这个例题中，将介绍如何使用具体的 FTP 协议来开发 FTP 客户端程序。因为用户每发出一个命令都是一个真正的 FTP 命令，而且需要真正了解 FTP 是如何工作的，并且要掌握许多相关理论。

读者如果真正掌握了这个例题，就可以学习到以下知识：

- VB 类编程。
- VB 事件编程。
- Winsock 对象编程。
- VB 回调函数的实现。
- FTP 协议。
- FTP 工作模式。
- 断点续传原理。
- 其他 VB 高级编程知识和网络知识。

因此，读者对于这个例题，需要下工夫，这样无论是在 VB 的高级编程还是网络高级编程方面都会有较大的进步。

8.6.1　建立工程项目

同上面的例题一样，由于这个例题比较大，使用的资源比较多，所以对于整个项目的资源、属性、名称就不一一介绍了，读者可以参见本书所附的光盘和书上的解说。对于 VB 基础比较差的读者，则需要学相关的 VB 知识，否则读起来会有点困难。读者可以打开源程序，通过单步调试来搞清楚整个程序的工作过程以及相关代码的功能。

该应用一共由 3 个窗体、4 个类模块、一个标准模块组成，其中最主要的是类“ CFtpConnection ”，其中所有的与 FTP 服务器进行协议通信的功能都是在这个类中实现的。

读者在学习的时候要多花些时间。为了后面解说方便，下面列出以上几种资源的名字：

- 类模块 CftpFile
- 类模块 CftpFiles
- 类模块 Ctimeout
- 标准模块 MFtpSupport
- 类模块 CftpConnection
- 窗体 frmMain
- 窗体 frmConnect

在后面的代码说明章节中就是按照这个顺序进行解说的。目的是在关键代码的解说过程中，主要是介绍一些比较重要的和比较难的部分。让读者学起来能更清楚，对 FTP 客户端程序的工作过程有更深的了解。介绍的时候主要以类和窗体为单位进行，通常是把被调用的代码放在前面介绍，这样读者在看后面程序的时候，如果里面调用了其他函数，这个函数在前面就已经讲解过了。读者在学习的时候一定要搞清楚函数之间的调用关系以及执行次序，否则可能会胡涂。对于搞不清的地方，可以打开源代码，然后在 VB 中单步调试，这样对于 FTP 的工作过程就能看得比较清楚了。程序中许多地方用到了无条件循环，因此在调试状态下，可能难以实现，因此代码中都在关键地方加了“ dubug.print ”来打印中间变量和数据，读者可以通过查看这些数据来分析，这样也能达到比较好的效果。

工作界面如图 8-21 所示，该界面为连接到一个站点上。

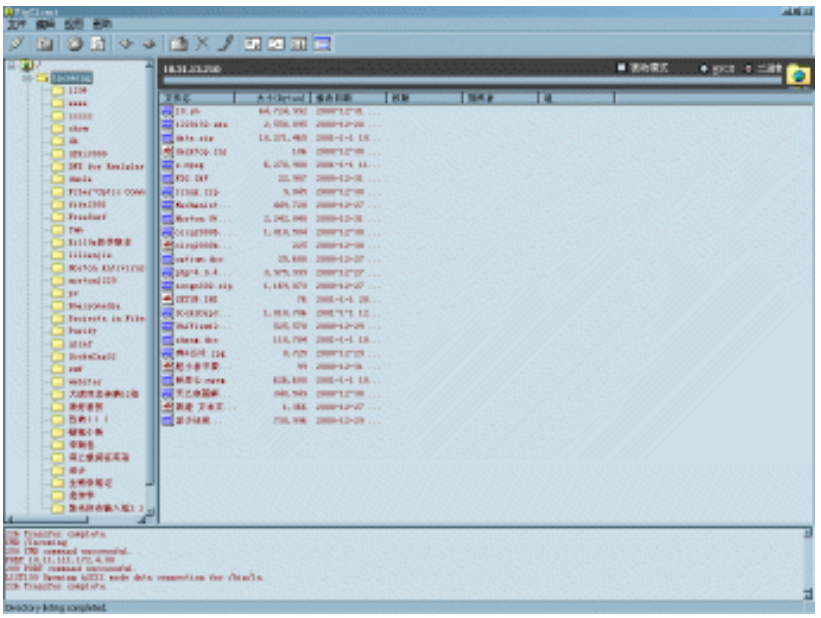


图 8-21　工作界面

该服务器有两个根目录，其中一个是“ incoming ”，还有一个是“ pub ”。通常目录“ incoming ”用于让用户上载或者下载文件，而目录“ pub ”用于让用户下载文件，但没有权限上载。图 8-22 为上载一个文件后的工作界面，用该程序上载了一个“ 000.txt ”的文件，出现在文件列表的第一项。

该程序中的应用非常多，其他的读者可以自己运行程序来分析。

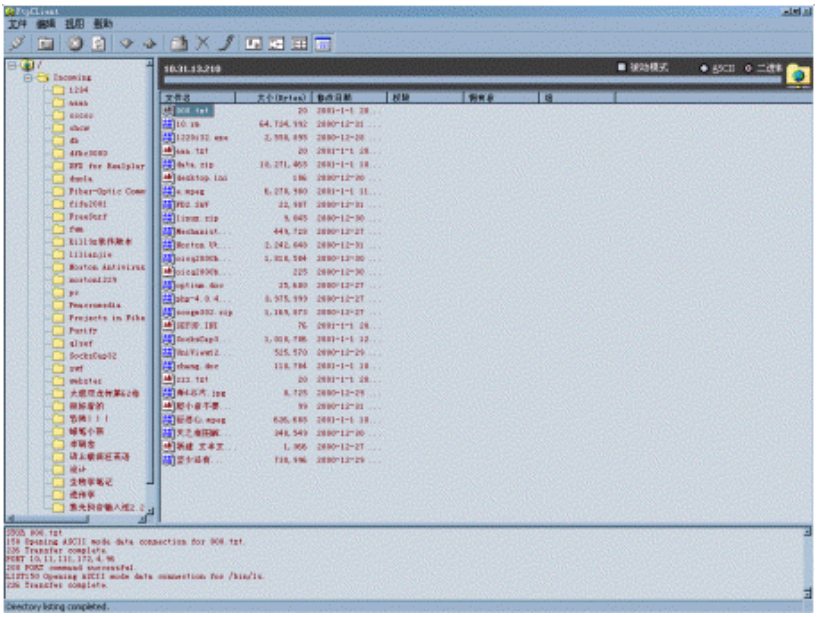


图 8-22　文件上载后的界面

8.6.2　关键代码分析

在代码解说中，主要也是解释和 FTP 相关的代码以及相关的理论，对于其他的通过各种控件来显示和处理功能的 VB 程序代码，读者可以自己去研究。

1. 类模块 CftpFile

该类是一个很简单的类，主要功能是存储一个文件的信息，比如文件大小、文件类型等。具体功能可以看程序代码解释。

```
                '申明类模块内部变量
Private mvarFileName         As String
Private mvarLastWriteTime     As Date
Private mvarFileSize          As Long
Private mvarIsDirectory      As Boolean
Private mvarFilePath         As String
'
Private mvarPermissions       As String
Private mvarOwner           As String
Private mvarGroup           As String
Public Property Let IsDirectory(ByVal vData As Boolean)
'设置类型、表示是目录还是文件
   mvarIsDirectory = vData
End Property
Public Property Get IsDirectory() As Boolean
'读取类型
   IsDirectory = mvarIsDirectory
```

```
End Property
Public Property Let FileSize(ByVal vData As Long)
'设置文件大小属性值
    mvarFileSize = vData
End Property
Public Property Get FileSize() As Long
'获得文件大小属性值
    FileSize = mvarFileSize
End Property
Public Property Let LastWriteTime(ByVal vData As Date)
'设置最近修改时间属性值
    mvarLastWriteTime = vData
End Property
Public Property Get LastWriteTime() As Date
'获得最近修改时间属性值
    LastWriteTime = mvarLastWriteTime
End Property
Public Property Let FileName(ByVal vData As String)
'设置文件名属性值
    mvarFileName = vData
End Property
Public Property Get FileName() As String
'获得文件名属性值
    FileName = mvarFileName
End Property
Public Property Let FilePath(ByVal vData As String)
'设置文件名属性值
    mvarFilePath = vData
End Property
Public Property Get FilePath() As String
'获得文件名属性值
    FilePath = mvarFilePath
End Property
Public Property Let Permissions(ByVal vData As String)
'设置权限属性值
    mvarPermissions = vData
End Property
ublic Property Get Permissions() As String
'获得权限属性值
```

```
    Permissions = mvarPermissions
End Property
Public Property Let Owner(ByVal vData As String)
'设置属主属性值
    mvarOwner = vData
End Property
Public Property Get Owner() As String
'获得属主属性值
    Owner = mvarOwner
End Property
Public Property Let Group(ByVal vData As String)
'设置组属性值
    mvarGroup = vData
End Property
Public Property Get Group() As String
'获得组属性值
    Group = mvarGroup
End Property
```

通过上面这个简单的类，可以让读者了解到类是如何工作的，以及它的一些属性如何设置和获得，当然还有其他的方法，不过不在本书的范围之内，读者可以查阅相关资料。这个类主要为了在后面显示文件列表时来获得每一个文件的信息，因此在从服务器得到数据的时候，先放在这个类中，然后统一显示出来。

2. **类模块** CftpFiles

这也是一个简单的类，主要是创建一个集合，这个集合是由类“ CftpFile **”组成的，用来存储所有的文件列表。其代码如下：**

```
'类模块内部变量，变量类型为 collection
Private mCol As Collection

Public Function Add(oFtpFile As CFtpFile, vKey As Variant) As CFtpFile
'向集合中加入一个 CftpFile 类的函数
    mCol.Add oFtpFile
    Set Add = oFtpFile
End Function

Public Property Get Item(vntIndexKey As Variant) As CFtpFile
'获得一个类成员的函数
    Set Item = mCol(vntIndexKey)
End Property
```

```
    Public Property Get Count() As Long
    '获得类成员数量的属性值
        Count = mCol.Count
    End Property

    Public Sub Remove(vntIndexKey As Variant)
    '删除一个类成员的过程
        mCol.Remove vntIndexKey
    End Sub

    Private Sub Class_Initialize()
    '类初始化事件
        Set mCol = New Collection
    End Sub

    Private Sub Class_Terminate()
    '类结束事件
        Set mCol = Nothing
    End Sub
```

这个类中，继续学习了一些类的知识，但是并没有和 FTP 拉上关系，因为在后面的代码中用到了该类，因此有必要介绍一下。这个类中增加了如何声明函数的应用，同时学习了如何使用两个事件来处理程序。读者应该养成使用类的好习惯，因为这样可以使程序有代码重复率低、更便于管理等优点。

3. 类模块 Ctimeout

这个类的功能就是通过 API 函数创建一个计时器，因为在连接服务器的时候，多数情况是需要用时间来限制的，如果时间得不到响应，应该做超时处理，这个时候就需要用到计时器来处理了。该类模块的代码如下：

```
    'SetTimerAPI 函数创建一个定时器
    Private Declare Function SetTimer Lib "user32" (ByVal hwnd As Long, ByVal nIDEvent
As Long, ByVal uElapse As Long, ByVal lpTimerFunc As Long) As Long
    'KillTimerAPI 函数销毁一个定时器
    Private Declare Function KillTimer Lib "user32" (ByVal hwnd As Long, ByVal
nIDEvent As Long) As Long

    Private m_lngTimerID As Long
    Private m_intTimeoutValue As Integer
    '


    '启动定时器函数
```

```
Public Sub StartTimer()
    If m_lngTimerID > 0 Then
        '如果已经创建定时器，则销毁
        KillTimer 0, m_lngTimerID
    End If
    MFtpSupport.p_intCounter = 0
    '创建新的定时器
    m_lngTimerID = SetTimer(0, 0, 1000, AddressOf MFtpSupport.TimerProc)
End Sub

'销毁定时器
Public Sub StopTimer()
    MFtpSupport.p_intCounter = 0
    KillTimer 0, m_lngTimerID
End Sub

'设置 TimeoutValue 属性值，用来设置时间限制
Public Property Let TimeoutValue(NewValue As Integer)
    m_intTimeoutValue = NewValue
End Property

'获得 TimeoutValue 属性值
Public Property Get TimeoutValue() As Integer
    TimeoutValue = m_intTimeoutValue
End Property

'获得 Timeout 属性值，用来判断是否已经超时
Public Property Get Timeout() As Boolean
                '为标准模块，intCounter 为一个全局变量，其直是通过回调函数来增加的
                '回掉函数在标准模块 MftpSupport 中
If MFtpSupport.p_intCounter > m_intTimeoutValue Then
            Timeout = True
            Call StopTimer
            Else
            Timeout = False
            End If
End Property
```

```
    '计数器回 0
    Public Sub Reset()
        MFtpSupport.p_intCounter = 0
    End Sub

    '类终止的时候同时销毁定时器
    Private Sub Class_Terminate()
        Call StopTimer
    End Sub
```

上面这个类的作用是实现定时器，其实对于初中级的读者，使用 VB 中自带的 Timer 控件就可以了。这里是通过 API 函数来实现的，而且还需要通过回调函数来实现。这个类中定义了一系列的函数和属性，主要是为了方便控制定时器。

4. 标准模块 MFtpSupport

这是一个标准的模块，其中定义了一个全局变量和一个函数，这两部分都是为计时器服务的。定时器的标记变量需要随时间变化，因此需要设定一个变量。函数是一个回调函数，这个函数有明确的格式，不能随意地写。创建一个计时器后，就可以通过这个函数进行一系列的处理。在本例题中，只是简单地将计数器的值增加一。对回调函数不熟悉读者，可以查看 MSDN。其原理就是将一个函数的地址传给另一个函数作为参数。可以将这个函数做为一个事件处理是因为每当经过一段时间后，系统就会处理这个函数，而不需要再在程序中调用，其工作是由 API 函数控制的。该模块的代码如下：

```
    '声明计数器变量
    Public p_intCounter As Integer
    'settimer 函数中的回调函数，每次将计数器变量增加 1
    Sub TimerProc(ByVal hwnd As Long, ByVal uMsg As Long, ByVal idEvent As Long,
ByVal dwTime As Long)
        p_intCounter = p_intCounter + 1
    End Sub
```

上面 TimerProc 函数后面的参数可以通过 MSDN 查到，不过没有 VB 格式，需要转化，API 的说明是用 C 语言来描述的，因此需要转化成 VB 的数据格式。

5. 类模块 CftpConnection

这个类程序是这个程序的核心，所有的 FTP 编程几乎都是在这个类中，因此读者要想搞清楚 FTP 协议的编程以及工作模式，就必须把这个类搞清楚。因为这个类的开发比较高级而且程序比较长，因此在代码解说的时候，将分成一段一段地解说。如果需要看完整的程序，可以参见本书所附光盘。

（1）变量和对象的声明

声明部分的代码如下：

```
    '*************************************************************************
********
```

```
    'CftpConnection 类
    '*************************************************************************
********
    '在该类中使用到以下几个类模块和标准模块
    'CFtpFile     类模块
    'CFtpFiles    类模块
    'CTimeout    类模块
    'MftpSupport  类模块

    '因为在该类中使用到 winsock 对象，所以应该在 REFERENCES 中加入对该对象的引用
    '*************************************************************************
********
    'winsock 对象的声明
    '*************************************************************************
********
    '声明用于进行控制连接的 winsock 对象
    Private WithEvents wscControl As MSWinsockLib.Winsock
    '声明用于进行数据连接的 winsock 对象
    Private WithEvents wscData As MSWinsockLib.Winsock
    '*************************************************************************
********
    '定义类模块内部的属性变量
    '*************************************************************************
********
    Private m_strUserName          As String         '用户名
    Private m_strPassword          As String          '密码
    Private m_varFtpServer         As Variant         '服务器地址
    Private m_strCurrentDirectory   As String           '服务器当前工作目录
    Private m_bPassiveMode         As Boolean       '定义连接模式
    Private m_bBusy               As Boolean       '表示程序正忙
    Private m_intTimeout           As Integer         '表示连接是否超时
    Private m_TransferMode         As FtpTransferModes '定义传输模式
    '*************************************************************************
********
    '公共枚举变量
    '*************************************************************************
********
    '定义连接服务器时的各种状态
    Public Enum FTP_CONNECTION_STATES
```

```
        FTP_CONNECTION_RESOLVING_HOST
    FTP_CONNECTION_HOST_RESOLVED
    …（详见源程序）
    End Enum
    'FTP 服务器返回的各种响应码
    Private Enum FTP_RESPONSE_CODES
        FTP_RESPONSE_RESTATRT_MARKER_REPLY = 110
        FTP_RESPONSE_SERVICE_READY_IN_MINUTES = 120
        FTP_RESPONSE_DATA_CONNECTION_ALREADY_OPEN = 125
        FTP_RESPONSE_FILE_STATUS_OK = 150
        FTP_RESPONSE_COMMAND_OK = 200
    FTP_RESPONSE_COMMAND_NOT_IMPLEMENTED_SUPERFLUOUS_AT_THIS_SITE = 202
    …（详见源程序）
    End Enum
    'ftp 传输模式
    Public Enum FtpTransferModes
        FTP_ASCII_MODE  'ASCII 模式
        FTP_IMAGE_MODE '二进制模式
    End Enum
    '*************************************************************************
********
    '各种错误信息
    '*************************************************************************
********
    Public Enum FtpErrors
        ERROR_FTP_WINSOCK_AddressInUse
        ERROR_FTP_WINSOCK_AddressNotAvailable
        ERROR_FTP_WINSOCK_AlreadyComplete
        ERROR_FTP_WINSOCK_AlreadyConnected
        ERROR_FTP_WINSOCK_BadState
    ERROR_FTP_WINSOCK_ConnectAborted
    …（详见源程序）
    End Enum
    '*************************************************************************
********
    '定义类事件
    '*************************************************************************
********
    Public Event StateChanged(State As FTP_CONNECTION_STATES)
```

```
    Public Event DownloadProgress(lBytes As Long)
    Public Event UploadProgress(lBytes As Long)
    Public Event ReplyMessage(ByVal sMessage As String)
    '*************************************************************************
********
    '类模块内部的变量和常数
    '*************************************************************************
********
    Const RESPONSE_CODE_LENGHT = 3        '定义返回码的长度为 3
    Private m_LastError           As FtpErrors   '定义最近一次错误
    Private m_strLastErrorDesc     As String      '定义最近一次错误的描述
    Private m_strWinsockBuffer     As String     '定义用来存放获得命令连接返回数据的
变量
    Private m_strDataBuffer        As String     '定义数据缓冲区变量
    Private m_strLocalFilePath     As String      '定义本地文件路径
    Private m_intLocalFileID       As Integer    '定义当前打开文件的文件号
    Private m_bTransferInProgress   As Boolean    '定义是否正在传输文件
    Private m_lDownloadedBytes     As Long    '定义已经下载的字节
    Private m_bUploadFile          As Boolean   '定义是否是上载文件
    Private m_lUploadedBytes       As Long     '定义已经上载的字节
    Private m_strLastServerResponse As String      '定义服务器最近一次的返回码
    Private m_objTimeOut           As Ctimeout  '定义是否已经超时
    Private m_bFileIsOpened        As Boolean   '定义文件是否已经被打开
```

在上面的定义代码中，读者要知道每个变量的基本含义，这样在看后面程序的时候会比较容易懂。而且通过上面的例子，读者要学会如何定义一个事件、一个枚举变量，声明一个类以及加入一个 REFERENCES 的对象应用。

（2）类属性的定义

下面定义的一些属性，主要是为了能在外部程序访问类内部的一些变量，从而能够进行判断和分析。

```
    '获得传输模式属性值
    Public Property Get TransferMode() As FtpTransferModes
        TransferMode = m_TransferMode
    End Property

    '设置传输属性值
    Public Property Let TransferMode(NewValue As FtpTransferModes)
        m_bBusy = True
        If Not (NewValue = m_TransferMode) Then
```

```
                    '首先发送改变模式命令，如果成功才设置新值，ProcessTYPECommand 函数
                    '将在后面介绍
            If ProcessTYPECommand(NewValue) Then
                m_TransferMode = NewValue
            End If
        End If
        m_bBusy = False
    End Property

    '获得连接模式属性值
    Public Property Get PassiveMode() As Boolean
        PassiveMode = m_bPassiveMode
    End Property

    '设置连接模式属性值
    Public Property Let PassiveMode(NewValue As Boolean)
        m_bPassiveMode = NewValue
    End Property

    '获得类是否正忙的属性
    Public Property Get Busy() As Boolean
        Busy = m_bBusy
    End Property

    '获得服务器地址属性值
    Public Property Get FtpServer() As Variant
        FtpServer = m_varFtpServer
    End Property

    '设置服务器地址属性值
    Public Property Let FtpServer(NewValue As Variant)
        m_varFtpServer = NewValue
    End Property

    '获得密码属性值
    Public Property Get Password() As String
        Password = m_strPassword
    End Property
```

```
'设置密码属性值
Public Property Let Password(NewValue As String)
    m_strPassword = NewValue
End Property

'获得用户名值
Public Property Get UserName() As String
    UserName = m_strUserName
End Property

'设置用户名属性值
Public Property Let UserName(NewValue As String)
    m_strUserName = NewValue
End Property

'获得超时时限值
Public Property Let Timeout(NewValue As Integer)
    m_intTimeout = NewValue
    m_objTimeOut.TimeoutValue = NewValue
End Property

'设置超时时限值
Public Property Get Timeout() As Integer
    Timeout = m_intTimeout
End Property
```

（3）类初始化事件和终止事件

在类初始化事件中，主要是创建一些其他类实例。类终止的时候释放一些资源，代码如下：

```
'类初始化事件
Private Sub Class_Initialize()
    Set wscControl = New MSWinsockLib.Winsock
    Set wscData = New MSWinsockLib.Winsock
    Set m_objTimeOut = New CTimeout
End Sub

'类终止事件
Private Sub Class_Terminate()
```

```
    Call BreakeConnection
    Set wscData = Nothing
    Set wscControl = Nothing

    m_objTimeOut.StopTimer
    Set m_objTimeOut = Nothing
End Sub
```

在类初始化的时候，创建了两个 winsock 对象的实例，一个用于控制连接，另一个用于数据连接，同时创建了一个计时器类实例。在类终止的时候首先关闭当前连接，然后释放每个类实例的资源。

（4）Winsock 对象的事件处理

在前面的学习中，读者已经了解到 FTP 的工作模式，也就是有两个 Socket 连接同时工作，一个用于进行控制连接，一个用于进行数据连接。因此在前面的声明中，声明了两个对象，这两个对象有几个事件，其中有一个事件是关键即 DataArrival，所有从服务器传来的数据都是从这个事件获得的。

控制连接的 DataArrival 事件处理代码如下：

```
Private Sub wscControl_DataArrival(ByVal bytesTotal As Long)
    On Error Resume Next
    Dim strData As String
    '
    '获得数据，将数据放在变量 strData 中
    wscControl.GetData strData
    '将数据叠加，因为服务器端数据可能不止作为一次发送到客户端
    m_strWinsockBuffer = m_strWinsockBuffer & strData
    m_strLastServerResponse = strData

'每次接收到数据的时候就将计时器清 0
    m_objTimeOut.Reset

    '分析服务器传回的数据，通过分析数据来获得返回码
    '如果返回码是 426，表示连接关闭、传输终止
    If GetResponseCode(strData) = 426 Then
        If m_bTransferInProgress Or m_bUploadFile Then
           '判断是否是在传输或者上载文件，如果是，则关闭数据连接
            wscData.Close
            Close m_intLocalFileID
            '初始化一些变量
```

```
            m_strDataBuffer = ""
            m_lDownloadedBytes = 0
            m_lUploadedBytes = 0
            m_bTransferInProgress = False
            m_bUploadFile = False
            m_bFileIsOpened = False
        End If
        wscControl.Close
        m_bBusy = False
    End If

    Debug.Print Left(strData, Len(strData) - 2)
    '然后激活一个事件
RaiseEvent ReplyMessage(Left(strData, Len(strData) - 2) & vbCrLf)
End Sub
```

上面代码的功能是在进行控制连接的时候处理从服务器传来的数据，这里要判断从服务器传来的数据。通常从服务器传来的数据，总是响应码后面跟着其他的信息，因此可以通过函数 GetResponseCode(strData)来分析返回码。返回码如果不是 426，则表示连接并没有断，否则表示连接已经关闭，传输终止，因此需要重新连接，同时需要关闭数据连接以及初始化一些数据。最后激活一个事件，让外界程序获得新接收到的数据。

数据连接的 DataArrival 事件处理代码如下：

```
Private Sub wscData_DataArrival(ByVal bytesTotal As Long)
    '该变量用来保存字符型的数据
    Dim strData As String
    '该变量用来保存二进制的数据
    Dim temparray() As Byte

    On Error Resume Next

    '如果接收到的数据数量为 0，则退出事件
    If bytesTotal = 0 Then Exit Sub

    '如果正在进行传输文件，主要是下载文件
    If m_bTransferInProgress Then
        '动态定义树组大小为实际传来数据的大小
        ReDim temparray(bytesTotal)
        '通过 getdata 方法来获得数据，注意要使用二进制格式接收
        wscData.GetData temparray(), vbByte
    Else
```

```
        '如果不是进行文件的确传输，则接收的其他数据，如接收到的目录数据
        '注意，如果是字符串，则可以以符串格式接收
            wscData.GetData strData
        End If

        '如果正在上传文件
        If m_bTransferInProgress And Not m_bUploadFile Then
            '如果文件没有打开，则打开文件，来接收从服务器传来的数据
            If Not m_bFileIsOpened Then
                '获得一个空的文件号
                m_intLocalFileID = FreeFile
                '以而进制方式打开文件
                Open m_strLocalFilePath For Binary As m_intLocalFileID

                '标志文件已经打开
                m_bFileIsOpened = True
              '初始化下载字节数为 0
                m_lDownloadedBytes＝0
            End If

            '将数据以二进制格式写入到文件
            Put m_intLocalFileID, , temparray
            m_lDownloadedBytes = m_lDownloadedBytes + bytesTotal
            '激发一个事件，让外部知道现在下载的字节数
            RaiseEvent DownloadProgress(m_lDownloadedBytes)
        Else
            '如果不是进行文件传输，则将数据写到缓冲区 m_strDataBuffer 中
            m_strDataBuffer = m_strDataBuffer & strData
        End If
        '计时器清 0
        m_objTimeOut.Reset
        Exit Sub
    End Sub
```

以上是数据连接的 DataArrival 的事件，在这个过程中，要处理来自服务器的数据，这个事件处理也是本程序的一个关键的地方，尤其是打印文件列表和进行下载文件的时候，都需要从这个事件中获得数据，而且在数据接收的时候一定要注意，对于不同的格式要用不同的方法。一般来说，文件的传输以二进制为主，因为二进制传输不但可以传输二进制的文件，还可以传输文本类型的文件。对于传来的字符型数据，处理起来很简单，直接将数据接收到一个字符串就可以了，而对于二进制的数据，用字符型接收，通常会出错。在许多的书上介

绍例题的时候，通常用“String=Space(bytesTotal)”来定义一个缓冲区，然后通过“GetData”方法来将数据写入缓冲区，但是这样的结果通常不对，笔者曾试过几个程序，下载文件并不完全。所以建议读者首先创建一个 byte 类型的数组，然后动态定义大小，这样一般是没有问题的。

在这个事件中，还要注意的一个地方就是，客户发出下载命令，在等待服务器从控制端口返回连接成功和传输已经开始的信息后，其实数据已经开始传送了，可以用两种方法来处理。第一种就是等待服务器返回开始传输信息后，打开文件，而此时虽然数据端口已经开始接收数据，但是如果文件没有打开，则不接收数据，一直等到文件打开，再将数据通过“GetData”方法获得。后一种方法就是判断是否为发送下载文件后得到的数据，如果肯定是下载文件的数据，则直接在该事件中打开文件，然后每次接收到数据，都将数据写入文件。本程序采用的是后者，即有数据到达时就打开文件并将数据写入文件。

当服务器发送完数据后，会自动关闭连接，所以客户端关闭文件的代码也可以发生在两个地方。一个就是在 Winsock 对象的 Close 事件中，因为服务器端的 Socket 关闭事件，会激发客户端的 Close 事件，或者可以等到从控制端口得到响应码为数据传输完毕，就可以关闭文件。

用于数据连接的 Winsock 对象，还有几个事件处理，这里一并介绍了。

```
'wscData 对象的 ConnectionRequest 事件
Private Sub wscData_ConnectionRequest(ByVal requestID As Long)
'如果以 PORT 方式建立连接，该事件用于接受对方的连接
If wscData.State <> sckClosed Then wscData.Close
    '通过 Accept 方法来建立一对一的数据连接
wscData.Accept (requestID)
 End Sub
```

在进行数据连接的时候，通过前面的介绍，读者可能已经知道连接方式有两种，一种是 Port 方式，另外一种是 Passive 方式。如果采用 Port 方式，则需要给服务器一个地址和端口，让服务器主动连接到本地机器上并且建立连接，因此当服务器主动连接时，便会产生该事件，然后客户端接受请求并建立连接。

```
' wscData 对象的 SendProgress 事件
Private Sub wscData_SendProgress(ByVal bytesSent As Long, ByVal bytesRemaining
As Long)
    m_lUploadedBytes = m_lUploadedBytes + bytesSent


    RaiseEvent UploadProgress(m_lUploadedBytes)
End Sub
```

该事件是 wscData 对象在发送数据的时候，会产生事件通知用户传输的字节数以及剩余的字节数，因此可以通过这个事件来统计传输速率。在许多的 FTP 客户端中，都会显示当前的传输速率以及已经传输的文件大小，通过这个事件就可以做到这两个功能。实现这两个功能并不难，只要掌握了它的本质或实现方法，就很简单了。

```
'wscData 对象的 SendComplete 事件
```

```
Private Sub wscData_SendComplete()
    '判断是否是正在进行文件的上载，如果是上载就调用 UploadData 函数继续上传
If m_bUploadFile Then
       Call UploadData(0)
    End If
    m_objTimeOut.Reset
End Sub
```

wscData 对象的 SendComplete 事件主要发生在 wscData 对象发送完一个数据包后，因此如果是在传输文件，则需要调用 UploadData 函数继续传送下一个数据包，直到整个文件传输完毕。

（5）与服务器协议通信

下面的代码和说明是这个类的核心，也是本程序的核心，只要读者掌握了这些命令的发送以及数据的接收，不但能够编写客户端程序，而且还能够编写服务器端的程序。因为在介绍 HTTP 协议的时候，就介绍了如何开发自己的 Web 服务器来与标准的浏览器通信，而且可以在服务器端做各种处理来满足自己的开发需求。如果是在局域网内部，就可以扩充自己的协议，或者扩充服务器端的功能，这样除了可以满足标准 FTP 客户连接，还可以让具有特定权限的人进行其他的操作，如协议好一些特殊的指令来执行特殊的功能。因此当读者学习完这本书以后，就会发现基于 TCP/IP 编程也都是基于协议的，而编程的本质并没有变。下面就对一些主要的命令进行说明，而没有说明的命令，读者可以查看源程序，原理都类似。

发送用户名命令，代码如下：

```
'该函数是进行用户名验证
Private Function ProcessUSERCommand() As FTP_RESPONSE_CODES
    Dim strData    As String
    On Error GoTo ProcessUSERCommand_Err_Handler
    '激发事件，通知外界正在进行用户名验证
    RaiseEvent StateChanged(FTP_CONNECTION_AUTHENTICATION)
                '分析用户名，确认是匿名的还是非匿名的
    m_strUserName = IIf(Len(m_strUserName) > 0, m_strUserName, "anonymous")
    '如果密码长度为 0
    If Len(m_strPassword) = 0 Then
        '则分析用户名是否为匿名，如果是，则密码用 E-mail 地址代替
        If m_strUserName = "anonymous" Then
            m_strPassword = "guest@unknown.com"
        Else
        End If
    End If
    '下面的这句代码是向服务器发送命令“USER”，然后跟着用户名和回车换行符
    '注意命令的发送格式
```

```
wscControl.SendData "USER " & m_strUserName & vbCrLf
    Debug.Print "USER " & m_strUserName
    '激发事件，通知外界正在发送用户名
    RaiseEvent ReplyMessage("USER " & m_strUserName & vbCrLf)
    '当发送完一个命令后，总是等待服务器端的响应
    '如果长时间不能接收到服务器的信息，则要进行超时处理
    '所以下面启动计时器
    m_objTimeOut.StartTimer
    '紧接着是一个无条件循环
    Do
        DoEvents
        '通过调用类 objTimeOut 的 Timeout 属性判断是否已经超时
        '如果超时，则退出循环
        If m_objTimeOut.Timeout Then
            m_LastError = ERROR_FTP_USER_TIMEOUT
            Exit Do
        End If
        '如果没有超时，则读取从控制接口传来的数据
        '如果数据大于 3，即 RESPONSE_CODE_LENGHT
        If Len(m_strWinsockBuffer) > RESPONSE_CODE_LENGHT Then
            '则表示能分析服务器的响应码，因为通常响应码总是再最前面的
            '将从控制接口的数据保存，然后清空缓冲数据
strData = m_strWinsockBuffer
            m_strWinsockBuffer = ""
            '退出循环
            Exit Do
        End If
    Loop
    '然后停止计时器
    m_objTimeOut.StopTimer

    '分析返回码
    Select Case GetResponseCode(strData)
        '如果返回的是 230，表示登录成功，则继续
        Case FTP_RESPONSE_USER_LOGGED_IN
            ProcessUSERCommand = FTP_RESPONSE_USER_LOGGED_IN
        '表示用户名验证成功，需要继续验证密码
        Case FTP_RESPONSE_USER_NAME_OK_NEED_PASSWORD
            ProcessUSERCommand = FTP_RESPONSE_USER_NAME_OK_NEED_PASSWORD
```

```
        Case Else
            ProcessFtpResponse GetResponseCode(strData)
    End Select

Exit_Label:
    Exit Function
'错误处理，通过调用对象 Err 来获得错误号和错误描述
ProcessUSERCommand_Err_Handler:
    If Not ProcessWinsockError(Err.Number, Err.Description) Then
        Err.Raise vbObjectError + 1000 + Err.Number,
"CFtpConnection.ProcessUSERCommand", Err.Description
    End If

End Function
```

上面的函数就是通过向服务器发送命令来进行登录的，这个过程和前面通过控件来实现是完全不同的，因为前面通过控件来实现，控件都已经把许多功能集成了，因此用户并没有多少灵活性来编写自己的程序或者作相应的处理。实际上，如果用户掌握了这些具体的协议以及具体的执行步骤，同样可以开发自己的 FTP 服务器端程序。当然如果要开发一个完整的服务器端程序，需要考虑许多的东西，比如安全性、权限设置等。

通常如果不是匿名登录，如果登录成功，下一步总是需要进行密码验证的。接下来看看密码验证的函数。

发送密码命令，代码如下：

```
'密码验证的函数，该函数的工作过程同上面的一样
Private Function ProcessPASSCommand() As FTP_RESPONSE_CODES
    Dim strResponse As String
    Dim strData    As String
    '
    On Error GoTo ProcessPASSCommand_Err_Handler
    '向服务器发送 PASS 命令，然后跟着具体的密码和回车符
    wscControl.SendData "PASS " & m_strPassword & vbCrLf
    Debug.Print "PASS " & m_strPassword
    RaiseEvent ReplyMessage("PASS " & "**********" & vbCrLf)
    '启动定时器
    m_objTimeOut.StartTimer
    Do
        DoEvents
        '判断是否已经超时
        If m_objTimeOut.Timeout Then
            m_LastError = ERROR_FTP_USER_TIMEOUT
```

```
            Exit Do
        End If
        '分析从服务器端传来的数据
        If Len(m_strWinsockBuffer) > RESPONSE_CODE_LENGHT Then
            strData = m_strWinsockBuffer
            Exit Do
        End If
    Loop
    m_objTimeOut.StopTimer
    '分析响应码
    If GetResponseCode(strData) = FTP_RESPONSE_USER_LOGGED_IN Then
        Do
            DoEvents
            '分析缓冲区中的字符串，如果包含有 230，则表示密码验证通过
            If InStr(1, m_strWinsockBuffer, "230 ") > 0 Then
                ProcessPASSCommand = FTP_RESPONSE_USER_LOGGED_IN
                '然后清除缓冲区
                m_strWinsockBuffer = ""
                Exit Function
            End If
        Loop
    Else
        ProcessFtpResponse GetResponseCode(strData)
    End If
    ProcessPASSCommand = GetResponseCode(strData)

Exit_Label:
    Exit Function
'出错处理
ProcessPASSCommand_Err_Handler:
    If Not ProcessWinsockError(Err.Number, Err.Description) Then
        Err.Raise vbObjectError + 1000 + Err.Number,
"CFtpConnection.ProcessPASSCommand", Err.Description
    End If
    GoTo Exit_Label
End Function
```

从上面的代码可以看到，工作过程同发送用户名是一样的，即发送完以后，启动定时器，然后分析从服务器传来的数据，根据返回不同的值作不同的处理。因为后面其他的命令工作过程也是一样，因此对于每个命令函数，只是列出其中关键的一段。详细代码参见本书所附

光盘中的程序。

发送 REST 命令的，代码如下：

```
'该函数是向服务器端发送 REST 命令
Private Function ProcessRESTCommand(lStartPoint As Long) As Boolean
    Dim strResponse As String
    Dim strData    As String

    On Error GoTo ProcessRESTCommand_Err_Handler
    '发送 REST 命令
    wscControl.SendData "REST " & lStartPoint & vbCrLf
    Debug.Print "REST " & lStartPoint
    RaiseEvent ReplyMessage("REST " & lStartPoint & vbCrLf)

    m_objTimeOut.StartTimer
    Do
        DoEvents
       ……（后面的代码省略）
End Function
```

该函数是断点续传的基础，因为发送该命令是让服务器设置传输文件的起始位置而已。其中“lStartPoint”就是起始传送的位置。通常的做法是，首先比较本地文件的长度和服务器端该文件的长度，如果服务器端的文件长度大于本地文件的长度，则提示用户是否要续传还是覆盖。如果用户选择续传，则向服务器发送“REST”命令，并传递起始位置即该文件的长度到服务器，服务器就从该长度以后获得文件内容然后发送给客户。

发送下载文件命令，代码如下：

```
Private Function ProcessRETRCommand(strFileName As String, lStartPoint As Long)
As Boolean
    Dim strResponse As String
    Dim strData    As String
    On Error GoTo ProcessRETRCommand_Err_Handler
    '清空数据连接缓冲区的内容
    m_strDataBuffer = ""
    '发送 RETR 命令
    wscControl.SendData "RETR " & strFileName & vbCrLf
    Debug.Print "RETR " & strFileName
    RaiseEvent ReplyMessage("RETR " & strFileName & vbCrLf)
    m_objTimeOut.StartTimer
    AllBytes = 0
    Do
        DoEvents
```

```
            '超时处理
            If m_objTimeOut.Timeout Then
                m_LastError = ERROR_FTP_USER_TIMEOUT
                Exit Do
            End If
            '如果不是再传输文件，则获得控制连接的数据
            If Not m_bTransferInProgress Then
                strData = m_strWinsockBuffer
                Exit Do
            End If

            If InStr(1, m_strWinsockBuffer, vbCrLf) > 0 Then
                If GetResponseCode(m_strWinsockBuffer) = 150 Or _
                    GetResponseCode(m_strWinsockBuffer) = 125 Then
                    '如果开始传送的位置是 0，同时被下载的文件已经存在，则首先删除该文件
                    '因为 lStartPoint 为 0 并且文件存在，表示不需要断点续传
                    If lStartPoint = 0 And FileExists(m_strLocalFilePath) Then
                        Kill m_strLocalFilePath
                    End If
```

'如果文件没有打开，则需要打开文件接收数据。不过根据笔者的经验，通常当控制'端接口获得响应码为 150 或者 125 的时候数据连接已经开始了，因此是在本程序中，发送下载文件命令后，如果数据端口

```
                    '接收到数据，则立即打开文件并接收数据
                    If Not m_bFileIsOpened Then
                        m_intLocalFileID = FreeFile
                        If m_bFileIsOpened Then
                            Open m_strLocalFilePath For Binary As m_intLocalFileID
                        End If
                        '如果是断点续传，则将文件的指针定位到起始位置的下一个字节
                        If lStartPoint > 0 Then
                            Seek m_intLocalFileID, lStartPoint + 1
                        End If
                        '然后修改标记，表示文件已经打开
                        m_bFileIsOpened = True
                        '初始化下载字节数为 0
                        m_lDownloadedBytes = 0
                    End If
                m_strWinsockBuffer    =    Mid$(m_strWinsockBuffer,    InStr(1,
```

```
m_strWinsockBuffer, vbCrLf) + 2)
                RaiseEvent StateChanged(FTP_TRANSFER_STARTING)
            Else
                '如果响应码不为 125 或者 150，则表示请求失败，退出循环
                strData = m_strWinsockBuffer
                m_strWinsockBuffer = ""
                Exit Do
            End If
        End If
    Loop
    m_objTimeOut.StopTimer
……
End Function
```

在下载文件的代码中，主要是向服务器发送下载文件的命令，而在打开文件后准备接收数据的代码并不一定能被执行到，因为当接收到响应码为 125 或者 150 的时候，数据连接已经建立，因此可以在数据连接中判断是否当前正处在下载文件的过程中，如果是，则表示接收到的数据是要下载的文件，可以在数据连接的 DataArrival 事件中打开文件并写入数据。

发送上载文件命令，代码如下：

```
'该函数的功能是向服务器发送 STOR 命令，让服务器准备接收一个来自数据连接的文件
Private Function ProcessSTORCommand(strLocalFileName As String,
strRemoteFileName As String, lStartPoint As Long) As Boolean
    Dim strResponse As String
    Dim strData    As String

    On Error GoTo ProcessSTORCommand_Err_Handler

    m_strDataBuffer = ""
    '发送上载命令，命令后接上载到服务器的路径和文件名
    wscControl.SendData "STOR " & strRemoteFileName & vbCrLf
    Debug.Print "STOR " & strRemoteFileName
    RaiseEvent ReplyMessage("STOR " & strRemoteFileName & vbCrLf)

    m_objTimeOut.StartTimer
    Do
        DoEvents
        '超时处理
        If m_objTimeOut.Timeout Then
            m_LastError = ERROR_FTP_USER_TIMEOUT
            Exit Do
```

```
        End If
        '同样也是等待控制连接的数据，如果控制连接的数据中包含有回车符
        '则表示有响应码返回
        If InStr(1, m_strWinsockBuffer, vbCrLf) > 0 Then
            '如果响应码为 150 或者 125，表示可以发送数据
            If GetResponseCode(m_strWinsockBuffer) = 150 Or _
                GetResponseCode(m_strWinsockBuffer) = 125 Then

                '清空控制连接缓冲区内容
                m_strWinsockBuffer = ""
                RaiseEvent StateChanged(FTP_TRANSFER_STARTING)
                '获得要上载的本地文件的信息
                m_strLocalFilePath = strLocalFileName
                '通过调用 UploadData 函数上载文件
                Call UploadData(lStartPoint)
            Else
                '如果响应码不是 125 或者 150，则表示请求失败
                strData = m_strWinsockBuffer
                m_strWinsockBuffer = ""
                Exit Do
            End If
        End If
    Loop
    m_objTimeOut.StopTimer
.....
End Function
```

这个程序的主要功能是向服务器发送上载命令，然后等待服务器的返回。如果客户有权限上载，则会返回相应的响应码，否则会返回请求失败的响应码。具体发送文件的功能是在一个函数 UploadData 中实现的。注意在上载命令“ STOR ”后的参数应该是远程服务器的文件信息，也就是上传到服务器后的路径和文件名。

前面学习了一些重要的命令，如上载和下载，同时它们也是一些基本的命令，是 FTP 的基本功能。FTP 协议中，还有其他许多的命令和协议，接下来继续介绍一些笔者认为比较重要的函数和命令。

设定传输模式，代码如下：

```
'该函数的功能是向服务器发送 TYPE 命令，表示传输模式
Private Function ProcessTYPECommand(vType As FtpTransferModes) As Boolean
    Dim strResponse As String
    Dim strData    As String
```

```
        On Error GoTo ProcessTYPECommand_Err_Handler
        '发送设置传输模式的命令，通过命令 TYPE 设置，A 表示 ASCII 码格式，而 I 表示二进制
        wscControl.SendData "TYPE " & IIf(vType = FTP_ASCII_MODE, "A", "I") & vbCrLf
        Debug.Print "TYPE " & IIf(vType = FTP_ASCII_MODE, "A", "I")
        RaiseEvent ReplyMessage("TYPE " & IIf(vType = FTP_ASCII_MODE, "A", "I")
& vbCrLf)

        m_objTimeOut.StartTimer
        Do
              DoEvents
           '超时处理
           If m_objTimeOut.Timeout Then
              m_LastError = ERROR_FTP_USER_TIMEOUT
              Exit Do
           End If
           '当控制连接缓冲区中的数据含有回车符的时候，取出缓冲区内容
           If InStr(1, m_strWinsockBuffer, vbCrLf) > 0 Then
              strData = m_strWinsockBuffer
              m_strWinsockBuffer = ""
              Exit Do
           End If
        Loop
        m_objTimeOut.StopTimer
        '分析缓冲区中的数据，如果返回响应码为常数 FTP_RESPONSE_COMMAND_OK
        '表示转换模式成功
        If GetResponseCode(strData) = FTP_RESPONSE_COMMAND_OK Then
           ProcessTYPECommand = True
        Else
           ProcessFtpResponse GetResponseCode(strData)
        End If
    ……
    End Function
```

通常在 FTP 的文件传输过程中，有两种格式：一种是基于字符传输的，也就是 ASCII 码；另外一种，也是最常用的一种传输模式。ASCII 格式只能传输文本文件，而二进制格式能传输各种格式，因此在传输模式的时候，通常选择二进制格式就可以了。其中如果是转换成 ASCII 格式，发送命令参数为“A”，否则发送“I（Image）”。如果响应码为 200，表示转换成功，否则失败。

下面的两个函数是设置连接模式的。只有对 FTP 的工作模式了解后才能真正了解这两段代码。先看这两个函数的具体代码，然后再解释。

连接模式设置，代码如下：

```
'该函数的功能是设置连接模式为被动模式
Private Function ProcessPASVCommand() As Boolean
    Dim strResponse As String
    Dim strData    As String

    On Error GoTo ProcessPASVCommand_Err_Handler

    RaiseEvent StateChanged(FTP_ESTABLISHING_DATA_CONNECTION)
    '发送 PASV 命令，将连接模式设定为被动模式
    wscControl.SendData "PASV" & vbCrLf
    Debug.Print "PASV"
    RaiseEvent ReplyMessage("PASV" & vbCrLf)

    m_objTimeOut.StartTimer
    Do
        DoEvents
        '超时处理
        If m_objTimeOut.Timeout Then
            m_LastError = ERROR_FTP_USER_TIMEOUT
            Exit Do
        End If
        '如果控制连接缓冲区中含有回车符，则表示有服务器返回响应码
        If InStr(1, m_strWinsockBuffer, vbCrLf) > 0 Then
            strData = m_strWinsockBuffer
            m_strWinsockBuffer = ""
            Exit Do
        End If
    Loop
    m_objTimeOut.StopTimer
    '如果转换被动模式成功，则调用函数 MakePassiveDataConnection 建立被动连接
    If GetResponseCode(strData) = FTP_RESPONSE_ENTERING_PASSIVE_MODE Then
        ProcessPASVCommand = MakePassiveDataConnection(strData)
    Else
        ProcessFtpResponse GetResponseCode(strData)
    End If
……
End Function
'该函数的功能是发送 PORT 命令，为数据连接指定一个 IP 地址和本地地址
```

```
Private Function ProcessPORTCommand() As Boolean
    Dim intPort        As Integer
    Dim strIPAddress    As String
    Dim colIPAddresses  As New Collection
    Dim strSend        As String
    Dim strData        As String

    On Error Resume Next

    RaiseEvent StateChanged(FTP_ESTABLISHING_DATA_CONNECTION)

    Do
        '通过函数 GetFreePort 获得一个自由端口
        intPort = GetFreePort
        '如果数据连接的 winsock 对象没有关闭，则关闭对象
        If wscData.State <> sckClosed Then wscData.Close
        '设置本地端口
        wscData.LocalPort = intPort
        '然后侦听端口的连接
        wscData.Listen
        If Not Err Then Exit Do
    Loop

    On Error GoTo ProcessPORTCommand_Err_Handler
    '获得本地机器的 IP 地址，可以通过 Winsock 对象的 LocalIP 属性获得 IP 地址
    strIPAddress = CStr(wscControl.LocalIP)
    '向服务器发送 PORT 命令，并将本地机的 IP 地址和端口号
    strSend = "PORT " & Replace(strIPAddress, ".", ",")
    strSend = strSend & "," & intPort \ 256 & "," & (intPort Mod 256)
    strSend = strSend & vbCrLf
    '向服务器发送命令
    wscControl.SendData strSend
    Debug.Print Left(strSend, Len(strSend) - 2)
    RaiseEvent ReplyMessage(Left(strSend, Len(strSend) - 2) & vbCrLf)

    m_objTimeOut.StartTimer
    Do
        DoEvents
        '超时处理
```

```
        If m_objTimeOut.Timeout Then
            m_LastError = ERROR_FTP_USER_TIMEOUT
            Exit Do
        End If
        '分析服务器响应码
        If InStr(1, m_strWinsockBuffer, vbCrLf) > 0 Then
            strData = m_strWinsockBuffer
            m_strWinsockBuffer = ""
            Exit Do
        End If
    Loop
    m_objTimeOut.StopTimer
    '如果响应码为 FTP_RESPONSE_COMMAND_OK，则表示 PORT 连接模式建立
    If GetResponseCode(strData) = FTP_RESPONSE_COMMAND_OK Then
        ProcessPORTCommand = True
        RaiseEvent StateChanged(FTP_DATA_CONNECTION_ESTABLISHED)
    Else
        ProcessFtpResponse GetResponseCode(strData)
    End If
……

End Function
```

函数 ProcessPASVCommand 和函数 ProcessPORTCommand 是建立两种不同的连接模式。其中一种是被动模式，一种是 PORT 模式。前面曾经介绍过，FTP 在建立数据连接的时候有这两种模式。当建立被动模式的时候，服务器会向客户发送服务器端 IP 地址和一个临时端口，然后客户连接到服务器，因此当返回响应码表示被动模式建立后，客户端要处理服务器端的数据以得到服务器端的地址和端口，通过函数 MakePassiveDataConnection 来建立连接。而 PORT 模式是本地机器作为服务器，向服务器发送本机地 IP 址和端口，然后等待服务器的连接，因此也就不需要调用额外的函数来建立连接。这两种模式工作的次序是刚好相反的，程序代码也有些区别。而且 FTP 服务器在传输文件的时候，端口都是临时的。每当传输完一个文件后，端口就会关闭，然后重新建立一个 SOCKET 连接。

由于 FTP 协议中的命令非常多，因此在本书中不一一介绍，以下列出其他命令发送代码中的一部分，并作简单解释。

发送命令 ABOR，以终止上一次 FTP 服务器命令以及数据传输。

```
'该函数的功能是终止上一次 ftp 服务器命令及相关数据传输
Private Function ProcessABORCommand() As Boolean
    Dim strResponse As String
    Dim strData    As String
```

```
    On Error GoTo ProcessABORCommand_Err_Handler
    wscControl.SendData "ABOR" & vbCrLf
    Debug.Print "ABOR"
    RaiseEvent ReplyMessage("ABOR" & vbCrLf)
……
End Function
```

发送 CDUP **命令，更改服务器当前目录为根目录。**

```
'该函数的功能是把当前目录改为远程文件系统的根目录而无需改变登录、帐号信息或传输参数
Private Function ProcessCDUPCommand() As Boolean
    Dim strResponse As String
    Dim strData    As String

    On Error GoTo ProcessCDUPCommand_Err_Handler

    wscControl.SendData "CDUP" & vbCrLf
    Debug.Print "CDUP"
    RaiseEvent ReplyMessage("CDUP" & vbCrLf)
    …(详见本书所附光盘中的源程序)
End Function
```

发送 CWD **命令，以改变当前目录为指定目录，后面的参数为指定的服务器目录。**

```
'该函数的功能是改变当前目录为指定的目录
Private Function ProcessCWDCommand(strNewDir As String) As Boolean
    Dim strResponse As String
    Dim strData    As String

    On Error GoTo ProcessCWDCommand_Err_Handler

    wscControl.SendData "CWD " & strNewDir & vbCrLf
    Debug.Print "CWD " & strNewDir
    RaiseEvent ReplyMessage("CWD " & strNewDir & vbCrLf)
End Function
```

发送 DELE **命令，以删除指定的文件，其中参数为指定的服务器文件路径和文件名。**

```
'该函数的功能是删除指定的文件
Private Function ProcessDELECommand(strFileName As String) As Boolean
    Dim strResponse As String
    Dim strData    As String
    '
    On Error GoTo ProcessDELECommand_Err_Handler
```

```
    wscControl.SendData "DELE " & strFileName & vbCrLf
    Debug.Print "DELE " & strFileName
    RaiseEvent ReplyMessage("DELE " & strFileName & vbCrLf)
……
End Function
```

发送 LIST 命令，功能是获得指定目录中的子目录、文件列表或者是指定文件的信息

```
'该函数的功能是向服务器发送 LIST 命令
Private Function ProcessLISTCommand() As Boolean
On Error GoTo ProcessLISTCommand_Err_Handler
    Dim strResponse As String
    Dim strData    As String
    wscControl.SendData "LIST" & vbCrLf
    Debug.Print "LIST"
    RaiseEvent ReplyMessage("LIST")
……
End Function
```

发送 MKD 命令，用来建立新的目录，其中参数为要新建的目录服务器路径和新建目录名字。

```
'该函数的功能是创建新的目录
Private Function ProcessMKDCommand(strDirName As String) As Boolean
    Dim strResponse As String
    Dim strData    As String
    '
    On Error GoTo ProcessMKDCommand_Err_Handler

    wscControl.SendData "MKD " & strDirName & vbCrLf
    Debug.Print "MKD " & strDirName
    RaiseEvent ReplyMessage("MKD " & strDirName & vbCrLf)
……
End Function
```

发送 PWD 命令，以打印当前服务器的工作目录。

注意，这里返回的当前目录不是从数据连接过来的，而是作为控制连接的一部分发送到客户端的。这一点要注意同命令 LIST 的区别，LIST 命令获得文件和目录信息是从数据连接发送过来的。

```
'该函数是获得当前工作目录的名称
Private Function ProcessPWDCommand() As Boolean

    Dim strResponse As String
```

```
    Dim strData    As String

    On Error GoTo ProcessPWDCommand_Err_Handler
    '发送 PWD 命令
    wscControl.SendData "PWD" & vbCrLf
    Debug.Print "PWD"
    RaiseEvent ReplyMessage("PWD" & vbCrLf)

    m_objTimeOut.StartTimer
    Do
        DoEvents
        '超时处理
        If m_objTimeOut.Timeout Then
            m_LastError = ERROR_FTP_USER_TIMEOUT
            Exit Do
        End If
        If InStr(1, m_strWinsockBuffer, vbCrLf) > 0 Then
        '当从服务器中得到的响应码中含有回车的时候，表示获得目录
            strData = m_strWinsockBuffer
            m_strWinsockBuffer = ""
            Exit Do
        End If
    Loop
    m_objTimeOut.StopTimer

    If GetResponseCode(strData) = FTP_RESPONSE_PATHNAME_CREATED Then
    '如果响应码为 257，表示已经创建"Path Name"
    '以下代码是分离出当前的服务器目录
        Dim intPosA As Integer, intPosB As Integer
        intPosA = InStr(1, strData, Chr$(34)) + 1
        intPosB = InStr(intPosA, strData, Chr$(34))
        If intPosA > 1 And intPosB > 0 Then
            m_strCurrentDirectory = Mid$(strData, intPosA, intPosB - intPosA)
            ProcessPWDCommand = True
        Else
            '未知的响应格式
        End If
    Else
        ProcessFtpResponse GetResponseCode(strData)
```

```
    End If
……
End Function
```

发送 QUIT 命令，该命令的功能是终止连接，注销用户。

```
'该函数的功能是终止连接，注销用户
Private Function ProcessQUITCommand() As Boolean
    Dim strResponse As String
    Dim strData    As String

    On Error GoTo ProcessQUITCommand_Err_Handler

    wscControl.SendData "QUIT" & vbCrLf
    Debug.Print "QUIT"
    RaiseEvent ReplyMessage("QUIT" & vbCrLf)
……
End Function
```

发送 RMD 命令，用来删除服务器的一个目录，后面的参数是准备删除的服务器路径和被删除的目录名。

```
'该函数的功能是删除一个目录
Private Function ProcessRMDCommand(strDirName As String) As Boolean
    Dim strResponse As String
    Dim strData    As String

    On Error GoTo ProcessRMDCommand_Err_Handler
    wscControl.SendData "RMD " & strDirName & vbCrLf
    Debug.Print "RMD " & strDirName
    RaiseEvent ReplyMessage("RMD " & strDirName & vbCrLf)
……
End Function
```

发送 RNFR 命令，用来指定将被更改的服务器文件的路径和文件名，其中参数是将被更名的服务器端文件路径和文件名。

```
'该函数的功能是要更改一个文件名，指定要更改的老文件名和路径
Private Function ProcessRNFRCommand(strFileName As String) As Boolean
    Dim strResponse As String
    Dim strData    As String
    On Error GoTo ProcessRNFRCommand_Err_Handler

    wscControl.SendData "RNFR " & strFileName & vbCrLf
```

```
    Debug.Print "RNFR " & strFileName
    RaiseEvent ReplyMessage("RNFR " & strFileName & vbCrLf)
……
End Function
```

发送 RNTO 命令，用来指定被更名文件的新路径和文件名。其中参数即新的文件路径和文件名。

注意这个命令要同前面的那个命令 RNFR 结合起来才能正确的更改文件名。先通过命令 RNFR 指定原来的文件路径和文件名，然后通过 RNTO 指定新的文件路径和文件名。

```
'该函数的功能是指定要更名文件的新的路径和文件名
Private Function ProcessRNTOCommand(strFileName As String) As Boolean
    Dim strResponse As String
    Dim strData    As String

    On Error GoTo ProcessRNTOCommand_Err_Handler

    wscControl.SendData "RNTO " & strFileName & vbCrLf
    Debug.Print "RNTO " & strFileName
    RaiseEvent ReplyMessage("RNTO " & strFileName & vbCrLf)
……
End Function
```

（6）重要函数

前面介绍的一系列的函数，是与服务器直接进行通信的函数，主要是发送服务器能够识别的各种命令，然后等待服务器的响应码。但是一个完整的 FTP 客户端程序，并不仅仅是这些函数就能完成的，它需要其他一些代码或者函数的调用。而且这些函数之间有一定的工作次序，如果次序错误，也会得到错误的结果或者失败的请求。因此有必要对另外一些使程序功能完整的重要函数作一个讲解。

首先介绍连接服务器的函数，当用户准备连接一个服务器的时候，总是首先设置好服务器的地址、用户名和密码，然后连接服务器。具体的代码见下面的函数。

```
'连接服务器的函数
Public Function Connect() As Boolean
On Error GoTo Connect_Err_Handler
Dim strData    As String
m_strWinsockBuffer = ""
'在进行一个操作之前，总是需要将变量 m_bBusy 设置为 True，这样表示程序正在工作
'这样做的目的是防止用户进行多次同样的操作，如果不限制，程序容易崩溃
m_bBusy = True
'首席判断服务器地址是否为空，如果不为空，就从控制接口连接服务器
If Len(m_varFtpServer) > 0 Then
```

```
    With wscControl
        '首先关闭当前的控制连接
        .Close
        .LocalPort = 0
        '连接服务器，控制端口默认的是 21
        .Connect m_varFtpServer, 21
        m_objTimeOut.StartTimer
        Do
            DoEvents
            '超时处理
            If m_objTimeOut.Timeout Then
                m_LastError = ERROR_FTP_USER_TIMEOUT
                Exit Do
            End If
            If .State = sckConnected Then
            '如果连接成功
                m_objTimeOut.StopTimer
                RaiseEvent StateChanged(FTP_CONNECTION_CONNECTED)
                m_objTimeOut.StartTimer
                Do
                    DoEvents
                    '超时处理
                    If m_objTimeOut.Timeout Then
                        m_LastError = ERROR_FTP_USER_TIMEOUT
                        Exit Do
                    End If
                    '分析是否已经有响应码返回
                    If Len(m_strWinsockBuffer) > (RESPONSE_CODE_LENGHT - 1) Then
                        strData = m_strWinsockBuffer
                        m_strWinsockBuffer = ""
                        Exit Do
                    End If
                Loop
                m_objTimeOut.StopTimer
                '分析响应码，通常是获得开始的 3 个字符
                Select Case GetResponseCode(strData)
                    '如果响应码是 220，则表示服务器已经为用户准备好
                    Case FTP_RESPONSE_SERVICE_READY_FOR_NEW_USER
                        '然后调用 ProcessUSERCommand 函数发送用户名
```

```
                    Select Case ProcessUSERCommand
                        Case FTP_RESPONSE_USER_LOGGED_IN
                        '如果返回 230，表示登录成功，不需要继续验证密码
                          Connect = True
                        Case FTP_RESPONSE_USER_NAME_OK_NEED_PASSWORD
                        '如果返回 331，表示用户名正确，但需要继续验证密码，因此
                        '需要调用函数 ProcessPASSCommand 发送密码，如果发送密码验
证成
                        '功，则表示连接服务器成功
                           If             ProcessPASSCommand             =
FTP_RESPONSE_USER_LOGGED_IN Then
                                Connect = True
                           End If
                    End Select
                    If Connect Then
                    '如果连接成功，则调用函数 ProcessPWDCommand 打印当前服务器目录
                        Call ProcessPWDCommand
                    End If
                Case FTP_RESPONSE_SERVICE_READY_IN_MINUTES
                '如果返回 120，则表示服务器将在 nnn 分钟后准备好
                   m_LastError                                              =
ERROR_FTP_PROTOCOL_SERVICE_READY_IN_MINUTES
                Case
FTP_RESPONSE_SERVICE_NOT_AVAILABLE_CLOSING_CONTROL_CONNECTION
                '如果返回 421，表示服务不可用，关闭连接
                   m_LastError                                              =
ERROR_FTP_PROTOCOL_SERVICE_NOT_AVAILABLE_CLOSING_CONTROL_CONNECTION
                End Select
                Exit Do
            '以下表示 Socket 连接没有建立，具体请看 Winsock 对象的介绍
            ElseIf .State = sckConnectAborted Then
                m_LastError = ERROR_FTP_WINSOCK_ConnectAborted
            ElseIf .State = sckResolvingHost Then
                RaiseEvent StateChanged(FTP_CONNECTION_RESOLVING_HOST)
            ElseIf .State = sckHostResolved Then
                RaiseEvent StateChanged(FTP_CONNECTION_HOST_RESOLVED)
            End If
        Loop
        m_objTimeOut.StopTimer
```

```
    End With
Else
    '如果服务器地址为空，则退出函数
    Connect = False
    Exit Function
End If
……
End Function
```

在 Connect 函数中，读者要掌握连接服务器是如何设置的，以及其连接次序如何。由于 FTP 连接本质上也是 Socket 连接，因此首先要通过 Socket 对象建立连接，先建立的是控制连接。控制连接所连接的就是默认的端口，通常是 21，要根据服务器的设置而定。这个端口一般设置完后，用户在连接的过程中是不会改变的。与服务器连接上以后，就需要发送用户名和密码，如果只有用户名，没有密码，只需要发送用户名验证。如果有密码，则在通过用户名验证后还需要发送密码进行验证，只有两种都通过后，才能真正建立连接。连接建立后，需要调用打印目录函数打印当前目录。

成功连接服务器后，首先需要列出当前目录下的所有的子目录和文件。这里有一个比较重要的函数 EnumFiles，通过这个函数，将获得一个目录下的子目录和文件的信息，现在来看看该函数。

```
'该函数的功能是获得当前目录下面的所有文件和目录
Public Function EnumFiles(oFiles As CFtpFiles) As Boolean
Dim bDataConnectionEstablished As Boolean
    m_bBusy = True
    If m_bPassiveMode Then
        '建立被动连接
        bDataConnectionEstablished = ProcessPASVCommand
    Else
        '建立 PORT 连接模式
        bDataConnectionEstablished = ProcessPORTCommand
    End If
    '如果连接模式成功建立
    If bDataConnectionEstablished Then
        RaiseEvent StateChanged(FTP_RETRIEVING_DIRECTORY_INFO)
        '调用 ProcessLISTCommand 函数发送 LIST 命令以获得目录下的信息
        If ProcessLISTCommand Then
            m_objTimeOut.StartTimer
            Do
                DoEvents
                '如果超时，判断是否已经获得信息
                If m_objTimeOut.Timeout Then
```

```
                    m_LastError = ERROR_FTP_USER_TIMEOUT
                    '分析响应码，然后确定是否已经获得文件信息
                    If  GetResponseCode(Left(m_strLastServerResponse,  3))  =
FTP_RESPONSE_CLOSING_DATA_CONNECTION Then
                        '调用函数 GetFileList(m_strDataBuffer)来列举文件和子目录信
息
                        Set oFiles = GetFileList(m_strDataBuffer)
                        EnumFiles = True
                        RaiseEvent StateChanged(FTP_DIRECTORY_INFO_COMPLETED)
                        m_strDataBuffer = ""
                    End If
                    Exit Do
                End If
                '如果数据连接正在关闭，也表示获得数据
                If wscData.State = sckClosing Or wscData.State = sckClosed Then
                    Set oFiles = Nothing
                    Set oFiles = GetFileList(m_strDataBuffer)
                    EnumFiles = True
                    RaiseEvent StateChanged(FTP_DIRECTORY_INFO_COMPLETED)
                    m_strDataBuffer = ""
                    Exit Do
                End If
            Loop
            m_objTimeOut.StopTimer
        Else
            '产生错误，不能建立连接
        End If
    Else
        '产生错误，不能建立连接
    End If
……
End Function
```

读者在学习这个函数的时候，要注意在发送 LIST 命令之前，首先要建立数据连接，因为 LIST 命令获得文件和目录信息是从数据端口返回到客户端的。因此要建立被动连接或者 PORT 连接模式。最后要知道什么时候服务器已经完成发送目录和文件的信息。当发送完毕后，就可以调用其他程序来分析返回的数据，以列出每个文件或者目录的信息。

当执行了上一个函数后，就能获得服务器返回的数据，目录数据是按照一定的格式发送到客户端的，因此在分离出各个目录和文件的时候，也要按照一定的格式。先看下面的函数 GetFileList。

```
'分析服务器返回的数据
Private Function GetFileList(strListing As String) As CFtpFiles
    Dim vFiles     As Variant
    Dim vFile      As Variant
    Dim vComponents As Variant
    Dim oFtpFile   As CFtpFile
    Dim oFtpFiles  As New CFtpFiles

    On Error Resume Next
    '释放类占用的资源
    Set GetFileList = Nothing
    '通常服务器返回的各个数据项之间是通过回车符格开的
    '因此可以通过 Split 函数隔开各个数据项
    vFiles = Split(strListing, vbCrLf)
    '获得每一个具体项
    For Each vFile In vFiles
        Set oFtpFile = New CftpFile
        '得到的每个数据项都是一个字符串，它们之间可能有许多空格，因此将多个连续的空格
        '替换成为单个空格
        For i = 15 To 2 Step -1
            vFile = Replace(vFile, Space(i), " ")
        Next
        '以下为获得每个文件的具体信息，如文件名、大小等
        If Len(vFile) > 0 Then
            '如果开始五个字符不是“total”
            If Not LCase(Left(vFile, 5)) = "total" Then
                '通过空格分离出各个具体属性
                vComponents = Split(vFile, " ")
                '如果属性项大于 7，通常会返回 8 个以上关于文件的属性
                If UBound(vComponents) > 7 Then
                    With oFtpFile
                        '如果第一个属性项的第一个字符是“d”，则表示是目录
                        If Left(vComponents(0), 1) = "d" Then
                            oFtpFile.IsDirectory = True
                        '或者字符串 vFile 的第一个字符为“l”则表示是路径
                        ElseIf Left(vFile, 1) = "l" Then
                            .FilePath = vComponents(10)
                        '如果路径的最后没有“.”，表示不是文件，因为文件一般总是含有
扩
```

```
    '展名的
                            If Not CBool(InStr(InStrRev(vComponents(10), "/") +
1, vComponents(10), ".")) Then
                                .IsDirectory = True
                            End If
                        End If
                        '然后根据一定的次序，设置相应的属性
                        .Permissions = vComponents(0)
                        .Owner = vComponents(2)
                        .Group = vComponents(3)
                        .FileSize = vComponents(4)
                        .FileName = vComponents(8)
                        .LastWriteTime=GetDate(vComponents(6),vComponents(5),
vComponents(7))
                        If Not (.FileName = "." Or .FileName = "..") Then
                            oFtpFiles.Add oFtpFile, oFtpFile.FileName
                        End If
                    End With
                '如果返回的属性项少于 8
                Else
                    With oFtpFile
                        '分析是否为目录或者是文件大小
                        If vComponents(2) = "<DIR>" Then
                            .IsDirectory = True
                        Else
                            .FileSize = CLng(vComponents(2))
                        End If
                        '获得文件名
                        If UBound(vComponents) > 3 Then
                            Dim strFile As String
                            For i = 3 To UBound(vComponents)
                               strFile = strFile & " " & vComponents(i)
                            Next i
                            strFile = Mid$(strFile, 2)
                        Else
                            strFile = vComponents(3)
                        End If
                        .FileName = strFile
                        .LastWriteTime  =  CDate(vComponents(0)  &  "  "  &
```

```
vComponents(1))
                        oFtpFiles.Add oFtpFile, oFtpFile.FileName
                    End With
                End If
                Set oFtpFile = Nothing
            End If
        End If
        strFile = ""
    Next

    Set GetFileList = oFtpFiles
    Set oFtpFiles = Nothing

End Function
```

上面的函数主要是分析从服务器返回的数据。可能不同的服务器返回的格式不一样，读者可以显示出返回的数据，然后一项一项地进行隔离。上面的程序是按照标准写的，读者可以参照返回的数据来具体分析。

在分析了以上的一些函数后，接下来介绍一个非常重要的函数就是文件上载函数。在通过 Socket 进行文件上载不像通过 Inet 控件上载那样简单，只需要发一个命令然后跟着一个文件路径就可以了。这里要文件上载必须读取文件的具体内容，然后按照一定的大小分成多个数据包发送。

```
    '该函数的功能是向服务器上传文件
    Public Function UploadFile(strLocalFileName As String, strRemoteFileName As
String, vTransferMode As FtpTransferModes, Optional lStartPoint As Long) As Boolean
        Dim bDataConnectionEstablished As Boolean
        '标记程序正忙
        m_bBusy = True
        '设定传输模式，如果传输模式改变，则通过函数 ProcessTYPECommand 改变传输模式
        If Not (vTransferMode = m_TransferMode) Then
            If ProcessTYPECommand(vTransferMode) Then
                m_TransferMode = vTransferMode
            Else
                '如果改变传输模式失败，则退出函数
                Exit Function
            End If
        End If
        '因为文件上载是通过数据端口进行的，所以要先建立连接模式
        If m_bPassiveMode Then
            bDataConnectionEstablished = ProcessPASVCommand
```

```
        Else
            bDataConnectionEstablished = ProcessPORTCommand
        End If
        '如果连接模式建立成功
        If bDataConnectionEstablished Then
            '通过 ProcessRESTCommand 函数设置初始点，也就是断点上传
            '目前通常的 FTP 客户端软件是不支持断点上传的
            If Not IsMissing(lStartPoint) Then
                '如果设置上传初始点错误，则退出函数
                If Not ProcessRESTCommand(lStartPoint) Then
                    UploadFile = False
                    Exit Function
                End If
            End If
            '获得要上传的文件
            m_strLocalFilePath = strLocalFileName
            '设置标志表示要上传文件
            m_bUploadFile = True
            '通过 ProcessSTORCommand 函数向服务器请求上传文件
            If      ProcessSTORCommand(strLocalFileName,      strRemoteFileName,
lStartPoint) Then
                m_objTimeOut.StartTimer
                Do
                    DoEvents
                    '超时处理
                    If m_objTimeOut.Timeout Then
                        m_LastError = ERROR_FTP_USER_TIMEOUT
                        Exit Do
                    End If
                    '如果数据连接的 Winsock 对象已经关闭，则退出循环
                    If wscData.State = sckClosing Or _
                        wscData.State = sckClosed Then
                        RaiseEvent StateChanged(FTP_TRANSFER_COMLETED)
                        Exit Do
                    End If
                Loop
                m_objTimeOut.StopTimer
                UploadFile = True
            End If
```

```
    End If
    m_bBusy = False
End Function
```

在函数 UploadFile 的学习中，最重要的是要掌握上载文件需要进行的步骤。如果在上载文件的过程中缺少了或者做错了某个步骤，就有可能导致文件上载出错。首先要通过函数 ProcessTYPECommand 向服务器发送改变传输模式的命令，然后是要建立数据连接，要根据用户的选择来建立被动模式或者 PORT 的模式。在建立了数据连接后，还需要设置断点上传的初始点。在经过了这一系列的过程之后，就可以通过函数 ProcessSTORCommand 来向服务器发送请求上载的命令。如果上载成功，则等待数据连接关闭。不过在这个程序中，没有具体涉及到发送文件的内容，而这个操作是在函数 ProcessSTORCommand 中进行的，其中使用了另外一个函数 UploadData。下面是对该函数的讲解。

```
'该函数的功能是是打开一个文件，然后把数据上传到服务器，每次传送 4k 字节
Private Sub UploadData(lStartPoint As Long)
'说明 :如果一次传送多于 4k 的字节，则有可能多次产生 wscFtpData_SendComplete 事件
    '定义一个数据包的大小
    Const CHANK_SIZE As Integer = 4096

    Static bFileIsOpen  As Boolean  '标记变量，表示文件是否已经打开
    Static lChanksCount As Long     '总共要传送的次数
    Static lCounter     As Long     '已经传送的次数
    Static intRemainder As Integer    '剩下的字节大小
    Dim strData()        As Byte   '用来存放被发送的数据

    On Error GoTo UploadData_Err_Handler
    '如果 bFileIsOpen = True,则表示文件已经被打开了
    If m_bFileIsOpened Then
        '上传下一个数据包
        If lCounter < lChanksCount And lCounter > 0 Then
            '首先开辟一个 4k 的缓存
            ReDim strData(CHANK_SIZE)
            '计数器加 1
            lCounter = lCounter + 1
            '从文件中读去数据到 strdata 中
            Get m_intLocalFileID, , strData()
            '发送数据
            wscData.SendData strData()
        Else
            '所有的数据已经上载
```

```
            If lCounter = 0 Then
                '关闭数据连接
                '连接已经完成
                wscData.Close
                '关闭本地文件
                Close #m_intLocalFileID
                RaiseEvent StateChanged(FTP_TRANSFER_COMLETED)
                '初始化一些数据
                m_lUploadedBytes = 0:      lChanksCount = 0: intRemainder = 0
                m_bFileIsOpened = False:       m_bUploadFile = False
            Else
                '现在发送剩余的数据
                'now we have to send the remainder
                ReDim strData(intRemainder)
                '将计数器清 0，下面是最后一批数据了
                lCounter = 0
                '从文件中读取数据
                Get m_intLocalFileID, , strData()
                '发送数据
                m_objTimeOut.StartTimer
                Do
                    DoEvents
                    If m_objTimeOut.Timeout Then
                        m_LastError = ERROR_FTP_USER_TIMEOUT
                        Exit Do
                    End If
                    If wscData.State = sckConnected Then
                        wscData.SendData strData
                        Exit Do
                    End If
                Loop
                m_objTimeOut.StopTimer
            End If
        End If
    Else
        '以下是第一次发送数据，需要打开文件
        m_bFileIsOpened = True  '做一个文件打开的标志
        m_intLocalFileID = FreeFile
        Open m_strLocalFilePath For Binary As m_intLocalFileID
```

```
        '如果开始上传的起始点大于 0，则表示是发送剩余的数据，类似与断点续传
        If lStartPoint > 0 Then
            Seek m_intLocalFileID, lStartPoint + 1
            m_lUploadedBytes = lStartPoint
            '获得剩余文件大小，并计算需要分成几个数据包发送
            lChanksCount = CLng((FileLen(m_strLocalFilePath) - lStartPoint) \
CHANK_SIZE)
            '获得不完整数据包即最后一个数据包的大小
            intRemainder = (FileLen(m_strLocalFilePath) - lStartPoint) Mod
CHANK_SIZE
        Else
            '如果是上传整个文件，则计算完整的数据包的个数
            lChanksCount = CLng(FileLen(m_strLocalFilePath) \ CHANK_SIZE)
            '获得剩余字节
            intRemainder = FileLen(m_strLocalFilePath) Mod CHANK_SIZE
        End If

        If lChanksCount = 0 Then
            '如果整个文件不足一个完整的数据包，即 4k，则创建一个实际文件大小的缓存区
            ReDim strData(intRemainder)
        Else
            '否则创建一个 4k 大小的内存空间
            ReDim strData(CHANK_SIZE)
            '将发送数据包的计数器设置为 1
            lCounter = 1
        End If

        '从文件中读取数据
        Get m_intLocalFileID, , strData
        '发送数据
              Do
                 DoEvents
                 If wscData.State = sckConnected Then
                    wscData.SendData strData
                    Exit Do
                 End If
              Loop
    End If
    Exit Sub
```

```
    Exit_Label:
        Exit Sub
    '出错处理
    UploadData_Err_Handler:
        If Not ProcessWinsockError(Err.Number, Err.Description) Then
            Err.Raise     vbObjectError     +     1000     +     Err.Number,
"CFtpConnection.UploadData", Err.Description
        End If
        '关闭文件
        Close #intFile
        GoTo Exit_Label
    End Sub
```

这是一个直接和文件以及通信相关的函数。因为发送文件通常是通过二进制的形式发送，所以在定义动态数组的时候，最好定义为 Byte 类型，而不要定义为 String 类型。为了防止数据的丢失，必须动态地申请数组大小为要发送数据的大小，否则如果差一个字节，文件就可能打不开。注意在申请数组大小的时候，不要使用“字符串变量＝Space（n）”来获得缓冲区，因为如果定义发送数据缓冲区为字符串类型的时候，只能发送文本文件，当文件以二进制的形式发送时，会导致数据丢失。因此可以统一使用二进制方法，二进制方法在传送文本和其他文件的时候都是适用的。每个发送数据包的大小，最好为 4k，当超过 4k 的时候，Winsock 对象将会产生多次的 SendComplete 事件，增加处理难度。如果定义为 4k，通常当一个数据包发送完的时候会产生 SendComplete 事件。注意上面的函数只发送一个数据包，对于一个大于 4k 的文件，要多次发送，当一个数据包发送完毕后，即产生 SendComplete 事件，然后在该事件中继续调用该函数，直到文件发送完毕。SendComplete 事件的详细程序请参看前面的 wscFtpData_SendComplete 事件代码。

由于类 CftpConnection 是一个非常综合的程序，里面还有其他的许多函数，具体内容请参见本书所附光盘，这里就不一一介绍了。在读这个类之前，最好能首先学习 FTP 协议的内容以及其工作模式，否则可能会有不少的困难。

6. 窗体 frmMain

窗体 frmMain 主要是解决一些界面的问题，如菜单、工具条、树形控件等一些操作。如果读者对 VB 很熟悉，这些是很容易看懂的，这里也就不对此一一说明了。比较关键的就是在程序中如何使用类进行编程，比如如何创建一个类、调用类的属性和方法。同时对于含有事件的类，可以利用类的事件来获得相关信息。有 VB 功底的读者应该不难理解。

7. 窗体 frmConnect

该窗体的功能非常简单，就是对用户登录时的服务器地址、用户名和密码设置。如果是匿名登录，则不需要输入用户名和密码。注意，在本例题中，该窗体被当作一个对象在窗体 frmMain 中使用。在 frmMain 中通过创建该窗体对象，可以直接引用它的一些属性。这也是 VB 的一个用法，读者可以参考相关书籍。

8. 总结

这个程序利用了许多 VB 的编程技巧，面向的是高级的 VB 读者，初级的读者学习起来会有一定的困难。这里关键的是要掌握类编程、Winsock 对象的使用、FTP 协议、文件传输过程中的接收发送以及 FTP 的工作次序。这些内容书中都做了比较详细的解释。在后面，还要为大家介绍另外一种 FTP 客户端程序的编写，是通过 Windows API 来实现的，可能更高级一点。当然其中对于协议的使用并不多，因此对于掌握 FTP 编程，通过 Winsock 对象来直接和服务器通信是最好的。

8.7　API 开发高级 FTP 客户端程序

在这个实例中，与 FTP 服务器的连接与对话是通过 API 函数来实现的，建立连接必须遵循一定的步骤和规则，才能正确地与 FTP 服务器取得连接和对话，总的说来，建立连接的步骤一般分为以下几步：

（1）打开 Internet 会话；
（2）建立 FTP 类型的 Internet 连接；
（3）设置 FTP 服务器的当前目录；
（4）查看 FTP 服务器上指定目录下的文件；
（5）从 FTP 服务器当前目录下载文件或上载文件；
（6）在 FTP 服务器上进行目录或文件操作，如创建新目录、删除目录、删除文件等；
（7）关闭与 FTP 服务器的连接；
（8）关闭 Internet 对话。

对于实现连接与对话的每一个步骤，API 函数至关重要，在整个交互对话过程中，只要熟悉了相应的 API 函数，就不难理解进行 FTP 操作的用法。至于 API 函数怎么样通过 Internet 与 FTP 进行会话的底层细节，可不必去考虑，这些都是由系统提供的。

8.7.1　建立工程项目

该应用主要由 4 个部分组成，具体如下：

- 标准模块 modWinInet.bas
- 标准模块 modInter.bas
- 窗体 frmFTPst
- 窗体 frmFTP2

在标准模块 modInter 中，主要声明该程序用到的 API 函数及其常量。在标准模块 modInter 中主要声明要用到的全局变量。窗体 frmFTPst 的主要功能是建立 Internet 对话，如果对话成功建立，则显示与 FTP 服务器连接及交互对话的对话窗体。窗体 frmFTP2 的主要功能是建立与 FTP 服务器的连接并与其交互对话，实现从 FTP 服务器上下载选择的文件、上载文件、删除文件、改变当前目录、创建新目录等功能，这些都是用模块 modWinInet 所声明的 API 函数实现的。

工作界面如图 8-23 所示，即启动界面。

图 8-23　启动界面

在图 8-23 中单击“打开 Internet 会话”按钮，则会连接到服务器，连接成功后工作界面如图 8-24 所示。

图 8-24　工作主界面

对于其他的工作界面，这里就不再一一演示了，希望读者能够自己去实践。

8.7.2　关键代码分析

在代码分析的过程中，主要是针对程序中的代码做相应分析和解释说明。让读者能够更加容易看懂程序和理解程序。由于本程序是利用 API 来实现的，因此并不需要读者对 FTP 的协议了解的很清楚，而只需要对其工作过程和每个函数的功能掌握清楚就可以了。

1. 标准模块 modWinInet

该模块主要是对是申明一些程序中使用到的和 FTP 相关的 API 函数和申明一些常数。具体的解说和程序代码如下。

```
'modWinInet 标准模块
Option Explicit
Public Const MAX_PATH = 260
Public Const NO_ERROR = 0

'在 WIN32_FIND_DATA 结构中的文件查找属性常数
Public Const FILE_ATTRIBUTE_READONLY = &H1  '只读文件
Public Const FILE_ATTRIBUTE_HIDDEN = &H2   '隐含文件
Public Const FILE_ATTRIBUTE_SYSTEM = &H4   '系统文件
Public Const FILE_ATTRIBUTE_DIRECTORY = &H10
```

```
'目录文件（所获取的文件类型是目录）
'在 FtpFindFirstFile 和 FtpFindNextFile 函数把文件和目录当作文件一同获取
Public Const FILE_ATTRIBUTE_ARCHIVE = &H20   '存档文件
Public Const FILE_ATTRIBUTE_NORMAL = &H80   '文件没有的其他属性值
Public Const FILE_ATTRIBUTE_TEMPORARY = &H100  '临时文件
Public Const FILE_ATTRIBUTE_COMPRESSED = &H800  '被压缩的文件和目录
Public Const FILE_ATTRIBUTE_OFFLINE = &H1000

''直接从远程服务器中获得数据，而不使用本地缓冲的数据
Public Const INTERNET_FLAG_RELOAD = &H80000000

'FTP 服务器的文件打开标记：只读或可写
Public Const GENERIC_READ = &H80000000
Public Const GENERIC_WRITE = &H40000000

'标示调用 Internet 会话的应用程序的名称
Public Const scUserAgent = "FTP CLIENT"

'按照预定义的方式打开连接 Internet 对话
Public Const INTERNET_OPEN_TYPE_PRECONFIG = 0

'直接连接 Internet
Public Const INTERNET_OPEN_TYPE_DIRECT = 1

'通过代理服务器连接
Public Const INTERNET_OPEN_TYPE_PROXY = 3

'使用与连接协议相应的端口号
Public Const INTERNET_INVALID_PORT_NUMBER = 0

'与 FTP 服务器之间的文件传输采取 ASCII（文本）方式
Public Const FTP_TRANSFER_TYPE_ASCII = &H1

'与 FTP 服务器之间的文件传输采取 Binary（二进制）方式
Public Const FTP_TRANSFER_TYPE_BINARY = &H1

'定义连接模式
```

```
Public Const INTERNET_FLAG_PASSIVE = &H8000000

'从服务器返回的一个附加错误
Public Const ERROR_INTERNET_EXTENDED_ERROR = 12003

'在与 Internet 上的一些常用的服务器程序连接时，常用的端口号是.
'FTP 协议服务器的端口号为 21
Public Const INTERNET_DEFAULT_FTP_PORT = 21

'GOPHER 协议服务器的端口号为 70
Public Const INTERNET_DEFAULT_GOPHER_PORT = 70

'HTTP 协议服务器的端口号为 80
Public Const INTERNET_DEFAULT_HTTP_PORT = 80

'连接 Internet 服务的常数
Public Const INTERNET_SERVICE_FTP = 1
Public Const INTERNET_SERVICE_GOPHER = 2
Public Const INTERNET_SERVICE_HTTP = 3

'用于装载文件时间的结构体
Type FILETIME
       dwLowDateTime As Long
       dwHighDateTime As Long
End Type

'这个结构用于装载与找到的文件有关的具体信息
Type WIN32_FIND_DATA
       dwFileAttributes As Long '文件的属性
       ftCreationTime As FILETIME '文件的创建时间
       ftLastAccessTime As FILETIME '文件的最后一次被读写的时间
       ftLastWriteTime As FILETIME '文件的最后一次修改的时间
       nFileSizeHigh As Long
       nFileSizeLow As Long
       dwReserved0 As Long
       dwReserved1 As Long
       cFileName As String * MAX_PATH  '用于存放文件名的字符串
       cAlternate As String * 14
End Type
```

```
    Public Const ERROR_NO_MORE_FILES = 18
    '*************************************************************************
*******
    '连接 FTP 服务器的操作函数
    '*************************************************************************
*******
    '打开连接 Internet 的会话
    Public Declare Function InternetOpen Lib "wininet.dll" Alias "InternetOpenA"
_
    (ByVal sAgent As String, ByVal lAccessType As Long, ByVal sProxyName As String,
_
    ByVal sProxyBypass As String, ByVal lFlags As Long) As Long
    'sAgent--要调用 Internet 对话的应用程序名
    'lAccessType--请求的访问的类型，包括：
    'INTERNET_OPEN_TYPE_PRECONFIG-预配置（缺省）
    'INTERNET_OPEN_TYPE_DIRECT-直接指向 Internet
    'INTERNET_OPEN_TYPE_PROXY-通过代理服务器连接
    'sProxyName--如果 lAccessType 被设置为 INTERNET_OPEN_TYPE_PROXY，该参数
    '为代理服务器的名字
    'sProxyBypass--包含一系列代理服务器地址的字符串
    'lFlags--会话的选项，可包括下列值：
    'INTERNET_FLAG_DONT_CACHE--不对数据进行本地缓冲或通过网关服务器缓冲
    'INTERNET_FLAG_ASYNC--
    '当操作完成时，将同 INTERNET_STATUS_REQUEST_COMPLETE 一起进行一个状态
    '回调
    'INTERNET_FLAG_OFFLINE--只通过永久缓冲进行下载操作

    '打开一个根据连接类型的 Internet 连接
    Public Declare Function InternetConnect Lib "wininet.dll" Alias
"InternetConnectA" _
    (ByVal hInternetSession As Long, ByVal sServerName As String, ByVal nServerPort
As Integer, _
    ByVal sUsername As String, ByVal sPassword As String, ByVal lService As Long,
_
    ByVal lFlags As Long, ByVal lContext As Long) As Long
    'hInternetSession--函数 InternetOpen（）打开 Internet 对话返回的值
    'sServerName--要连接的服务器的名称或 IP
    'nServerPort--该连接的 Internet 端口
```

```
    'sUsername--登录的用户帐号
    'sPassword--登录的口令
    'lService--要连接的服务器类型（这里是连接 FTP 服务器，连接的类型为常数
'INTERNET_SERVICE_FTP）

    '关闭 Internet 连接
    Public Declare Function InternetCloseHandle Lib "wininet.dll" _
    (ByVal hInet As Long) As Integer
    'hInet--InternetConnect () 函数返回的值

    '*************************************************************************
*******
    '文件查找操作函数
    '*************************************************************************
*******

    '在 FTP 连接中查找一个文件。在调用此函数后取得第一个文件后，可以接着调用 FtpFindNextFile
()
    '获得下一个查到的 FTP 服务器上的文件。
    Public Declare Function FtpFindFirstFile Lib "wininet.dll" Alias
"FtpFindFirstFileA" _
    (ByVal hFtpSession As Long, ByVal lpszSearchFile As String, _
    lpFindFileData As WIN32_FIND_DATA, ByVal dwFlags As Long, ByVal dwContent As
Long) As Long
    'lpszSearchFile--指向要搜索的文件名。可以通配符*.*的形式。
    'lpFindFileData--用于装载与找到的文件有关的具体信息
    'dwFlags--数据传输的方式，通常设置为 0（ASCII 方式）

    '继续由 FtpFindFirstFile () 函数发起的文件搜索操作
    Public Declare Function InternetFindNextFile Lib "wininet.dll" Alias _
     "InternetFindNextFileA"    (ByVal hFind As Long,  _
    lpvFindData As WIN32_FIND_DATA) As Long
    'hFind--FtpFindFirstFile () 函数的返回值
    'lpvFindData--用于装载与找到的文件有关的具体信息

    '目录操作函数
    '改变 FTP 服务器的当前目录。
    Public Declare Function FtpSetCurrentDirectory Lib "wininet.dll" Alias _
     "FtpSetCurrentDirectoryA"    (ByVal hFtpSession As Long,  _
```

```
    ByVal lpszDirectory As String) As Boolean
    'lpszDirectory--包含要到达的目录名。可以是相对的或绝对的路径。

    '在 FTP 服务器上创建目录
    Public Declare Function FtpCreateDirectory Lib "wininet.dll" Alias
"FtpCreateDirectoryA"
    (ByVal hFtpSession As Long, ByVal lpszDirectory As String) As Boolean
    'lpszDirectory--包含要创建目录的字符串，可以是一个相对路径或绝对路径

    '取得 ftp 当前的目录的名字
    Public Declare Function FtpGetCurrentDirectory Lib "wininet.dll" Alias
"FtpGetCurrentDirectoryA" _
    (ByVal hFtpSession As Long, lpszCurrentDirectory As String,
lpdwCurrentDirectory As Long) As Boolean
    'lpszCurrentDirectory--存放目录名字的字符串
    'lpdwCurrentDirectory--目录名字字符串的字节数

    '删除 FTP 服务器的一个目录
    Public Declare Function FtpRemoveDirectory Lib "wininet.dll" Alias
"FtpRemoveDirectoryA" _
       (ByVal hFtpSession As Long, ByVal lpszDirectory As String) As Boolean
    'lpszDirectory--要删除的目录名称，可以是相对路径或绝对路径

    '文件操作函数

    '从 FTP 服务器上取得一个文件并保存在本地机器上，此函数包括了与从一个 FTP 服务器中读取一个
文件
    '并在本地保存等操作有关的所有功能

    Public Declare Function FtpGetFile Lib "wininet.dll" Alias "FtpGetFileA" _
    (ByVal hFtpSession As Long, ByVal lpszRemoteFile As String, _
    ByVal lpszNewFile As String, ByVal fFailIfExists As Boolean, ByVal
dwFlagsAndAttributes As Long, ByVal dwFlags As Long, ByVal dwContext As Long) As
Boolean
    'lpszRemoteFile--包含 FTP 服务器中要读的文件名
    'lpszNewFile--在本地机器中要创建的文件名
    'fFailIfExists--当为 TRUE 时，如果文件已经存在，则调用失败
    'dwFlagsAndAttributes--文件的属性
    'dwFlags--文件的传输方式可能包括下列值：
```

```
    'FTP_TRANSFER_TYPE_ASCII--使用可以将控制和格式信息转换到本地对应文件的 ASCII 传输方式
    'FTP_TRANSFER_TYPE_BINARY--使用把文件作为一个连续数据流传输的 FTP 图像传输方式，这种方式不区别在文件内部数据结构之间的任何分界（如行结束的回车标记）
    'dwContext--要取回的文件的描述表标识符

    '把一个文件上载到 FTP 服务器上。其中包括了要将一个文件上载到 FTP 服务器上所有的有关操作
    Public Declare Function FtpPutFile Lib "wininet.dll" Alias "FtpPutFileA" _
    (ByVal hFtpSession As Long, ByVal lpszLocalFile As String, _
         ByVal lpszRemoteFile As String, _
         ByVal dwFlags As Long, ByVal dwContext As Long) As Boolean
    'lpszLocalFile--要发送的文件名
    'lpszRemoteFile--在 FTP 服务器上要创建的文件名
    'dwFlags--文件的传输方式
    'dwContext--要取回的文件的描述表标识符

    '在 FTP 服务器上打开一个文件以进行读或写操作
    Public Declare Function FtpOpenFile Lib "wininet.dll" Alias _
           "FtpOpenFileA" (ByVal hFtpSession As Long, _
           ByVal sFileName As String, ByVal lAccess As Long, _
           ByVal lFlags As Long, ByVal lContext As Long) As Long
    'sFileName--要打开的文件名
    'lAccess--文件访问标记，可以是 GENERIC_READ 或 GENERIC_WRITE
    'lFlags--文件传输方式（文本或二进制）
    'lContext--要打开的文件的描述表标识符

    '在 FTP 服务器上删除一个指定的文件
    Public Declare Function FtpDeleteFile Lib "wininet.dll" _
       Alias "FtpDeleteFileA" (ByVal hFtpSession As Long, _
       ByVal lpszFileName As String) As Boolean
    'lpszFileName--要删除的文件名。可以是相对路径或绝对路径。
    '取得最后一次与 Internet 交互的响应信息
    Public Declare Function InternetGetLastResponseInfo Lib "wininet.dll" Alias
"InternetGetLastResponseInfoA" ( _
       lpdwError As Long, _
       ByVal lpszBuffer As String, _
    lpdwBufferLength As Long) As Boolean
```

由于上面的代码解说很清楚，因此不具体对某个函数展开叙述。读者关键要掌握每个函数的用途。

2. **标准模块** modInter

该模块中主要声明要用到的全局变量。

```
Public bActiveSession As Boolean
'用以标记已经建立了 FTP 连接
Public hOpen As Long
'用 InternetOpen()函数建立 Internet 对话后返回的句柄
Public hConnection As Long
'用 InternetConnect()函数建立 FTP 连接后返回的句柄

Public dwType As Long
Public EnumItemNameBag As New Collection
'用以存放查找到的 FTP 目录下的文件名
Public EnumItemAttributeBag As New Collection
'用以存放与 EnumItemNameBag 中相对应的文件名的属性
```

3. **窗体** frmFtpst

在此窗体中，建立 Internet **对话。如果对话成功建立，则显示与** FTP **服务器连接及交互对话的对话窗体。下面就对一些重要的代码程序进行讲解。**

建立 Internet **对话**

```
Private Sub cmdInternetOpen_Click()
'在建立一定类型的 Internet 连接前，必须首先建立 Internet 对话，如果成功建立
'Internet 对话，则返回一个句柄作为建立相应类型的连接（在这里是 FTP 连接）
'InternetConnect（）的参数
   If Len(txtProxy.Text) <> 0 Then
      hOpen = InternetOpen(scUserAgent, INTERNET_OPEN_TYPE_PROXY,  _
                     txtProxy.Text, vbNullString, 0)
   Else
      hOpen = InternetOpen(scUserAgent, INTERNET_OPEN_TYPE_DIRECT,  _
                     vbNullString, vbNullString, 0)
   End If
   '打开 Internet 连接的线程后，根据打开的结果设置各个按钮的状态
   If hOpen = 0 Then
   MsgBox "不能打开 Internet 连接", vbCritical
   cmdInternetOpen.Enabled = False
   cmdClosehOpen.Enabled = True
   Else
   frmFTPst.cmdClosehOpen.Enabled = True
   frmFTPst.cmdInternetOpen.Enabled = False
   '如果打开成功，以模式对话框显示主窗体 frmFTP
   frmFTP.Show vbModal
```

```
    End If
End Sub
```

关闭 INTERNET **对话**

```
Private Sub cmdClosehOpen_Click()
'关闭连接，将所有的 Internet 连接的句柄关闭
'清除所有的对象集合和窗体中的文本框的内容
    If hConnection <> 0 Then
    InternetCloseHandle (hConnection)
    End If
    If hOpen <> 0 Then
    InternetCloseHandle (hOpen)
    End If
    hConnection = 0
    hOpen = 0
    bActiveSession = False
    cmdInternetOpen.Enabled = True
End Sub
```

由于上面所调用的 API **函数在前面的章节内都已经说明过，这里主要是让读者明白在什么情况下应该使用哪个** API **函数。**

4. **主窗体** frmFTP

此窗体的主要功能是建立与 FTP **服务器的连接，并与其交互对话，实现从** FTP **服务器上下载选择的文件、上载文件、删除文件、改变当前目录、创建新目录等功能。这些都是用模块** modWinInet **所声明的** API **函数实现的。下面就对一些具体实现某个功能的代码作详细说明。**

（1**）连接** FTP **服务器**

```
'按钮 cmdConnect 事件
Private Sub cmdConnect_Click()
'连接 FTP 服务器，只有当前不处于连接状态时连接
    If Not bActiveSession And hOpen <> 0 Then
        If txtServer.Text = "" Then
            MsgBox "请键入要连接的 FTP 服务器的域名或 IP 地址!"
            Exit Sub
        End If
        Dim nFlag As Long
        If chkPassive.Value Then
        '设置 FTP 会话是被动模式的
```

```
    nFlag = INTERNET_FLAG_PASSIVE
Else
    nFlag = 0
End If
Dim strUser As String
Dim strPassword As String
If chkAnon.Value = 1 Then
 '匿名登录连接时，登录用户名及口令均为 NULL
    strUser = ""
    strPassword = ""
Else
'用用户名及口令连接
    strUser = txtUser.Text
    strPassword = txtPassword.Text
End If
Dim Port As Integer
If Val(txtPort.Text) < 0 Then
    Port = INTERNET_INVALID_PORT_NUMBER
Else
    Port = Val(txtPort.Text)
End If
'根据打开 Internet 连接的函数 Internetopen（）返回的句柄、用户名、口令
'打开与 FTP 服务器的连接
hConnection = InternetConnect(hOpen, txtServer.Text, Port, _
strUser, strPassword, INTERNET_SERVICE_FTP, nFlag, 0)
If hConnection = 0 Then
    '与 FTP 服务器的连接失败，显示连接错误信息
    bActiveSession = False
    ErrorOut Err.LastDllError, "InternetConnect"
Else
'与 FTP 服务器的连接成功，设置各个控件的 Enabled 属性
    bActiveSession = True
    chkPassive.Enabled = True
    chkAnon.Enabled = False
    cmdConnect.Enabled = False
    cmdDisconnect.Enabled = True
    txtServer.Enabled = False
    txtPort.Enabled = False
    txtUser.Enabled = False
```

```
            txtPassword.Enabled = False
            EnableUI (True)
            FillTreeViewControl (txtServer.Text)
            'FtpEnumDirectory()--遍历服务器当前目录下的文件及文件夹，并将其
            '添加到 Treeview 中
            '在此子程序中使用了 FtpFindFirstFile
            'FtpFindNextFile 函数，可将当前目录下的文件全部找出来
            FtpEnumDirectory ("")
            If EnumItemNameBag.Count = 0 Then Exit Sub
            '集合 EnumItemNameBag 为空，遍历目录下的文件数为 0
            FillTreeViewControl (txtServer.Text)
        End If
    End If
End Sub
```

（2）删除目录

```
Private Sub cmdDelDir_Click()
Dim temp
temp = MsgBox("在本地计算机删除目录请按 “是”" & vbCrLf _
      & "在 FTP 服务器上删除目录请按“否”", vbYesNoCancel)
If temp = vbCancel Then Exit Sub
If temp = vbYes Then
    RmDir (Dir1.Path)
    Dir1.Refresh
    Exit Sub
End If
Dim bRet As Boolean
Dim szFileRemote As String, szDirRemote As String, szFileLocal As String
Dim szTempString As String
Dim nPos As Long, nTemp As Long
Dim nodX As Node
Dim strDelDir As String
strDelDir = TreeView1.SelectedItem.Text
Set nodX = TreeView1.SelectedItem.Parent
'以下程序的作用是在 FTP 服务器删除目录文件夹
If bActiveSession Then
    If nodX Is Nothing Then
        MsgBox "请选择 FTP 服务器上一个要删除的目录!"
        Exit Sub
```

```
    End If
    If nodX.Image = "leaf" Then
        MsgBox "请选择 FTP 服务器上一个要删除的目录!"
        Exit Sub
    End If
    '取得要删除的目录名
    strDelDir = Right(strDelDir, Len(strDelDir) - Len(txtServer.Text))
    Dim bMake As Boolean
    '根据目录名删除目录
    bMake = FtpRemoveDirectory(hConnection, strDelDir)
    If bMake Then
        MsgBox "文件夹删除成功！"
        '将 FTP 服务器当前目录下的节点去除
        Dim nodChild As Node, nodNextChild As Node
        Set nodChild = nodX.Child
        '删除 TreeView 中所有的节点
        Do
            If nodChild Is Nothing Then Exit Do
            Set nodNextChild = nodChild.Next
            TreeView1.Nodes.Remove nodChild.Index
            If nodNextChild Is Nothing Then Exit Do
            Set nodChild = nodNextChild
        Loop
        '重新取得当前 FTP 目录下的文件及目录，加入到 TreeView 中
        FtpEnumDirectory (nodX.Text)
        FillTreeViewControl (nodX.Text)
    End If
End If
End Sub
```

（3）删除文件

```
Private Sub cmdDelFile_Click()
'此子程序的作用是在 FTP 服务器删除文件
Dim bRet As Boolean
Dim szFileRemote As String, szDirRemote As String, szFileLocal As String
Dim szTempString As String
Dim nPos As Long, nTemp As Long
Dim nodX As Node
Dim strDelFile As String
```

```
strDelFile = TreeView1.SelectedItem.Text
Set nodX = TreeView1.SelectedItem.Parent
If bActiveSession Then
    If TreeView1.SelectedItem.Image <> "leaf" Then
        MsgBox "请选择 FTP 服务器中要删除的文件!"
        Exit Sub
    End If
    strDelFile = Right(strDelFile, Len(strDelFile) - Len(txtServer.Text))
    Dim bMake As Boolean
    bMake = FtpDeleteFile(hConnection, strDelFile)
    If bMake Then
        MsgBox "文件删除成功！"
        '将 Treeview 中 FTP 服务器当前目录下的节点去除
        Dim nodChild As Node, nodNextChild As Node
        Set nodChild = nodX.Child
        Do
            If nodChild Is Nothing Then Exit Do
            Set nodNextChild = nodChild.Next
            TreeView1.Nodes.Remove nodChild.Index
            If nodNextChild Is Nothing Then Exit Do
            Set nodChild = nodNextChild
        Loop
        '重新取得当前 FTP 服务器目录下的文件，加入到 TreeView 相应的节点中
        FtpEnumDirectory (nodX.Text)
        FillTreeViewControl (nodX.Text)
    End If
End If
End Sub
```

（4）获得服务器端文件和目录

```
Private Sub FtpEnumDirectory(strDirectory As String)
'取得 FTP 服务器当前目录下的文件夹和文件
'清除集合的各元素
ClearBag
Dim hFind As Long
Dim nLastError As Long
Dim dError As Long
Dim ptr As Long
Dim pData As WIN32_FIND_DATA
```

```
    '设置 FTP 当前目录为 strDirectory,并取得此目录下的第一个文件
    If Len(strDirectory) > 0 Then rcd (strDirectory)
        pData.cFileName = String(MAX_PATH, 0)
        hFind = FtpFindFirstFile(hConnection, "*.*", pData, 0, 0)
        nLastError = Err.LastDllError
        If hFind = 0 Then
            If (nLastError = ERROR_NO_MORE_FILES) Then
                MsgBox "This directory is empty!"
            Else
                ErrorOut nLastError, "FtpFindFirstFile"
            End If
            Exit Sub
        End If
        dError = NO_ERROR
        Dim bRet As Boolean
        Dim strItemName As String
        '将取得的第一个文件名及其属性加入相应集合中
        EnumItemAttributeBag.Add pData.dwFileAttributes
        strItemName = Left(pData.cFileName, InStr(1, pData.cFileName, String(1,
0), _
                    vbBinaryCompare) - 1)
        EnumItemNameBag.Add strItemName
        'do....loop 是查找此目录下的所有文件，并把他们的属性及名字加入相应的集合
        '中
        Do
            pData.cFileName = String(MAX_PATH, 0)
            bRet = InternetFindNextFile(hFind, pData)
            If Not bRet Then
                dError = Err.LastDllError
                If dError = ERROR_NO_MORE_FILES Then
                    Exit Do
                Else
                    ErrorOut dError, "InternetFindNextFile"
                    InternetCloseHandle (hFind)
                   Exit Sub
                End If
            Else
                EnumItemAttributeBag.Add pData.dwFileAttributes
                strItemName = Left(pData.cFileName, InStr(1, pData.cFileName,  _
```

```
                        String(1, 0), vbBinaryCompare) - 1)
            EnumItemNameBag.Add strItemName
        End If
    Loop
    InternetCloseHandle (hFind)
End Sub
```

（5）下载文件

```
Private Sub cmdGet_Click()
'下载选中的在 FTP 服务器上的文件
Dim bRet As Boolean
Dim szFileRemote As String, szDirRemote As String, szFileLocal As String
Dim szTempString As String
Dim nPos As Long, nTemp As Long
Dim nodX As Node
Set nodX = TreeView1.SelectedItem
If bActiveSession Then
    If nodX Is Nothing Then
        MsgBox "请选择要下载的文件!"
        Exit Sub
    End If
    szTempString = TreeView1.SelectedItem.Text
    szFileRemote = szTempString
    nPos = 0
    nTemp = 0
    Do
        nTemp = InStr(1, szTempString, "/", vbBinaryCompare)
        If nTemp = 0 Then Exit Do
        szTempString = Right(szTempString, Len(szTempString) - nTemp)
        nPos = nTemp + nPos
    Loop
    'do....loop 的语句的作用是把文件的地址中的文件夹的路径的位置
    szDirRemote = Left(szFileRemote, nPos)
    '抽取文件夹的路径
    szFileRemote = Right(szFileRemote, Len(szFileRemote) - nPos)
    '抽取文件名
    szFileLocal = File1.Path
    rcd szDirRemote
    '将 FTP 服务器的当前目录设置为 szDirRemote
```

```
        bRet = FtpGetFile(hConnection, szFileRemote, szFileLocal & "/" &
szFileRemote, _
    False, INTERNET_FLAG_RELOAD, dwType, 0)
        '下载文件到本地机的当前目录
        File1.Refresh
        If bRet = False Then ErrorOut Err.LastDllError, "FtpGetFile"
    Else
        MsgBox "Not in session"
    End If
    End Sub
```

（6）上载文件

```
    Private Sub cmdPut_Click()
    '此子程序的作用是将本地机的文件上传到 FTP 服务器
    Dim bRet As Boolean
    Dim szFileRemote As String, szDirRemote As String, szFileLocal As String
    Dim szTempString As String
    Dim nPos As Long, nTemp As Long
    Dim nodX As Node
    Set nodX = TreeView1.SelectedItem
    If bActiveSession Then
        If nodX Is Nothing Then
            MsgBox "请选择上载文件的目录!"
            Exit Sub
        End If
        If nodX.Image = "leaf" Then
            MsgBox "请选择上载文件的目录!"
            Exit Sub
        End If
        If File1.FileName = "" Then
            MsgBox "请在本地计算机选择要上载的文件"
            Exit Sub
        End If
        szTempString = nodX.Text
        szDirRemote = Right(szTempString, Len(szTempString) - Len(txtServer.Text))
        szFileRemote = File1.FileName
        szFileLocal = File1.Path & "\" & File1.FileName
        If (szDirRemote = "") Then szDirRemote = "\"
        rcd szDirRemote
```

```
        '以上是先取得要上载文件的 FTP 服务器的目录，并设置为 ftp 当前目录
        '再用 FtpPutFile 上载
        bRet = FtpPutFile(hConnection, szFileLocal, szFileRemote, _
         dwType, 0)
        If bRet = False Then
           ErrorOut Err.LastDllError, "FtpPutFile"
           Exit Sub
        End If
        '将 FTP 服务器当前目录下的节点去除
        Dim nodChild As Node, nodNextChild As Node
        Set nodChild = nodX.Child
        Do
          If nodChild Is Nothing Then Exit Do
          Set nodNextChild = nodChild.Next
            TreeView1.Nodes.Remove nodChild.Index
            If nodNextChild Is Nothing Then Exit Do
            Set nodChild = nodNextChild
        Loop
        If nodX.Image = "closed" Then
           nodX.Image = "open"
        End If
        '查找 nodX.text 目录下的所有文件将其名字及属性加入集合中
        FtpEnumDirectory (nodX.Text)
        '重新根据集合中元素设置 Treeview 中目录 nodx.text 下的各节点
        FillTreeViewControl (nodX.Text)
    End If
    End Sub
```

5. **总结**

下面就 API **开发** FTP **客户端程序作一个总结，让读者了解工作过程以及所要用到的** API **函数。总的说来，建立连接的步骤一般分为以下几步。**

（1**）打开** Internet **对话**

在进行 Internet **连接之前，不管是采用** FTP**，还是使用其他协议如** HTTP**、**Gopher **等，首先使用** API **函数** InternetOpen()**打开** Internet **会话。具体做法如下：**

- **直接连接**

```
    hOpen = InternetOpen(scUserAgent, INTERNET_OPEN_TYPE_DIRECT, vbNullString,
vbNullString, 0)
```

（scUserAgent **可以是标示该应用程序的任何字符串）**

- **通过代理服务器连接**

```
    hOpen = InternetOpen(scUserAgent, INTERNET_OPEN_TYPE_PROXY, txtProxy.Text,
vbNullString, 0)
```

（其中 txtProxy.Text 为存放代理服务器地址的字符串）

如果连接成功，将返回非 0 的句柄 hOpen，此返回值可以作为建立 FTP 连接时的必需的参数之一。

（2）建立 FTP 类型的 Internet 连接

在成功地打开 Internet 对话后，接着就可以连接一定协议的 Internet 服务器了，使用到的 API 函数是 InternetConnect()。具体做法如下：

```
    hConnection = InternetConnect(hOpen, txtServer.Text, Port, _
    strUser, strPassword, INTERNET_SERVICE_FTP, nFlag, 0)
```

这里使用到打开 Internet 对话的 API 函数 InternetOpen()所返回的句柄 hOpen，根据服务器的地址（txtServer.Text）和端口号（Port），进行连接。在要求验证身份的私有的服务器上进行登录，必须使用正确的帐号（strUser）和口令（strPassword）。

此外，使用 InternetOpen()函数，还可进行其他协议的服务器连接，如 HTTP、Gopher 等服务器。只要将 INTERNET_SERVICE_FTP 改为以下常数即可。

Gopher 服务器： INTERNET_SERVICE_GOPHER （为常数 2）

HTTP 服务器： INTERNET_SERVICE_HTTP （为常数 3）

（3）设置 FTP 服务器的当前目录

在与 FTP 服务器进行对话之前，可设置用户在 FTP 服务器上的当前目录，使得在下一步对 FTP 进行目录或文件的操作时，不必指明目录或文件的绝对路径，只要指明目录或文件的名称，即相对于当前目录的相对路径即可。设定 FTP 服务器的当前目录用 API 函数 FtpSetCurrentDirectory()实现，具体用法如下：

```
    FtpSetCurrentDirectory(hConnection, sPathFromRoot)
```

其中 hConnection 为 FTP 连接函数 InternetConnect()返回的值。SPathFromRoot 为相对于服务器根目录的要设定的当前目录的字符串。如果成功地设定了当前目录，则上面的函数返回值为 TRUE

（4）查看 FTP 服务器上指定目录下的文件

要从 FTP 服务器中下载有用的文件，查找文件是必需的，查找文件的操作是一件比较麻烦的事，所幸的是 Windows 提供了两个这样的 API 函数：FtpFindFirstFile() 和 InternetFindNextFile()，利用这两个函数，就可以将 FTP 服务器指定目录下的目录和文件的名称查找出来。具体做法如下：

```
    hFind = FtpFindFirstFile(hConnection, "*.*", pData, 0, 0)
```

先用函数FtpFindFirstFile()把当前目录下的第一个文件查找出来。其中：hConnection为InternetConnect()返回标示与FTP服务器连接的值。"*.*"为查找文件的通配符，这里是将当前目录下所有的文件都查找出来，要查找特定类型的文件，可用相应的通配符。如设定为"a*.txt"可以将文件名的第一个字母为a的、扩展名为txt的文件找出来。

pData 为 WIN32_FIND_DATA 结构，用于装载查找出来的文件的信息及属性。定义为：

```
Dim pData As WIN32_FIND_DATA
```

如果成功地查找到了当前目录下的第一个文件，此函数的返回值 hFind 为查找文件的标示值（可称之为查找文件的句柄），用于查找下一个文件的函数 InternetFindNextFile（）的参数。查找下一个文件的函数的用法为：

```
bRet = InternetFindNextFile(hFind, pData)
```

其中：hFind 为 FtpFindFirstFile()的返回值，在这个值非零时才能用，表示已经成功地找到了第一个文件。pData 为查找到的文件的属性及其他信息。此函数的查找条件与 FtpFindFirstFile()的一致。如果函数成功地找到了下一个文件，则返回值为 True。要将 FTP 服务器当前目录下的所有指定类型文件和目录查找出来，只要设定一个 Do While … Loop 的循环调用 InternetFindNextFile()直到返回值为 FALSE 即可。

注意：用这两个函数查找的“文件”的结果包括文件和目录，查找的结果装载在 pData（WIN32_FIND_DATA 结构），可根据该结构中 dwFileAttributes 值来确定查找结果是哪种类型的“文件”。以下标示是文件类型的常数：

```
FILE_ATTRIBUTE_READONLY = &H1        只读文件
FILE_ATTRIBUTE_HIDDEN = &H2          隐含文件
FILE_ATTRIBUTE_SYSTEM = &H4          系统文件
FILE_ATTRIBUTE_DIRECTORY = &H10      目录文件（所获取的文件类型是目录）
FILE_ATTRIBUTE_ARCHIVE = &H20        存档文件
FILE_ATTRIBUTE_NORMAL = &H80         文件没有的其他属性值
FILE_ATTRIBUTE_TEMPORARY = &H100     临时文件
FILE_ATTRIBUTE_COMPRESSED = &H800    被压缩的文件和目录
```

（5）从 FTP 服务器当前目录下载文件或上载文件以及其他文件操作

FTP 服务器所提供的最重要的服务就是下载文件和上载文件了，在与服务器取得连接之后，用户的大部分操作都是与文件下载或上载有关，与文件下载和上载操作有关的 API 函数为 FtpGetFile()和 FtpPutFile()。这两个函数的具体用法如下。

■ 文件下载

```
bRet = FtpGetFile(hConnection, szFileRemote, szFileLocal, False, _
INTERNET_FLAG_RELOAD, dwType, 0)
```

其中：hConnection 为与 FTP 连接的函数 InternetConnect（）返回的值。szFileRemote 为要取得的在 FTP 服务器上的文件，可以是绝对路径或相对于当前目录的相对路径。szFileLocal 为下载文件后存在本地计算机的文件名。INTERNET_FLAG_RELOAD 表示从远程 FTP 服务器或本地 Cache 中取得数据。dwType 为文件传输的方式，其方式可以是文本方式（FTP_TRANSFER_TYPE_ASCII）或二进制方式（FTP_TRANSFER_TYPE_BINARY）。如果成功地调用该函数，函数的返回值 bRet 为 True，否则为 False。

■ 文件上载

```
bRet = FtpPutFile(hConnection, szFileLocal, szFileRemote, dwType, 0)
```

其中：该函数中的参数与 FtpGetFile（）中的参数的用法一致。如果成功调用了该函数，

函数的返回值 bRet 为 True，否则为 False。

（6）在 FTP 服务器上进行目录或文件操作

在 FTP 服务器上进行目录或文件操作，如创建新目录、删除目录、删除文件等。在一般的用匿名方式登录的 FTP 服务器，用户通常不具有创建新目录、删除目录、删除文件的权限，对于一些使用帐号和口令登录连接的 FTP 服务器，用户通常是具有这些权限的。Windows 的 API 函数为我们提供了一些这样的函数，如创建新目录的函数 FtpCreateDirectory()、删除目录的函数 FtpRemoveDirectory()删除文件的函数 FtpDeleteFile()等。这些函数的具体用法如下。

■　创建新目录

```
bMake = FtpCreateDirectory(hConnection, strRemoteDir)
```

其中：hConnection 为函数 InternetOpen()的返回值。strRemoteDir 为存放要创建的目录名的字符串，创建的目录名可以是绝对路径或相对于 FTP 服务器当前的目录的相对路径。如果在 FTP 服务器上成功创建目录，该函数的返回值为 True，否则为 False。

■　删除目录

```
bMake = FtpRemoveDirectory(hConnection, strRemoteDir)
```

其中：hConnection 的含义同上。strRemoteDir 为在 FTP 服务器上要删除的目录名，此目录名可以是绝对路径或相对于 FTP 服务器当前目录的相对路径。如果删除目录成功，则函数的返回值为 True，否则为 False。

■　删除文件

```
bMake = FtpDeleteFile(hConnection, strRemoteFile)
```

其中：hConnection 的含义同上。strRemoteFile 为在 FTP 服务器上要删除的文件名，此文件名可以是绝对路径或相对于 FTP 服务器当前目录的相对路径。如果删除文件成功，则函数的返回值为 True，否则为 False。

（7）关闭与 FTP 服务器的连接

在与 FTP 服务器对话完成之后，关闭与 FTP 服务器的连接，所用的 API 函数为 InternetCloseHandle()，具体用法如下：

```
InternetCloseHandle(hConnection)
```

其中：hConnection 的含义同上。如果成功的关闭与 FTP 服务器的连接，该函数的返回值为 True，否则为 False。

（8）关闭 Internet 对话

最后一步是关闭与 Internet 的对话，所用的 API 函数同关闭 FTP 服务器连接的函数一致，也为 InternetCloseHandle()，但是所用的参数不一样，用法如下：

```
InternetCloseHandle(hOpen)
```

其中：hOpen 为 InternetOpen()返回的用来标示 Internet 对话的值（句柄）。如果成功的关闭与 Internet 的对话，该函数的返回值为 True，否则为 False。

8.8 3 种 FTP 客户端程序开发方法的比较

以上所介绍的 3 种 FTP 客户端程序开发方法各有优缺点。具体在实际开发程序的时候，应该根据开发者的实际开发水平和具体的需求环境来选择。尽量选择简单的同时又能完成各项功能的方法。

通过 Internet Transfer 控件来开发 FTP 程序是最简单的。它只需要开发者掌握该控件的基本方法和属性，就能够进行 FTP 客户端的开发。它的功能有限，所以作为开发一个完整的 FTP 程序的工具是不能胜任的，但是如果作为某个大程序中集成一个 FTP 上载或者下载的功能，这是最方便的，读者只要简单地利用几个方法和属性就可以安全地完成文件的上传或者下载。

通过 Winsock 对象开发 FTP 客户端程序是比较高级的，除了对读者的 VB 水平要求比较高以外，还要对协议能了解得比较透彻。但是这个方法在 3 个方法里面是最灵活的，它涉及到 FTP 协议的底层工作模式，能够捕捉到各个状态，有效地进行各种错误处理。而且它还能自定义协议，只要服务器增加了新的协议，就能开发扩展 FTP 程序。因此该方法适合用来开发如同 Cute Ftp 这样完美的 FTP 客户端程序。

通过 API 开发程序相对更加高级，涉及到一些并不为大多数 VB 程序员所熟悉的知识。它也不是很灵活，缺乏自定义的功能，只能适用于支持标准协议的 FTP 服务器，它的使用比较繁琐，要使用各种 API 来完成。但是它的执行效率比第一种使用控件要高许多，因此对于开发一些要求不高的 FTP 客户端程序，应该是可以的。

上面对开发 FTP 常用的 3 种方法进行了简单的比较，读者可以根据自己的水平以及需要来选择适当的方法。但是读者的目的如果是学习网络开发，这里建议一定要掌握第二种方法。

第 9 章　RAS 高级编程

远程访问服务（RAS）是 Windows 9x/NT/2000 操作系统提供的系统服务之一，通过电话线可以使单独的计算机接入网络，或通过两个 RAS 对拨使两个局域网互连。 此项服务的功能可以使远程的计算机以较低的费用同网络连接，而且一旦建立了 RAS 连接，则可以使用其他的几乎所有的网络函数，对用户来说，实际上和通过网卡在局域网中进行数据传输是一样的。

本章主要介绍如何使用 Windows 的拨号程序以及如何开发自己的 RAS 拨号程序。在介绍这些知识的时候，都是结合程序介绍理论，这样读者学到的不仅仅是程序本身，而是对于 RAS 原理也有更深的了解。掌握了底层的应用，除了开发自己的程序以外，还可以开发出 DLL 或者 Activex 来供别人使用。通常如果使用第三方控件需要交好几十美元的购买，而且使用还会受限制。

9.1　RAS 客户机

微软提供的 Windows 操作系统中都提供 RAS 的服务，因此只要客户机中安装了 Windows 的系列产品，就可以很容易地进行 RAS 远程访问。它允许将客户端的计算机和一台远程计算机相连。

通过这种方式连入了 Internet 以后，客户并不需要知道自己是在什么环境中连入 Internet 的，可以认为是通过网卡连接而不是通过 Modem 拨号然后通过电话线连接出去的。一旦搭建了这个平台以后，就可以使用 Windows 的所有网络函数和网络软件。如可以使用 Winsock API 进行网络编程，也可以使用 FTP、IE 进行网上传输等等。

通常的情况下，RAS 客户机利用连接了电话线的一个调制解调器，通过拨号的方式呼叫远程的计算机（其特色就是一个远程访问服务器组件）。因此有时候，RAS 的客户机也称为“拨号联网(DUN)客户机”。

客户机要接入 Internet，在进行拨号以前，必须要有服务器能够提供服务接入功能，即等候 DUN 连接的服务。RAS 客户机能够和多种类型的远程访问服务器建立通信。RAS 其实是通过使用工业标准分帧协议建立通信。

例如，有如下的协议：

- 点到点的协议（PPP），可以传输 IP/IPX/NetBEUI 通信协议。
- 串行线路网际协议（SLIP），这种协议只能传输 IP 通信协议。
- 异步 NetBEUI，只能传输 NetBEUI 协议。

这些协议就是一些的数据传输的标准，它描述了怎么样通过 RAS 链接进行数据传输，指明 RAS 链接采用何种网络通信协议进行通信。至于采用何种协议建立链接，主要取决于服务器采用或者能够支持几种以上的协议。如果服务器支持上面的某种协议，则客户端 RAS 便可

建立一个基于这种协议的链接。在微软提供的一系列的操作系统中，RAS 服务器组件能够支持前面所说的任何一种分帧协议。

RAS 客户机和服务器之间的链接建立以后，网络协议堆栈（与所用的分帧协议有关）就通过这个 RAS 链接，与远程计算机通信，就像在局域网中一样。不过有一点要注意的是，通过 Modem 链接的速度比局域网的速度是要慢了许多，目前通常 Modem 使用的数据传输速率是 56kbps，前面在 Modem 编程的章节里面，对 Modem 的技术特征有详细说明。

RAS 服务器接收到一次拨号链接请求时，首先处理前面列出的一种分帧协议，然后便与客户机开始通信。分帧协议一旦确立，RAS 就会对客户端的接入进行身份验证。如果读者经常在家中拨号上网，那一定会熟悉拨号时出现的各种状态，一旦建立链接以后就会弹出对话框让用户输入用户名和密码，现在 163 和 169 都有通用的帐号和密码。如 163 的通用帐号和密码分别时 163、163；169 的通用帐号和密码是 169、169。RAS 客户机会为 RAS 服务器指定用户名、密码和域登录凭证。如果 Windows NT/2000 等 RAS 服务器接收到这条信息的时候，就会里同 Windows NT/2000 的域安全访问控制验证登录凭证。注意，RAS 服务器没有进入客户机的登录 Windows NT/2000 域；相反，它采用客户机凭证验证是否允许用户建立 RAS 链接。RAS 链接的过程和 Windows NT/2000 域的登录过程不太一样。RAS 链接成功建立以后，计算机就可以登录到 Windows NT/2000 域。Windows 98/95 平台上，RAS 链接通过电话簿条条目中可用的选项，经过验证后，RAS 便可自动进入一台机器，登录到一个域。

9.2 建立拨号连接

对于许多的 Windows 用户来说，接入 Internet 网络最方便地就是直接调用 Windows 的拨号面板来实现了，这样既方便又简单。本书的目的重在开发，所以只是简单介绍一下如何使用 Windows 自带的拨号程序来进行拨号。

下面的步骤是在 Windows 98 系统下进行的，在 Windows NT/2000 系统下面，基本步骤差不多。

（1）在桌面上打开“我的电脑”，选择“拨号网络”图标，如图 9-1 所示。

图 9-1 进入我的电脑

（2）双击“拨号网络”选项，进入欢迎使用拨号网络对话框，如图 9-2 所示。

图 9-2　欢迎使用拨号网络

（3）单击“下一步”按钮，进入安装新的调制解调器，如图 9-3 所示。

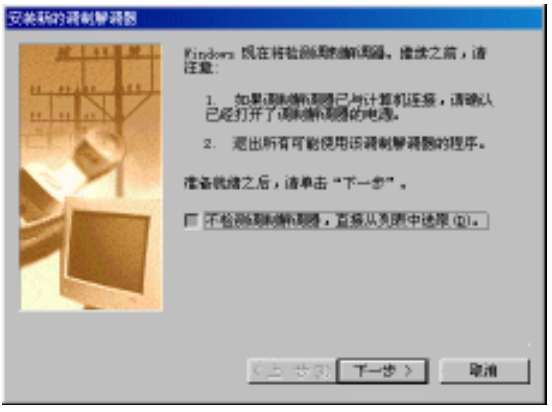


图 9-3　检测新的调制解调器

如果让系统自动检测 Modem 则不需要选中检查框，如果不需要自动检测，则不选。

（4）单击“下一步”按钮，进入安装调试解调器界面，用户可以根据自己的 Modem 型号直接安装或者从磁盘安装，如图 9-4 所示。

图 9-4　安装新的调制解调器

（5）单击“下一步”按钮，选择通信端口，如果系统自动检测，会自动设置端口，如图

9-5 所示。

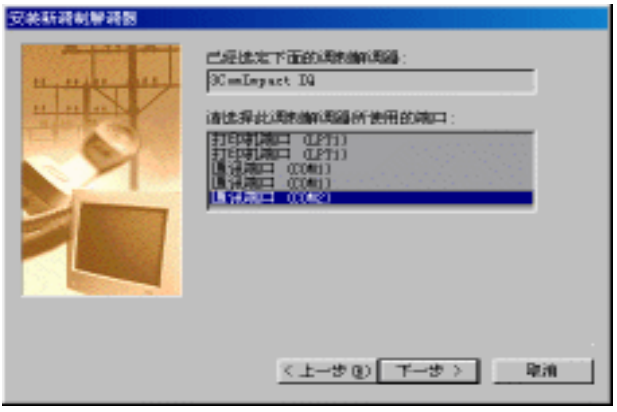


图 9-5　选择通信端口

（6）单击“ 下一步 ”按钮，完成安装，如图 9-6 所示。

图 9-6　完成调制解调器安装

如果已经安装调制解调器，则不会出现步骤（2）~（6）。

（7）单击完成按钮，弹出建立拨号连接的对话框如图 9-7 所示。

图 9-7　新建拨号连接

其中对方计算机名称可以随便取。

（8）单击“ 下一步 ”按钮，进入设置远程计算机电话号码的对话框，如图 9-8 所示。

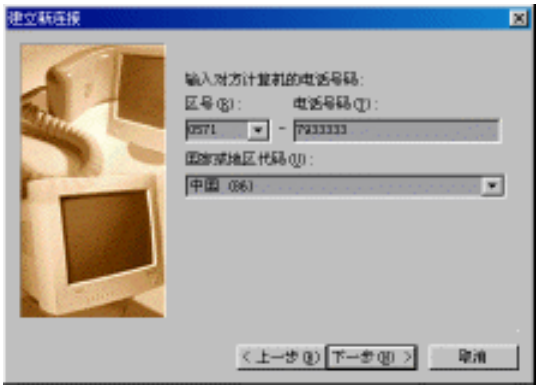


图 9-8　设置远程计算机电话号码和地区

（9）单击“下一步”按钮，成功完成新的连接建立，如图 9-9 所示。

图 9-9　成功创建拨号网络连接

注意，只有第一次才会按照这个次序安装，如果已经安装了调制解调器，则可以直接进入到图 9-10 中，单击新建连接来建立新的连接，可以建立多个拨号连接。比如多用户可以每人建立一个拨号连接。

（10）单击“完成”按钮，“我的连接”已经成功创建，如图 9-10 所示。

图 9-10　成功创建拨号连接

（11）双击“我的连接”图标，就可以进行拨号连接了，如图 9-11 所示。

图 9-11　进入最后拨号界面

如果需要额外的设置，可以单击拨号属性进行各项设置。在用户名和密码中输入相应的值，然后单击连接便可进入拨号状态。如果用户名和密码正确，以及电话线和服务器不忙，则可以成功实现拨号上网，当连通以后，就可以实现各种功能了，比如通过浏览器浏览页面、进行 FTP 传输、以及进行各种网路编程，就如同专线上网一样，但速度稍慢。

9.3　RAS 简单拨号程序

在介绍高级开发以前，首先向读者介绍一种 RAS 简单拨号程序。本程序很简单，只有几句代码，主要是为了让读者了解一种用法，而且对于有些读者，可能不需要开发很复杂的程序来实现 RAS。

本程序是通过调用系统已经建立好的电话簿来进行拨号，但有不方便之处就是会弹出连接界面，如果读者觉得没关系，这就是最好的选择了。

由于本程序非常简单，所以不详细列出各种资源。整个程序就一个窗体，具体代码如下：

```
Private Sub Apply_Click()
Dim rtn
'通过调用程序 rundll32.exe 来执行拨号控制的 rnaui.dll
rtn = Shell("rundll32.exe rnaui.dll,RnaDial " & Connection.Text, 0)
End Sub

Private Sub Cancel_Click()
'退出程序
Unload Me
End Sub

Private Sub Form_Load()
'使界面居中
```

```
    Move (Screen.Width - Width) \ 2, (Screen.Height - Height) \ 2
    End Sub
```

具体程序可以参见本书所附光盘。上面简单的几句语句就实现了拨号，具体的功能实现就是通过了 rnaui.dll 来执行一个已经建立起来的连接。工作界面如图 9-12 所示。

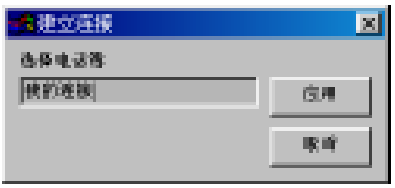


图 9-12　程序运行界面

在上面要注意的是，一定要在电话簿中输入一个建立起来的电话簿，不能输入错误。建立的步骤在上一章已经介绍了。如果我们输入上一节建立的“ 我的连接 ”，则会弹出如图 9-11 所示的界面。这样给用户带来的好处就是，如果要在程序中让用户拨号，不需要让用户去到拨号连接中进行连接，只需要单击一个按钮就可以弹出拨号对话框了。

✍ 该程序只能在 Windows 98 下通过（笔者在 Windows NT 环境下测试失败），而且在输入电话簿的时候不能输入错误，否则会出错。

9.4　RAS 重要函数说明

在本章节中，要实现拨号远程登录的功能，基本上使用的就是 RAS API，它是一个非常复杂的 API 函数，下面就分别对这些函数作一个简单的说明。在 9.2 节中，我们介绍了如何通过 Windows 工具建立拨号连接，那也是一个必要的手段，通常安装了拨号网络以后，机器才安装 RAS API，否则程序不能调用 RAS 来进行拨号。安装了拨号网络以后，才能在网络设置里面增加拨号协议。如何增加拨号协议，将在 9.6 节中介绍。

9.4.1　连接函数

（1）函数 RasDial

该函数的调用格式如下（函数声明在 RAS_GLB.bas 中）：

```
    Public Declare Function RasDial Lib "RasApi32.DLL" Alias "RasDialA"
(lpRasDialExtensions As Any, ByVal lpszPhonebook As String, lpRasDialparams As
Any, ByVal dwNotifierType As Long, lpvNotifier As Long, lphRasConn As Long) As
Long
```

其中参数 lpRasDialExtensions 为结构体 RASDIALEXTENSIONS 类型，lpszPhonebook 为字符串类型，lpRasDialparams 为结构体 RASDIALPARAMS 类型，而其他几个参数为 Long 类型。（在后面介绍以上提到的结构体）

结构体 RASDIALEXTENSIONS

```
    Public Type RASDIALEXTENSIONS
```

```
    'set dwsize to 16
    dwSize As Long
    dwfOptions As Long
    hwndParent As Long
    reserved As Long
End Type
```

✍ **说明：该结构体被** RasDial **函数使用，有了这个结构体，应用程序就可以使用** RasDial **函数的扩展特性了。在** Windows 95 **、** Windows 98 **和** Windows CE **中这个参数被设置位** NULL**，被忽略。**

下面对以上各个变量进行解释。

DwSize：**必须设置为** RASDIALEXTENSIONS **结构体的长度（按字节计算）。**

DwfOptions **：允许为使用** RasDial **扩展特性而设置位标志。表** 9-1 **对这些标志进行了说明。**

HwndParent：**不用，应该设置为** NULL **字符。**

Reserved：**不用，应该设置为** 0**。**

✍ **以上结构体是适用于** Windows 98 **和** Windows.NT **系统，对于现在新的** Windows 2000 **系统，该结构体又增加了两项，如下，如果读者在** Windows 2000 **系统下开发** RAS **程序，则需要考虑增加这两个变量**

Windows 2000 **系统增加的两个变量如下：**

```
reserved1 as Long
RasEapInfo as Long
```

reserved1：**保留，供以后** Windows 2000 RAS **扩展特性使用，应该设置为** 0**。**

RasEapInfo：**在** Windows 2000 **操作系统上，允许指定“扩展性身份验证协议”（**EAP**）信息。如表** 9-1 **所示。**

表 9-1　　RasDial **位标志**

标志	**说明**
RDEOPT_UsePrefixSuffix	**令** RasDial **使用指定拨号设备关联的前缀和后缀**
RDEOPT_PausedStates	**允许** RasDial **进入暂停操作状态，这样用户可以重试登录、修改密码和设置回拨号码**
RDEOPT_IgnoreModemSpeaker	**令** RasDial **忽略电话簿中的** Modem **喇叭设置**
RDEOPT_SetModemSpeaker	**如果设置了** RDEOPT_IgnoreModemSpeaker **标志，那么利用这个标志就可以使** Modem **喇叭打开**
RDEOPT_IgnoreSoftwareCompression	**令** RasDial **忽略软件压缩设置**
RDEOPT_SetSoftwarecompression	**如果设置了** RDEOPT_IgnoreSoftwareCompression**，则该标志可以打开软件压缩**
RDEOPT_PauseOnScript	**供** RasDialDlg **内部使用。不能设置这个标志**

结构体 RASDIALPARAMS：

```
Public Type RASDIALPARAMS
    'set dwsize to 736 unless winver >= 400 then set to 1052
    dwSize As Long
    szEntryName(RAS_MaxEntryName) As Byte
    szPhoneNumber(RAS_MaxPhoneNumber) As Byte
```

```
    szCallbackNumber(RAS_MaxCallbackNumber) As Byte
    szUserName(UNLEN) As Byte
    szPassword(PWLEN) As Byte
    szDomain(DNLEN) As Byte
End Type
```

✍ RasDial 函数利用该结构体建立一个远程连接，该结构体定义了拨号和用户的身份验证。

DwSize：应该设置为结构体 RASDIALPARAMS 的长度（按照字节计算）。有了这个标志，RAS 便可以对程序使用的操作系统平台版本进行内部判断。因为不用的操作系统，这个结构体是不一样的。

SzEntryName：一个字符串，允许标志一个电话簿条目，该条目包含在 RasDial 函数的 lpszPhonebook 参数中列出的电话簿文件内。该参数比较重要，因为电话簿条目能够使用户更好地指定 RAS 连接属性。如选定一个 Modem 或者一个分帧协议等等。但是，为了使用 RasDial 而指定一个电话簿条目并不是非用不可。可以根据实际情况选择使用。如果该字段为空，RasDial 就会选择系统上安装的第一个可以使用的 Modem，并依据下一个参数 szPhoneNumber 来拨叫连接。

SzPhoneNumber：一个字符串，代表一个电话号码，这个号码优先于 SzEntryName 字段中指定的电话簿条目内包含的电话号码。

SzCallbackNumber：允许指定一个电话号码，RAS 服务器可以根据这个号码回拨。如果 RAS 服务器允许有一个回拨号码，它就会中断原来的连接，利用指定的回拨号码回拨。这个特性非常有用，因为这样服务器可以知道连接的用户来自何处。

SzUserName：一个字符串，标志 RAS 服务器上的用户进行身份验证时所用的登录名。

SzPassword：一个字符串标志 RAS 服务器上的用户进行身份验证所用的密码。

SzDomain：可选项，指定最初的电话簿子条目拨叫一个 RAS 多链路连接（在后面将会介绍）。

✍ 以上的这些字段是在非服务器版中的设置，如果在 Windows NT 或者 Windows 2000 操作系统中，还需要增加两个字段。并且，如果增加字段，字段 dwSize 的长度要改变。

下面为额外的字段：

dwSubEntry：标志用户帐号所在的 Windows 2000 或者 Windows NT 区域。

dwCallbackId：允许把一个应用程序定义的值投递到一个 RasDidalFunc2 回拨函数中，这个字段没有使用。

参数 lpszPhonebook

RasDial 函数中有一个参数为 lpszPhonebook，该参数的作用为在 Windows 2000/NT 操作系统中识别到一个电话簿文件的路径。如在 Windows 95、Windows 98、Windows CE 上，这个参数必须设为 NULL。因为电话簿是保存在系统的注册表里面的。所谓电话簿，就是一个 RAS 拨号属性的集合，这些属性对如何设置一个 RAS 连接进行了定义。但并不要求利用这个电话簿建立一个 RAS 连接。RasDial 的特色在于它的丰富的拨号参数，允许设置一个基本的连接。

参数 dwNotifierType 和 lpvNotifier

这两个参数用于对 RasDial 操作模式进行判断可以用同步调用还是异步调用。

参数 lphRasConn

它是一个 Long 类型的参数，该参数用于保存函数的返回值，在开始调用 RasDial 函数的时候，必须赋给空值，如果函数调用成功，则返回一个指向该 RAS 连接的句柄。

（2）函数 RasHangUp

通过 RasDial 建立一个拨号连接以后，用户要能够对它进行控制，通常如果用户在利用 Windows 自带的拨号程序拨号连接成功以后，会在任务栏的最右端托盘处显示一个图标，表示正在连接，同时会显示连接的速度以及接收和发送的字节。此时用户只需要右键单击该图标，然后在弹出菜单上选择断开连接就可以了。当然也可以通过程序来实现这个功能。实现该功能的方法就是调用函数 RasHangUp。

Public Declare Function RasHangUp Lib "RasApi32.DLL" Alias "RasHangUpA" (ByVal hRasConn As Long) As Long

该函数一共就只有一个参数，即一个指向当前正在连接的句柄，也就是在上一个函数中的最后一个参数 hRasConn 所得到的值。该函数的使用非常简单，但为了能让读者更深地了解其连接的实质，我们再多介绍一些。连接在利用一个调制解调器端口时，如果连接关闭，这个端口需要花时间重新建立连接。因此，应该一直等下去，直到连接的端口完全关闭为止。为了要重新建立连接，可以使用另外一个函数来判断连接是否完全关闭。该函数为 RasGetConnectStatus。

（3）函数 RasGetConnectStatus

该函数的调用格式如下：

```
Public Declare Function RasGetConnectStatus Lib "RasApi32.DLL" Alias "RasGetConnectStatusA" (ByVal hRasConn As Long, lpRASCONNSTATUS As Any) As Long
```

而参数 lpRASCONNSTATUS 为一个结构体 RASCONNSTATUS。它取得当前连接的状态。

结构体 RASCONNSTATUS

```
Public Type RASCONNSTATUS
    'set dwsize to 64 unless winver >= 400 then set to 288
    dwSize As Long
    rasconnstate As Long                        'RASCONNSTATE Enumeration
    dwError As Long
    szDeviceType(RAS_MaxDeviceType) As Byte
    szDeviceName(RAS_MaxDeviceName) As Byte
End Type
```

下面对各个成员的含义简单说明。

DwSize：该字段为该结构体的大小，即 RASCONNSTATUS 的长度（按照字节计算）。

Rasconnstate：获得一种连接状态，连接状态有很多种，详见表 9-2。

DwError：如果函数返回值不是 0，就取得一个具体的 RAS 错误代码。

SzDeviceType：该字段取得一个字符串，该字符串代表连接所用的设备类型。

SzDeviceName：取得当前设备的名称。

学习了上面的这个函数，读者在建立一个新的连接以前，要尽可能地调用这个函数判断当前的状态，以便能够成功地建立一个新的连接。

该函数一共有两个函数，其中第一个参数很容易理解，就是指向连接的句柄，与关闭连接函数的参数含义相同。

表 9-2　　RAS 连接活动

标志	状态	说明
RASCS_OpenPort	运行	即将打开一个通信端口
RASCS_PortOpened	运行	通信端口已经打开
RASCS_ConnectDevice	运行	准备一个设备连接
RASCS_DeviceConnected	运行	已经成功与设备连接
RASCS_AllDevicesConected	运行	物理链接已经建立
RASCS_Authenticate	运行	RAS 身份验证进程已经开始
RASCS_AuthNotify	运行	身份验证事件已经发生
RASCS_AuthRetry	运行	服务器已经尝试请求另外一个身份验证
RASCS_AuthCallback	运行	服务器请求一个回拨号码
RASCS_AuthChangePassword	运行	客户机已经请求改变 RAS 帐号上的密码
RASCS_AuthProject	运行	协议发送准备进行
RASCS_AuthLinkSpeed	运行	链接速度正在计算过程中
RASCS_AuthAck	运行	身份验证请求正在确认过程中
RASCS_ReAuthenticate	运行	回拨之后的身份验证进程准备开始
RASCS_Authenticated	运行	客户机已经成功结束身份验证
RASCS_PrepareForCallback	运行	线路即将取消连接，准备回拨
RASCS_WaitForModemReset	运行	准备回拨之前，客户机等待 Modem 的重新设置
RASCS_WaitForCallBack	运行	客户机等待服务器发出的接入拨号
RASCS_Projected	运行	协议已经发送完成
RASCS_StartAuthentication	运行	用户身份验证已经开始或者已经完成(只适用于 9x 操作系统)
RASCS_CallbackComplete	运行	已经回拨客户机（只适用于 95/98 操作系统）
RASCS_LogonNetwork	运行	客户机正在登录远程网络（只适用于 95/98 操作系统）
RASCS_SubEntryConnected	运行	多链路电话簿条目的子条目已经接入。RasDialFunc2 的 dwSubEntry 参数中将包含一个已经连接的子条目的索引。
RASCS_SubEntryDisconnected	运行	多链路电话簿条目已经取消连接。RasDialFunc2 的 dwSubEntry 参数中将包含这个已经被取消的连接的索引。
RASCS_RetryAuthentication	暂停	RasDial 正在等待新的用户凭据
RASCS_CallBackSetByCaller	暂停	RasDial 正在等待客户机的回拨号码
RASCS_PasswordExpired	暂停	RasDial 希望用户提供一个新密码
RASCS_InvokeEapUI	暂停	Windows 2000 平台上，RasDial 等待自定义用户接口，以便获得 EAP 信息
RASCS_Connected	中止	RAS 连接成功，处于活动状态
RASCS_Disconnected	中止	RAS 连接失败或者处于不活动状态

连接有 3 种活动状态：运行、暂停和中止。

■　运行状态

表示 RasDial 调用仍然处于进程当中，每一处运行状态活动都将提供进程状态分析。

■　暂停状态

表示 RasDial 需要更多的信息来建立连接，默认设置是取消暂停状态。在 RASDIALEXTENSIONS 结构中设置 RDEOPT_PausedStates 标志，便可以启动通知进程。当连接处于暂停状态时，可能有如下原因：

① 因为身份验证失败，用户需要提供新的登录凭证。

② 因为密码过期需要提供一个新的密码

③ 用户密码需要提供一个回拨号码。

■ 终止状态

表示 RasDial 拨号失败，或者表示 RasHangUp 函数关闭了连接。

9.4.2 连接管理函数

在 RAS API 中提供了几个非常有用的连接管理函数，它们可以获得系统上已经建立的各个连接的属性。这些函数分别是 RasEnumConnections、RasGetSubEntryHandle、RasGetProjectionInfo。

（1）函数 RasEnumConnections

该函数的功能是列出所有活动的 RAS 连接，具体调用格式如下：

```
Public Declare Function RasEnumConnections Lib "RasApi32.DLL" Alias "RasEnumConnectionsA" (lprasconn As Any, lpcb As Long, lpcConnections As Long) As Long
```

函数 RasEnumConnections 有 3 个参数，第一个参数是一个结构体，另外两个是一个数值型变量。

lprasconn 为一个应用程序缓冲区，将接收到一个 RASCONN 结构数组。

结构体 RASCONN

```
Public Type RASCONN
    'set dwsize to 32
    dwSize As Long
    hRasConn As Long
    szEntryName(RAS_MaxEntryName) As Byte
End Type
```

✍ 对于不同的操作系统，该结构体的结构是不一样的，因此 dwsize 的长度也要动态的改变。以上的几个字段是基本的，即任何系统都需要。下面针对不同的操作系统需要增加的字段。

Windows 95、Windows 98 增加字段如下：

```
szDeviceType(RAS_MaxDeviceType) As Byte
szDeviceName(RAS95_MaxDeviceName) As Byte
```

Windows NT 系统（包括 Windows 95 增加字段）：

```
SzPhonebook(MAX_PATH) As Byte
DwSubEntry As Long
```

Windows 2000 系统（包括 Windows 95 和 NT 增加字段）：

```
    GuidEntry As Long
```

接下来对以上结构体中的各个字段进行解释：

dwSize：获得结构体长度，注意要根据操作系统的不同，动态设置该变量。

hRasConn：获得建立的连接句柄。

SzEntryName：取得用来建立连接的电话簿条目。如果采用了空字符串，这个字段就回返回一个带有句号的（.）字符串，后面还有全部的电话号码。

SzDeviceType：获得描述连接所有的设备类型。

SzDeviceName：获得用于连接的设备名称。

SzPhonebook：为了建立连接取得电话簿的完整路径。

DwSubEntry：取得一个多链路电话簿条目的子条目索引。

GuidEntry：在 Windows 2000 平台上，为了建立连接所用的电话簿条目取得一个 GUID。

✍ 对于函数 RasEnumConnections 而言，需要把它投到一个够大的缓冲区，使得能够容纳若干个 RASCONN 结构，否则该函数可能失败，并返回 ERROR_BUFFER_TOO_SMALL 的错误。而且在缓冲区中，第一个 RASCONN 结构中的 dwSize 字段必须被设置成为自己的字节长。

■　参数 lpcb

该参数为一个地址变量，必须把这个变量设置为自己的 lprasconn 的结构体数组的长度。当函数返回的时候该变量就会包含所列举的所有连接时所需要的一切字节。即使没有提供足够大的缓冲区，也可以用该参数变量返回的正确的缓冲区长度再试一次。

■　参数 lpcConnections

该参数也是一个地址变量，取得被写成 lprasconn 的 RASCONN 结构体的个数。

（2）函数 RasGetSubEntryHandle

该函数允许为多链接的连接一个具体的字条目取得一个连接句柄。由于多链接在本书中不作重点，也就不多作说明。

当有了上面两个管理连接的 RAS 函数后，就可以获得网络协议特有的信息。该信息用于一个已经建立的 RAS 连接。这个新的网络协议特有的信息就是通常所说的“Projection information”(投影信息)。远程访问服务器利用这个投影信息来表示网络上的一个远程客户机。比如说，在一个分帧协议上建立一个连接的时候，这个连接采用的时 IP 协议，IP 配置信息（如分配得到的 IP 地址）就会从 RAS 服务器到客户之间建立起来。要想获得 PPP（点对点）分帧协议上的协议投影信息，就可以使用管理连接函数中的第三个函数 RasGetProjectionInfo。

（3）函数 RasGetProjectionInfo

函数的调用格式如下：

```
   Public Declare Function RasGetProjectionInfo Lib "RasApi32.DLL" Alias
"RasGetProjectionInfoA" (ByVal hRasConn As Long, ByVal rasprojection As Long,
lpprojection As Any, lpcb As Long) As Long
```

该函数有 4 个参数，其中 hRasConn 为建立的连接句柄。rasprojection 为一个枚举类型，允许指定一个协议来接收连接信息。lpprojection 为取得一个数据结构，这个结构和

Rasprojection 中指定的枚举类型有关联。lpcb 参数为一个变量，必须把这个变量设置成为自己 lpprojection 结构体的长度。该函数返回时，变量 lpcb 会得到包含获得投影信息所需要的缓冲区长度。

在下面的 RASPROJECTION 枚举类型允许取得连接信息：

- RASP_Amb
- RASP_PppNbf
- RASP_PppIpx
- RASP_PppIp
- 结构体 RASAMB

当指定 RASP_Amb，就会得到一个 RASAMB 结构体，具体如下：

```
Public Type RASAMB
    'set dwsize to 28
    dwSize As Long
    dwError As Long
    szNetBiosError(NETBIOS_NAME_LEN) As Byte
    bLana As Byte
End Type
```

dwSize：结构体的长度

dwError：从 PPP 协商进程中取得一个错误代码。

szNetBiosError：如果在 PPP 身份验证进程中，发生了名字冲突，就回收到一个 NetBIOS 名。如果 dwError 返回的 ERROR_NAME_EXISTS_ON_NET，szNetBiosError 就会收到导致这一错误的名字。

bLana：对于用于建立远程访问连接的 NetBIOS LAN 适配器(LANA)的编号进行识别。

结构体 RASPPPNBF

当指定 RASP_PppNbf，就会得到一个 RASPPPNBF 结构体，具体如下：

```
Public Type RASPPPNBF
    'set dwsize to 48
    dwSize As Long
    dwError As Long
    dwNetBiosError As Long
    szNetBiosError(NETBIOS_NAME_LEN) As Byte
    szWorkstationName(NETBIOS_NAME_LEN) As Byte
    bLana As Byte
End Type
```

该结构体比 RASAMB 结构体多了两个字段，而其他字段意思一样。

szNetBiosError：该字段收到 NetBIOS 名，这个名用来识别网络上你正在连接的那个工作站。

szWorkstationName:：该字段得到的是发生的 NetBIOS 错误。

结构体 RASPPPIP

当指定 RASP_ PppIpx，就会得到一个 RASPPPIPX 结构体，具体如下：

```
Public Type RASPPPIPX
    'set dwsize to 32
    dwSize As Long
    dwError As Long
    szIpxAddress(RAS_MaxIpxAddress) As Byte
End Type
```

字段 dwSize 和 dwError 的意思同上面的函数。

szIpxAddress：该字段返回代表客户机的 IPX 地址。

当指定 RASP_ PppIp，就会得到一个 RASPPPIP 结构体，具体如下：

```
Public Type RASPPPIP
    'set dwsize to 40
    dwSize As Long
    dwError As Long
    szIpAddress(RAS_MaxIpAddress) As Byte
    szServerAddress(RAS_MaxIpAddress) As Byte
End Type
```

SzIpAddress：获得一个代表客户机的 IP 地址

szServerAddress：获得一个代表服务器 IP 地址。

以上对连接管理的各个 API 函数做了比较详细的解释，至于具体的应用，在后面的一个综合性例题里面会有体现。

9.4.3　电话簿和用户凭证管理

利用电话簿条目对建立连接所需要的属性进行保存和管理。RAS 便是通过这种方式来设置同远程服务器的通信。电话簿通常是一个 RASENTRY 结构的集合。在这些结构中包含了电话号码、数据速率、用户身份验证信息和其他的连接信息。在 Windows 95、Windows 98 和 Windows CE 平台上，电话簿保存在系统的注册表中，而在 Windows NT 和 Windows 2000 中，电话簿一般是保存在扩展名为.PBK 的文件中。

在调用 RAS API 时，需要传递一个参数 lpszPhonebook，用于识别电话簿的路径。通常由于在 Windows 95\98\CE 系统中，电话簿保存在注册表中，所以必须设置为控制字符 NULL，而在其他平台如 Windows NT 或者 Windows 2000 平台上，可以指定一个具体的路径。在 NT 或者 Windows 2000 平台上，默认的电话簿路径是“ 系统根目录\system32\Ras\RasPhone.pbk ”。如果指定电话簿为 NULL，则表示使用默认的电话簿文件。

在 RAS API 中，提供了若干个函数用于管理电话簿和用户凭证，接下来对其中的一部分进行说明。

一共有 4 个函数可以用于用户凭证管理，它们分别是 RasGetCredentials、RasSetCredentials、RasGetEntryDialParams 和 RasSetEntryDialParams。这 4 个函数允许对一个电话条目关联的用户安全凭证进行管理。RasGetCredentials 和 RasSetCredentials 函数是在 Windows NT 中引入的（同样适用于 Windows 2000 系统。这两个函数其实是取代了

RasGetEntryDialParams 和 RasSetEntryDialParams 函数。但是前面的两个函数不能通用，后面的两个函数可以适用于各个平台，因此开发的时候，如果是要适用于多平台，则可以考虑使用后者。

（1）函数 RasGetEntryDialParams

该函数的调用格式如下：

```
Public Declare Function RasGetEntryDialParams Lib "RasApi32.DLL" Alias "RasGetEntryDialParamsA" (ByVal lpszPhonebook As String, lpRasDialparams As Any, lpfPassword As Long) As Long
```

参数 lpszPhonebook 指向文件簿路径，lpRasDialparams 为结构体 RASDIALPARAMS，lpfPassword 为一个布尔型的标志，如果在 lpRasDialparams 结构体中获得用户密码，就会返回 True。

（2）函数 RasSetEntryDialParams

该函数调用格式如下：

```
Public Declare Function RasSetEntryDialParams Lib "RasApi32.DLL" Alias "RasSetEntryDialParamsA" (ByVal lpszPhonebook As String, lpRasDialparams As Any, ByVal fRemovePassword As Long) As Long
```

lpszPhonebook 和 lpRasDialparams 参数意思同上一个函数，而参数 fRemovePassword 为一个布尔型标志，如果被设置为 True，则是要求 RasSetEntryDialParams 删除与指定电话簿条目相关的秘密，这个条目是在 lpRasDialparams 结构体中识别的。

9.4.4 拨号方式

在拨号的时候，可以采用两种模式来进行，一种是同步方式，另外一种是异步方式。

（1）同步方式

在 RasDial 函数中，有一个参数 lpvNotifier，如果被设置成为 NULL，RasDial 就会进入同步模式。如果 lpvNotifier 被设置成为 NULL，参数 dwNotifierType 参数就会被忽略。同步调用函数使用非常简单，但是有一个缺点就是不能对连接的状态进行检测。

（2）异步方式

相对于同步方式来说，异步方式要复杂许多。在 RasDial 函数中，如果 lpvNotifier 参数没有被设置成为 NULL，RasDial 函数就会进入异步方式。这表明函数在进行连接的同时，函数调用立即返回。因此读者在开发 RAS 程序的时候，最好能够选择异步拨号方式，因为这样可以随时对程序进行监控并且作出相应的处理。注意 lpvNotifier 参数即可以是一个指针指向一个在 RasDial 中发生连接活动时被调用的函数，又可以是一个窗口句柄（通过 Windows 消息接收进程通知）。RasDial 函数的 dwNotifierType 参数用于判断函数类型或者被传递的窗口句柄。DwNotifierType 可以取的值如表 9-3 所示。

表 9-3　　RasDial 异步通知方法

通知类型	说明
0	LpvNotifier 参数令 RasDial 利用 RasDialFunc 函数指针管理连接事件
1	LpvNotifier 参数令 RasDial 利用 RasDialFunc1 函数指针管理连接事件
2	LpvNotifier 参数令 RasDial 利用 RasDialFunc2 函数指针管理连接事件
0xFFFFFFF	LpvNotifier 参数令 RasDial 在连接事件期间发送一个窗口消息

在本书后面的实际例题中，既有异步拨号模式和同步拨号模式，但是在进行异步拨号的时候都是传递窗口句柄来实现的，因为在 VB 中实现回调函数比较复杂。因此关于函数 RasDialFunc、RasDialFunc1、RasDialFunc2 的使用，在下面将给出 VC 的原型，读者可以自己使用 VC 开发，或者用 VC 编写动态连接库，然后在 VB 中调用来实现。当然最好的方式是能够采用回调函数来实现。因为通过窗体句柄传递还需要增加一个计时器控件来扫描拨号状态。

RasDialFunc 函数原型

```
VOID WINAPI RasDialFunc(
              UINT unMsg,
              RASCONNSTATE rasconnstate,
              DWORD dwError
);
```

unMsg：该参数接收已经发生的事件类型。当前的这个事件可能是 WM_RASDIALEVENT，它意味着这个参数没有用。

rasconnstate：该参数接收 RasDial 函数即将发生了连接活动。这些活动在表 9-2 中描述。

dwError：如果连接失败，dwError 参数就会收到 RAS 错误代码。

RasDialFunc1 函数原型

```
VOID WINAPI RasDialFunc1(
              HRASCONN hrasconn,
UINT unMsg,
              RASCONNSTATE rascs,
              DWORD dwError,
              DWORD dwExtendedError
);
```

RasDialFunc1 函数和 RasDialFunc 函数比较相似，只是增加了两个参数 hrasconn 和 dwExtendedError。

hrasconn：一个指向连接的句柄。这个连接句柄是又 RasDial 拨号成功后返回的。

dwExtendedError：运行在连接期间获得 dwError 参数所发生错误的扩展错误信息。

① ERROR_SERVER_NOT_RESPONDING，表示接收到一个 NetBIOS 特有的错误代码。

② ERROR_NETBIOS_ERROR，表示接收到一个 NetBIOS 特有的错误代码。

③ ERROR_AUTH_INTERNAL，表示接收到一个内部诊断性错误代码。

④ ERROR_CANNOT_GET_LANA，接收到一个路由 RAS 特有的错误代码

RasDialFunc2 函数原型

```
VOID WINAPI RasDialFunc2(
              DWORD dwCallbackId,
              DWORD dwSubEntry,
              HRASCONN hrasconn,
UINT unMsg,
              RASCONNSTATE rascs,
              DWORD dwError,
              DWORD dwExtendedError
);
```

RasDialFunc2 函数同 RasDialFunc1 相比，增加了两个参数。dwCallbackId 参数中包含了一个应用程序定义的值。该值是在 RASDIALPARAMS 结构体的 dwCallbackId 字段中。dwSubEntry 参数接收子条目电话簿索引，这个索引会再次回调 RasDialFunc2。

9.5 RAS 高级程序开发实例

在 9.4 节，主要介绍了 RAS 的一些函数声明和 RAS 工作原理。但是仅仅介绍些枯燥的理论也许并不能帮助读者真正理解 RAS 的工作模式和工作原理。通过学习 RAS 实例是最好的学习办法。

9.5.1 建立工程项目

该例程是一个综合性的例题，里面涉及到了许多 VB 的应用，比如类的应用、API 的应用等。由于本书的关键是向读者介绍一个理论如何实现，而不在于 VB 的知识点，所以有关 VB 的知识就不多介绍了，读者可以参考相关书籍。

本工程可以实现以下功能：建立电话簿、编辑电话簿、查看电话簿内容、建立新的连接、查看连接信息、同步拨号、异步拨号、状态检测、挂断拨号。

该工程由以下几个模块组成：

窗体 frmAsyncDial
窗体 frmCreateConn
窗体 frmRASMAIN
基本模块 RAS_GLB
类模块 Connection
类模块 Connections
类模块 PhoneEntries
类模块 PhoneEntry]
类模块 RASEngine
类模块 RASError

9.5.2　程序运行结果图

为了让读者更好地看到本程序的工作状况，利用下面一系列的图说明了整个操作步骤，这样读者可以理解工作步骤以后去学习下面的程序。

（1）程序启动界面，如图 9-13 所示。

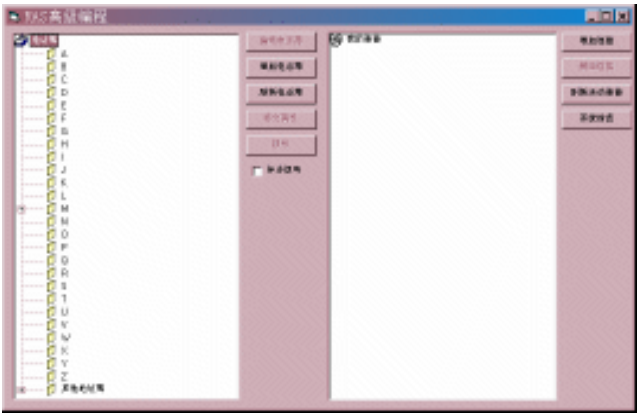


图 9-13　程序运行结果图

从图中可以看到整个程序被划分成两个区域左边是对电话簿进行管理的，而右边是对活动连接进行管理。接下来一步一步地测试程序，当单击按钮“增加电话簿”(中间一排第二个按钮)时，会弹出建立电话簿的对话框，如图 9-7 所示，可以一步一步按照向导建立一个电话簿，这里我们建立一个电话簿为“wxp”。当建立好一个连接以后，会在图的左侧出现一个“+”号，是在以“w”开始的位置，因为界面安排电话簿的管理是按照字符顺序来进行的，点开“+”号，可以展开，从展开的树中可以看到电话簿的各项属性。如图 9-14 所示。

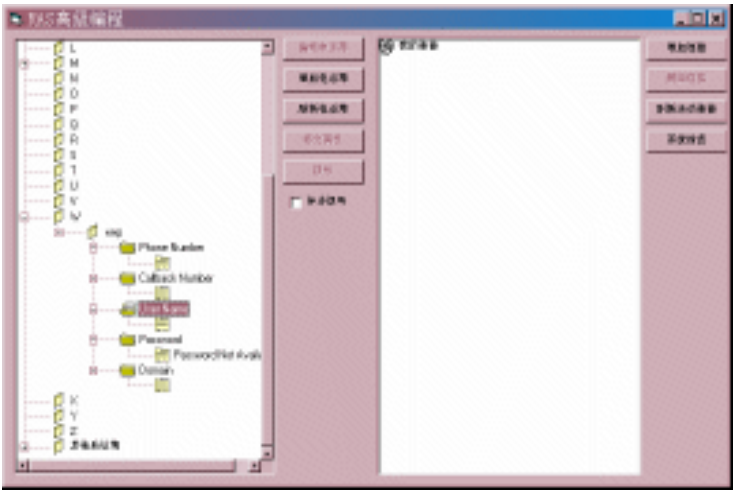


图 9-14　查看一个电话簿的信息

如果要改变其中的一个属性，比如电话号码，可以用鼠标单击，然后按钮“提交属性”（从上数第 4 隔按钮），可以进行修改。如图 9-15 所示。

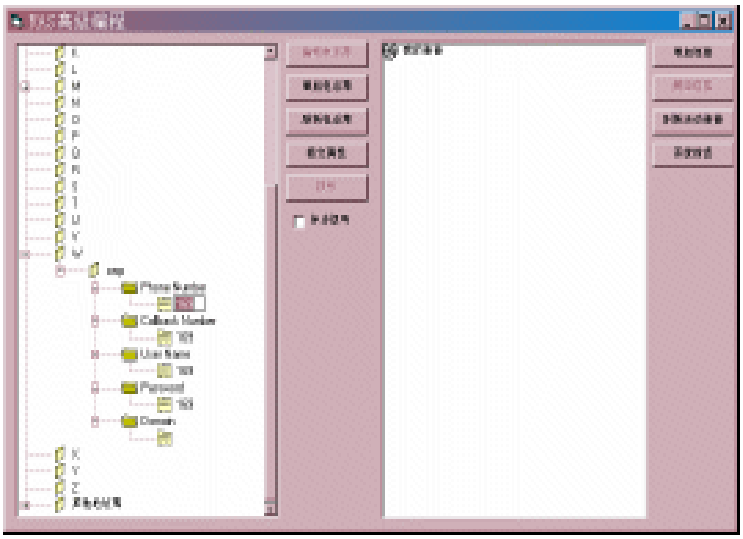


图 9-15　修改电话簿属性

电话簿的管理是 RAS 的一个重要的功能，但对于读者来说拨号才是最关键的，在程序界面的右边区域就是管理拨号连接的。在右边空白区域列出了所有活动的连接，通常在一般的机器上只能建立一个活动拨号连接。因此可以首先建立一个活动 RAS。单击右边一排按钮的第一个“增加连接”，界面如图 9-16 所示。

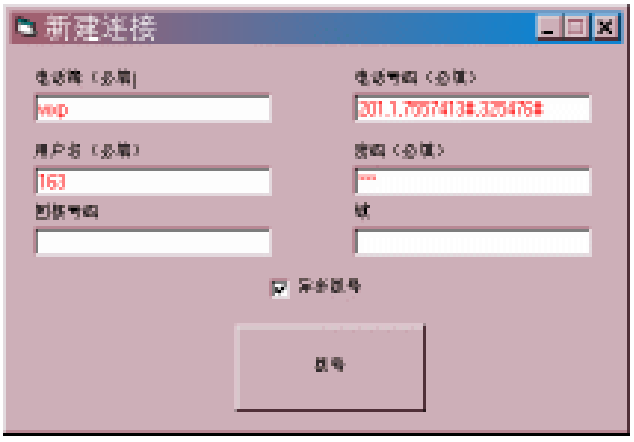


图 9-16　新建连接界面

在图 9-16 中输入一些参数就可以进行拨号了。在电话簿里面，输入刚才建立的“wxp”，然后输入远程主机的电话号码、用户名、密码。现在对于 163、169 都有缺省的帐号和密码在前面已经介绍，即 163、163 和 169、169。

✍ 在拨号的时候一定要正确输入电话簿的名称，或者用下拉表框列出所有的已经在机器里面建立的电话簿，这个功能在读者学习了后面的的程序以后很容易实现。如果机器里面一个电话簿都没有，可以让用户新增电话簿。

在进行拨号以前，可以选择异步拨号或者同步拨号，如果选择异步，只需要将检查框选

中即可。单击“拨号”按钮进入状态检测界面，如图 9-17 所示。

图 9-17　异步拨号状态检测界面

状态显示正在连接，如图 9-18 所示。

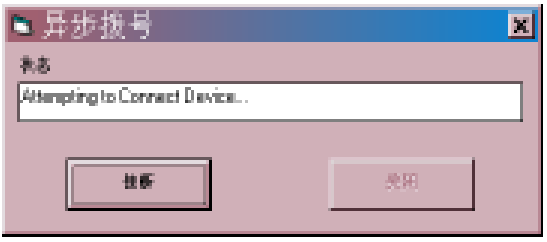


图 9-18　线路正忙状态

显示验证用户名和密码完毕状态，如图 9-19 所示。

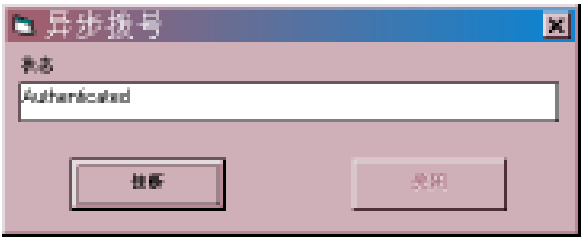


图 9-19　状态验证完毕

连接成功状态，如图 9-20 所示。

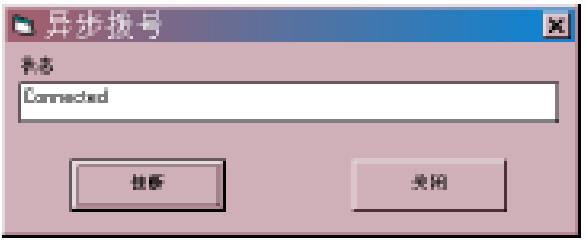


图 9-20　状态验证完毕

查看连接属性，如图 9-21 所示。

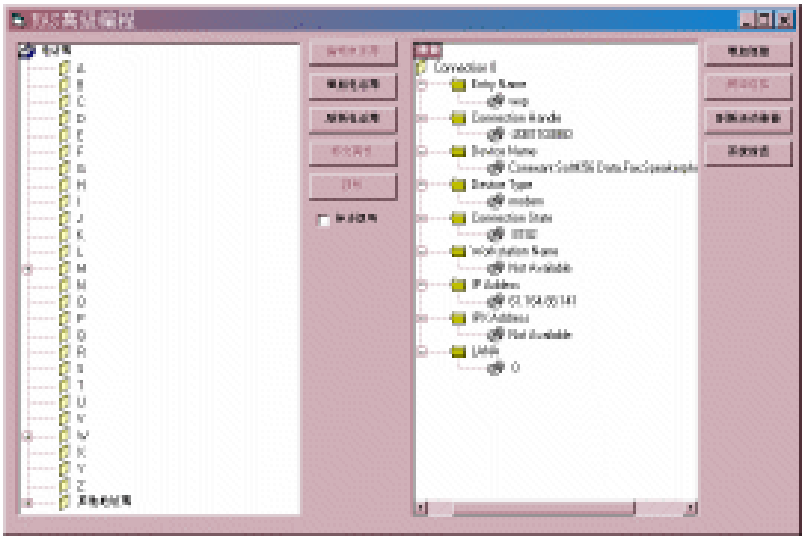


图 9-21　状态验证完毕

在图 9-21 中，可以看到右边区域列出了刚才创建的连接，从展开树可以查看一些属性，如电话簿“ Entry Name=wxp ”，就是在开始拨号的时候设置的电话簿名称（如图 9-16 所示）、“ IP Address=61.164.88.141 ”为拨号成功以后获得动态 IP 地址，如果用户想获得拨号成功以后本机的 IP 地址，就可以通过改方法获得了，还有其他的一些属性，读者可以自己去测试。

当连接成功以后，就可以使用各种网络函数以及网络资源，图 9-22 所示的是使用“ PING ”函数 PING 通一个主机地址“ www.sina.com（新浪网） ”。

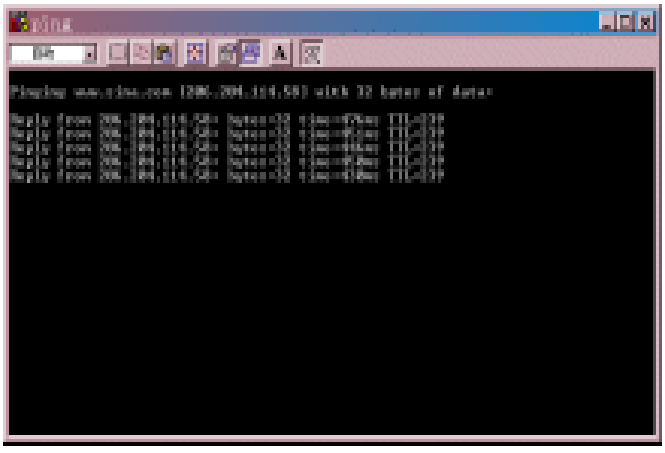


图 9-22　状态验证完毕

图 9-23 演示的是通过浏览器连接新浪网的主页。

图 9-23　新浪网主页

至此，整个程序的基本工作过程和主要界面都给读者介绍了，当然整个程序不止这些功能，读者朋友可以自己去测试。

✍ 本程序在 Windows 98 系统下测试成功，笔者没有在 NT 系统下测试，读者可以自己去测试。最后演示了两个网络应用，其实可以使用几乎多所有的网络应用，比如可以使用 FTP 下载文件，发送 E-mail，以及共享文件等等。同专线上网的感觉是一样的，可能速度上有些区别，毕竟现在的 Modem 的传输速度还不是很快。

9.5.3　关键代码分析

在下面的章节中，将对程序中的关键代码进行分析和说明，以帮助读者理解程序。如果还不能理解，可以打开附录光盘中的源代码进行调试来分析。下面为了说明方便以及前后安排，将按照程序运行时所用到的代码的引用次序来说明，被其他程序调用的程序将先介绍，这样，读者更容易理解。

1. 类模块 Connections

该类模块的功能是保存一个连接的所有信息，但是拨号的功能并不是在这个类模块中实现的。如果已经建立了一个连接以后，客户端可以得到许多关于这个连接的信息，比如客户端动态分配的 IP 地址、所使用的设备名称、电话号码等信息。

```
'类模块 Connections
Option Explicit
'connection 类的内部变量
Private intIndex As Integer
Private lnghRasConn As Long
Private strEntryName As String
Private lngRASConnState As Long
Private strDeviceType As String
Private strDeviceName As String
```

```
Private strPhoneNumber As String
Private strCallbackNumber As String
Private strUserName As String
Private strDomain As String
Private strPassword As String
Private strWorkstationName As String
Private strIPXAddress As String
Private strIPAddress As String
Private bytLANA As Byte

'获得拨号连接句柄
Public Property Get hRasConn() As Long
   hRasConn = lnghRasConn
End Property

'设置拨号句柄
Public Property Let hRasConn(hNewConn As Long)
  Dim lngRetCode As Long
  '如果允许重新设置
If boolAllowUpdate Then
      lnghRasConn = hNewConn
      lngRetCode = fcnRASGetConnectionStatus()
      lngRetCode = fcnRasGetProjectionInfo()
   Else
      lngRASErrorNumber = 1111
      strRASDescription = "Property Not Updateable"
      Err.Raise vbObjectError + 1111, "Property Not Updateable", "RAS Failure"
   End If
End Property

'获得电话簿名
Public Property Get EntryName() As String
   EntryName = strEntryName
End Property

'设置电话簿名
Public Property Let EntryName(strNewName As String)
   Dim lngRetCode As Long
   '如果允许修改属性
```

```
      If boolAllowUpdate Then
         strEntryName = strNewName
         If lngRetCode Then
            Err.Raise vbObjectError + lngRetCode, "EntryName Set Failed", "RAS
Failure"
         End If
      Else
         lngRASErrorNumber = 1111
         strRASDescription = "Property Not Updateable"
         Err.Raise vbObjectError + 1111, "Property Not Updateable", "RAS Failure"
      End If


   End Property

   '获得拨号索引，主要是因为可能有多个活动连接同时存在
   Public Property Get Index() As Integer
      Index = intIndex
   End Property

   '设置拨号索引
   Public Property Let Index(intNewIndex As Integer)
   If boolAllowUpdate Then
         intIndex = intNewIndex
      Else
         lngRASErrorNumber = 1111
         strRASDescription = "Property Not Updateable"
         Err.Raise vbObjectError + 1111, "Property Not Updateable", "RAS Failure"
      End If
   End Property

   Private Function fcnRASGetConnectionStatus() As Long
   '获得连接状态
      Dim lngRetCode As Long

      If lngWindowVersion = 2 Then
         'Windows NT 操作系统
         Dim lpRASCONNSTATUS As RASCONNSTATUS
         lpRASCONNSTATUS.dwSize = 64
         lngRetCode = RasGetConnectStatus(lnghRasConn, lpRASCONNSTATUS)
```

```
        If lngRetCode Then
           '失败
           strDeviceName = "Not Available"
           strDeviceType = "Not Available"
           lngRASErrorNumber = lngRetCode
           strRASDescription = lpRASError.fcnRASErrorString()
           fcnRASGetConnectionStatus = lngRetCode
        Else
           '成功
           lngRASConnState = lpRASCONNSTATUS.rasconnstate
           strDeviceName = fcnTrimNulls(StrConv(lpRASCONNSTATUS.szDeviceName(),
vbUnicode))
           strDeviceType = fcnTrimNulls(StrConv(lpRASCONNSTATUS.szDeviceType(),
vbUnicode))
           fcnRASGetConnectionStatus = 0
        End If
      Else
        'Windows 95/98 操作系统
        Dim lpRASCONNSTATUS95 As RASCONNSTATUS95
        lpRASCONNSTATUS95.dwSize = 160
        lngRetCode = RasGetConnectStatus(lnghRasConn, lpRASCONNSTATUS95)
        If lngRetCode Then
           strDeviceName = "Not Available"
           strDeviceType = "Not Available"
           lngRASErrorNumber = lngRetCode
           strRASDescription = lpRASError.fcnRASErrorString()
           fcnRASGetConnectionStatus = lngRetCode
        Else
           '成功
           lngRASConnState = lpRASCONNSTATUS95.rasconnstate
           strDeviceName                                                    =
fcnTrimNulls(StrConv(lpRASCONNSTATUS95.szDeviceName(), vbUnicode))
           strDeviceType                                                    =
fcnTrimNulls(StrConv(lpRASCONNSTATUS95.szDeviceType(), vbUnicode))
           fcnRASGetConnectionStatus = 0
        End If
      End If
   End Function
```

```
'获得连接状态
Public Property Get State() As Long
State = lngRASConnState
End Property

'获得设备类型
Public Property Get DeviceType() As String
   DeviceType = strDeviceType
End Property

'获得设备名称
Public Property Get DeviceName() As String
   DeviceName = strDeviceName
End Property

'获得网络信息
Public Property Get LANA() As Byte
   LANA = bytLANA
End Property

'获得工作站名称
Public Property Get WorkstationName() As String
   WorkstationName = strWorkstationName
End Property
'获得 IP 地址
Public Property Get IPXAddress() As String
   IPXAddress = strIPXAddress
End Property
Private Function fcnRasGetProjectionInfo() As Long
'获得投影信息
'具体的结构体信息，在书中已经有说明
   Dim lngRetCode As Long
   Dim lprasamb As RASAMB
   Dim lpraspppnbf As RASPPPNBF
   Dim lpraspppipx As RASPPPIPX
   Dim lpraspppip As RASPPPIP
   Dim lpcb As Long
   Dim rasprojection As Long
```

```
      '仅仅在 PPP 协议下适用
      '以 RASAMB 结构体开始
      rasprojection = RASP_Amb
      lprasamb.dwSize = 28
      lpcb = 28
      lngRetCode = RasGetProjectionInfo(lnghRasConn, rasprojection, lprasamb,
lpcb)
      If lngRetCode Then
        bytLANA = 0
        lngRASErrorNumber = lngRetCode
        strRASDescription = lpRASError.fcnRASErrorString()
        fcnRasGetProjectionInfo = lngRetCode
      Else
        bytLANA = lprasamb.bLana
        fcnRasGetProjectionInfo = 0
      End If
      'Net BIOS
      rasprojection = RASP_PppNbf
      lpraspppnbf.dwSize = 48
      lpcb = 48
      lngRetCode = RasGetProjectionInfo(lnghRasConn, rasprojection, lpraspppnbf,
lpcb)
      If lngRetCode Then
        bytLANA = 0
        strWorkstationName = "Not Available"
        lngRASErrorNumber = lngRetCode
        strRASDescription = lpRASError.fcnRASErrorString()
        fcnRasGetProjectionInfo = lngRetCode
      Else
        bytLANA = lpraspppnbf.bLana
        strWorkstationName                                                    =
fcnTrimNulls(StrConv(lpraspppnbf.szWorkstationName, vbUnicode))
        fcnRasGetProjectionInfo = 0
      End If
      'IPX
      rasprojection = RASP_PppIpx
      lpraspppipx.dwSize = 32
      lpcb = 32
      lngRetCode = RasGetProjectionInfo(lnghRasConn, rasprojection, lpraspppipx,
```

```
lpcb)
      If lngRetCode Then
         strIPXAddress = "Not Available"
         lngRASErrorNumber = lngRetCode
         strRASDescription = lpRASError.fcnRASErrorString()
         fcnRasGetProjectionInfo = lngRetCode
      Else
         strIPXAddress     =     fcnTrimNulls(StrConv(lpraspppipx.szIpxAddress,
vbUnicode))
         fcnRasGetProjectionInfo = 0
      End If
      'TCP/IP
      rasprojection = RASP_PppIp
      lpraspppip.dwSize = 40
      lpcb = 40
      lngRetCode = RasGetProjectionInfo(lnghRasConn, rasprojection, lpraspppip,
lpcb)
      If lngRetCode Then
         strIPAddress = "Not Available"
         lngRASErrorNumber = lngRetCode
         strRASDescription = lpRASError.fcnRASErrorString()
         fcnRasGetProjectionInfo = lngRetCode
      Else
         strIPAddress = fcnTrimNulls(StrConv(lpraspppip.szIpAddress, vbUnicode))
         fcnRasGetProjectionInfo = 0
      End If
   End Function
```

在该类模块中，设置一些属性来让客户设置和得到其相关的值。一旦一个连接建立以后，其相关的信息都保存在该类中，因此如果要获得这些值，可以通过各种方法来实现，典型的方法就是设置全局变量来实现，但是通过 Property Get **以及** Property Let **来实现更好一点。**

类中有一个 Index **属性，该属性主要是为了标志某一个活动连接的。因为可能同时会有多个活动连接存在，因此为了让用户能够知道哪一个状态对应哪一个连接，有必要对这些连接进行编号。而在下面一个类** Connections **中，将会创建一个** Connection **类的数组，这样就可以很方便地对连接进行管理，比如查看信息、修改信息以及删除连接等。**

在该类中有一个重要的函数就是 fcnRASGetConnectionStatus**，该函数通过调用** RAS API **函数** RasGetConnectStatus **来定时查询目前连接的状态。在** VB **中由于使用回调函数比较麻烦，因此要想捕获拨号状态，最方便的办法就是通过定时器每隔一个时间段就查询一次当前的状态。获得的状态保存在一个结构体中。可以通过该结构体获得连接的状态以及设备名称、设备类型等。而在** VC **中可以直接调用回调函数来实现，该回调函数有** 3 **个，在前面的章节已**

经有过说明，这里就不详细介绍。

该类中还有一个重要的函数就是 fcnRasGetProjectionInfo。该函数获得连接的投影信息，信息是通过调用 RAS API 函数 RasGetProjectionInfo 来实现的。因此要想获得 PPP 分帧协议上的协议投影信息，就必须调用该函数。在执行的时候，可以依次对某个具体的协议调用该函数来获得相关信息。比如要获得建立在 IP 基础上的连接的 IP 地址（通常为动态 IP 地址），就可以使用语句如下：

```
strIPAddress = fcnTrimNulls(StrConv(lprasppp ip.szIpAddress, vbUnicode))
```

注意在调用函数的时候，要针对某个协议，至于每个结构体如何定义以及某个字段的大小如何，在书中和程序中都有定义。

同时，在写程序的时候，可能考虑到读者有可能在 Windows NT 系统下运行，因此在程序开发的时候增加了一些代码，用来区别是 Windows 98 系统还是 Windows NT 系统。它们主要的区别就是某些结构体中的字段不一样，要注意的一点就是，结构体变动，其代表结构体大小的字段在赋值的时候也应该动态变化，否则可能出错。如果用户不知道该结构体的大小，可以通过函数 Len(结构体名)来试一下，一般情况下可以获得正确的大小。由于 Window 2000 操作下面的拨号还有一些不同，笔者在程序中没有加入，但是相关的结构体构成在前面的函数介绍中已经介绍了，读者可以自己动手开发。

2. 类模块 Connections

该类模块的实现的功能比较关键，其中拨号的主要任务就是由这个类模块来实现的，因此读者朋友应该好好读懂这个程序。该类其实是上一个类的集合，通过创建一个类数组来保存每个活动的连接，这样就可以很容易地来进行连接地管理以及查看连接地信息。

```
'类模块 Connections
Option Explicit

'连接的个数
Private intCCount As Integer

Public Property Get Count() As Integer
   '获得连接的个数
   Dim lngRetCode

   lngRetCode = fcnRASEnumConnections()
   If lngRetCode Then
      Err.Raise vbObjectError + lngRetCode, "Connections.Count Failure", "RAS
Failure"
   Else
      Count = intCCount
   End If
End Property
```

```
'类模块实现了列出所有活动连接的功能
Private Function fcnRASEnumConnections() As Long
   Dim lngRetCode As Long
   Dim lpcb As Long
   Dim lpcConnections As Long
   Dim intArraySize As Integer
   Dim intLooper As Integer

   '定义 256 个连接，如果不够，可以再增加
   intArraySize = 255
   If lngWindowVersion = 2 Then
      Windows NT 系统
      ReDim lprasconn(intArraySize) As RASCONN
      lprasconn(0).dwSize = 32
      lpcb = 256 * lprasconn(0).dwSize
      lngRetCode = RasEnumConnections(lprasconn(0), lpcb, lpcConnections)
   Else
      'Windows 95/98 系统
      '定义结构体数组
      ReDim lprasconn95(intArraySize) As RASCONN95
      lprasconn95(0).dwSize = 412
      lpcb = 256 * lprasconn95(0).dwSize
      '获得各个活动连接
      lngRetCode = RasEnumConnections(lprasconn95(0), lpcb, lpcConnections)
   End If
   Select Case lngRetCode
      Case SUCCESS
         '如果成功
         If lpcConnections > 0 Then
            ReDim arrConnection(lpcConnections - 1) As Connection
            If lngWindowVersion = 2 Then
               'Windows NT 系统
               For intLooper = 0 To UBound(arrConnection())
                  Set arrConnection(intLooper) = New Connection
                  '允许修改电话簿
                  boolAllowUpdate = True
                  arrConnection(intLooper).hRasConn                    =
lprasconn(intLooper).hRasConn
                  arrConnection(intLooper).EntryName                   =
fcnTrimNulls(StrConv(lprasconn(intLooper).szEntryName, vbUnicode))
```

```
                    arrConnection(intLooper).Index = intLooper
                    boolAllowUpdate = False
                Next intLooper
            Else
                    'Windows 95/98 系统
                    For intLooper = 0 To UBound(arrConnection())
                    Set arrConnection(intLooper) = New Connection
                    '允许修改电话簿
                    boolAllowUpdate = True
                    arrConnection(intLooper).hRasConn                    =
lprasconn95(intLooper).hRasConn
                    arrConnection(intLooper).EntryName                   =
fcnTrimNulls(StrConv(lprasconn95(intLooper).szEntryName, vbUnicode))
                    arrConnection(intLooper).Index = intLooper
                    boolAllowUpdate = False
                Next intLooper
            End If
         End If
         '设置连接的数量
         intCCount = CInt(lpcConnections)
         fcnRASEnumConnections = 0
      Case ERROR_BUFFER_TOO_SMALL
         '设置更大的缓冲区
         If lngWindowVersion = 2 Then
            'running NT
            intArraySize = lpcb / lprasconn(0).dwSize
         Else
            'running 95
            intArraySize = lpcb / lprasconn95(0).dwSize
         End If
      Case Else
         lngRASErrorNumber = lngRetCode
         strRASDescription = lpRASError.fcnRASErrorString()
         fcnRASEnumConnections = lngRetCode
   End Select
End Function

'该函数的功能是建立拨号连接，是该类的核心程序
Public Function AddConnection(strNewEntryName As String, strNewPhoneNumber As
```

```
String, strNewCallbackNumber As String, strNewUsername As String, strNewPassword
As String, strNewDomain As String, boolAsync As Boolean) As Connection
   '该函数返回一个 Connection 对象
      Dim lngRetCode As Long
      Dim hRasConn As Long
      Dim lngRetlstrcpy As Long
      Dim intLooper As Integer
      Dim lngRetHangUp As Long

      '再 vb 中实现异步拨号，不过如果要直接实现异步拨号，要用到回调函数，而再 VB 中要实现回调函
   '数比较困难
      '因此在调用 RASDIAL 函数的时候，可以传递一个模态对话框的句柄，然后通过
   'RASGetConnectionStatus 函数来
         '获取各种状态
         If lngWindowVersion = 2 Then
            'Windows NT 系统
            Dim lpRasDialparams As RASDIALPARAMS
            lpRasDialparams.dwSize = 736
            '最好使用函数 lstrcpy，因为如果使用 StrConv，可能失败
            lngRetlstrcpy       =       lstrcpy(lpRasDialparams.szEntryName(0),
strNewEntryName)
            lngRetlstrcpy      =      lstrcpy(lpRasDialparams.szPhoneNumber(0),
strNewPhoneNumber)
            lngRetlstrcpy    =    lstrcpy(lpRasDialparams.szCallbackNumber(0),
strNewCallbackNumber)
            lngRetlstrcpy       =       lstrcpy(lpRasDialparams.szUserName(0),
strNewUsername)
            lngRetlstrcpy       =       lstrcpy(lpRasDialparams.szPassword(0),
strNewPassword)
            lngRetlstrcpy = lstrcpy(lpRasDialparams.szDomain(0), strNewDomain)
            '调用 RasDial
            If boolAsync Then
               '异步调用同时忽略 RASDIALEXTENSIONS
               '在参数传递是，其中一个是窗体的句柄
               Load frmAsyncDial
               lngRetCode = RasDial(ByVal APINULL, vbNullString, lpRasDialparams,
&HFFFFFFFF, frmAsyncDial.hWND, hRasConn)
            Else
               '同步拨号,传递了一个 APINULL
```

```
            Screen.MousePointer = vbHourglass
            lngRetCode = RasDial(ByVal APINULL, vbNullString, lpRasDialparams,
APINULL, ByVal APINULL, hRasConn)
            Screen.MousePointer = vbDefault
         End If
         '判断是否成功
         If lngRetCode Then
            lngRASErrorNumber = lngRetCode
            strRASDescription = lpRASError.fcnRASErrorString()
            lngRetHangUp = RasHangUp(hRasConn)
            Err.Raise vbObjectError + lngRetCode, "Connections AddConnection
Failed", "RAS Failure"
         Else
            '可以返回连接成功后的句柄
         End If
      Else
         '95/98 系统 (lngWindowVersion =1)
         Dim lpRasDialparams95 As RASDIALPARAMS95
         lpRasDialparams95.dwSize = 1052
         lngRetlstrcpy      =      lstrcpy(lpRasDialparams95.szEntryName(0),
strNewEntryName)
         lngRetlstrcpy     =     lstrcpy(lpRasDialparams95.szPhoneNumber(0),
strNewPhoneNumber)
         lngRetlstrcpy    =    lstrcpy(lpRasDialparams95.szCallbackNumber(0),
strNewCallbackNumber)
         lngRetlstrcpy      =      lstrcpy(lpRasDialparams95.szUserName(0),
strNewUsername)
         lngRetlstrcpy      =      lstrcpy(lpRasDialparams95.szPassword(0),
strNewPassword)
         lngRetlstrcpy = lstrcpy(lpRasDialparams95.szDomain(0), strNewDomain)
         '调用 RasDial
         If boolAsync Then
            '异步调用
            '弹出对话框 frmAsyncDial
            Load frmAsyncDial
            DoEvents
            lngRetCode = RasDial(ByVal APINULL, vbNullString, lpRasDialparams95,
&HFFFFFFFF, frmAsyncDial.hWND, hRasConn)
         Else
```

```
            '同步调用
            Screen.MousePointer = vbHourglass
            lngRetCode=RasDial(ByVal APINULL, vbNullString, lpRasDialparams95,
APINULL, ByVal APINULL, hRasConn)
            Screen.MousePointer = vbDefault
         End If
         '错误处理
         If lngRetCode Then
            lngRASErrorNumber = lngRetCode
            strRASDescription = lpRASError.fcnRASErrorString()
            lngRetHangUp = RasHangUp(hRasConn)
            Err.Raise vbObjectError + lngRetCode, "Connections AddConnection
Failed", "RAS Failure"
         Else
            '调用成功
            DoEvents
         End If
      End If

      '可以在需要的时候返回连接句柄
      '或者可以遍历所有的连接，这样可以更准确地返回各个句柄
      If boolAsync Then
         '通过窗体的 Tag 属性传递 hRasConn 连接句柄
         frmAsyncDial.Tag = Hex$(hRasConn)
         frmAsyncDial.Show 1
      Else
         '如果是同步则不用处理
      End If
      '刷新活动连接
      lngRetCode = fcnRASEnumConnections()
      If lngRetCode Then Err.Raise vbObjectError + lngRetCode, "Connections
AddConnection Failed", "RAS Failure"
      '循环获得每一个连接，如果其中一个连接等于最新建立地连接，则可以返回该连接
      '注意返回的是 Connection 对象
      For intLooper = 0 To intCCount - 1
         If hRasConn = arrConnection(intLooper).hRasConn Then
            Set AddConnection = arrConnection(intLooper)
         Else
            Set AddConnection = Nothing
         End If
```

```
        Next intLooper

    End Function

    Public Sub RemoveConnection(lngIndexToRem As Long)
    '删除一个连接
      Dim lngRetCode As Long
      Dim hRasConnToRem As Long

      '在删除以前，首先要获得句柄
      hRasConnToRem = arrConnection(lngIndexToRem).hRasConn

      '然后挂断该连接
      lngRetCode = RasHangUp(hRasConnToRem)
      If lngRetCode Then
           lngRASErrorNumber = lngRetCode
           strRASDescription = lpRASError.fcnRASErrorString()
           Err.Raise vbObjectError + lngRetCode, "Connections RemoveConnection
Failed", "RAS Failure"
        Else
           '重新刷新活动连接
           lngRetCode = fcnRASEnumConnections()
           If lngRetCode Then Err.Raise vbObjectError + lngRetCode, "Connections
RemoveConnection Failed", "RAS Failure"
      End If
    End Sub

    '类初始化事件
    Private Sub Class_Initialize()
      Dim lngSuccess As Long
      '在初始化时获得所有活动连接
      lngSuccess = fcnRASEnumConnections()
    End Sub
```

该类中有几个重要的函数需要解释一下。在上一个类中提到为了能够管理所有的活动连接，需要创建一个类数组来保存每一个连接的信息。但是如何才能获得每一个活动的连接呢？许多时候可能在用户进行远程访问的时候，需要获得其他应用程序创建的连接，有必要的情况下还必须把连接断开，然后建立自己的连接。而当自己的应用程序启动的时候，并不都可以获得其他活动连接的句柄，因此还是需要其他函数的帮忙，在本类中函数 fcnRASEnumConnections 就是实现了这个功能。该函数是调用了 RAS API 函数 RasEnumConnections(lprasconn95(0), lpcb, lpcConnections)来完成得。其中参数 lprasconn95(0)

是一个结构体数组，开始定义缓冲大小为 256，通常不会超标，因为一般不可能同时有这么多的活动连接存在。当函数调用成功，lpcConnections 是获得具体的连接数目。如果给定的缓冲区大小不够，可以通过错误处理重新设置缓冲区的大小。如下代码：intArraySize = lpcb / lprasconn95(0).dwSize，其中 lpcb 也是函数的返回后获得的值，其为需要缓冲区的大小，因此可以动态改变数组的大小。这样在第二次调用的时候就不会出错了。获得具体的连接数量以后，就可以创建相同数量的 Connection 类了，为了设置每一个类的相关属性，必须从结构体数组 lprasconn95(0)中获得相关信息，具体操作，读者可以看上面的源程序。

在这个类中还有一个更加重要的类就是拨号的类，其实本章节的核心内容就是说明如何建立远程登录，而核心程序就是拨号了。

具体函数为：

```
Public Function AddConnection(strNewEntryName As String, strNewPhoneNumber As String, strNewCallbackNumber As String, strNewUsername As String, strNewPassword As String, strNewDomain As String, boolAsync As Boolean) As Connection
```

在这个函数体中实现的办法就是调用 RasDial API 函数，在开始的时候已经说过，拨号存在两种方式，同步的和异步的，因此它们的调用格式是不一样的。

对于异步拨号，格式如下：

```
lngRetCode = RasDial(ByVal APINULL, vbNullString, lpRasDialparams, &HFFFFFFFF, frmAsyncDial.hWND, hRasConn)
```

这其中注意一个参数 frmAsyncDial.hWND，在函数介绍的时候已经说明过，如果要实现异步拨号，该参数不能为空，可以是一个模态窗体的句柄或者是回调函数。在 VB 中采用了窗体的句柄传递。然后通过定时器来获得状态。而在同步连接中，该参数为空，如下：

```
lngRetCode = RasDial(ByVal APINULL, vbNullString, lpRasDialparams, APINULL, ByVal APINULL, hRasConn)
```

至于是否调用成功，可以根据返回值来分析，如果返回值 lngRetCode 为正数，则表示失败，否则表示成功。参数 hRasConn 即为获得拨号连接的句柄。

同时注意的是，在拨号的时候也是分成针对两个系统来实现的，如果用户要开发特定系统下的 RAS 程序，就可以不用根据系统判断了。

在拨号参数传递的时候，注意一个参数 szEntryName 是不可省略的，szEntryName 就是我们通过程序或者拨号连接中的新建连接建立的电话簿名称，RasDial 必须通过一个电话簿来实现拨号，可以是任意的一个电话簿的名字。

在给拨号结构体赋值的时候注意，通常使用 API 函数 lstrcpy 而不使用 StrConv，因为使用 StrConv 函数通常会出错，希望读者注意。

最后要注意的是函数 AddConnection 返回的是一个 Connection 类，不是拨号句柄，主要是为了更好的管理一个连接而已。

在该类中，还有其他的一些函数，如删除一个连接，删除一个连接很简单，只需要挂断一个连接，然后刷新连接就可以了，刷新连接就是重新调用一遍遍历连接的函数 fcnRASEnumConnections 而已。

3. 类模块 PhoneEntry

该类的功能是保存一个电话簿的所有信息，以便用户方便地进行管理和编辑修改，因为

当你建立了一个电话簿后，总会有许多相关信息的，有许多信息是在拨号的时候要用的。

```
   '类模块 PhoneEntry
   Option Explicit
   '类内部变量
   Private strEntryName As String
   Private strPhoneNumber As String
   Private strCallbackNumber As String
   Private strUserName As String
   Private strPassword As String
   Private strDomain As String
   Private intIndex As Integer
   '标志密码是否获得变量
   Private lngGotPassword As Long

   Public Property Get EntryName() As String
   '获得电话簿名称
      EntryName = strEntryName
   End Property

   Public Property Let EntryName(strNewName As String)
   '设置电话簿名称
      Dim lngRetCode As Long
      '如果允许修改
      If boolAllowUpdate Then
         strEntryName = strNewName
         lngRetCode = fcnRasGetEntryDialParams()
         If lngRetCode Then
            Err.Raise vbObjectError + lngRetCode, "EntryName Set Failed", "RAS
Failure"
         End If
      Else
         lngRASErrorNumber = 1111
         strRASDescription = "Property Not Updateable"
         Err.Raise vbObjectError + 1111, "Property Not Updateable", "RAS Failure"
      End If
   End Property

   '对一个电话簿拨号
   Public Function DialEntry(boolAsync As Boolean) As Connection
```

```
    Set DialEntry = lpConnections.AddConnection(strEntryName, "", "", "", "",
"", boolAsync)
  End Function

  '编辑电话簿
  Public Sub EditEntry()
    Dim lngRetCode As Long

    '编辑电话簿，注意在 Windows NT 下会失败，但在 Windows 95/98 下面可以成功
    lngRetCode = RasEditPhonebookEntry(APINULL, vbNullString, strEntryName)
     If lngRetCode Then
       lngRASErrorNumber = lngRetCode
       strRASDescription = lpRASError.fcnRASErrorString()
       Err.Raise vbObjectError + lngRetCode, "EditEntry Method Failed", "RAS
Failure"
     End If
  End Sub

  '获得电话簿的各项参数
  Private Function fcnRasGetEntryDialParams() As Long
    Dim lngRetCode As Long
    Dim lngRetlstrcpy As Long
    '在 Windows NT 下会失败
    Dim lpRasDialparams As RASDIALPARAMS95

    '设置结构体
    lpRasDialparams.dwSize = 1052
    lngRetlstrcpy = lstrcpy(lpRasDialparams.szEntryName(0), strEntryName)
    lngRetCode   =   RasGetEntryDialParams(vbNullString,   lpRasDialparams,
lngGotPassword)
    Select Case lngRetCode
      Case SUCCESS
        '如果成功
        strUserName   =   fcnTrimNulls(StrConv(lpRasDialparams.szUserName,
vbUnicode))
        strPhoneNumber = fcnTrimNulls(StrConv(lpRasDialparams.szPhoneNumber,
vbUnicode))
        strCallbackNumber                                                =
fcnTrimNulls(StrConv(lpRasDialparams.szCallbackNumber, vbUnicode))
        If lngGotPassword = 1 Then
```

```
            strPassword  =  fcnTrimNulls(StrConv(lpRasDialparams.szPassword,
vbUnicode))
          Else
            strPassword = "Password Not Available"
          End If
          strDomain     =     fcnTrimNulls(StrConv(lpRasDialparams.szDomain,
vbUnicode))
          fcnRasGetEntryDialParams = 0
        Case NOT_SUPPORTED
          '如果是 NT 系统，设置为 "Not Available"
          strUserName = "Not Available"
          strPhoneNumber = "Not Available"
          strCallbackNumber = "Not Available"
          strPassword = "Not Available"
          strDomain = "Not Available"
          fcnRasGetEntryDialParams = 0
        Case Else
          '其他情况
          strUserName = "Not Available"
          strPhoneNumber = "Not Available"
          strCallbackNumber = "Not Available"
          strPassword = "Not Available"
          strDomain = "Not Available"
          lngRASErrorNumber = lngRetCode
          lngRASErrorNumber = lngRetCode
          strRASDescription = lpRASError.fcnRASErrorString()
          fcnRasGetEntryDialParams = lngRetCode
      End Select
   End Function

   '重新设置电话电话簿的各项参数
   Private Function fcnRasSetEntryDialParams() As Long

      Dim lngRetCode As Long
      Dim lngRetlstrcpy As Long
      Dim lngAttemptOrder As Long
      'Windows NT 下会失败
      Dim lpRasDialparams As RASDIALPARAMS95
```

```
      lpRasDialparams.dwSize = 1052
      lngRetlstrcpy = lstrcpy(lpRasDialparams.szEntryName(0), strEntryName)
      lngRetlstrcpy = lstrcpy(lpRasDialparams.szUserName(0), strUserName)
      lngRetlstrcpy = lstrcpy(lpRasDialparams.szDomain(0), strDomain)
      lngRetlstrcpy = lstrcpy(lpRasDialparams.szPhoneNumber(0), strPhoneNumber)
      lngRetlstrcpy       =       lstrcpy(lpRasDialparams.szCallbackNumber(0),
strCallbackNumber)
      '获得密码
      If lngGotPassword = 1 Then
        lngRetlstrcpy = lstrcpy(lpRasDialparams.szPassword(0), strPassword)
      Else
        lngRetlstrcpy = lstrcpy(lpRasDialparams.szPassword(0), "")
      End If
      lngRetCode = RasSetEntryDialParams(vbNullString, lpRasDialparams, 0&)
      Select Case lngRetCode
        Case SUCCESS
          strUserName    =    fcnTrimNulls(StrConv(lpRasDialparams.szUserName,
vbUnicode))
          strPhoneNumber = fcnTrimNulls(StrConv(lpRasDialparams.szPhoneNumber,
vbUnicode))
          strCallbackNumber                                                =
fcnTrimNulls(StrConv(lpRasDialparams.szCallbackNumber, vbUnicode))
          If lngGotPassword Then
            strPassword  =  fcnTrimNulls(StrConv(lpRasDialparams.szPassword,
vbUnicode))
          Else
            strPassword = "Not Available"
          End If
          strDomain     =     fcnTrimNulls(StrConv(lpRasDialparams.szDomain,
vbUnicode))
          fcnRasSetEntryDialParams = 0
        Case NOT_SUPPORTED
          'Windows NT 系统会失败
          strUserName = "Not Available"
          strPhoneNumber = "Not Available"
          strCallbackNumber = "Not Available"
          strPassword = "Not Available"
          strDomain = "Not Available"
          lngRASErrorNumber = lngRetCode
          strRASDescription = lpRASError.fcnRASErrorString()
```

```
            fcnRasSetEntryDialParams = lngRetCode
         Case Else
            '设置为原来的值
            lngAttemptOrder = fcnRasGetEntryDialParams()
            lngRASErrorNumber = lngRetCode
            lngRASErrorNumber = lngRetCode
            strRASDescription = lpRASError.fcnRASErrorString()
            fcnRasSetEntryDialParams = lngRetCode

      End Select
   End Function

   '获得电话号码
   Public Property Get PhoneNumber() As String
      PhoneNumber = strPhoneNumber
   End Property

   '设置电话号码
   Public Property Let PhoneNumber(strNewNumber As String)
      Dim lngRetCode As Long
      strPhoneNumber = strNewNumber
      lngRetCode = fcnRasSetEntryDialParams()
      If lngRetCode Then
         Err.Raise vbObjectError + lngRetCode, "PhoneNumber Set Failed", "RAS
Failure"
      End If
   End Property

   Public Property Get CallbackNumber() As String
      CallbackNumber = strCallbackNumber
   End Property

   '设置回拨电话号码
   Public Property Let CallbackNumber(strNewNumber As String)
      Dim lngRetCode As Long
      strCallbackNumber = strNewNumber
      lngRetCode = fcnRasSetEntryDialParams()
      If lngRetCode Then
         Err.Raise vbObjectError + lngRetCode, "CallbackNumber Set Failed", "RAS
```

```
Failure"
      End If
   End Property

   '得到用户名
   Public Property Get UserName() As String
      UserName = strUserName
   End Property

   '重新设置用户名
   Public Property Let UserName(strNewUser As String)
      Dim lngRetCode As Long
      strUserName = strNewUser
      lngRetCode = fcnRasSetEntryDialParams()
      If lngRetCode Then
         Err.Raise vbObjectError + lngRetCode, "UserName Set Failed", "RAS Failure"
      End If
   End Property

   '获得密码
   Public Property Get Password() As String
      Password = strPassword
   End Property

   '设置密码
   Public Property Let Password(strNewPassword As String)
      Dim lngRetCode As Long

      strPassword = strNewPassword
      lngRetCode = fcnRasSetEntryDialParams()
      If lngRetCode Then
         Err.Raise vbObjectError + lngRetCode, "Password Set Failed", "RAS Failure"
      End If
   End Property

   '得到域
   Public Property Get Domain() As String
      Domain = strDomain
   End Property
```

```
'设置域
Public Property Let Domain(strNewDomain As String)
   Dim lngRetCode As Long

   strDomain = strNewDomain
   lngRetCode = fcnRasSetEntryDialParams()
   If lngRetCode Then
      Err.Raise vbObjectError + lngRetCode, "Domain Set Failed", "RAS Failure"
   End If
End Property

'得到该电话簿的索引，因为可能有多个电话簿，为了方便管理而建立索引
Public Property Get Index() As Integer
   Index = intIndex
End Property

'设置索引
Public Property Let Index(intNewIndex As Integer)

   If boolAllowUpdate Then
      intIndex = intNewIndex
   Else
      lngRASErrorNumber = 1111
      strRASDescription = "Property Not Updateable"
      Err.Raise vbObjectError + 1111, "Property Not Updateable", "RAS Failure"
   End If
End Property
```

该类是记录一个电话簿的所有信息，如果是做一个完整的 RAS 程序，这部分是必要的，但是如果只是做一个简单的隐含式拨号程序，这个类可以省略了。该类主要是设置和获取一些属性，另外有一个比较重要的函数就是 fcnRasSetEntryDialParams，该函数调用了 RAS API 函数 lngRetCode = RasSetEntryDialParams(vbNullString, lpRasDialparams, 0&)来获得每一个电话簿的相关信息，并保存在类中。其中调用函数所得到的值保存在 lpRasDialparams 结构体中，因此获取信息的时候，直接读取这个结构体中的数据就可以了。

4. 类模块 PhoneEntries

该类模块的功能如果 Connections 类模块，是对所有的电话簿进行管理。通过创建该类，就可以在窗体上方便地进行操作。

```
'类模块 PhoneEntries
Option Explicit
```

```
    '*************************************************************************
**
    '电话簿的个数
    Private intPCount As Integer

    Public Property Get Count() As Integer
    '获得电话簿的个数
      Dim lngRetCode
      '通过函数 fcnRASEnumEntries 获得电话簿的个数
      lngRetCode = fcnRASEnumEntries()
      If lngRetCode Then
        Err.Raise vbObjectError + lngRetCode, "Connections.Count Failure", "RAS
Failure"
      Else
        Count = intPCount
      End If
    End Property

    '获得所有地电话簿
    Private Function fcnRASEnumEntries() As Long
      Dim lngRetCode As Long
      Dim lpszreserved As String
      Dim lpszPhonebook As String
      Dim lpcb As Long
      Dim lpcEntries As Long
      Dim intArraySize As Integer
      Dim intLooper As Long

      lpszreserved = vbNullString
      lpszPhonebook = vbNullString
      '定义最大 256 个电话簿，如果有必要可以增加
      intArraySize = 255
      If lngWindowVersion = 2 Then
        'Windows NT 系统
        '定义结构体数组
        ReDim lprasentryname(intArraySize) As RASENTRYNAME
        lprasentryname(0).dwSize = 28
        lpcb = 256 * lprasentryname(0).dwSize
        lngRetCode     =     RasEnumEntries(lpszreserved,     lpszPhonebook,
lprasentryname(0), lpcb, lpcEntries)
```

```
      Else
         'Windows 95/98 操作系统 (lngWindowVersion =1)
         ReDim lprasentryname95(intArraySize) As RASENTRYNAME95
         lprasentryname95(0).dwSize = 264
         lpcb = 256 * lprasentryname95(0).dwSize
         lngRetCode     =     RasEnumEntries(lpszreserved,     lpszPhonebook,
lprasentryname95(0), lpcb, lpcEntries)
      End If
      Select Case lngRetCode
         Case SUCCESS
            If lpcEntries > 0 Then
               '定义数组大小以获得电话簿的个数
               ReDim arrPEntry(lpcEntries - 1) As PhoneEntry
               If lngWindowVersion = 2 Then
                  'Windows NT 操作系统
                  For intLooper = 0 To UBound(arrPEntry())
                     Set arrPEntry(intLooper) = New PhoneEntry
                     '允许直接在树视图里面修改属性
                     boolAllowUpdate = True
                     arrPEntry(intLooper).EntryName                              =
fcnTrimNulls(StrConv(lprasentryname(intLooper).szEntryName, vbUnicode))
                     arrPEntry(intLooper).Index = intLooper
                     boolAllowUpdate = False
                  Next intLooper
               Else
                  ' Windows 95/98 操作系统
                  For intLooper = 0 To UBound(arrPEntry())
                     Set arrPEntry(intLooper) = New PhoneEntry
                     '允许修改属性
                     boolAllowUpdate = True
                     arrPEntry(intLooper).EntryName                              =
fcnTrimNulls(StrConv(lprasentryname95(intLooper).szEntryName, vbUnicode))
                     arrPEntry(intLooper).Index = intLooper
                     boolAllowUpdate = False
                  Next intLooper
               End If
            End If
            intPCount = CInt(lpcEntries)
            fcnRASEnumEntries = 0
```

```
        Case ERROR_BUFFER_TOO_SMALL
           '如果缓冲区太小，则扩大缓冲区
           If lngWindowVersion = 2 Then
              'running NT
              intArraySize = lpcb / lprasentryname(0).dwSize
           Else
              'running 95
              intArraySize = lpcb / lprasentryname95(0).dwSize
           End If
        Case Else
           lngRASErrorNumber = lngRetCode
           strRASDescription = lpRASError.fcnRASErrorString()
           fcnRASEnumEntries = lngRetCode
     End Select
  End Function

  '类初始化事件
  Private Sub Class_Initialize()
     Dim lngSuccess As Long
     '类初始化的时候获得所有的电话簿
     lngSuccess = fcnRASEnumEntries()
  End Sub

  '增加电话簿
  Public Sub AddEntry()
     Dim lngRetCode As Long

     '通过 RasCreatePhonebookEntry 函数增加电话簿
     lngRetCode = RasCreatePhonebookEntry(APINULL, "")
        If lngRetCode Then
           lngRASErrorNumber = lngRetCode
           strRASDescription = lpRASError.fcnRASErrorString()
           Err.Raise vbObjectError + lngRetCode, "PhoneEntries AddEntry Failed",
"RAS Failure"
        Else
           '如果成功，则重新刷新电话簿树视图
           lngRetCode = fcnRASEnumEntries()
           If lngRetCode Then Err.Raise vbObjectError + lngRetCode, "PhoneEntries
AddEntry Failed", "RAS Failure"
        End If
```

```
End Sub
```

该类中有两个函数有必要解释一下，其中第一个函数就是 fcnRASEnumEntries，因为在一个机器里面，可能每个用户都建立了自己的电话簿。为了对这些电话簿进行统一的管理和查看，就有必要罗列出所有的电话簿。FcnRASEnumEntries 函数中调用了 RAS API 函数来实现，具体格式如下：

```
lngRetCode = RasEnumEntries(lpszreserved, lpszPhonebook, lprasentryname95(0),
lpcb, lpcEntries)
```

其中是函数调用成功返回的缓冲区大小，如果开始定义的 lprasentryname95 结构体数组不够大，可能产生错误，可以利用该参数重新设置数组大小。lpcEntries 参数为函数调用成功返回的电话簿的数量。lpszPhonebook 在 Windows NT 系统和 Windows 98 系统中都被设置成空，在 98 系统中没有具体的电话簿文件，因为电话簿信息保存在注册表中，而在 NT 中如果设置为空，表示的是默认的路径。

还有一个函数就是增加电话簿了，不过这里增加电话簿时调用了系统的功能来增加地址簿，要弹出新建向导，如果读者不想使用该功能，可以使用其他的函数来隐含式地建立。这里增加电话簿的函数为 AddEntry，其中调用了 RAS API 函数 RasCreatePhonebookEntry，具体格式如下：

```
lngRetCode = RasCreatePhonebookEntry(APINULL, "")
```

该函数会弹出建立向导，具体可以参见前面的程序运行结果图。

5. 类模块 RASError

该类的作用是为了能够让程序顺利地运行，而进行错误代码分析。读者如果要开发一个完整的 RAS 程序，是需要增加这部分内容的。

```
'类模块 RASError
Option Explicit
'该类用于错误处理
'*************************************************************************
***************
Public Function fcnRASErrorString() As String

   Dim lngRetCode As Long
   '所有的错误描述一般都不超过 256 个字节
   Dim strRASErrorString As String

   '首先创建一个空字符串
   strRASErrorString = Space$(256)
   '获得拨号产生的错误码
   Select Case lngRASErrorNumber
      Case Is >= 600
         lngRetCode = RasGetErrorString(lngRASErrorNumber, strRASErrorString,
256&)
         If lngRetCode Then
```

```
            '通常不会出现这种情况
            fcnRASErrorString = "Call To RasGetErrorString Failed. Error
Retrieving The Error String!!!"
         Else
            '返回错误描述
            fcnRASErrorString = strRASErrorString
         End If
      Case NOT_SUPPORTED
         fcnRASErrorString = "Function Is Not Supported On This Version of
Windows."
      Case Else
         fcnRASErrorString = "Unexpected Error. See WIN32 SDK More Information
On This Error Number!!!"
   End Select
End Function
```

在这个类中主要就是这个函数，主要是通过调用函数 RasGetErrorString **根据错误代码来获得具体的错误描述。**

6. **类模块** RASEngine

该类模块是核心模块，以上介绍的 5 **个类模块基本上都需要在这个类模块中被引用，因此在介绍的时候我们首先介绍了前面几个类，然后介绍这个类。该类管理并根据需要调用其他的几个类的方法和属性。**

```
Option Explicit
'该类是本程序的主类
'通过这个类调用其他的几个类

'*************************************************************************
***************
'定义结构体，获得系统信息
Private lpOSInfo As OSVERSIONINFO
'定义变量来保存系统的版本
Private sngOSVersion As Single
'保存拨号的创建的对象，也就是如果有多个拨号连接，则保存连接个数
Private hRASDLLInstance As Long

Private Sub Class_Initialize()
   '在类初始化的时候获得系统版本
   Dim lngErrNum As Long
   Dim lngbugfix As Long
```

```
      '创建几个类实例
      Set lpRASError = New RASError
      lngbugfix = lpRASError.ErrorNumber
      Set lpPhoneEntries = New PhoneEntries
      lngbugfix = lpPhoneEntries.Count
      Set lpConnections = New Connections
      lngbugfix = lpConnections.Count
      '初始化 RAS，该函数是装入 RAS 动态库
      lngErrNum = fcnLoadandCheckRAS()
      If lngErrNum Then
         '如果不为 0，表示错误
         Err.Raise vbObjectError + 1911, "RAS.RASEngine", "RAS Could Not Be
Initialized. WIN32 Error: " & Str$(lngErrNum)
      Else
         '否则表示已经将该 RAS 载入
         '设置结构体字段值
         lpOSInfo.dwOSVersionInfoSize = 148
         If (GetVersionEx(lpOSInfo)) Then
            '获得系统版本，并保存在一个全局变量中
            lngWindowVersion = lpOSInfo.dwPlatformId
            '合并两个字段来表示系统版本名
            sngOSVersion = CSng(lpOSInfo.dwMajorVersion) + CSng(Val("." &
Str$(lpOSInfo.dwMinorVersion)))
         Else
            lngErrNum = GetLastError()
            '错误处理
            Err.Raise vbObjectError + 1912, "RAS.RASEngine", "GetVersionEx Failed
With WIN32 Error: " & Str$(lngErrNum)
         End If
      End If
   End Sub

   Public Property Get OSVersion() As Single
      '获得系统版本
      OSVersion = sngOSVersion
   End Property

   '获得系统版本号
   Public Property Get OSBuildNumber() As Long
```

```
   OSBuildNumber = lpOSInfo.dwBuildNumber
End Property

'获得系统的类型
Public Property Get OSType() As Long
   'WIN32s = 0
   'WIN 95 = 1
   'WIN NT = 2
   OSType = lpOSInfo.dwPlatformId
End Property

'返回一个错误处理类
Public Function RASError() As RASError
   'set up a new pointer to Error object
   Set RASError = lpRASError
End Function

'该函数返回一个连接或者连接集合
Public Function Connections(Optional ByVal Index As Variant) As Object

   If IsMissing(Index) Then
      '如果没有指定 index，则返回整个 connections 集合
      Set Connections = lpConnections
   Else
      '返回一个连接
      Set Connections = arrConnection(Index)
   End If
End Function

'返回电话簿或者所有电话簿
Public Function PhoneEntries(Optional ByVal Index As Variant) As Object
   If IsMissing(Index) Then
      '如果省略 index，获得电话簿集
      Set PhoneEntries = lpPhoneEntries
   Else
      '获得某一个电话簿
      Set PhoneEntries = arrPEntry(Index)
   End If
End Function
```

```
'该函数调用 API 函数 LoadLibrary 载入 RAS 动态库
Private Function fcnLoadandCheckRAS() As Long
   '载入动态连接库
   hRASDLLInstance = LoadLibrary("RASAPI32.DLL")
   If hRASDLLInstance Then
      fcnLoadandCheckRAS = 0
   Else
      fcnLoadandCheckRAS = GetLastError()
   End If
End Function

'该函数调用 API 函数 FreeLibrary 来卸载动态库资源
Private Function fcnUnloadRAS() As Long
   Dim lngRetCode As Long

   '释放 RAS 动态库
   lngRetCode = FreeLibrary(hRASDLLInstance)
   If lngRetCode Then
      fcnUnloadRAS = 0
   Else
      fcnUnloadRAS = GetLastError()
   End If
End Function

Private Sub Class_Terminate()
'类终止事件
   Dim lngErrNum As Long
   Dim intLooper As Long

   '终止所有的活动连接
   On Error Resume Next
   For intLooper = 0 To lpConnections.Count - 1
      lpConnections.RemoveConnection (intLooper)
   Next intLooper
   On Error GoTo 0

   '释放类资源
   Set lpPhoneEntries = Nothing
   Set lpConnections = Nothing
```

```
    Set lpRASError = Nothing
    lngErrNum = fcnUnloadRAS()
    If lngErrNum Then Err.Raise vbObjectError + 1913, "RAS.RASEngine", "RAS was
not properly uninitialized. WIN32 Error: " & Str$(lngErrNum)

End Sub
```

该类为主类，但是没有具体的功能，只是在开始的时候生了其他几个类的实例并且调用了其他相关的函数。注意在类开始的时候调用了 API **函数** hRASDLLInstance = LoadLibrary("RASAPI32.DLL")**和** lngRetCode = FreeLibrary(hRASDLLInstance)**，主要是为了加载** RAS API **动态库的，好像不使用也可以，读者可以自己试试。**

注意在类中止的时候，要释放所有的资源。

7. **窗体** frmRASMAIN

上面介绍了很多的类，但归根结底还是要在窗体中调用。该窗体是主窗体，当整个程序启动的时候，就是从这个窗体开始启动。如果想在其他的程序中使用本程序中的所有类，可以编译成动态链接库，然后在其他程序里面使用就可以了。

```
'窗体 frmRASMAIN
Public objRASEng As RASEngine
'增加连接按钮单击事件
Private Sub cmdAddConnection_Click()
   Dim boolSuccess As Boolean
   '弹出 frmCreateConn 对话框
   frmCreateConn.Show 1
   '刷新连接树视图
   boolSuccess = fcnRefreshConnectionTree()
End Sub

'增加电话簿按钮单击事件
Private Sub cmdAddEntry_Click()
'增加电话簿
   On Error GoTo errhand
      '通过类 RASEngine 来增加电话簿
      objRASEng.PhoneEntries.AddEntry
      boolSuccess = fcnRefreshPhonebookTree()
   On Error GoTo 0
   Exit Sub
errhand:
   Call fcnErrorHandler
   Resume Next
End Sub
```

```
   Private Sub cmdDialNumber_Click()
   '该事件拨号
      Dim boolSuccess As Boolean
      Dim intTempIndex As Integer
      Dim objRASConn As Connection

      On Error GoTo errhand
      '至于拨号的内部细节请参看各个类的具体描述
      intTempIndex      =      CInt(Val(Right$(trvPhonebook.SelectedItem.Key,
Len(trvPhonebook.SelectedItem.Key) - 3)))
      Set                              objRASConn                              =
objRASEng.PhoneEntries(CVar(intTempIndex)).DialEntry(chkDialAsync.Value)
      boolSuccess = fcnRefreshConnectionTree()
      On Error GoTo 0
      Exit Sub
   errhand:
      Call fcnErrorHandler
      Resume Next
   End Sub

   Private Sub cmdEditEntry_Click()
   '编辑电话簿
      Dim boolSuccess As Boolean
      Dim intTempIndex As Integer

      On Error GoTo errhand
      intTempIndex      =      CInt(Val(Right$(trvPhonebook.SelectedItem.Key,
Len(trvPhonebook.SelectedItem.Key) - 3)))
      objRASEng.PhoneEntries(CVar(intTempIndex)).EditEntry
      '在树视图中，打开当前要编辑的电话簿
      trvPhonebook.SelectedItem.Text                                          =
objRASEng.PhoneEntries(CVar(intTempIndex)).EditEntry
      On Error GoTo 0
      Exit Sub
   errhand:
      Call fcnErrorHandler
      Exit Sub
   End Sub
```

```
   Private Sub cmdGetWinVer_Click()
   ’得到操作系统信息
      Select Case objRASEng.OSType
         Case 2 ’VER_PLATFORM_WIN32_NT
            MsgBox "Running: Windows NT" & vbCrLf & "Version: " & Str$(objRASEng.OSVersion) & vbCrLf & "Build: " & Str$(objRASEng.OSBuildNumber), , "Operating System Information"
         Case 1 ’VER_PLATFORM_WIN32_WINDOWS
            MsgBox "Running: Windows 95" & vbCrLf & "Version: " & Str$(objRASEng.OSVersion) & vbCrLf & "Build: " & Str$(objRASEng.OSBuildNumber), , "Operating System Information"
         Case 0 ’VER_PLATFORM_WIN32s
            MsgBox "Running: Windows 3.1 with WIN32s" & vbCrLf & "Version: " & Str$(objRASEng.OSVersion) & vbCrLf & "Build: " & Str$(objRASEng.OSBuildNumber), vbCritical, "Warning: Not A Supported OS"
         Case Else ’其他未知操作系统
            MsgBox "Running: Unknown OS" & vbCrLf & "Version: " & Str$(objRASEng.OSVersion) & vbCrLf & "Build: " & Str$(objRASEng.OSBuildNumber), vbCritical, "Warning: Not A Supported OS"
      End Select
   End Sub

   ’刷新树视图，以便查看最新的电话簿信息
   Private Sub cmdRefresh_Click()
      Dim boolSuccess As Boolean
      ’刷新树视图
      On Error GoTo errhand
      boolSuccess = fcnRefreshPhonebookTree()
      On Error GoTo 0
      Exit Sub
   errhand:
      Call fcnErrorHandler
      Resume Next
   End Sub

   ’刷新连接视图，以便查看最新的连接消息
   Private Sub cmdRefreshConnTree_Click()
      Dim boolSuccess As Boolean
```

```
      '刷新连接视图,通过函数 fcnRefreshConnectionTree
      On Error GoTo errhand
      boolSuccess = fcnRefreshConnectionTree()
      On Error GoTo 0
      Exit Sub
   errhand:
      Call fcnErrorHandler
      Resume Next
   End Sub

   Private Sub cmdRemoveConnection_Click()
      Dim boolSuccess As Boolean
      '删除被选中的连接
      Select Case trvConnections.SelectedItem.Image
         Case 2, 3
            On Error GoTo errhand
            objRASEng.Connections.RemoveConnection
CInt(Right$(trvConnections.SelectedItem.Key,
Len(trvConnections.SelectedItem.Key) - 4))
            '为了能够使计算机能够响应关闭连接的事件，调用 DoEvents
            DoEvents
            boolSuccess = fcnRefreshConnectionTree()
            On Error GoTo 0
         Case Else
            '不做处理
      End Select
      Exit Sub
   errhand:
      Call fcnErrorHandler
      Exit Sub
   End Sub

   '提交电话簿的修改
   Private Sub cmdUpdateEntry_Click()
      trvPhonebook.StartLabelEdit
   End Sub

   Private Function fcnRefreshPhonebookTree() As Boolean
   '刷新视图的函数
```

```
    Dim intEntryLooper As Integer
    Dim strTemp As String

    fcnRefreshPhonebookTree = False
    '首先删除所有的树节点
    trvPhonebook.Nodes.Clear
    '建立起始节点
      trvPhonebook.Nodes.Add , , "PHTopNode", "电话簿", 1, 1
    '首先按照字符顺序 A、B、C....Z 等顺序建立字节点，为了方便用户寻找不同的电话簿，电话簿如
    '果名字以 A 开头，则可以在树视图 A 的下面查看
    For intEntryLooper = 65 To 90
    '循环建立
    trvPhonebook.Nodes.Add  "PHTopNode",  tvwChild,  Chr$(intEntryLooper), Chr$(intEntryLooper), 2, 3
    Next intEntryLooper
    '增加其他电话簿，如果名字不是以字符开头，则列在该项下面
    trvPhonebook.Nodes.Add "PHTopNode", tvwChild, "NumericNode", "其他电话簿", 2, 3
    '填充整个视图，首先遍历计算机中所有的电话簿，并在相关的字节点下建立相关子节点
    For intEntryLooper = 0 To objRASEng.PhoneEntries.Count - 1
    '循环得到每一个电话簿
        '所有电话簿的信息都是从 PhoneEntries 类中得到的
        With objRASEng.PhoneEntries(CVar(intEntryLooper))
          '得到电话簿的名称
          strTemp = .EntryName
          If Asc(UCase$(Left$(strTemp, 1))) >= 65 And Asc(UCase$(Left$(strTemp, 1))) <= 90 Then
          '分析电话簿的名称首字符，以便加入相关的字节点中
          '把每个电话簿的相关信息如电话号码、用户名、用户密码等加入
            trvPhonebook.Nodes.Add  UCase$(Left$(strTemp,  1)),  tvwChild, "key" & intEntryLooper, strTemp, 2, 3
            trvPhonebook.Nodes.Add "key" & intEntryLooper, tvwChild, "phn" & intEntryLooper, "Phone Number", 7, 8
            trvPhonebook.Nodes.Add "phn" & intEntryLooper, tvwChild, "phonen" & intEntryLooper, .PhoneNumber, 4, 4
            trvPhonebook.Nodes.Add "key" & intEntryLooper, tvwChild, "cbn" & intEntryLooper, "Callback Number", 7, 8
            trvPhonebook.Nodes.Add "cbn" & intEntryLooper, tvwChild, "callba" & intEntryLooper, .CallbackNumber, 4, 4
```

```
                trvPhonebook.Nodes.Add "key" & intEntryLooper, tvwChild, "usn"
& intEntryLooper, "User Name", 7, 8
                trvPhonebook.Nodes.Add "usn" & intEntryLooper, tvwChild, "userna"
& intEntryLooper, .UserName, 4, 4
                trvPhonebook.Nodes.Add "key" & intEntryLooper, tvwChild, "pwd"
& intEntryLooper, "Password", 7, 8
                trvPhonebook.Nodes.Add "pwd" & intEntryLooper, tvwChild, "passwo"
& intEntryLooper, .Password, 4, 4
                trvPhonebook.Nodes.Add "key" & intEntryLooper, tvwChild, "dmn"
& intEntryLooper, "Domain", 7, 8
                trvPhonebook.Nodes.Add "dmn" & intEntryLooper, tvwChild, "domain"
& intEntryLooper, .Domain, 4, 4
             Else '如果是以数字或者其他字符开始
                '程序省略，因为结构同上
             End If
          End With
      Next intEntryLooper
      '设置相关控件的属性
      trvPhonebook.Nodes(2).EnsureVisible
      cmdEditEntry.Enabled = False
      cmdDialNumber.Enabled = False
      cmdUpdateEntry.Enabled = False
      fcnRefreshPhonebookTree = True
   End Function

   '窗体启动
   Private Sub Form_Load()
      Dim boolSuccess As Boolean
      On Error GoTo errhand
         frmRASMAIN.MousePointer = 11
         '程序启动的时候，创建一个主类 RASEngine
         Set objRASEng = New RASEngine
         '将视图控件和图象列表控件相关
         trvPhonebook.ImageList = imgTreeIcons
         trvConnections.ImageList = imgTreeIcons
         '刷新视图
         boolSuccess = fcnRefreshPhonebookTree()
         boolSuccess = fcnRefreshConnectionTree()
         frmRASMAIN.MousePointer = 0
```

```
      On Error GoTo 0
      Exit Sub
   errhand:
      Call fcnErrorHandler
      Unload frmmain
      Resume Next
   End Sub

   '电话簿树形控件节点单击事件
   Private Sub trvConnections_NodeClick(ByVal Node As Node)
      '根据选中的节点来使得删除按钮是否有效
       Select Case Node.Image
         Case 2, 3
            cmdRemoveConnection.Enabled = True
         Case Else
            cmdRemoveConnection.Enabled = False
       End Select
   End Sub

   Private Sub trvPhonebook_AfterLabelEdit(Cancel As Integer, NewString As String)
   '如果直接在树形视图中修改属性，则需要将新的属性提交
   Dim boolSuccess As Boolean
      Dim intTempIndex As Integer
      On Error GoTo errhand
        Select Case Left$(trvPhonebook.SelectedItem.Key, 6)
           Case "callba"
              intTempIndex   =   CInt(Val(Right$(trvPhonebook.SelectedItem.Key,
Len(trvPhonebook.SelectedItem.Key) - 6)))
              objRASEng.PhoneEntries(CVar(intTempIndex)).CallbackNumber      =
NewString
              trvPhonebook.SelectedItem.Text                                 =
objRASEng.PhoneEntries(CVar(intTempIndex)).CallbackNumber
           Case "phonen"
              intTempIndex   =   CInt(Val(Right$(trvPhonebook.SelectedItem.Key,
Len(trvPhonebook.SelectedItem.Key) - 6)))
              objRASEng.PhoneEntries(CVar(intTempIndex)).PhoneNumber         =
NewString
              trvPhonebook.SelectedItem.Text                                 =
objRASEng.PhoneEntries(CVar(intTempIndex)).PhoneNumber
           Case "userna"
```

```
            intTempIndex   =   CInt(Val(Right$(trvPhonebook.SelectedItem.Key,
Len(trvPhonebook.SelectedItem.Key) - 6)))
            objRASEng.PhoneEntries(CVar(intTempIndex)).UserName = NewString
            trvPhonebook.SelectedItem.Text                                    =
objRASEng.PhoneEntries(CVar(intTempIndex)).UserName
         Case "passwo"
            intTempIndex   =   CInt(Val(Right$(trvPhonebook.SelectedItem.Key,
Len(trvPhonebook.SelectedItem.Key) - 6)))
            objRASEng.PhoneEntries(CVar(intTempIndex)).Password = NewString
            trvPhonebook.SelectedItem.Text                                    =
objRASEng.PhoneEntries(CVar(intTempIndex)).Password
         Case "domain"
            intTempIndex   =   CInt(Val(Right$(trvPhonebook.SelectedItem.Key,
Len(trvPhonebook.SelectedItem.Key) - 6)))
            objRASEng.PhoneEntries(CVar(intTempIndex)).Domain = NewString
            trvPhonebook.SelectedItem.Text                                    =
objRASEng.PhoneEntries(CVar(intTempIndex)).Domain
         Case Else
            Cancel = True
      End Select
   On Error GoTo 0
   Exit Sub
errhand:
   Call fcnErrorHandler
   Cancel = True
   Resume Next
End Sub

Private Sub trvPhonebook_NodeClick(ByVal Node As Node)
   '根据不同的节点级别来设置是否能够更改
   Select Case Node.Image
      Case 2, 3
         '如果节点关键字前 3 位为 key，则设置按钮的相关属性
         If Left$(Node.Key, 3) = "key" Then
            cmdEditEntry.Enabled = True
            cmdDialNumber.Enabled = True
            cmdUpdateEntry.Enabled = False
         Else
            cmdEditEntry.Enabled = False
```

```
            cmdDialNumber.Enabled = False
            cmdUpdateEntry.Enabled = False
         End If
      Case 4
         '改变属性，属于下一级节点
         cmdEditEntry.Enabled = False
         cmdDialNumber.Enabled = False
         cmdUpdateEntry.Enabled = True
      Case Else
         cmdEditEntry.Enabled = False
         cmdDialNumber.Enabled = False
         cmdUpdateEntry.Enabled = False
    End Select
End Sub

Private Function fcnRefreshConnectionTree() As Boolean
'刷新连接树视图
   Dim intConnLooper As Integer
   fcnRefreshConnectionTree = False
   '首先删除树视图所有节点
   trvConnections.Nodes.Clear
   '设置顶节点
   trvConnections.Nodes.Add , , "CTopNode", "我的连接", 5, 5
   '显示所有的连接
   For intConnLooper = 0 To objRASEng.Connections.Count - 1
        With objRASEng.Connections(CVar(intEntryLooper))
           '建立连接节点
           trvConnections.Nodes.Add   "CTopNode",   tvwChild,   "conn"   &
intConnLooper, "Connection " & intConnLooper, 2, 3
           '设定节点的名称和值
           trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "entry"
& intConnLooper, "Entry Name", 7, 8
           trvConnections.Nodes.Add  "entry"  &  intConnLooper,  tvwChild,
"entryval" & intConnLooper, .EntryName, 6, 6
           trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "hconn"
& intConnLooper, "Connection Handle", 7, 8
           trvConnections.Nodes.Add  "hconn"  &  intConnLooper,  tvwChild,
"hconnval" & intConnLooper, Str$(.hRasConn), 6, 6
           trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "devnm"
& intConnLooper, "Device Name", 7, 8
```

```
            trvConnections.Nodes.Add  "devnm"  &  intConnLooper,  tvwChild,
"devnmval" & intConnLooper, .DeviceName, 6, 6
            trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "devtp"
& intConnLooper, "Device Type", 7, 8
            trvConnections.Nodes.Add  "devtp"  &  intConnLooper,  tvwChild,
"devtpval" & intConnLooper, .DeviceType, 6, 6
            trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "state"
& intConnLooper, "Connection State", 7, 8
            trvConnections.Nodes.Add  "state"  &  intConnLooper,  tvwChild,
"stateval" & intConnLooper, Str$(.State), 6, 6
            trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "wksnm"
& intConnLooper, "Workstation Name", 7, 8
            trvConnections.Nodes.Add  "wksnm"  &  intConnLooper,  tvwChild,
"wksnmval" & intConnLooper, .WorkstationName, 6, 6
            trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "IPadd"
& intConnLooper, "IP Addess", 7, 8
            trvConnections.Nodes.Add  "IPadd"  &  intConnLooper,  tvwChild,
"IPaddval" & intConnLooper, .IPAddress, 6, 6
            trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "IPXad"
& intConnLooper, "IPX Address", 7, 8
            trvConnections.Nodes.Add  "IPXad"  &  intConnLooper,  tvwChild,
"IPXadval" & intConnLooper, .IPXAddress, 6, 6
            trvConnections.Nodes.Add "conn" & intConnLooper, tvwChild, "hlana"
& intConnLooper, "LANA", 7, 8
            trvConnections.Nodes.Add  "hlana"  &  intConnLooper,  tvwChild,
"hlanaval" & intConnLooper, Str(CInt(.LANA)), 6, 6
          End With
      Next intConnLooper
      trvConnections.Nodes(1).EnsureVisible
      If trvConnections.Nodes(1).Children Then
        trvConnections.Nodes(2).EnsureVisible
      End If
      fcnRefreshConnectionTree = True
      cmdRemoveConnection.Enabled = False
   End Function

   Public Sub fcnErrorHandler()
   '错误处理
      Select Case Err.Number
```

```
        'RAS 连接错误
        Case vbObjectError + 600 To vbObjectError + 750
          MsgBox  objRASEng.RASError.Description,  vbCritical,  "Error:  "  &
objRASEng.RASError.ErrorNumber
          objRASEng.RASError.Clear
        'RAS WIN32 错误
        Case vbObjectError + 120
          MsgBox  objRASEng.RASError.Description,  vbCritical,  "Error:  "  &
objRASEng.RASError.ErrorNumber
          objRASEng.RASError.Clear
        'RAS 初始化错误
        Case vbObjectError + 1911
          MsgBox "Ras is not Properly Configured on this machine", vbCritical,
"Error: 1911"
        'RAS 不能获得版本错误
        Case vbObjectError + 1912
          MsgBox "Could not determine Windows version on this machine", vbCritical,
"Error: 1912"
        '不能创建 RAS 实例
        Case vbObjectError + 1913
          MsgBox "RAS was not properly deinitialized", vbCritical, "Error: 1913"
        Case vbObjectError + 6
          MsgBox objRASEng.RASError.Description & vbCrLf & "Often The Result Of
An Invalid Connection Handle" & vbCrLf & "Refresh Connections And Try Again If
Necessary", vbCritical, "Error: " & objRASEng.RASError.ErrorNumber
          objRASEng.RASError.Clear
        '其他以外错误
        Case Else
          MsgBox Err.Description, vbCritical, "Error: " & Err.Number
      End Select
   End Sub
```

该类主要是对一些基本控件的使用以及使用上面的一些类，解释比较全，不再多说。

8. **窗体** frmCreateConn

该窗体就是创建一个新的连接，用户只要输入相关的一些必须字段，就可以进行同步或者异步的拨号了。

```
   '窗体 frmCreateCon
   Option Explicit
   '*****************************************
   '该窗体的功能是新建一个拨号连接
   '*****************************************
```

```
Private Sub cmdDial_Click()
   '创建一个 Connection 类
   Dim objNewConnection As Connection
     On Error GoTo errhand
   '调用主窗体的 Connections 类对象的 AddConnection 来创建一个拨号
   Set                          objNewConnection                          =
frmRASMAIN.objRASEng.Connections.AddConnection(txtEntryName.Text,
txtPhoneNumber.Text, txtCallbackNumber.Text, txtUserName.Text, txtPassword.Text,
txtDomain.Text, chkAsync.Value)
   '卸载窗体
   Unload frmCreateConn
   On Error GoTo 0
   Exit Sub
errhand:
   Call frmRASMAIN.fcnErrorHandler
   Resume Next
End Sub

Private Sub Form_Load()
'调整窗体位置
frmCreateConn.Top    =    frmRASMAIN.Top    +    frmRASMAIN.Height    -
frmRASMAIN.ScaleHeight
frmCreateConn.Left    =    frmRASMAIN.Left    +    frmRASMAIN.Width    -
frmRASMAIN.ScaleWidth
End Sub
```

该窗体是要输入拨号的一些属性设置，比如电话簿（一定要填一个已经建立的电话簿）、电话号码、用户名、密码等。然后选择是同步还是异步拨号。

9. **窗体** frmAsyncDial

该窗体的作用就是在进行异步拨号的时候，对连接的各种状态进行连接。

```
Option Explicit
Private hRasConn As Long

'关闭窗体按钮事件
Private Sub cmdClose_Click()
   Unload frmAsyncDial
End Sub

'挂断连接按钮事件
Private Sub cmdHangUp_Click()
```

```
    Dim lngRetCode As Long
    '挂断连接，调用函数 RasHangUp
    lngRetCode = RasHangUp(hRasConn)
    '卸载窗体
    Unload frmAsyncDial
End Sub

Private Sub Form_Activate()
    '改变鼠标形状
    Screen.MousePointer = vbArrowHourglass
    '是关闭按钮无效
    cmdClose.Enabled = False
    '设置文本框属性
    txtCallStatus.FontBold = False
    txtCallStatus.ForeColor = 0 'black

    '获得正在连接的句柄，注意是通过窗体的 Tag 属性传递的
    hRasConn = "&H" & frmAsyncDial.Tag
    '设置时钟控件可用
    tmrGetConnStatus.Enabled = True
End Sub

Private Sub Form_Load()
    '设定窗体位置
    frmAsyncDial.Top = ((Screen.Height / 2) - (frmAsyncDial.Height / 2))
    frmAsyncDial.Left = ((Screen.Width / 2) - (frmAsyncDial.Width / 2))
End Sub

Private Sub Form_Unload(Cancel As Integer)
    '停止时钟
    tmrGetConnStatus.Enabled = False
    '改变鼠标形状为默认
    Screen.MousePointer = vbDefault
End Sub

Private Sub tmrGetConnStatus_Timer()
'该时钟控件处理函数实现的是获得异步拨号时的各种状态
  Dim lngRetCode As Long
  Dim lngRASConnState As Long
  Dim lngRASError As Long
```

```
If lngWindowVersion = 2 Then
   '如果使用的是 Windows NT 系统
   Dim lpRASCONNSTATUS As RASCONNSTATUS
   lpRASCONNSTATUS.dwSize = 64
   '调用 RAS API 来获得各种状态
   lngRetCode = RasGetConnectStatus(hRasConn, lpRASCONNSTATUS)
   If lngRetCode Then
      '如果返回值不等于 0，表示有错误
      lngRASErrorNumber = lngRetCode
      '改变字体颜色
      txtCallStatus.FontBold = True
      txtCallStatus.ForeColor = RGB(255, 0, 0)
      txtCallStatus.Text = lpRASError.fcnRASErrorString()
      '挂断连接
      lngRetCode = RasHangUp(hRasConn)
      '允许用户退出窗体
      cmdClose.Enabled = True
      '中止时钟控件
      tmrGetConnStatus.Enabled = False
      lngRASError = 10000
   Else
      '如果成功
      '通过结构体获得状态
      lngRASConnState = lpRASCONNSTATUS.rasconnstate
      lngRASError = lpRASCONNSTATUS.dwError
   End If
Else
   '使用的是 Windows 95/98 系统
   Dim lpRASCONNSTATUS95 As RASCONNSTATUS95
   lpRASCONNSTATUS95.dwSize = 160
   lngRetCode = RasGetConnectStatus(hRasConn, lpRASCONNSTATUS95)
   If lngRetCode Then
      '返回值不为 0，获得错误码
      lngRASErrorNumber = lngRetCode
      '设置文本颜色
      txtCallStatus.FontBold = True
      txtCallStatus.ForeColor = RGB(255, 0, 0)
      txtCallStatus.Text = lpRASError.fcnRASErrorString()
```

```
        lngRetCode = RasHangUp(hRasConn)
        '使关闭按钮有效
        cmdClose.Enabled = True
        '使时钟控件无效
        tmrGetConnStatus.Enabled = False
        lngRASError = 10000
    Else
        '如果成功，则获得状态
        lngRASConnState = lpRASCONNSTATUS95.rasconnstate
        lngRASError = lpRASCONNSTATUS95.dwError
    End If
End If

'如果有错误，则显示错误，否则显示正确的连接状态
Select Case lngRASError
    Case SUCCESS, PENDING
        'Update connection
        Select Case lngRASConnState
            Case RASCS_OpenPort
                txtCallStatus = "Attempting To Open Port..."
            Case RASCS_PortOpened
                txtCallStatus = "Port Successfully Opened"
            Case RASCS_ConnectDevice
                txtCallStatus = "Attempting to Connect Device..."
            Case RASCS_DeviceConnected
                txtCallStatus = "Device Opened"
            Case RASCS_AllDevicesConnected
                txtCallStatus = "All Devices Connected"
            Case RASCS_Authenticate
                txtCallStatus = "Authenticating..."
            Case RASCS_AuthNotify
                txtCallStatus = "Athentication Notification"
            Case RASCS_AuthRetry
                txtCallStatus = "Retrying Authentication..."
            Case RASCS_AuthCallback
                txtCallStatus = "Authentication Callback"
            Case RASCS_AuthChangePassword
                txtCallStatus = "Change Password"
            Case RASCS_AuthProject
                txtCallStatus = "Authenticating Project.."
```

```
            Case RASCS_AuthLinkSpeed
               txtCallStatus = "Authenticating Link Speed.."
            Case RASCS_AuthAck
               txtCallStatus = "Athentication Acknowlegement"
            Case RASCS_ReAuthenticate
               txtCallStatus = "ReAuthentication..."
            Case RASCS_Authenticated
               txtCallStatus = "Authenticated"
            Case RASCS_PrepareForCallback
               txtCallStatus = "Prepare For Callback"
            Case RASCS_WaitForModemReset
               txtCallStatus = "Waiting For Modem Rest..."
            Case RASCS_WaitForCallback
               txtCallStatus = "Waiting For Callback..."
            Case RASCS_Projected
               txtCallStatus = "Network Completely Configured"
            Case RASCS_StartAuthentication    'Windows 95 only
               txtCallStatus = "Attempting to Open Port"
            Case RASCS_CallbackComplete        'Windows 95 only
               txtCallStatus = "Callback Completed"
            Case RASCS_LogonNetwork           'Windows 95 only
               txtCallStatus = "Logging On To Network"
            Case RASCS_Interactive
               txtCallStatus = "Interactive"
            Case RASCS_RetryAuthentication
               txtCallStatus = "Retry Authentication"
            Case RASCS_CallbackSetByCaller
               txtCallStatus = "CallBack Set By Caller"
            Case RASCS_PasswordExpired
               txtCallStatus = "Password Expired"
            Case RASCS_Connected
               txtCallStatus = "Connected"
               cmdClose.Enabled = True
            Case RASCS_Disconnected
               txtCallStatus = "Disconnected"
            Case Else
               txtCallStatus = "Unknown State"
         End Select
      Case 10000
```

```
            '如果为 1000 表示获取状态的函数失败
        Case Else
            '获得错误代码
            lngRASErrorNumber = lngRASError
            '设置文本框属性
            txtCallStatus.FontBold = True
            txtCallStatus.ForeColor = RGB(255, 0, 0)
            txtCallStatus.Text = lpRASError.fcnRASErrorString()
            '挂断连接
            lngRetCode = RasHangUp(hRasConn)
            '使得关闭按钮有效
            cmdClose.Enabled = True
            '中止时钟
            tmrGetConnStatus.Enabled = False
    End Select
End Sub
```

该窗体是在异步拨号的时候检测拨号状态，在程序中很容易看懂，不多解释。重点是要看懂前面的几个类。

9.5.4　RAS 编程小结

上面的这个工程是一个功能比较强大的程序，可以实现各种功能，因此从根本上说可以满足用户的需要。在程序中，几乎所有的功能都是调用 API 函数来实现的，但是由于在 VB 中，同 API 的结合并不是十分密切，所以编程有一定的难度，需要参考比较多的资料。比如许多结构体需要在程序中声明等。而且在程序中，异步拨号的状态检测是通过时间控件定时调用 API 来检测，这样并不是最好的办法，其实可以通过使用回调函数来实现，回调函数的格式在前面已经用 VC 的格式介绍了。由于在 VB 中使用回调函数并不十分方便，所以这里笔者并没有编程实现。有兴趣的读者可以尝试用 VC 来开发程序。

同时在程序的开发过程中，新建电话簿要是调用系统的创建向导来实现的，其实这也并不是最好的解决办法。也可以通过其他的 API 函数来实现。因为在拨号的时候，必须要调用一个建立好的电话簿来进行拨号，因此如果一个新的系统，如果没有建立一个电话簿，则可能不成功。但是如果调用系统向导建立，则需要弹出恼人的界面，这可能也是读者不希望出现的。实际上在 Windows NT 或者 Windows 2000 系统里面提供了几个函数可以用来建立、增加和管理簿，而不需要向导。这几个函数是：RasValidateEntry、RasEnumDevices、RasGetCountryInfo、RasSetEntry、RasGetEntryProperties、RasRenameEntry 和 RasDeleteEntry。至于 Windows 98 系统由于电话簿是保存在注册表中，所以这些函数可能不是很适用，不过直接写注册表应该可以实现。

9.6 RAS 应用实例——远程文件共享

许多时候，用户可能在外地或者其他用户要需要拷贝一些自己家中机器的文件或者进行删除安装等等一系列的操作。如果仅仅是为了传输文件，通常的作法需要在两台机器上安装客户端软件，然后一方拨号，另一方自动应答，然后需要手工传输一些需要的文件，而这需要两边都有人在操作，而且功能很弱，只能传输文件。在本书的前面章节里面已经详细地解说了如何通过电话线来实现两台机器的文件传输以及信息发送。其实如果通过 RAS，用户可以进行更多的操作，可以操纵远程机器上的许多资源。具体的操作很简单，通过下面的步骤就可以实现这个功能。以下步骤在 Windows 98 操作系统上实现的。

（1）安装拨号服务器

为了能够实现 RAS 的功能，必须把一台机器做为服务器，另外一台机器作为客户端，作为服务器的机器必须首先要安装拨号服务器，最好是两台机器都能够安装拨号服务器。安装拨号服务器的步骤如图 9-24、图 9-25 所示。

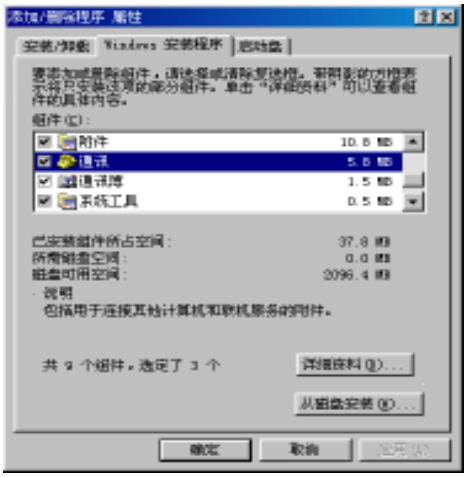


图 9-24 进入 Windows 安装程序

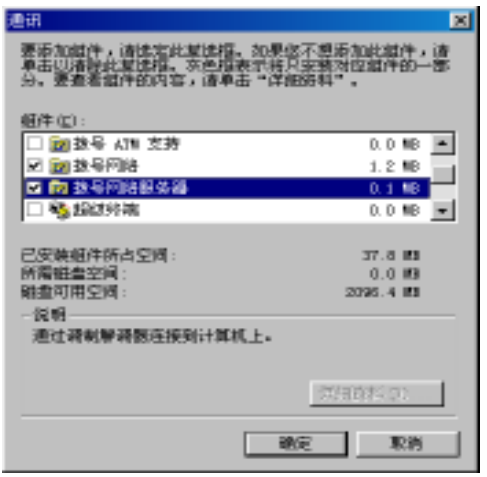


图 9-25 安装拨号网络服务器

图 9-24 是进入控制面板，选择“增加/删除程序”后，再选择“Windows 安装程序”属性页后的界面，因为拨号服务器是 Windows 98 自带的一个程序。然后选中“通讯”选项，单击“详细资料”按钮，会出现如图 9-25 的界面。可以看到里面有许多通讯服务程序，选择需要安装的“拨号网络服务器”，单击确定。可能此时需要插入 Windows 98 的安装盘。安装完毕重新启动系统。

✍ 其实我们通过 169、163 拨号出去，也是远程服务器上安装了拨号服务器的结果，只是它们的服务器上接了很多的 Modem，这样就可以让很多的客户同时上网了。

（2）安装协议

在两台计算机上同时增加相关协议，具体步骤如图 9-26～9-29 所示。

图 9-26　网络配置界面

图 9-26 的界面可以在桌面上右键单击网络邻居，然后在弹出菜单上选择属性，即可进入。单击“ 添加 ”按钮，可进入如图 9-27 所示的对话框。

图 9-27　选择安装协议

在图 9-27 中，选择协议，然后单击添加，可以进入到图 9-28，然后选择安装具体的协议。

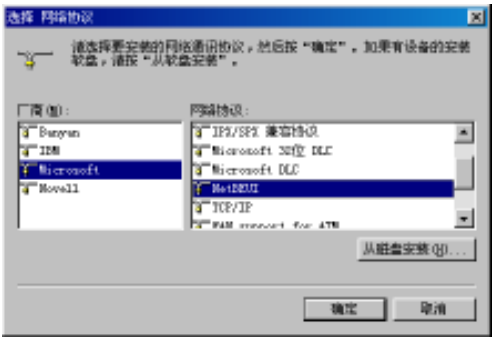


图 9-28　安装具体协议

在图 9-28 中，首先选择“ Microsoft ”，然后选择“ NetBEUI ”，然后单击确定，有可能要用户插入 Windows 98 安装盘。安装完协议后，如图 9-29 所示。

图 9-29　安装完协议

（3）设置共享

在图 9-29 中单击“文件及打印共享”按钮，弹出对话框如图 9-30 所示。为了能够让对方机器寻找，必须设置机器名和标志。因此只要在图 9-29 中选择“标志”属性页，然后设置如图 9-31，双方都需要设置计算机名。

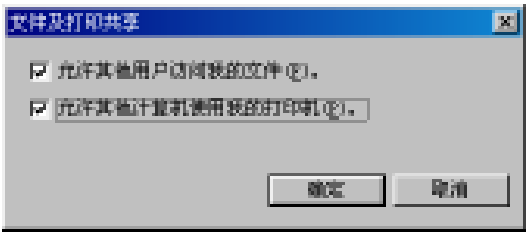


图 9-30　文件及打印共享

图 9-31　设置计算机标志

经过以上的设置以后，就是共享文件，共享文件的操作很简单，一般读者都会，就如同在局域网中共享一样，在一个文件夹中右键单击，然后选择共享，就可以完成。设置共享的时候可以设置密码，这样对方访问该文件夹的时候就必须提供密码。同时还可以设置权限，如只读、完全等，只读比较安全，只能读。而完全共享则对方既可以拷贝，也可是上载，还可以删除，因此要慎重。

（4）配置拨号服务器

为了能够让对方能够拨号，必需要配置拨号服务器。首先进入我的电脑，然后进入拨号网络，选择菜单上的的“连接选项”，再选择子菜单“拨号服务器”，出现如图 9-32 所示的界面。

图 9-32　拨号服务器

在图 9-32 上选择允许拨入，然后单击“服务器类型”按钮，进入图 9-33 所示。

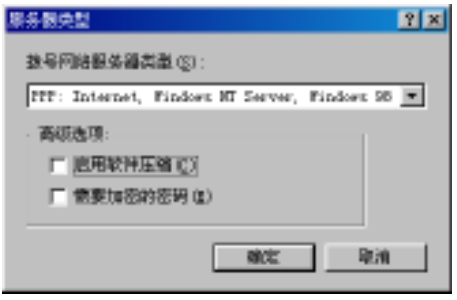


图 9-33　服务器类型

在服务器类型中选择点对点协议 PPP。然后取消下面的两个检查框。

（5）开始拨号并查找对方

建立拨号连接，在 9.2 节已经介绍，只需要在图 9-11 中输入对方电话号码，然后去掉用户名和密码就可以了，拨号成功以后，就可以双击网上邻居查找对方或者右键网上邻居，然后选择查找计算机，在计算机名中输入对方的计算机名。计算机名在前面已经教读者设置了。

如果一切顺利，就可以进入对方的计算机，然后可以看到对方共享的文件夹了。这样通过 RAS 来共享文件的步骤基本完成。注意以上的步骤对两台机器上最好都设置。

第 10 章　串口通信高级编程

在第 2 章中介绍了 MSComm 控件的使用，在这一章中就着重介绍如何使用该控件和 Windows API 来进行串口通信的高级开发。

10.1　串口通信中字符传输

串口通信中交换的数据大多是字符串的数据，而数据的交换必须要有格式，通信的双方才能根据一定的数据格式作出解释。

10.1.1　ASCII 控制字符

串口通信中一般使用许多 ASCII 中的控制字符作为数据格式的前导字符或者分别字符。ASCII 码的控制字符有很多，以下进行简单的介绍。

- NUL（Null）：该字符的含义是空项。它可以从 ASCII 字符序列中删除或插入，并不影响字符序列的意义，删除或插入 NUL 字符可以产生所希望的信息格式或设备控制。
- SOH（Start of Heading）：表头的开始字符，这是一个通信控制字符。在异步传输中用来分开不作任何处理的多个串行文件的传输。在多个串 行传输文件中用 SOH 字符表示每个文件的文件名开始，而后才是文件的内容。
- STX（Start of Text）：文本开始，通信控制字符。在二进制进制同步协议中使用。它表示表头头结束，信息数据开始。
- ETX（End of Text）：文本结束，通信控制字符。它表示信息数据的结束，数据检测字符开始，说明后面的信息是用来检查通信错误的。
- EOT（End of Transmission）：发送结束，通信控制字符。用于通知接收设备内文中的数据已全部发送完毕，同时也向网络中的其他设备要求注意检测，看看是否有发向它们的内文。EOT 与 SOH 相对应。
- EOB（End of Block）：发送区块结束，通信控制字符。用于表示发送数据群的结束。当发送的是两个或多个区块，而不是一个连续群时，二进制同步通信规定用这个字符代替 ETX 字符。
- ENQ（Enquiry）：查询，通信控制字符。用于查询通信接收站的情况。它可用来查询设备的名称或者判断发送的状态。收到 ENQ 字符后要作出响应，接收设备可以用成功收到区块的编号作为回答。
- ACK（Acknowledge）：确认，通信控制字符。用来告诉发送端接收端所接收数据正确与否，作为错误检测和流量控制。当接收端接收到正确的数据后，便向发送端送出 ACK 字符，以表示收到数据，而发送端在收到 ACK 字符后便接着发送下

一次的数据。

- BEL（Bell）：警示，专用 ASCII 字符。可以包括在文本文件中，在两个设备间发送，用于引起通信双方必要的注意。
- BS（Backspace）：退格，格式控制字符。通常仅在交谈方式的数据发送中使用。
- HT（Horizontal Tabulation）：水平制表，格式控制字符。此字符能使打印机打印头移到下一个预定的位置。它可以包括在本文件中。
- LF（Line Feed）：换行，格式控制字符。使打印头移动到下一行的同一列的位置。此字符通常接在 CR 字符后。
- VT（Vertical Tabulation）：垂直制表，格式控制字符。移到规定的行上。
- FF（Form Feed）：换页，格式控制字符。将打印头移到下一页的顶部或指定行上。
- CR（Carriage Return）：回转键，格式控制字符。
- SO（Shift Out）：移出，代码扩充控制字符，用于扩充标准图形字符集。打印机收此字符后，会以双倍宽度做打印的工作，直到收到 DC4 控制字符为止。
- SI（Shift In）：移入，代码扩充控制字符。用于把接收设备恢复为标准 ASCII 字符集。
- DLE（Data Link Escape）：数据链跳离，通信控制字符。用来修改后面一定数量字符的意义。
- DC1、DC2、DC3、DC4：设备控制字符。DCI 用作 XON，重新激活由 XOFF 所中断的传输。DCZ 用来终止压缩打印模式，并清空打印缓冲区。DC3 用作 XOFF，与 DCI 相互搭配，作为流量控制用。DC4 通常由厂商自行决定其用途。
- NAK（Negative Acknowledge）：否认，通信控制字符。用于表示发送端和接收端间的通信错误。此字符一般也由发送端发送，而出现错误时就以此字符激活发送端的重送程序。
- SYN（Synchronous）：同步，通信控制字符。用于激活或保持不发送数据时的通信同步信号。此字符类似于异步传输中的停止字符。
- CAN（Cancel）：取消，专用控制字符，用来指示数据传输中的误差，指出接收到的数据应不予考虑。
- EM（End of　Median）：媒体用毕，专用控制字符，用来指示数据媒体用毕。
- SUB（Substitute）：取代字符，用作数据通信的准确度控制用。
- ESC（Escape）：跳脱字符，代码扩充类控制字符。特别是用在与打印机的通信中。
- FS（File Separator）：文件分隔符，属信息分隔类字符。作为被发送文件之间的逻辑边界。
- GS（Group Separator）：组分隔字符，属于信息分隔类字符。用于表示被发送数据组之间的逻辑边界。
- RS（Record Separator）：记录分隔字符，是第 3 个信息分隔字符。
- US（Unit Separator）：单位分隔字符，是最后一个信息分隔字符。用于表示不同数据单位之间的逻辑边界。
- DEL（Delete）：删除字符。

不管命令字符串或是响应字符串，固定的格式是传输双方必须遵守的，这种格式约定提

升到一定高度，就是协议。

10.1.2 通信中的字符和字节

1. 字符和字节

早期的计算机只识别英文，英文只需使用到 ASCII 码的前 128 个位足以表达全部的字符；可是中文就无法仅以这些码来表示，它们必须以两个字节才足以表达完整；为了解决这个问题，几家美国的厂商就联手制订了 Unicode，这个 Unicode 涵盖了世界上所有国家的字符码，每个文字用一个唯一的内码来表示，但是此种 Unicode 的特点就是所有的字符是以两个字节表示的，不仅中文使用两个字节连英文也使用两个字节。

如果英文也是两个字节，那么会造成原本是一个字节（7 位 ASCII）表示的英文保存要多浪费空间。在 Windows 98/95 中使用的是 ANSI/DBCS，即使用英文时使用 ANSI，使用中文或者其他时使用 DBCS；Windows NT/2000 支持 Unicode 。

在 VB 环境下，使用 Len 函数获得的是字符数，包括中文和英文数。

例如：

```
Dim  s%
s=Len("NC 开始")
```

可以看出 s 等于 4。

如果要使用英文用一个字节表示，中文用两个字节表示，要用到 LenB 函数和 StrConv 函数组合处理。

LenB 函数返回整个该字符串的字节数，在 VB 中是以 Unicode 字符集的，因而把英文字符也作为双字节处理的，这也不能达到的目的。

StrConv 函数的功能十分强大，可以转换字符串到特定的格式。StrConv 函数回使用到前两个参数——String 以及 Conversion，其中的 String 指的是要转换的字符串，Conversion 的值如表 10-1 所示。

表 10-1　　通信事件常数设定值

常数	值	描述
vbUpperCase	1	将字符串转换成大写
vbLowerCase	2	将字符串转换成小写
vbProperCase	3	将字符串中字的开头转换成大写
vbWide*	4	将字符串中单字节转换成双字节
vbNarrow*	8	将字符串中双字节转换成单字节
vbUnicode	64	将系统的默认字符码对应页转换成 Unicode
vbFromUnicode	128	将 Unicode 转换成系统的默认字符码对应页

可以用 LenB(StrConv(str, vbFromUnicode))获得字节的长度，进行传输。

2. MSComm 中的字节传输

上一节介绍的串口传输是使用字符的传输。而一般的串行控制的确也以字符为主要的传输方式。不过，串行数据传送的设定中若使用 8 个位代表一个字节，则有可能传送超过 ASCII 码 128 以上的字符（例如，汉字的双字节的前一个字节），而这些字符均属于不可见字符，

这些不可见字符与可见字符将使得串行通信产生一些特别的现象。

在介绍 MSComm 控件时，有一个 InputMode 属性，此属性的默认值是 0（Text 文本模式），也就是接收的方式是将输入的数据当作文字予以解读；但某些情况下，传送的却不全是纯文字型态的数据，而是数据或文件在传送，在此种情形下，位在传输线上的数据就是一个字节接着一个字节地被传送过来，接收端收到这些数据后，再按照组合形成数据资料或文件，这种情形就是 1（Binary 二进制数据）的传送。

可以使用如下的代码进行组合。

```
Dim Buffer As Variant
Buffer = MSComm1.Input
Debug.Print "Receive - " & StrConv(Buffer, vbUnicode )
ShowData txtTerm, (StrConv(Buffer, vbUnicode ))
```

10.2　MSComm 控件编程实例

10.2.1　建立工程项目

下面将会结合一个利用 MSComm 控件编写的高级实例 CommTerm 介绍 MSComm 控件编程，让读者能够更深入的了解怎样对 MSComm 控件编程。

该实例所能实现的功能：

- 配置串口的各种属性。
- 按键传输。
- 传送文本文件。
- 定时机制自动接受和监视功能。

该实例建立的 VB 工程一共有 3 个窗体和一个模块，两个类模块：

- 窗体 frmMSCommDemo、frmCancelSend 和 frmProperties。
- 模块 Module1。

frmMSCommDemo 窗口是这个应用程序的主窗口，运行时的界面如图 10-1 所示。

10.2.2　代码分析

1. 窗体 frmMSCommDemo 文件

该窗体上有一个 TextBox 控件，用于显示和输入传递的的数据，还有一个 Timer 控件，其他的还有一个 Timer 控件，用于定时刷新输入和输出的数据，更重要的还有一个 MSComm 控件，它的 CommPort 为 1，InputMode 设为 1（二进制数据模式），Rthreshold 设为 1，即有输入就触发接收事件。请读者打开相应的工程项目文件查看。

```
Option Explicit
Dim Ret As Integer
Dim Temp As String
```

```
    Dim hLogFile As Integer
    Dim StartTime As Date
```

上面的代码用定义所用到的一些全局变量。

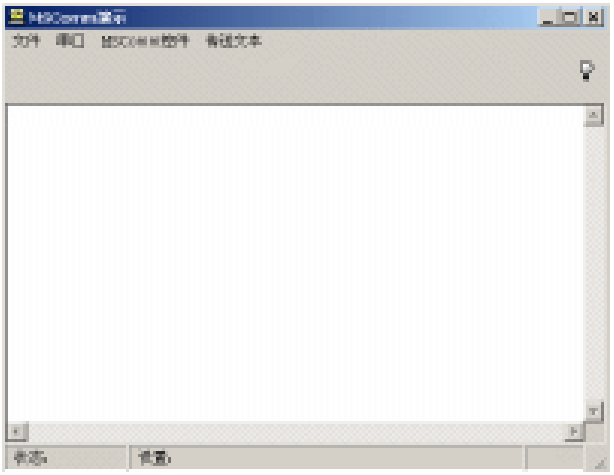


图 10-1　CommTerm **运行界面**

```
    Private Sub Form_Load()
        Dim CommPort As String, Handshaking As String, Settings As String
        On Error Resume Next
        txtTerm.SelLength = Len(txtTerm)
        txtTerm.SelText = ""
        txtTerm.ForeColor = vbBlue
        '设置标题
        App.Title = "MSComm 演示"
        ' 设置不连接的灯显示
        imgNotConnected.ZOrder
        frmMSCommDemo.Move (Screen.Width - Width) / 2, (Screen.Height - Height)
/ 2
         ' 从注册表中装载上次运行的记录
        Settings = GetSetting(App.Title, "Properties", "Settings", "")
        If Settings <> "" Then
            MSComm1.Settings = Settings
            If Err Then
                MsgBox Error$, 48
                Exit Sub
            End If
        End If
        CommPort = GetSetting(App.Title, "Properties", "CommPort", "")
        If CommPort <> "" Then MSComm1.CommPort = CommPort
```

```
    Handshaking = GetSetting(App.Title, "Properties", "Handshaking", "")
    If Handshaking <> "" Then
       MSComm1.Handshaking = Handshaking
       If Err Then
          MsgBox Error$, 48
          Exit Sub
       End If
    End If
    On Error GoTo 0
End Sub
```

上面的代码用于在窗体载入时，利用 API **函数** GetSetting **从注册表中获得上次运行该程序的串口配置信息，以便应用程序能够继承上次的串口配置。**

```
Private Sub Form_Resize()
If frmMSCommDemo.ScaleWidth > 0 Then
 txtTerm.Move 0, tbrToolBar.Height, frmMSCommDemo.ScaleWidth,
frmMSCommDemo.ScaleHeight - sbrStatus.Height - tbrToolBar.Height
 Frame1.Left = ScaleWidth - Frame1.Width * 1.5
 End If
End Sub
```

上面的代码用于在窗体串口尺寸变化时，响应是 txtTerm **文本框要产生的变化。注意在极小化应用程序时，**txtTerm **文本框不用产生变化。**

```
Private Sub mnuSendText_Click()
  Dim hSend, BSize, LF&
  On Error Resume Next
  mnuSendText.Enabled = False
  OpenLog.DialogTitle = "发送的文本文件"
  OpenLog.Filter = "文本文件(*.TXT)|*.txt|All Files (*.*)|*.*"
  Do
    OpenLog.CancelError = True
    OpenLog.FileName = ""
    OpenLog.ShowOpen
    If Err = cdlCancel Then
      mnuSendText.Enabled = True
      Exit Sub
    End If
    Temp = OpenLog.FileName
    '没有，返回
    Ret = Len(Dir$(Temp))
    If Err Then
```

```
            MsgBox Error$, 48
            mnuSendText.Enabled = True
            Exit Sub
         End If
         If Ret Then
            Exit Do
         Else
            MsgBox Temp + " not found!", 48
         End If
      Loop
       hSend = FreeFile
      Open Temp For Binary Access Read As hSend
      If Err Then
         MsgBox Error$, 48
      Else
         '显示可以取消对话框
         CancelSend = False
         frmCancelSend.Label1.Caption = "正在传送文本文件 " + Temp
         frmCancelSend.Show
          '读进输出缓冲区
         BSize = MSComm1.OutBufferSize
         LF& = LOF(hSend)
         Do Until EOF(hSend) Or CancelSend
            ' 读取余额
            If LF& - Loc(hSend) <= BSize Then
               BSize = LF& - Loc(hSend) + 1
            End If
           ' 读取一块数据
            Temp = Space$(BSize)
            Get hSend, , Temp
            '传送一块
            MSComm1.Output = Temp
            If Err Then
               MsgBox Error$, 48
               Exit Do
            End If
           '等待全部的数据输出
            Do
               Ret = DoEvents()
```

```
        Loop Until MSComm1.OutBufferCount = 0 Or CancelSend
      Loop
    End If
    Close hSend
    mnuSendText.Enabled = True
    tbrToolBar.Buttons("TransmitTextFile").Enabled = True
    CancelSend = True
    frmCancelSend.Hide
End Sub
```

上面的代码用于单击“传送文本文件”事件的响应过程。该程序用来在串口打开时传输文本文件。程序先弹出一个选择“发送的文本文件”的对话框，选定一个文本文件，以二进制方式打开，然后把文件分割成输出缓冲区大小的二进制数据块，分块地传送到接受端。在传输的过程中，可以取消剩余的传输。

选择文件传输对话框如图 10-2 所示。

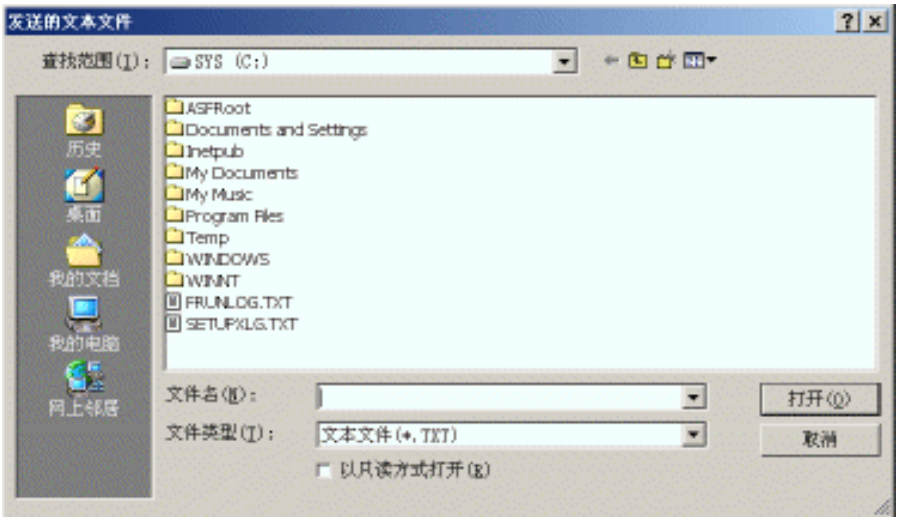


图 10-2　选择传输文件

执行文件传输的界面如图 10-3 所示。

```
Private Sub imgConnected_Click()
    '点击显示灯，等于点击“打开串口”
    Call mnuOpen_Click
End Sub

Private Sub imgNotConnected_Click()
   '点击显示灯，等于点击“打开串口”
    Call mnuOpen_Click
End Sub
```

以上的代码表明点击连接的显示等相当于单击串口“打开”菜单功能。

图 10-3 文件传输过程

```
'DTREnabled 属性控制.
Private Sub mnuDTREnable_Click()
    MSComm1.DTREnable = Not MSComm1.DTREnable
    mnuDTREnable.Checked = MSComm1.DTREnable
End Sub

'CDHolding 属性控制
Private Sub mnuHCD_Click()
    If MSComm1.CDHolding Then
        Temp = "True"
    Else
        Temp = "False"
    End If
    MsgBox "CDHolding = " + Temp
End Sub

'CTSHolding 属性控制.
Private Sub mnuHCTS_Click()
    If MSComm1.CTSHolding Then
        Temp = "True"
    Else
        Temp = "False"
    End If
```

```
        MsgBox "CTSHolding = " + Temp
    End Sub

    'DSRHolding 属性控制
    Private Sub mnuHDSR_Click()
        If MSComm1.DSRHolding Then
            Temp = "True"
        Else
            Temp = "False"
        End If
        MsgBox "DSRHolding = " + Temp
    End Sub
    'InputLen 属性控制.
    Private Sub mnuInputLen_Click()
        On Error Resume Next

        Temp=InputBox$("Enter New InputLen:","InputLen",Str$(MSComm1.InputLen))
        If Len(Temp) Then
            MSComm1.InputLen = Val(Temp)
            If Err Then MsgBox Error$, 48
        End If
    End Sub

    Private Sub mnuParRep_Click()
    ' ParityReplace 属性控制
    On Error Resume Next
    Temp=InputBox$("Enter eplace Character", "ParityReplace",
frmMSCommDemo.MSComm1.ParityReplace)
       frmMSCommDemo.MSComm1.ParityReplace = Left$(Temp, 1)
        If Err Then MsgBox Error$, 48
    End Sub

    ' RThreshold 属性控制
    Private Sub mnuRThreshold_Click()
        On Error Resume Next
        Temp = InputBox$("Enter New RThreshold:", "RThreshold",
Str$(MSComm1.RThreshold))
        If Len(Temp) Then
            MSComm1.RThreshold = Val(Temp)
```

```
        If Err Then MsgBox Error$, 48
      End If
   End Sub

   ' SThreshold property 属性控制.
   Private Sub mnuSThreshold_Click()
      On Error Resume Next
     Temp = InputBox$("Enter New SThreshold Value", "SThreshold",
Str$(MSComm1.SThreshold))
      If Len(Temp) Then
         MSComm1.SThreshold = Val(Temp)
         If Err Then MsgBox Error$, 48
      End If
   End Sub
```

以上是响应 MSComm 控件属性的响应函数，可以用它们来修改 MSComm 的各个属性，但要注意的是，修改以后可以无法做出正确的响应。

```
   'CommPort 属性窗口
   Private Sub mnuProperties_Click()
      frmProperties.Show vbModal
   End Sub
```

以上的是打开串口属性的响应过程，显示一个串口属性配置对话框配置串口属性。

```
   ' 打开和关闭串口
   Private Sub mnuOpen_Click()
      On Error Resume Next
      Dim OpenFlag
      MSComm1.PortOpen = Not MSComm1.PortOpen
      If Err Then MsgBox Error$, 48
      OpenFlag = MSComm1.PortOpen
      mnuOpen.Checked = OpenFlag
      mnuSendText.Enabled = OpenFlag
      tbrToolBar.Buttons("TransmitTextFile").Enabled = OpenFlag
      If MSComm1.PortOpen Then
         imgConnected.ZOrder
         sbrStatus.Panels("Settings").Text = "设置： " & MSComm1.Settings
         StartTiming
      Else
         imgNotConnected.ZOrder
         sbrStatus.Panels("Settings").Text = "设置： "
```

```
        StopTiming
    End If
End Sub
```

以上的是打开和关闭串口的响应过程。如果串口没有被打开，该过程作用是打开该串口，并设置相应的显示；如果串口如果已经被该程序打开，该过程作用是关闭该串口，并设置相应的显示。如果要打开一个已被别的程序打开的串口，则程序会显示如图 10-4 所示的错误对话框。

图 10-4　串口打开错误提示框

```
' OnComm 事件控制.
Private Static Sub MSComm1_OnComm()
    Dim EVMsg$
    Dim ERMsg$
    '根据事件分发处理
    Select Case MSComm1.CommEvent
        ' Event messages.
        Case comEvReceive
            Dim Buffer As Variant
            Buffer = MSComm1.Input
            Debug.Print "Receive - " & StrConv(Buffer, vbUnicode )
            ShowData txtTerm, (StrConv(Buffer, vbUnicode ))
        Case comEvSend
        Case comEvCTS
            EVMsg$ = "Change in CTS Detected"
        Case comEvDSR
            EVMsg$ = "Change in DSR Detected"
        Case comEvCD
            EVMsg$ = "Change in CD Detected"
        Case comEvRing
            EVMsg$ = "The Phone is Ringing"
        Case comEvEOF
            EVMsg$ = "End of File Detected"
        ' 错误信息
        Case comBreak
            ERMsg$ = "Break Received"
```

```
        Case comCDTO
            ERMsg$ = "Carrier Detect Timeout"
        Case comCTSTO
            ERMsg$ = "CTS Timeout"
        Case comDCB
            ERMsg$ = "Error retrieving DCB"
        Case comDSRTO
            ERMsg$ = "DSR Timeout"
        Case comFrame
            ERMsg$ = "Framing Error"
        Case comOverrun
            ERMsg$ = "Overrun Error"
        Case comRxOver
            ERMsg$ = "Receive Buffer Overflow"
        Case comRxParity
            ERMsg$ = "Parity Error"
        Case comTxFull
            ERMsg$ = "Transmit Buffer Full"
        Case Else
            ERMsg$ = "Unknown error or event"
    End Select

    If Len(EVMsg$) Then
        '显示
        sbrStatus.Panels("Status").Text = "状态：" & EVMsg$
        Timer2.Enabled = True
    ElseIf Len(ERMsg$) Then
        '显示 错误信息
        sbrStatus.Panels("Status").Text = "状态：" & ERMsg$

        Beep
        Ret = MsgBox(ERMsg$, 1, "Click Cancel to quit, OK to ignore.")
         If Ret = 2 Then
            MSComm1.PortOpen = False    '关闭串口，退出
        End If
          Timer2.Enabled = True
    End If
End Sub
```

以上代码是 MSComm 的 OnComm 事件处理过程。本程序是采用分发事件方式处理的，

如果是出错信息，则显示到主界面的状态条中；如果是接受事件，则调用 StrConv(Buffer, vbUnicode)将二进制字节数据转换为 Unicode 字符串，显示到文本框中。

```
' 显示数据，过滤和控制特殊字符：回退键、回车键、换行字符，监视有没有超过最大量
Private Static Sub ShowData(Term As Control, Data As String)
    On Error GoTo Handler
    Const MAXTERMSIZE = 16000
    Dim TermSize As Long, i
    '确保没有超过最大量
    TermSize = Len(Term.Text)
    If TermSize > MAXTERMSIZE Then
       Term.Text = Mid$(Term.Text, 4097)
       TermSize = Len(Term.Text)
    End If
    Term.SelStart = TermSize
    '过滤和控制特殊字符：回退键、回车键、换行字符
    Do
       i = InStr(Data, Chr$(8))
       If i Then
          If i = 1 Then
             Term.SelStart = TermSize - 1
             Term.SelLength = 1
             Data = Mid$(Data, i + 1)
          Else
             Data = Left$(Data,i-2)&Mid$(Data,i+1)
          End If
       End If
    Loop While i
    Do
       i = InStr(Data, Chr$(10))
       If i Then
          Data = Left$(Data, i - 1) & Mid$(Data, i + 1)
       End If
    Loop While i
    i = 1
    Do
       i = InStr(i, Data, Chr$(13))
       If i Then
          Data = Left$(Data, i) & Chr$(10) & Mid$(Data, i + 1)
          i = i + 1
```

```
        End If
    Loop While i
    '添加过滤后的数据
    Term.SelText = Data
    Term.SelStart = Len(Term.Text)
Exit Sub
Handler:
    MsgBox Error$
    Resume Next
End Sub
```

以上代码是用来过滤控制字符，回退键、回车键、换行字符，监视有没有超过最大量，这样有利于数据的显示。

```
'捕捉 KEY_DOWN 消息
Private Sub txtTerm_KeyPress(KeyAscii As Integer)
       If MSComm1.PortOpen Then
       MSComm1.Output = Chr$(KeyAscii)
    End If
End Sub
```

以上代码是 KEY_DOWN 事件的响应过程。该过程的处理是将文本框输入的字符（不管是中文还是英文），都输出到串口中。KEY_DOWN 事件的传输如图 10-5 所示。

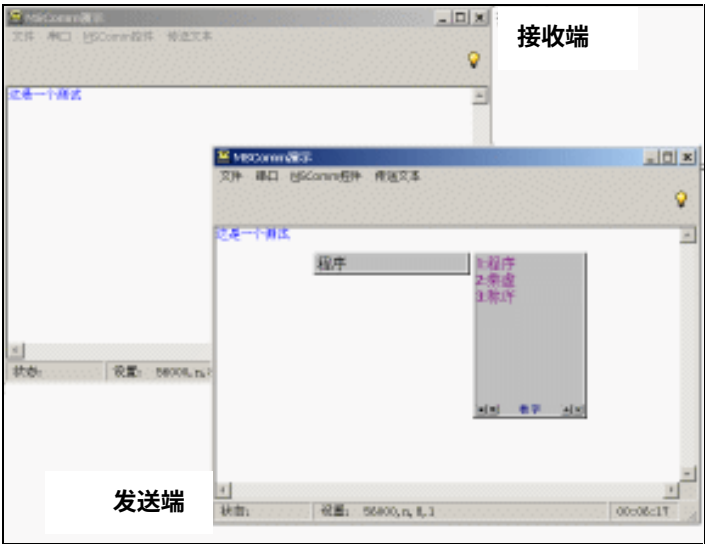


图 10-5　数据传输界面

```
Private Sub Timer1_Timer()
    ' Display the Connect Time
    sbrStatus.Panels("ConnectTime").Text = Format(Now - StartTime, "hh:nn:ss")
& " "
End Sub
```

```
Private Sub Timer2_Timer()
sbrStatus.Panels("Status").Text = "状态"
Timer2.Enabled = False
End Sub

' 连接计时器开始
Private Sub StartTiming()
    StartTime = Now
    Timer1.Enabled = True
End Sub
'停止计时器工作
Private Sub StopTiming()
    Timer1.Enabled = False
    sbrStatus.Panels("ConnectTime").Text = ""
End Sub
```

以上 4 个过程是响应在串口打开和关闭时的计时问题，串口一打开就开始计算时间，串口关闭是取消计时。

```
Private Sub mnuFileExit_Click()
    Form_Unload Ret
End Sub
Private Sub Form_Unload(Cancel As Integer)
    Dim Counter As Long
    If MSComm1.PortOpen Then
       '等待传送十秒钟
       Counter = Timer + 10
       Do While MSComm1.OutBufferCount
          Ret = DoEvents()
          If Timer > Counter Then
             Select Case MsgBox("数据还没传送完闭", 34)
                ' Cancel.
                Case 3
                   Cancel = True
                   Exit Sub
                ' Retry.
                Case 4
                   Counter = Timer + 10
                ' Ignore.
                Case 5
```

```
                Exit Do
            End Select
          End If
        Loop
        MSComm1.PortOpen = 0
     End If
   End
End Sub
```

以上 mnuFileExit_Click **是“退出”菜单的响应。**Form_Unload **是响应窗口退出事件的，退出时如果有文件在传输，则应该等待一段时间，而后询问是否继续等待还是退出，这样的处理比较友好。**

2. **窗体** frmCancelSend

frmCancelSend **窗体作用在于在传送文件过程中可以取消剩余的传送，该窗体如图** 10-6 **所示。**

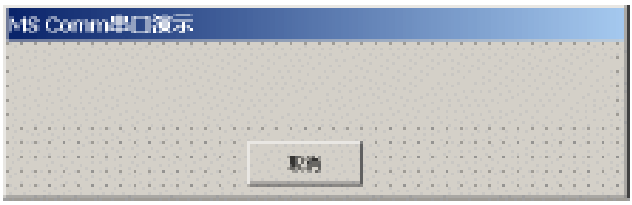


图 10-6　**取消传输窗口**

```
DefInt A-Z
Option Explicit

Const SWP_NOMOVE = &H2
Const SWP_NOSIZE = &H1

Private Sub Command1_Click()
   CancelSend = True
End Sub

Private Sub Form_Activate()
   SetWindowPos hWnd, -1, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE
End Sub

Private Sub Form_Deactivate()
   If Not CancelSend Then
      frmCancelSend.Show
   End If
End Sub
```

frmCancelSend 窗体只有在传输文件过程中才会出现。以上的代码说明了单击“取消”按钮，CancelSend 变量将会变为 TRUE，这时文件传输过程将推出分块传输循环，传输结束，窗体被隐藏。

3. 窗体 Properties

Properties 窗体显示串口的属性，包括选择串口，修改数据传输率，修改连接属性，以及修改握手协议等。Properties 窗体运行时界面如图 10-7 所示。

图 10-7　配置串口属性窗口

```
Private iFlow As Integer
Sub LoadPropertySettings()
Dim i As Integer, Settings As String, Offset As Integer
'加载串口设定
For i = 1 To 16
   cboPort.AddItem "Com" & Trim$(Str$(i))
Next i
'加载速度项选项
cboSpeed.AddItem "110"
cboSpeed.AddItem "300"
cboSpeed.AddItem "600"
cboSpeed.AddItem "1200"
cboSpeed.AddItem "2400"
cboSpeed.AddItem "4800"
cboSpeed.AddItem "9600"
cboSpeed.AddItem "14400"
cboSpeed.AddItem "19200"
cboSpeed.AddItem "28800"
cboSpeed.AddItem "38400"
cboSpeed.AddItem "56000"
```

```
    cboSpeed.AddItem "57600"
    cboSpeed.AddItem "115200"
    cboSpeed.AddItem "128000"
    cboSpeed.AddItem "256000"
    ' 加载数据位选项
    cboDataBits.AddItem "4"
    cboDataBits.AddItem "5"
    cboDataBits.AddItem "6"
    cboDataBits.AddItem "7"
    cboDataBits.AddItem "8"
    '加载奇偶检验选项
    cboParity.AddItem "Even"
    cboParity.AddItem "Odd"
    cboParity.AddItem "None"
    cboParity.AddItem "Mark"
    cboParity.AddItem "Space"
    '加载停止位选项
    cboStopBits.AddItem "1"
    cboStopBits.AddItem "1.5"
    cboStopBits.AddItem "2"
    '默认的设置
    Settings = frmMSCommDemo.MSComm1.Settings
    If InStr(Settings, ".") > 0 Then
       Offset = 2
    Else
       Offset = 0
    End If
    cboSpeed.Text = Left$(Settings, Len(Settings) - 6 - Offset)
    Select Case Mid$(Settings, Len(Settings) - 4 - Offset, 1)
    Case "e"
       cboParity.ListIndex = 0
    Case "m"
       cboParity.ListIndex = 1
    Case "n"
       cboParity.ListIndex = 2
    Case "o"
       cboParity.ListIndex = 3
    Case "s"
       cboParity.ListIndex = 4
```

```
    End Select
    cboDataBits.Text = Mid$(Settings, Len(Settings) - 2 - Offset, 1)
    cboStopBits.Text = Right$(Settings, 1 + Offset)
    cboPort.ListIndex = frmMSCommDemo.MSComm1.CommPort - 1
    optFlow(frmMSCommDemo.MSComm1.Handshaking).Value = True
    End Sub
```

以上代码被 Form_Load **出来函数调用，用于的在** Properties **窗体打开时显示当前的串口配置。**

```
    '按 cancel 按钮不进行修改
    Private Sub cmdCancel_Click()
    Unload Me
    End Sub

    '按 ok 按钮进行修改
    Private Sub cmdOK_Click()
    Dim OldPort As Integer, ReOpen As Boolean
    On Error Resume Next
    OldPort = frmMSCommDemo.MSComm1.CommPort
    NewPort = cboPort.ListIndex + 1
    If NewPort <> OldPort Then                ' If the port number changes, close
the old port.
       If frmMSCommDemo.MSComm1.PortOpen Then
             frmMSCommDemo.MSComm1.PortOpen = False
             ReOpen = True
       End If
       frmMSCommDemo.MSComm1.CommPort = NewPort         ' Set the new port number.
          If Err = 0 Then
          If ReOpen Then
              frmMSCommDemo.MSComm1.PortOpen = True
              frmMSCommDemo.mnuOpen.Checked = frmMSCommDemo.MSComm1.PortOpen
              frmMSCommDemo.mnuSendText.Enabled =
frmMSCommDemo.MSComm1.PortOpen
              frmMSCommDemo.tbrToolBar.Buttons("TransmitTextFile").Enabled =
frmMSCommDemo.MSComm1.PortOpen
          End If
       End If
              If Err Then
          MsgBox Error$, 48
          frmMSCommDemo.MSComm1.CommPort = OldPort
```

```
            Exit Sub
        End If
    End If
    frmMSCommDemo.MSComm1.Settings = Trim$(cboSpeed.Text) & "," &
Left$(cboParity.Text, 1) _
        & "," & Trim$(cboDataBits.Text) & "," & Trim$(cboStopBits.Text)
    If Err Then
        MsgBox Error$, 48
        Exit Sub
    End If
    '提示错误
    frmMSCommDemo.MSComm1.Handshaking = iFlow
    If Err Then
        MsgBox Error$, 48
        Exit Sub
    End If
    '保存到注册表
    SaveSetting App.Title, "Properties", "Settings",
frmMSCommDemo.MSComm1.Settings
    SaveSetting App.Title, "Properties", "CommPort",
frmMSCommDemo.MSComm1.CommPort
    SaveSetting App.Title, "Properties", "Handshaking",
frmMSCommDemo.MSComm1.Handshaking
    Unload Me
    End Sub
```

以上代码被分别响应“ Cancel **”按钮和“** OK **”按钮事件。点击“** cancel **”按钮** Sub cmdCancel_Click()**过程将不进行任何修改。点击“** OK **“按钮进行修改** cmdOK_Click()**将修改当前的配置，并** SaveSetting **将所做的修改保存到注册表中。**

```
    '装载设置
    Private Sub Form_Load()
    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 2
    fraSettings.ZOrder
    LoadPropertySettings
    End Sub
    '更改数据流控制
    Private Sub optFlow_Click(Index As Integer)
    iFlow = Index
    End Sub
```

以上代码是说明在 Form_Load 事件中装载设置，以及修改握手协议的响应过程。

4. 模块 Module1

Module1 模块文件用来声明了几个 API 以及一个全局变量。本项目可以在两部计算机上同时执行，执行时，须将两台计算机上的串口通过串口线相连接；也可以用一台计算机的两个串口通过串口线相连。连接完成后，便可以执行操作了，需要注意的是：两个串口的数据传输率等必须设置为一样。

10.3　Windows API 串口通信高级实例

Windows 环境下的串口编程跟 DOS 环境下的串口编程有很大的不同。Windows 最大的特征之一就是设备无关性，它通过设备驱动程序将 Windows 应用程序与不同的输出设备隔离，最典型的例子是输出到打印机、显示器、绘图仪上的代码可以完全相同。

Windows 封装了 Windows 的通信机制为通信 API，Windows 程序员可以利用 Windows 通信 API 进行编程，不用对硬件直接进行操作。

在第 2 章中已经详细介绍了串口通信 API 函数的内容，在本章主要通过这些函数来进行串口的通信和数据传输。

10.3.1　VB 中调用 Windows API

VB 自身提供了数量众多的内部函数，但是，在编程的过程中，经常需要访问操作系统的一些核心的内容，这就需要访问 Windows 操作系统中的一些系统函数，这些函数是由 Win32 应用程序接口（Application Program Interface，API）提供的。

1. API 声明

API（应用程序编程接口）由函数、消息、数据结构、数据类型以及语句组成，它们在创建在 Windows 下运行的应用程序中使用。API 中使用最多的部分是 Windows 中调用 API 函数的代码元素，包括过程声明（Windows 函数）、用户自定义类型的定义（用来传递到函数中的数据结构），以及常数声明（传递给函数以及从函数中返回的值）。

Win32 函数是作为动态链接库（DLL）提供给用户的，可以在任何语言中调用。 DLL 是一种过程库，应用程序可以在运行时链接并使用它。需要使用 VB 核心语言和控件未包含的功能，可以直接调用 Windows 系统动态链接库中的过程。通过 DLL，程序员可以访问构成 Windows 操作系统主体的成千上万个过程。VB 还可以使用 DLL 使用其他语言（例如 VC++）编写的各种例程。

2. API 的使用

VB 应用程序中能够通过声明外部过程，访问 Win32 API（以及其它的外部 DLL）。在声明了过程之后，调用它的方法与 Visual Basic 内部的过程相同。声明 API 函数要使用 Declare 语句。

Win32 中包含成千上万的函数、例程、类型和常数定义，在 VB 程序中可以声明并使用它们。但是，这些过程是用 C 语言编写的，在 VB 中使用之前必须先进行声明。DLL 过程的声明是比较复杂的，最简单的办法是使用 VB 专门提供的预定义 API 声明。

位于 Visual Basic 主目录下的＼Winapi 目录中的 Win32api.txt 文件中包含 VB 中经常使用的许多 Windows API 的过程声明。使用时只需从该文件将函数、类型等定义复制到 VB 模块中即可。可以使用 VB API Viewer 应用程序，也可以使用文本编辑器来复制 API。如图 10-8 所示。

使用 VB API Viewer 的步骤如下：

（1）从“外接程序”菜单中，打开外接程序管理器并加载“VB API Viewer”。

（2）从“外接程序”菜单中选择打开“VB API Viewer”。

（3）加载文本文件或者数据库文件。

（4）选择 API 的类型。

（5）查找 API，将查找关键词添加到选定项里。

（6）复制或者插入选定项到 VB 项目中。

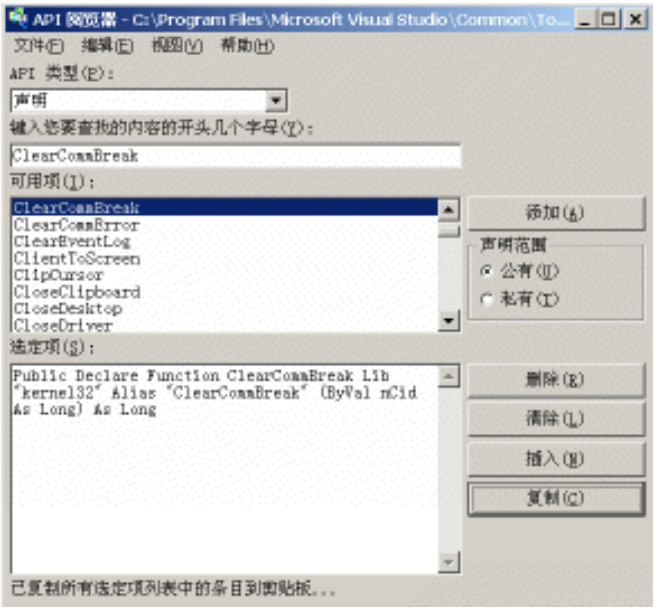


图 10-8　API 浏览器界面

也可以用下列 Declare 语句来声明其他的外部函数：

```
    Declare Function name Lib "libname" [Alias "aliasname"] [([[ByVal] variable [As type] [, [ByVal] variable [As type] ]..])] As Type
```

如果函数没有返回值，可将其声明为 Sub，语法如下：

```
    Declare Sub name Lib "libname" [Alias "aliasname"] [([[ByVal] variable [As type] [, [ByVal] variable [As type] ]..])]
```

10.3.2　建立工程项目

下面将会介绍一个串口实例：串口通信演示 democom，介绍利用 API 建立串口通信。本

项目类似一个简单的超级终端。

democom 所能实现的功能是：

- 程序开始时打开串口。
- 配制串口。
- 异步发送数据。
- 定时检测数据读写。
- 定时检测错误信息。
- 程序结束时关闭串口。

该实例建立的 VB 工程一共有两个类模块（dwCom 模块类和 dwDCB 类模块），两个窗体（frmCommDemo 窗体和 frmCommcfg 窗体）和一个模块（CommInfo 模块）。

其中 frmCommDemo 为主窗体，运行时界面如图 10-9 所示。

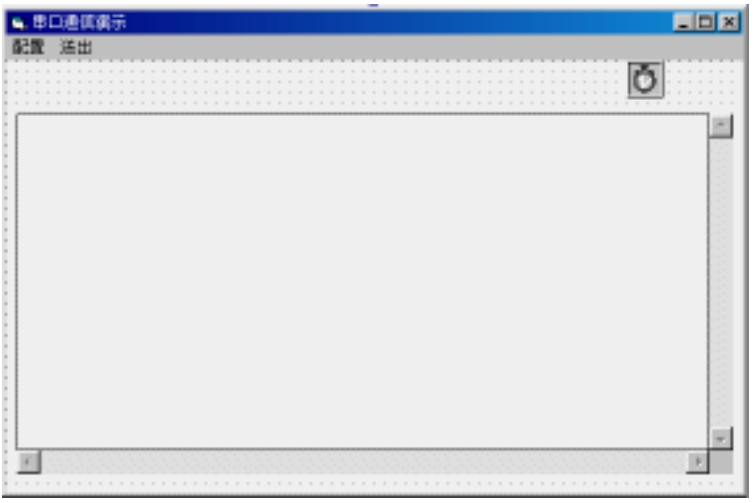


图 10-9　程序的主界面

10.3.3　代码分析

1. 类模块 dwDCB

dwDCB 类定义了一个设备控制块类，目的在于分解 DCB 的 fBitFields 属性和设定获得串口状态，以及检测各个属性的合法性等功能。

```
'dwDCB -设备控制块类 目的：分解 fBitFields 属性和设定获得串口状态
Option Explicit

Private Type DCB
        DCBlength As Long
        BaudRate As Long
        fBitFields As Long 'See Comments in Win32API.Txt
        wReserved As Integer
        XonLim As Integer
```

```
        XoffLim As Integer
        ByteSize As Byte
        Parity As Byte
        StopBits As Byte
        XonChar As Byte
        XoffChar As Byte
        ErrorChar As Byte
        EofChar As Byte
        EvtChar As Byte
        wReserved1 As Integer 'Reserved; Do Not Use
End Type

'定义了一个 DCB 成员
Private DCBStr As DCB

'定义例如缓冲区长度
Private BufferSize As Integer

Private Const ERR_INVALIDPROPERTY = 31000
Private Const CLASS_NAME$ = "dwDCB"

'与 fBitFields 属性相与的标志，参见 fBitFields 属性的说明
Private Const FLAG_fBinary& = &H1
Private Const FLAG_fParity& = &H2
Private Const FLAG_fOutxCtsFlow = &H4
Private Const FLAG_fOutxDsrFlow = &H8
Private Const FLAG_fDtrControl = &H30
Private Const FLAG_fDsrSensitivity = &H40
Private Const FLAG_fTXContinueOnXoff = &H80
Private Const FLAG_fOutX = &H100
Private Const FLAG_fInX = &H200
Private Const FLAG_fErrorChar = &H400
Private Const FLAG_fNull = &H800
Private Const FLAG_fRtsControl = &H3000
Private Const FLAG_fAbortOnError = &H4000
Private Declare Function apiSetCommState Lib "kernel32" Alias "SetCommState"
(ByVal hCommDev As Long, lpDCB As DCB) As Long
Private Declare Function apiGetCommState Lib "kernel32" Alias "GetCommState"
(ByVal nCid As Long, lpDCB As DCB) As Long
```

上面的代码用于声明所用到的 API 函数，变量和结构，其中 FLAG_fBinar 等的 API 常数将用于分解分解 DCB 的 fBitFields 属性到具体的某个属性。

```
Private Sub Class_Initialize()
    '设置长度
    DCBStr.DCBlength = Len(DCBStr)
    ' 默认值
    BufferSize = 2048
    fParity = False
    fOutxCtsFlow = True
    fOutxDsrFlow = True
    fDtrControl = 1
    fDsrSensitivity = True
    fTXContinueOnXoff = True
    fOutX = True
    fInX = True
    fErrorChar = True
    fNull = True
    fRtsControl = 1
    fAbortOnError = True
    DCBStr.XonLim = 100
    DCBStr.XoffLim = BufferSize - 100
    DCBStr.ByteSize = 8
    DCBStr.Parity = 0
    DCBStr.StopBits = 0
    DCBStr.XonChar = 17
    DCBStr.XoffChar = 19
    DCBStr.ErrorChar = Asc("~")
    DCBStr.EofChar = 26 ' ^Z
    DCBStr.EvtChar = 255
    DCBStr.BaudRate = 2400
End Sub
```

上面的代码说明了类初始化时，串口默认的值，该值一般是 Windows 推荐的使用的。

```
Public Property Get BaudRate() As Long
    BaudRate = DCBStr.BaudRate
End Property

'检查波特率的合法性
Public Property Let BaudRate(vNewValue As Long)
    Select Case vNewValue
```

```
            Case 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000,
57600, 115200, 128000, 256000
                DCBStr.BaudRate = vNewValue
            Case Else
                Err.Raise vbObjectError + ERR_INVALIDPROPERTY, CLASS_NAME, "无效
的波特率设定"
        End Select
    End Property
```

以上代码说明了 DCB 结构的数据传输率的控制,因为波特率只有在设置为 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000 时有效，不然会产生不可预料的错误，因而波特率的合法性要检查，如果出错，要将出错信息返回给用户。

```
    '分解 fBitFields 结构
    Public Property Get fParity() As Boolean
        If DCBStr.fBitFields And FLAG_fParity Then
            fParity = True
        End If
    End Property

    Public Property Let fParity(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fParity)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or FLAG_fParity
    End Property

    Public Property Get fOutxCtsFlow() As Boolean
        If DCBStr.fBitFields And FLAG_fOutxCtsFlow Then
            fOutxCtsFlow = True
        End If
    End Property

    Public Property Let fOutxCtsFlow(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fOutxCtsFlow)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or
FLAG_fOutxCtsFlow
    End Property

    Public Property Get fOutxDsrFlow() As Boolean
        If DCBStr.fBitFields And FLAG_fOutxDsrFlow Then
```

```
            fOutxDsrFlow = True
        End If
    End Property
    Public Property Let fOutxDsrFlow(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fOutxDsrFlow)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or
FLAG_fOutxDsrFlow
    End Property

    Public Property Get fDtrControl() As Integer
        Dim ival&
        ival = DCBStr.fBitFields And FLAG_fDtrControl
        fDtrControl = ival \ 16    ' Shift right 4 bits
    End Property

    ' 0 to disable, 1 to enable, 2 for handshake mode
    Public Property Let fDtrControl(vNewValue As Integer)
      If vNewValue < 0 Or vNewValue > 2 Then
        Err.Raise vbObjectError + ERR_INVALIDPROPERTY, CLASS_NAME, "无效的 DTR 控
制设定"
      End If
      DCBStr.fBitFields = DCBStr.fBitFields And FLAG_fDtrControl
      DCBStr.fBitFields = DCBStr.fBitFields Or (vNewValue * 16)
    End Property

    Public Property Get fDsrSensitivity() As Boolean
        If DCBStr.fBitFields And FLAG_fDsrSensitivity Then
            fDsrSensitivity = True
        End If
    End Property

    Public Property Let fDsrSensitivity(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fDsrSensitivity)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or
FLAG_fDsrSensitivity
    End Property


    Public Property Get fTXContinueOnXoff() As Boolean
```

```
        If DCBStr.fBitFields And FLAG_fTXContinueOnXoff Then
            fTXContinueOnXoff = True
        End If
    End Property

    Public Property Let fTXContinueOnXoff(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fTXContinueOnXoff)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or
FLAG_fTXContinueOnXoff
    End Property

    Public Property Get fOutX() As Boolean
        If DCBStr.fBitFields And FLAG_fOutX Then
            fOutX = True
        End If
    End Property

    Public Property Let fOutX(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fOutX)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or FLAG_fOutX
    End Property

    Public Property Get fInX() As Boolean
        If DCBStr.fBitFields And FLAG_fInX Then
            fInX = True
        End If
    End Property

    Public Property Let fInX(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fInX)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or FLAG_fInX
    End Property

    Public Property Get fErrorChar() As Boolean
        If DCBStr.fBitFields And FLAG_fErrorChar Then
            fErrorChar = True
        End If
    End Property
```

```
    Public Property Let fErrorChar(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fErrorChar)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or FLAG_fErrorChar
    End Property

    Public Property Get fNull() As Boolean
        If DCBStr.fBitFields And FLAG_fNull Then
            fNull = True
        End If
    End Property

    Public Property Let fNull(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fNull)
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or FLAG_fNull
    End Property

    Public Property Get fRtsControl() As Integer
        Dim ival&
        ival = DCBStr.fBitFields And FLAG_fRtsControl
        fRtsControl = ival \ &H1000    ' Shift right 4 bits
    End Property

    Public Property Let fRtsControl(vNewValue As Integer)
       If vNewValue < 0 Or vNewValue > 3 Then
         Err.Raise vbObjectError + ERR_INVALIDPROPERTY, CLASS_NAME, "无效 RTS 控制
设定"
       End If
       DCBStr.fBitFields = DCBStr.fBitFields And FLAG_fRtsControl
       DCBStr.fBitFields = DCBStr.fBitFields Or (vNewValue * &H1000)
    End Property

    Public Property Get fAbortOnError() As Boolean
        If DCBStr.fBitFields And FLAG_fAbortOnError Then
            fAbortOnError = True
        End If
    End Property

    Public Property Let fAbortOnError(vNewValue As Boolean)
        DCBStr.fBitFields = DCBStr.fBitFields And (Not FLAG_fAbortOnError)
```

```
        If vNewValue Then DCBStr.fBitFields = DCBStr.fBitFields Or
FLAG_fAbortOnError
    End Property
```

以上代码是 dwDCB **类存在的关键所在；通过这段代码将** DCB **结构的** fBitFields **成员分解为** dwDCB **类的多个成员，这样，应用程序可以通过直接访问** dwDCB **类的某个成员来控制串口的某种属性。**

```
    Public Property Get XonLim() As Integer
        XonLim = DCBStr.XonLim
    End Property

    Public Property Let XonLim(vNewValue As Integer)
        DCBStr.XonLim = vNewValue
    End Property

    Public Property Get XoffLim() As Integer
        XoffLim = DCBStr.XoffLim
    End Property

    Public Property Let XoffLim(vNewValue As Integer)
        DCBStr.XoffLim = vNewValue
    End Property

    Public Property Get ByteSize() As Byte
        ByteSize = DCBStr.ByteSize
    End Property

    Public Property Let ByteSize(vNewValue As Byte)
        If vNewValue < 4 Or vNewValue > 8 Then
            Err.Raise vbObjectError + ERR_INVALIDPROPERTY, CLASS_NAME, "无效的字
节长设定"
        End If
        DCBStr.ByteSize = vNewValue
    End Property

    Public Property Get Parity() As Byte
        Parity = DCBStr.Parity
    End Property

    ’ 0 - 4 = No, odd, even, mark, space
```

```
    Public Property Let Parity(vNewValue As Byte)
      If vNewValue < 0 Or vNewValue > 4 Then
        Err.Raise vbObjectError + ERR_INVALIDPROPERTY, CLASS_NAME, "无效的奇偶检
验设定"
      End If
      DCBStr.Parity = vNewValue
    End Property

    Public Property Get StopBits() As Byte
       StopBits = DCBStr.StopBits
    End Property

    ' 0 = 1, 1 = 1.5, 2 = 2
    Public Property Let StopBits(vNewValue As Byte)
       If vNewValue < 0 Or vNewValue > 4 Then
         Err.Raise vbObjectError + ERR_INVALIDPROPERTY, CLASS_NAME, "无效的停止
位设定"
       End If
       DCBStr.StopBits = vNewValue
    End Property

    Public Property Get XonChar() As String
       XonChar = Chr$(DCBStr.XonChar)
    End Property

    Public Property Let XonChar(vNewValue As String)
       DCBStr.XonChar = Asc(vNewValue)
    End Property

    Public Property Get XoffChar() As String
       XoffChar = Chr$(DCBStr.XoffChar)
    End Property

    Public Property Let XoffChar(vNewValue As String)
       DCBStr.XoffChar = Asc(vNewValue)
    End Property

    Public Property Get ErrorChar() As String
       ErrorChar = Chr$(DCBStr.ErrorChar)
```

```
End Property

Public Property Let ErrorChar(vNewValue As String)
    DCBStr.ErrorChar = Asc(vNewValue)
End Property

Public Property Get EofChar() As String
    EofChar = Chr$(DCBStr.EofChar)
End Property

Public Property Let EofChar(vNewValue As String)
    DCBStr.EofChar = Asc(vNewValue)
End Property

Public Property Get EvtChar() As String
    EvtChar = Chr$(DCBStr.EvtChar)
End Property

Public Property Let EvtChar(vNewValue As String)
    DCBStr.EvtChar = Asc(vNewValue)
End Property
```

以上代码说明了 DCB **结构的其他几个成员的控制，包括** XonLim、XoffLim、ByteSize、Parity，StopBits、XonChar、XoffChar、ErrorChar、EofChar **和** EvtChar **等，并加上了相应的合法性检查。**

```
' 获得串口状态
Public Function GetCommState(Comm As dwCom) As Boolean
    Dim res&
       If Comm.hCommDev = 0 Then Exit Function
    res = apiGetCommState(Comm.hCommDev, DCBStr)
    GetCommState = res <> 0
End Function

'设定串口
Public Function SetCommState(Comm As dwCom) As Boolean
    Dim res&
    If Comm.hCommDev = 0 Then Exit Function
    res = apiSetCommState(Comm.hCommDev, DCBStr)
    SetCommState = res <> 0
End Function
```

以上代码是获取和配置串口状态，通过 SetCommState 函数和 GetCommState 函数的调用实现。

2. 类模块 dwCom

dwCom 类定义了是一个串口通信类，串口的打开、关闭、定时读写和定时检测时间都在此类中完成。其中 OpenComm、CommOutput 时打开串口和发送数据的函数，以 Detect 开头的函数和过程是定时查询功能。该函数中所使用的函数正是前面所介绍的精华，读者应该仔细的阅读和领会。

```
' dwCom-串口通信类
Option Explicit
'超时结构
Private Type COMMTIMEOUTS
        ReadIntervalTimeout As Long
        ReadTotalTimeoutMultiplier As Long
        ReadTotalTimeoutConstant As Long
        WriteTotalTimeoutMultiplier As Long
        WriteTotalTimeoutConstant As Long
End Type
' 私有超时成员
Private timeouts As COMMTIMEOUTS
'重叠结构
Private Type OVERLAPPED
        Internal As Long
        InternalHigh As Long
        offset As Long
        OffsetHigh As Long
        hEvent As Long
End Type
'Com1 和 Com2 以及其他的一些兼容的设备（如串口卡等）；
Private devname$
' Comm 句柄 Comm handle
Private handle As Long
' 共有成员
Public dcbclass As dwDCB
'Current state indicators
Private PendingOutput$
Private CurrentEventMask&
'输入输出缓冲区
Private CurrentInputBuffer&
```

```
    Private CurrentOutputBuffer&
    '数据传递数目
    Private DataWritten&
    Private DataRead&
    Private EventResults&
    Private TriggeredEvents&    '装载事件结果的变量
    ' 重叠结构的数组
    ' 0 ：读操作, 1 = 写操作, 2 = 等待事件
    Private overlaps(2) As OVERLAPPED
    ' 表明后台操作是否在进行
    ' 0 ：读操作, 1 = 写操作, 2 = 等待事件
    Private inprogress(2) As Boolean
    Private CallbackObject As Object
```

以上代码首先申明和定义了超时结构，用于异步函数的时间控制。而后声明了 3 个重叠结构成员，定义了 3 个重叠结构，是因为有 3 种状态，读写或者等待事件状态，所以需要 3 个重叠结构进行控制。同时还定义了后台操作是否在进行的布尔变量。

```
    ' API 的声明
    Private Declare Function lstrcpyFromBuffer Lib "kernel32" Alias "lstrcpynA"
(ByVal lpString1 As String, ByVal buffer As Long, ByVal iMaxLength As Long) As
Long
    Private Declare Function lstrcpyToBuffer Lib "kernel32" Alias "lstrcpynA"
(ByVal buffer As Long, ByVal lpString2 As String, ByVal iMaxLength As Long) As
Long
```

以上代码声明 lstrcpyToBuffer 和 lstrcpyFromBuffer 函数，用于将输出字符串拷贝到输出缓冲区，将输入缓冲区内的字符拷贝输出字符串中。

```
    Private Declare Function lstrlen Lib "kernel32" Alias "lstrlenA" (ByVal lpString
As String) As Long
    Private Declare Function GetLastError Lib "kernel32" () As Long
    Private Declare Function CreateEvent Lib "kernel32" Alias "CreateEventA" (ByVal
lpEventAttributes As Long, ByVal bManualReset As Long, ByVal bInitialState As Long,
ByVal lpName As String) As Long
    Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle
As Long, ByVal dwMilliseconds As Long) As Long
```

以上代码声明了用于事件处理函数以及获得错误信息的函数。

```
    Private Declare Function WriteFile Lib "kernel32" (ByVal hFile As Long, ByVal
lpBuffer As Long, ByVal nNumberOfBytesToWrite As Long, lpNumberOfBytesWritten As
Long, lpOverlapped As OVERLAPPED) As Long
    Private Declare Function ReadFile Lib "kernel32" (ByVal hFile As Long, ByVal
lpBuffer As Long, ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long,
```

```
lpOverlapped As OVERLAPPED) As Long
    Private Declare Function SetCommMask Lib "kernel32" (ByVal hFile As Long, ByVal
dwEvtMask As Long) As Long
    Private Declare Function ClearCommError Lib "kernel32" (ByVal hFile As Long,
lpErrors As Long, ByVal l As Long) As Long
    Private Declare Function WaitCommEvent Lib "kernel32" (ByVal hFile As Long,
lpEvtMask As Long, lpOverlapped As OVERLAPPED) As Long
    Private Declare Function apiSetCommTimeouts Lib "kernel32" Alias
"SetCommTimeouts" (ByVal hFile As Long, lpCommTimeouts As COMMTIMEOUTS) As Long
    Private Declare Function apiGetCommTimeouts Lib "kernel32" Alias
"GetCommTimeouts" (ByVal hFile As Long, lpCommTimeouts As COMMTIMEOUTS) As Long
    Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long)
As Long
    Private Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" (ByVal
lpFileName As String, ByVal dwDesiredAccess As Long, ByVal dwShareMode As Long,
ByVal lpSecurityAttributes As Long, ByVal dwCreationDisposition As Long, ByVal
dwFlagsAndAttributes As Long, ByVal hTemplateFile As Long) As Long
    Private Declare Function SetupComm Lib "kernel32" (ByVal hFile As Long, ByVal
dwInQueue As Long, ByVal dwOutQueue As Long) As Long
    Private Declare Function GetCommModemStatus Lib "kernel32" (ByVal hFile As
Long, lpModemStat As Long) As Long
```

以上代码声明了本应用程序用到的通信 API 函数。

```
    Private Declare Function GlobalAlloc Lib "kernel32" (ByVal wFlags As Long,
ByVal dwBytes As Long) As Long
    Private Declare Function GlobalFree Lib "kernel32" (ByVal hMem As Long) As
Long
```

以上代码声明了两个为读写函数的输入输出缓冲区分配和释放内存空间的函数。程序启动是应该分配足够的内存空间该输入输出缓冲区，但是千万别忘记在程序结束时释放这些内存空间。

```
    'API 常量的声明
    Private Const GENERIC_READ = &H80000000
    Private Const GENERIC_WRITE = &H40000000
    Private Const OPEN_EXISTING = 3
    Private Const FILE_FLAG_OVERLAPPED = &H40000000
    Private Const INVALID_HANDLE_VALUE = -1
    Private Const GMEM_FIXED = &H0
    Private Const ClassBufferSizes% = 1024
    Private Const ERROR_IO_PENDING = 997 '  dderror
    Private Const WAIT_TIMEOUT = &H102&
```

```
Private Const CLASS_NAME$ = "dwCom"
Private Const ERR_NOCOMMACCESS = 31010
Private Const ERR_UNINITIALIZED = 31011
Private Const ERR_MODEMSTATUS = 31012
Private Const ERR_READFAIL = 31013
Private Const ERR_EVENTFAIL = 31014

Private Const EV_RXCHAR = &H1              ' Any Character received
Private Const EV_RXFLAG = &H2              ' Received certain character
Private Const EV_TXEMPTY = &H4             ' Transmitt Queue Empty
Private Const EV_CTS = &H8                 ' CTS changed state
Private Const EV_DSR = &H10                ' DSR changed state
Private Const EV_RLSD = &H20               ' RLSD changed state
Private Const EV_BREAK = &H40              ' BREAK received
Private Const EV_ERR = &H80                ' Line status error occurred
Private Const EV_RING = &H100              ' Ring signal detected
Private Const EV_PERR = &H200              ' Printer error occured
Private Const EV_RX80FULL = &H400          ' Receive buffer is 80 percent full
Private Const EV_EVENT1 = &H800            ' Provider specific event 1
Private Const EV_EVENT2 = &H1000           ' Provider specific event 2

Private Const CE_RXOVER = &H1              ' Receive Queue overflow
Private Const CE_OVERRUN = &H2             ' Receive Overrun Error
Private Const CE_RXPARITY = &H4            ' Receive Parity Error
Private Const CE_FRAME = &H8               ' Receive Framing error
Private Const CE_BREAK = &H10              ' Break Detected
Private Const CE_TXFULL = &H100            ' TX Queue is full
Private EmptyString As String * 1
```

以上代码声明了本应用程序用到的通信 API 常量，在 VB 调用 API 就是这样一步一步调用的。

```
'初始化类模块
Private Sub Class_Initialize()
    Dim olnum%
    Set dcbclass = New dwDCB
    CurrentInputBuffer = GlobalAlloc(GMEM_FIXED, ClassBufferSizes + 1)
    CurrentOutputBuffer = GlobalAlloc(GMEM_FIXED, ClassBufferSizes + 1)
    CurrentEventMask = EV_ERR
    EmptyString = Chr$(0)
```

```
    ' 创建重叠结构事件对象
    For olnum = 0 To 2
        overlaps(olnum).hEvent = CreateEvent(0, True, False, vbNullString)
    Next olnum
End Sub

Private Sub Class_Terminate()
    Dim olnum
    '关闭已打开的
    Call CloseComm
    ' 关闭事件对象的句柄
    For olnum = 0 To 2
        Call CloseHandle(overlaps(olnum).hEvent)
    Next olnum
    Set dcbclass = Nothing   ' Be sure dcbclass is free
    Call GlobalFree(CurrentInputBuffer)
    Call GlobalFree(CurrentOutputBuffer)
 End Sub
```

以上 Class_Initialize **过程是初始化** dwCom **类，程序首先先声明一个** dwDCB **类，用于串口信息，而后分配输入输出缓冲区内存空间大小，创建重叠结构事件对象。**Class_Terminate(**过程是程序结束时的清扫工作，首先如果串口已经打开，就关闭它，接着销毁重叠结构事件对象句柄，然后无效** dwDCB **类，最后释放输入输出缓冲区内存空间；这样本应用程序就完成了内存的释放工作。**Class_Initialize **过程还定义了一个** EV_ERR **错误掩膜事件，它表示只要发生了线路状态错误，即** CE_FRAME**（帧错误）、** CE_OVERRUN**（接收缓冲区超限）和** CE_RXPARITY**（奇偶校验错误），就有通信事件产生。**

```
' 错误消息
Private Sub DeviceNotOpenedError()
    Err.Raise vbObjectError + ERR_UNINITIALIZED, CLASS_NAME, "串口设备不能打开
"
End Sub
```

以上 DeviceNotOpenedError **过程当串口句柄无效时显示的错误消息，一般是硬件问题造成的。**

```
'-----------------------------------------------
'超时数据获取
'-----------------------------------------------
Public Property Get ReadIntervalTimeout() As Long
    ReadIntervalTimeout = timeouts.ReadIntervalTimeout
End Property
```

```
Public Property Let ReadIntervalTimeout(vNewValue As Long)
    timeouts.ReadIntervalTimeout = vNewValue
End Property

Public Property Get ReadTotalTimeoutMultiplier() As Long
    ReadTotalTimeoutMultiplier = timeouts.ReadTotalTimeoutMultiplier
End Property

Public Property Let ReadTotalTimeoutMultiplier(vNewValue As Long)
    timeouts.ReadTotalTimeoutMultiplier = vNewValue
End Property

Public Property Get ReadTotalTimeoutConstant() As Long
    ReadTotalTimeoutConstant = timeouts.ReadTotalTimeoutConstant
End Property

Public Property Let ReadTotalTimeoutConstant(vNewValue As Long)
    timeouts.ReadTotalTimeoutConstant = ReadTotalTimeoutConstant
End Property

Public Property Get WriteTotalTimeoutMultiplier() As Long
    WriteTotalTimeoutMultiplier = timeouts.WriteTotalTimeoutMultiplier
End Property

Public Property Let WriteTotalTimeoutMultiplier(vNewValue As Long)
    timeouts.WriteTotalTimeoutMultiplier = WriteTotalTimeoutMultiplier
End Property

Public Property Get WriteTotalTimeoutConstant() As Long
    WriteTotalTimeoutConstant = timeouts.WriteTotalTimeoutConstant
End Property

Public Property Let WriteTotalTimeoutConstant(vNewValue As Long)
    timeouts.WriteTotalTimeoutConstant = WriteTotalTimeoutConstant
End Property
```

以上代码是超时结构成员的控制。

```
'赋句柄
Public Property Get hCommDev() As Long
```

```
        hCommDev = handle
    End Property

    '赋设备
    Public Property Get DeviceName() As String
        DeviceName = devname
    End Property
```

以上代码是串口句柄的控制，API 串口操作时，总要用到句柄的，因而句柄对串口函数的调用很重要。

```
    Public Sub GetCommTimeouts()
        ' 如果超时报告超时错误时
        If handle = 0 Then Exit Sub
        Call apiGetCommTimeouts(handle, timeouts)
    End Sub
    '设置超时
    Public Function SetCommTimeouts() As Long
        If handle = 0 Then Exit Function
        SetCommTimeouts = apiSetCommTimeouts(handle, timeouts) <> 0
    End Function
```

GetCommTimeouts 过程和 SetCommTimeouts()过程只是简单的调用了 API 的超时结构控制函数，功能没有变化。

```
    '打开串口以及初始化操作
    Public Function OpenComm(CommDeviceName As String, Notify As Object, Optional
cbInQueue, Optional cbOutQueue) As Long
        ' Close an existing port when reopening
        If handle <> 0 Then CloseComm
        devname = CommDeviceName
        Set CallbackObject = Notify
        handle = CreateFile(devname, GENERIC_READ Or GENERIC_WRITE, 0, 0,
OPEN_EXISTING, FILE_FLAG_OVERLAPPED, 0)
        If handle = INVALID_HANDLE_VALUE Then Err.Raise vbObjectError +
ERR_NOCOMMACCESS, CLASS_NAME, "不能打开串口"
        ' 如果没设置串口的输入输出缓冲区，就设置串口的输入输出缓冲区
        If Not (IsMissing(cbInQueue) Or IsMissing(cbOutQueue)) Then
            Call SetupComm(handle, cbInQueue, cbOutQueue)
        Else
            Call SetupComm(handle, 8192, 1024)
        End If
        ' 获得超时结构
```

```
    GetCommTimeouts
    '设置超时
    timeouts.ReadIntervalTimeout = 1
    timeouts.ReadTotalTimeoutMultiplier = 0
    timeouts.ReadTotalTimeoutConstant = 10
    timeouts.WriteTotalTimeoutMultiplier = 1
    timeouts.WriteTotalTimeoutConstant = 1
    SetCommTimeouts
    ' 设定串口类
    Call dcbclass.GetCommState(Me)
   ' 设置通信事件掩膜
    Call SetCommMask(handle, CurrentEventMask)
    StartInput
End Function
```

OpenComm 函数是一个重要的函数，它起的作用就是用 CreateFile 打开串口，用 SetupComm 设定串口缓冲大小，用 SetCommTimeouts 设置超时结构，用 GetCommState 获得打开的串口信息，以及用 SetCommMask 设置 EV_ERR 掩膜事件。

✍ 注意，如果打开已经打开的串口，应用程序将会显示“不能打开串口”错误。如图 10-10 所示。

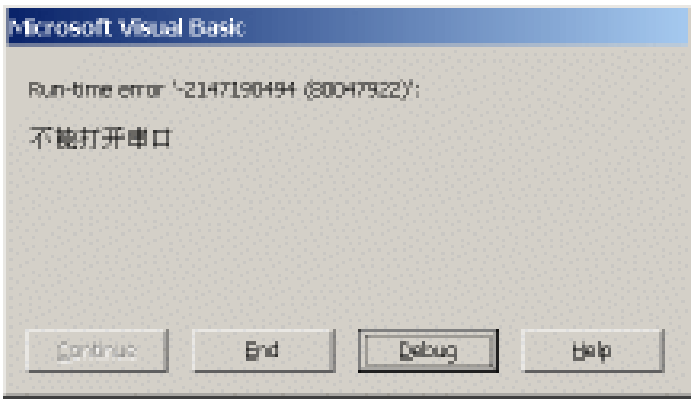


图 10-10 不能打开串口错误提示框

```
'关闭串口
Public Function CloseComm() As Long
    If handle = 0 Then Exit Function
    Set CallbackObject = Nothing
    Call CloseHandle(handle)
    handle = 0
End Function
```

CloseComm 用于关闭串口，关闭串口实际上很简单，就是调用 CloseHandle 关闭打开的

串口句柄。

```
' 获得串口状态
Public Function GetCommState() As Long
    If handle = 0 Then DeviceNotOpenedError
    GetCommState = dcbclass.GetCommState(Me)
End Function

'设置串口状态
Public Function SetCommState() As Long
    If handle = 0 Then DeviceNotOpenedError
    SetCommState = dcbclass.SetCommState(Me)
End Function
```

以上 GetCommState 函数和 SetCommState 函数用于设置和配置串口信息，它们通过直接调用通信 API 函数实现。

```
' 输出数据函数
Public Function CommOutput(outputdata As String) As Long
    Dim bytestosend&
    Dim res&
    Dim addnull As Boolean
    If handle = 0 Then DeviceNotOpenedError
    PendingOutput = PendingOutput & outputdata
    If inprogress(1) Then    ' 写操作正在进行
        CommOutput = True    '退出
        Exit Function
    End If
    ' 开始一个新的写操作
    bytestosend = Len(PendingOutput)
    '无数据输出，退出
    If bytestosend = 0 Then
        CommOutput = True
        Exit Function
    End If
    '防止缓冲区过界
    If bytestosend > ClassBufferSizes Then bytestosend = ClassBufferSizes
    ' null 字符，也送
    If lstrlen(PendingOutput) < bytestosend Then
        bytestosend = lstrlen(PendingOutput)
        addnull = True '表明送出 null 字符
    End If
```

```
        '复制到缓冲区
       If bytestosend > 0 Then Call lstrcpyToBuffer(CurrentOutputBuffer,
PendingOutput, bytestosend + 1)
       If bytestosend = Len(PendingOutput) Then
           PendingOutput = ""
        Else
           PendingOutput = Mid(PendingOutput, bytestosend + 1)
        End If
        If addnull Then bytestosend = bytestosend + 1
        '送出数据
        res = WriteFile(handle, CurrentOutputBuffer, bytestosend, DataWritten,
overlaps(1))
        If res <> 0 Then
           '正常结束，处理后续
           ProcessWriteComplete
           CommOutput = True
        Else
           '函数返回时操作任在继续
           If GetLastError() = ERROR_IO_PENDING Then
              inprogress(1) = True
              CommOutput = True
              #If DEBUGMODE Then
                 Debug.Print "后台正在写"
              #End If
           End If
        End If
    End Function

    '写后处理
    Public Sub ProcessWriteComplete()
        inprogress(1) = False
        '通过 CommOutput 函数退出写操作
        Call CommOutput("")
    End Sub
```

以上 CommOutput **函数用于输出数据，该函数的输出操作采用** WriteFile **函数实现，但在输出前，所要输出的字符串要拷贝到输出缓冲区；**WriteFile **函数调用后，要判断返回值，如果是返回不为零，则函数执行成功，并将设定标识写后台操作是否在进行的布尔变量为** FALSE**。如果为零，则要判断是不是后台输出状态，如果是，设定标识写后台操作是否在进行的布尔变量为** True**。**

```
'检测所有的后台操作
Public Sub Detect()
    DetectWrite
    DetectRead
    DetectEvent
End Sub
```

Detect 过程被定时器定时调用，用于定时检测数据读写，定时检测错误信息，该函数调用下面将要介绍的 DetectWrite、DetectRead、DetectEvent 过程来实现这些功能。

```
'定时查看写操作
Public Sub DetectWrite()
    Dim res&
'检查后台是不是正在写
    If Not inprogress(1) Then Exit Sub
    '检查事件
    res = WaitForSingleObject(overlaps(1).hEvent, 0)
    ' 如果没有结束信号，就退出
    If res = WAIT_TIMEOUT Then Exit Sub
    '否则数据已经写出，调用写后处理
    ProcessWriteComplete
End Sub
```

以上代码是定时检测数据写的过程,因为写操作是对用户程序是主动的,因而 DetectWrite 过程的主要目的是处理完成异步写操作。DetectWrite 过程的处理步骤是，先判断是不是后台正在写，如果不是，则该过程无事可做，退出；如果后台正在写，则该过程检查以下写重叠结构的事件信号，判断写操作有没有完成，如果没有，退出，否则调用 ProcessWriteComplete 进行一些设置工作。

```
' 控制数据的传递
Private Sub StartInput()
    Dim res&
    ' 写操作正在进行，退出
    If inprogress(0) Then Exit Sub
    If handle = 0 Then DeviceNotOpenedError
    '读数据
    res = ReadFile(handle, CurrentInputBuffer, ClassBufferSizes, DataRead,
overlaps(0))
    If res <> 0 Then
        '读后处理
        ProcessReadComplete
    Else
       '如果函数返回时操作任在继续
```

```
            If GetLastError() = ERROR_IO_PENDING Then
                inprogress(0) = True
                #If DEBUGMODE Then
                   Debug.Print "后台正在读"
                #End If
            '如果出错
            Else
                Err.Raise vbObjectError + ERR_READFAIL, CLASS_NAME, "串口读操作失败
"
            End If
        End If
    End Sub
```

以上代码是数据读操作的过程，处理时，先判断是不是后台正在读，如果是，则该过程只有退出；如果没有后台读，则该过程可以使用 ReadFile 从串口中读入数据，如果返回不为零，则函数执行成功，并将设定标识读后台操作是否在进行的布尔变量为 FALSE。；如果为零，则要判断是不是后台读入状态，如果是，设定标识读后台操作是否在进行的布尔变量为 True；否则 ReadFile 函数出错，显示错误信息。

```
    Public Sub DetectRead()
        Dim res&
        If Not inprogress(0) Then
            StartInput
            Exit Sub
        End If

        '检查读事件
        res = WaitForSingleObject(overlaps(0).hEvent, 0)
        ' 如果上次读操作还没结束，则退出
        If res = WAIT_TIMEOUT Then Exit Sub
        ' 读后处理
        ProcessReadComplete
    End Sub
```

以上代码是定时检测数据读操作的过程，因为读操作是对用户程序是被动的，因而 DetectRead 过程的目的开始或完成异步读操作。DetectRead 过程的处理步骤是，先判断有没有后台读操作，如果没有，则调用 StartInput 开始读入，然后退出；如果后台有读操作，则该过程检查以下写重叠结构的事件信号，判断写操作有没有完成，如果没有，则退出；否则调用 ProcessReadComplete 进行一些设置工作。

```
    ' 读后处理
    Public Sub ProcessReadComplete()
        Dim resstring$
```

```
    Dim copied&
    '读结束了，从重叠结构中获取所读的数目
    If inprogress(0) Then
       DataRead = overlaps(0).InternalHigh
       inprogress(0) = False
    End If
    If DataRead <> 0 Then
      #If DEBUGMODE Then
       Debug.Print "读出" & DataRead & " 字节"
      #End If
       resstring$ = String$(DataRead + 1, 0)
       copied = lstrcpyFromBuffer(resstring, CurrentInputBuffer, DataRead +
1)
       Debug.Print resstring
       '输出到界面上
       If Not (CallbackObject Is Nothing) Then
          Call CallbackObject.ShowInput(Me, Left$(resstring, DataRead))
       End If
    End If
End Sub
```

ProcessReadComplete **函数的主要功能是将输入缓冲区的数据转化为字符串，用来显示在主界面上。输入的长度是利用标识读重叠结构的** InternalHigh **成员获得的。**

```
'检测串口错误
Private Sub StartEventWatch()
    Dim res&
    ' 如果检测已经在进行，则退出本次检测
    If inprogress(2) Then Exit Sub
    If handle = 0 Then DeviceNotOpenedError
    EventResults = 0
    '检测串口
    res = WaitCommEvent(handle, EventResults, overlaps(2))
    If res <> 0 Then
        '返回正确时
       ProcessEventComplete
    Else
       '判断有没有后台 I/O 正在进行
       If GetLastError() = ERROR_IO_PENDING Then
          inprogress(2) = True
          #If DEBUGMODE Then
```

```
                Debug.Print "后台正在等待事件"
            #End If
         '出错
         Else
            Err.Raise vbObjectError + ERR_EVENTFAIL, CLASS_NAME, "串口设备出错
"
         End If
      End If
   End Sub
```

以上代码是监视通信事件的操作的过程，因为已经设置了 EV_ERR 掩膜事件，这样实际上监视的是线路错误时间。该过程调用 WaitCommEvent 来监视错误事件，如果返回不为零，则函数执行成功，并将设定标识监视事件后台操作是否在进行的布尔变量为 FALSE；如果为零，则要判断是不是正在等待监视事件状态，如果是，设定监视事件后台操作是否在进行的布尔变量为 True；否则，就是串口设备出错，应该显示错误信息。

```
   '事件结束
   Private Sub ProcessEventComplete()
      Dim errors&
      If inprogress(2) Then ' Was overlapped
         inprogress(2) = False
      End If
      If EventResults <> 0 Then
         #If DEBUGMODE Then
          Debug.Print "事件值 " & Hex$(EventResults)
          #End If
         If Not (CallbackObject Is Nothing) Then
            Call ClearCommError(handle, errors, 0)
           '输出到界面的文本框
         If (errors And CE_RXOVER) <> 0 Then
   Call CallbackObject.CommEvent(Me, "接受缓冲区满错误")
         If (errors And CE_OVERRUN) <> 0 Then
   Call CallbackObject.CommEvent(Me, "接受超速出错")
         If (errors And CE_RXPARITY) <> 0 Then
    Call CallbackObject.CommEvent(Me, "奇偶检验出错")
   If (errors And CE_FRAME) <> 0 Then
    Call CallbackObject.CommEvent(Me, "帧出错")
         If (errors And CE_BREAK) <> 0 Then
    Call CallbackObject.CommEvent(Me, "探测到中断")
         If (errors And CE_TXFULL) <> 0 Then
   Call CallbackObject.CommEvent(Me, "输出满")
```

```
        End If
    End If
End Sub
```

以上 ProcessEventComplet 函数的功能是，如果有错误，就将错误转化为提示文本，输出到主界面的文本框。

```
Private Sub DetectEvent()
    Dim res&
    If Not inprogress(2) Then
        StartEventWatch
        Exit Sub
    End If
    '后台操作时，检测串口事件有没有结束
    res = WaitForSingleObject(overlaps(2).hEvent, 0)
    '没有结束,退出
    If res = WAIT_TIMEOUT Then Exit Sub
    ' '事件结束
    ProcessEventComplete
End Sub
```

以上代码是定时检测错误的过程，因为检测错误操作是对用户程序是被动的，因而 DetectEvent 过程的目的开始或完成异步写操作。DetectEvent 过程的处理步骤是，先判断有没有后台监视事件操作，如果没有，则调用 StartEventWatch 开始监视，然后退出；如果后台有监视事件操作，则该过程检查监视事件的重叠结构的事件信号，判断监视事件的操作有没有完成，如果没有，退出；否则调用 ProcessEventComplete 进行一些设置工作。

3. 窗体 frmCommDemo

该窗体上有一个 TextBox 控件，用于显示和输入传递的的数据，还有一个 Timer 控件，其他的还有一个 Timer 控件，用于定时刷新输入和输出的数据。详细内容请读者打开相应的工程项目文件查看。

```
Option Explicit
Private Sub Form_Load()
    '配置串口对话框
    ' 打开串口
    frmCommcfg.Show 1
    ShowText.Move 0, 0, ScaleWidth, ScaleHeight
End Sub
```

上面的代码用于在开始执行时显示串口配置对话框，以便应用程序打开相应的串口，进行读写，以及事件检测等。

```
Private Sub Form_Resize()
    '重新调整输出和接入文本框的大小
```

```
    ShowText.Move 0, 0, ScaleWidth, ScaleHeight
End Sub
```

上面的代码用于在将 TEXTBOX 控件居中显示，利于用户操作。

```
Private Sub Form_Unload(Cancel As Integer)
    '卸载通信类
Set Comm = Nothing
End Sub
```

上面的代码是响应窗体卸载时释放所分配的串口类的内存空间。

```
'定时器刷新
Private Sub Timer1_Timer()
     If Not (Comm Is Nothing) Then Comm.Detect '串口通信类探测
End Sub
```

上面代码是响应定时器刷新的操作，调用 dwCom 的 Detect 函数进行检测。

```
'显示配置对话框
Private Sub MenuConfigure_Click()
    frmCommcfg.Show 1
End Sub
```

上面代码是是响应点击“配置”菜单事件，弹出串口配置对话框，用户可以进行配置。

```
'文本框中输入的任何一个字符都会发送出去
Private Sub ShowText_KeyPress(KeyAscii As Integer)
    If Not (Comm Is Nothing) Then
        Comm.CommOutput (Chr$(KeyAscii))
    End If
    KeyAscii = 0
End Sub
```

上面代码是是响应在文本框里输入 ASCII 字符的事件，这时程序立刻输出这些字符。按键字符传输如图 10-11 所示。

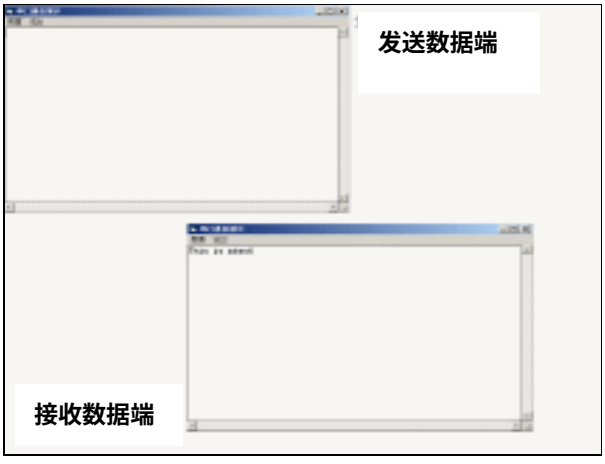


图 10-11　按键传输示意图

```
'发送一个句子
Private Sub SendMenu_Click()
    Dim dnum$
    If Not (Comm Is Nothing) Then
        ShowText.Text = ""
        dnum$ = InputBox$("写入操作者要发送的语句", "串口演示")
        Comm.CommOutput dnum$
    End If
End Sub
```

上面代码是是响应“送出”菜单事件，弹出一个输入字符串框，接受输入字符串，单击“OK”按钮后，程序将输入字符串输出。

```
'接受输入的数据
Public Sub ShowInput(thiscomm As dwCom, commdata As String)
    Dim use$
    Dim cpos%
    If commdata <> "" Then
        For cpos% = 1 To Len(commdata$)
           Select Case Asc(Mid$(commdata$, cpos%))
               Case 13
                   use$ = use$ + Chr$(13) + Chr$(10)
               Case 10
                   '舍弃换行符
               Case Else
                   use$ = use$ + Mid$(commdata$, cpos%, 1)
           End Select
       Next cpos%
       ShowText.SelStart = Len(ShowText.Text)
       ShowText.SelLength = 0
       ShowText.SelText = use$
       If Len(ShowText.Text) > 4096 Then
           ShowText.Text = Right$(ShowText.Text, 2048)
       End If
       ShowText.SelStart = Len(ShowText.Text)
     End If
End Sub
```

上面代码是是接受输入数据的处理。首先，将输入字符串中的回车符转换为换行符，输入字符串中的换行符被丢弃，然后将输入的转化到文本框中显示。

```
'显示串口出错事件
Public Sub CommEvent(thiscomm As dwCom, ev As String)
```

```
    ShowText.SelStart = Len(ShowText.Text)
    ShowText.SelText = vbCrLf & ev & vbCrLf
    ShowText.SelStart = Len(ShowText.Text)
End Sub
```

上面代码是是添加出错信息到文本框中的处理

4. **窗体** frmCommcfg

frmCommcfg **窗体是用来配置串口的窗口，它的界面如图** 10-12 **所示。**

图 10-12　democom **的配置串口窗口**

```
Option Explicit
Private CurrentBaudSetting&
'
Private Sub CmdOk_Click()
    Dim baudtouse&
    Dim DeviceName$
    '选择波特率
    Select Case CurrentBaudSetting
        Case 0
            baudtouse = 1200
        Case 1
            baudtouse = 2400
        Case 2
            baudtouse = 9600
        Case 3
            baudtouse = 14400
        Case 4
            baudtouse = 28800
        Case 5
            baudtouse = 56000
    End Select
    '选择串口
```

```
    If OptionCom1.Value Then DeviceName = "COM1" Else DeviceName = "COM2"
    '如果串口改变
    If Not (Comm Is Nothing) Then
             If DeviceName <> Comm.DeviceName Then
                Set Comm = Nothing
                Set Comm = New dwCom
             Call Comm.OpenComm(DeviceName, frmCommDemo)
       End If
    Else
       Set Comm = New dwCom
       Call Comm.OpenComm(DeviceName, frmCommDemo)
    End If
    '设置波特率等 DCB 参数
    Comm.dcbclass.BaudRate = baudtouse
    Comm.dcbclass.fNull = True
    Comm.dcbclass.fErrorChar = True
    Comm.dcbclass.ErrorChar = "~"
    '设置串口 DCB
    Call Comm.SetCommState
    Unload Me
End Sub
```

上面代码是响应“OK”按钮事件，如果改变了串口，重新初始化串口并打开串口，配置串口；如果没有改变，则重新配置串口参数。

```
Private Sub Form_Load()
    If Comm Is Nothing Then
       CurrentBaudSetting = 5
    Else
       Select Case (Comm.dcbclass.BaudRate)
          Case 1200
                CurrentBaudSetting = 0
          Case 2400
                CurrentBaudSetting = 1
          Case 9600
                CurrentBaudSetting = 2
          Case 14400
                CurrentBaudSetting = 3
          Case 28800
                CurrentBaudSetting = 4
          Case 56000
```

```
                CurrentBaudSetting = 5
            Case Else
                CurrentBaudSetting = 5
        End Select
    End If
    OptionBaud(CurrentBaudSetting).Value = True
End Sub

Private Sub OptionBaud_Click(Index As Integer)
    CurrentBaudSetting = Index
End Sub
```

上面代码是响应窗体装载和改变波特率事件。

5. **模块** CommInfo

CommInfo **模块声明了一些必须的** API **常量和应用程序变量，这里不在列出，请读者打开项目工程阅读。**

本项目可以在两部计算机上同时执行，执行时，须将两台计算机上的串口通过串口线相连接；也可以用一台计算机的两个串口通过串口线相连。连接完成后，便可以执行操作了，需要注意的是：两个串口的波特率等必须设置为一样。