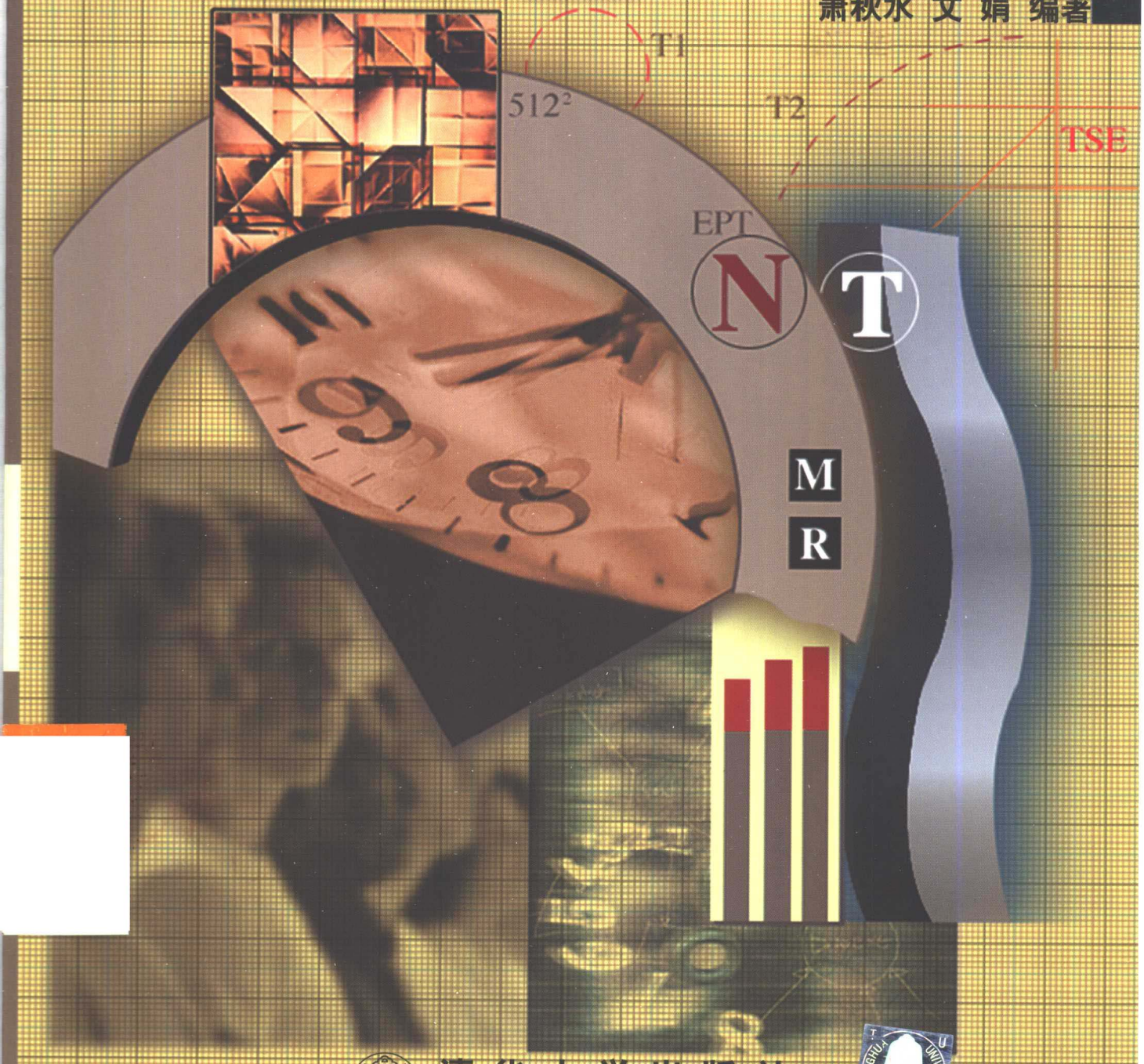
网络编程实例导学系列

Windows 网络编程之 Delphi 篇

萧秋水 文 娟 编著



清华大学出版社
http://www.tup.tsinghua.edu.cn

网络编程实例导学系列

Windows 网络编程之 Delphi 篇

萧秋水　文　娟　编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

Internet 技术无疑是当今计算机技术的最大热点。本书以当今最为流行的 RAD 软件之一的 Delphi 的最新版本为开发工具,尽最大可能包容现在网络的流行协议,讲解网络应用程序的开发原理以及在 Windows 平台下的实现方法。

本书以编程实例为主线,辅以必要的技术要点,详细地介绍了网络编程中的各个方面,从内容上覆盖了网络通讯中使用的多数协议,包括网上聊天、网络参数的获取、电子邮件的收发、Ping、FTP 客户机、Web 服务器与浏览器、Telnet 服务器与客户机、RAS 拨号上网以及 TAPI 编程等。

本书适用于使用 Delphi 进行 Windows 网络编程的中高级程序员。当然,初级读者也可从中受益。

书　名: Windows 网络编程之 Delphi 篇
作　者: 萧秋水　文　娟
出版者: 清华大学出版社(北京清华大学学研大厦,邮编:100084)
http://www.tup.tsinghua.edu.cn
印刷者: 清华大学印刷厂
发行者: 新华书店总店北京发行所
开　本: 787 × 1092 1/16　**印张**:17.5　**字数**: 423 千字
版　次: 2001 年 1 月第 1 版　　2001 年 1 月第 1 次印刷
书　号: ISBN 7-900630-94-5
印　数: 0001 ~ 5000
定　价: 35.00 元

前　言

Windows是个优秀的图形化操作系统，如今已经深入人心，成了PC市场上的主流操作系统。Internet的发展则是计算机技术的发展热点。如何在Windows平台下开发出出色的网络应用程序便成了很值得讨论的一个问题。有鉴于此，我们推出了Windows网络编程实例导学系列，本书便是其中的一本。

目前市面上讲解网络编程的书籍多是以内容为标准分章论节，本书则从一个崭新的角度向读者介绍网络编程的知识，这个角度对于读者来说也是最为实用的角度，即以编程实例为主线贯串全书。如果读者正在进行一个网络程序项目的开发工作，本书是最适当不过的了。本书虽然基于实例，但是我们在介绍过程中又不囿于实例，而是拓宽知识面，详细介绍了该实例所需的各个技术要点，以及除了实例中方法之外的其他可选方法。更为有用的是，书中给出了实现项目的步骤，读者针对某一内容找到相应实例就可以立即开始动手编程了。读者还可以以本书提供的代码为基础，不用一切都从头开始。

本书采用的开发工具是Delphi的最新版本。在内容上，本书力图覆盖Windows网络编程的大部分内容，适用于Windows 95/98/NT/2000平台。读者在使用本书时，既可以循序渐进地阅读，也可以直接查阅自己感兴趣的部分，下面对本书的内容做一个简单地概述。

实例1　介绍目前已经成为时尚的聊天程序的编写方法。聊天程序用WinSock实现，它既可以充当聊天服务器，也可以胜任聊天客户端的角色。

实例2　通过一个可融入Windows桌面的示例程序，不仅演示了如何获取本地计算机的主机名和IP地址的编程方法，而且传授给用户融入桌面的界面编程技巧。

实例3　介绍了通过Netbios获取网卡物理地址的知识和技巧。

实例4　为Internet的文件传输协议实现了一个客户端软件。使用这个工具，可以从Internet的FTP服务器上自由地下载想要的文件。

实例5　手把手地帮助用户创建两个实用工具。使用其中的Ping工具，可以检测远程计算机的可达性；而通过Trace Route工具，则可以列举出到达远程主机经过的所有计算机。

实例6　介绍了远程登录的编程原理，并通过一个TELNET服务器和一个TELNET客户机，演示了TELNET的具体实现步骤。

实例7　给出了电子邮件收发程序的编程原理及具体的实现方法，其中给出的源码很有参考价值。

实例8　使用Finger协议实现了一个客户端软件，用以获取特定服务器上的用户信息。

实例9　为用户讲解了RAS(远程访问服务)客户机的编程原理。它是拨号上网的各种编程方式中最为简单也是最为有效的一种方法，是拨号编程必须掌握的一项技能。

实例10　介绍了Web服务器和WWW浏览器的实现原理，并演示了它们的编写方法。当前，WWW已经成为Internet发展最快的一个热点。

实例 11　与“实例 10”相辅相成，它引入了 Web 服务程序，该服务程序扩展了 Web 服务器，使得 Web 服务器能够满足用户的商业规则。

实例 12　演示了如何使用底层的电话编程接口进行拨号程序的开发。它与 RAS 编程相比，有利有弊，如何取舍，得看程序的需要了。

本书的一大特点是编程实例丰富，不仅在所附光盘中给出了作者在平时编程中积累的许多源代码，而且在书中给予了详细讲解，引导读者逐步了解 Windows 网络编程的奥妙。

本书适用于使用 Delphi 进行 Windows 网络编程的中高级程序员。当然，初级读者也可以从中受益。

本书主要由第一作者萧秋水编著，文娟完成了本书大部分内容的整理工作，蒋永波、唐明、刘政伟等也参与了本书的编写工作。限于作者水平，难免在内容选择和叙述上有不当之处，欢迎广大读者批评指正。

作 者

目　录

实例 1　用 WinSock 实现网上聊天 …… 1

主要内容 …… 1

本例提要 …… 1

技术专题 …… 2

※ WinSock 简介 …… 2

※ WinSock API 主要函数的使用 …… 2

※ ScktComp 单元对 WinSock API 的封装 …… 5

步骤——实现聊天程序 …… 19

建立一个新项目 …… 20

启动时进入监视状态 …… 20

连接到聊天服务器 …… 21

客户机与服务器的对话 …… 22

实例 2　获取主机名和 IP 地址 …… 23

主要内容 …… 23

本例提要 …… 23

技术专题 …… 23

※ 获取主机名和 IP 地址的原理 …… 23

步骤——获取主机名和 IP 地址 …… 26

建立一个新项目 …… 26

提供两个核心功能函数 …… 26

创建非正常窗口并从 Taskbar 隐藏 …… 27

完成其他界面编程 …… 28

实例 3　网卡物理地址 …… 30

主要内容 …… 30

本例提要 …… 30

技术专题 …… 30
※ Netbios 网络编程接口 …… 30
※ Netbios 的基本概念 …… 31
※ Netbios 编程 …… 32
※ 使用 Netbios 获取网络参数 …… 38
步骤——获取网卡物理地址 …… 40
创建一个新项目 …… 40
窗体创建时枚举可用的 LANA 编号 …… 40
重置选中的 LANA 编号 …… 42
获取物理地址 …… 43

实例 4 FTP 客户程序 …… 45
主要内容 …… 45
本例提要 …… 45
技术专题 …… 46
※ FTP 协议及其实现方法 …… 46
※ 使用 TNMFTP 组件 …… 48
※ 使用 Microsoft Internet Transfer 组件 …… 53
※ 直接使用 WinInet API …… 56
步骤——实现 FTP 客户程序 …… 61
创建一个新组件 TDragDropFTP …… 61
安装该组件 …… 69
使用该组件实现 FTP 客户程序 …… 69
需要改进的地方 …… 70

实例 5 Ping 与 Trace Route …… 71
主要内容 …… 71
本例提要 …… 71
技术专题 …… 72
※ Internet 控制报文协议 ICMP …… 72
※ ICMP 报文格式 …… 73
※ ICMP 报文的分类 …… 74
步骤之一——实现 Ping 实用工具 …… 78

创建一个组件 TICMP …… 78
从 TICMP 派生 TPing 组件 …… 80
使用 TPing 组件建立 Ping 实用工具 …… 86
步骤之二——实现 Trace Route 实用工具 …… 87
从 TICMP 派生 TTraceRoute 组件 …… 87
使用 TTraceRoute 组件建立 Trace Route 实用工具 …… 90

实例 6 编程实现 TELNET …… 91
主要内容 …… 91
本例提要 …… 91
技术专题 …… 93
※ TELNET(远程登录) …… 93
步骤之一——实现 TELNET 服务器 …… 95
创建一个通用组件 TWSocket …… 95
实现 TELNET 端口的监视 …… 96
与客户机远程交互 …… 97
步骤之二——实现 TELNET 客户机 …… 101
创建一个 TELNET 客户端组件 TTnCnx …… 101
使用 TTnCnx 组件完成 TELNET 客户机 …… 104

实例 7 邮件收发程序 …… 106
主要内容 …… 106
本例提要 …… 106
技术专题 …… 107
※ 邮件格式 …… 107
※ POP3 协议简介 …… 108
※ SMTP 协议简介 …… 109
※ Delphi 中 POP3 协议的实现 …… 111
※ Delphi 中 SMTP 协议的实现 …… 114
步骤之一——实现邮件接收程序 …… 116
建立一个新项目 …… 116
实现服务器登录 …… 116
获取邮件列表 …… 118

接收指定邮件 …… 119
断开与服务器的连接 …… 121
处理其他的事件 …… 121
步骤之二——实现邮件发送程序 …… 122
建立一个新项目 …… 122
实现服务器登录 …… 122
发送邮件 …… 124
验证用户的存在 …… 125
扩展邮件列表 …… 126
断开与服务器的连接 …… 126
处理其他的事件 …… 127

实例 8 Finger 查询 …… 129
主要内容 …… 129
本例提要 …… 129
技术专题 …… 129
※ Finger 协议 …… 129
步骤——实现 Finger 客户机 …… 131
建立一个新项目 …… 131
在一个单独的线程中进行 Finger 查询 …… 131
完成其他辅助性的界面编程 …… 135

实例 9 RAS 拨号上网 …… 136
主要内容 …… 136
本例提要 …… 136
技术专题 …… 139
※ RAS 简介 …… 139
※ 拨号与挂断 …… 140
※ 连接管理 …… 149
※ 电话簿管理 …… 153
步骤——实现 RAS 客户机 …… 162
建立一个新项目并引入 RAS 库 …… 163

实现电话簿管理功能 …… 163
完成拨号与挂断 …… 173

实例 10 Web 服务器与浏览器 …… 180
主要内容 …… 180
本例提要 …… 180
技术专题 …… 181
※ World Wide Web …… 181
※ 统一资源定位符 …… 182
※ 超文本传送协议 …… 185
※ 使用 Microsoft WebBrowser 控件 …… 189
※ TNMHTTP 组件和 THTML 组件 …… 191
步骤之一——实现简单的 WWW 浏览器 …… 202
建立一个新项目 …… 202
完成程序的浏览任务 …… 202
步骤之二——实现 Web 服务器 …… 204
建立一个新项目 …… 204
启动 Web 服务器 …… 204
Web 服务器主循环 …… 206
THTTPServerThread 线程类 …… 208

实例 11 Web 服务程序 …… 214
主要内容 …… 214
本例提要 …… 214
技术专题 …… 214
※ Web 服务程序工作原理与分类 …… 214
※ CGI 简介 …… 215
※ ISAPI 简介 …… 218
※ Delphi 对 Web 服务程序的支持 …… 219
※ Delphi 为 Web 服务程序提供的组件类 …… 220
※ 调试 Web 服务器程序 …… 250
步骤之一——实现 ISAPI 服务程序 …… 251
建立一个新的 Web 服务程序项目 …… 251

设置默认的 Web 模块 …… 252
添加并处理动作项 …… 252
步骤之二——实现 CGI 服务程序 …… 257

实例 12 TAPI 编程 …… 258
主要内容 …… 258
本例提要 …… 258
技术专题 …… 259
※ TAPI 提供的服务 …… 259
※ TAPI 的分级 …… 260
※ 通信过程描述 …… 261
步骤之一——实现一个电话拨号程序 …… 263
建立一个新项目并实现其主窗体 …… 263
实现电话呼叫窗体 …… 264
实现拨号监测窗体 …… 268

实例 1　用 WinSock 实现网上聊天

主要内容

本例提要

随着 Internet 技术的发展，网上聊天已经成了一大时尚，相信大家已经有过这样的经历。所谓网上聊天，就是指很多人登录到一个聊天服务器，通过服务器的中转，可以看到所有人向服务器发送的内容。

在这个实例中，我们将自己动手，使用 WinSock 实现一个聊天程序。该程序既可以作为服务器，也可以作为客户机使用，运行时的界面如图 1-1 所示。

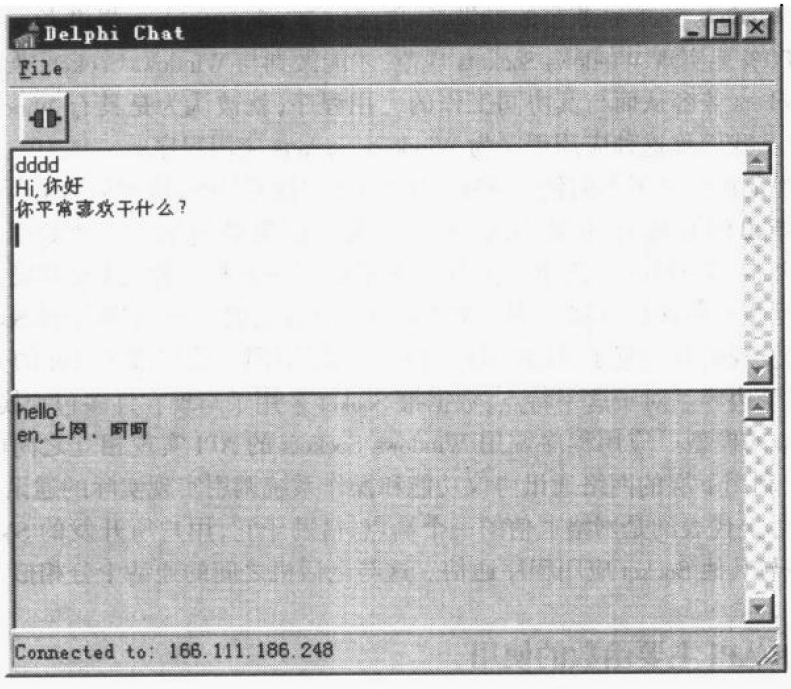


图 1-1　聊天程序

在一台计算机上启动该程序，它将自动作为服务器运行，接着用户就可以从网上的其他机器上启动该程序作为客户端连接到服务器上了。

技术专题

※ WinSock 简介

Socket(套接字)最初是由加利福尼亚大学 Berkeley(伯克利)分校为 UNIX 操作系统开发的网络通信接口,随着 UNIX 操作系统的广泛使用,Socket 成为当前最流行的网络通信应用程序接口之一。20 世纪 90 年代初,由 Sun Microsystems, JSB, FTP software, Microdyne 和 Microsoft 等几家公司共同制定了一套标准,即 Windows Sockets 规范,简称 WinSock。

Windows Sockets API 是 Microsoft Windows 的网络程序设计接口,它在继承了 Berkeley Sockets 主要特征的基础上,对其进行了重要扩充。这些扩充主要体现在它提供了一些异步函数,并增加了符合 Windows 消息驱动机制的网络事件异步选择方式。这些扩充有利于程序员编写符合 Windows 编程模式的软件。WinSock 使得在 Windows 下开发高性能的网络通信程序成为可能。

Windows Sockets 规范定义了程序员能够使用,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法。遵守这套 Windows Sockets 规范的网络软件,称之为与 Windows Sockets 兼容,而实现 Windows Sockets 兼容的提供者,称之为 Windows Sockets 提供者。一个网络软件供应商必须 100% 地实现 Windows Sockets 规范,才能做到与 Windows Sockets 兼容。任何能够与 Windows Sockets 兼容从而与其协同工作的应用程序,就被认为是具有 Windows Sockets 接口的应用程序。我们称这种应用程序为 Windows Sockets 应用程序。

此外,该规范还定义了如何使用 API 与 Internet 协议族(IPS,现在最为流行的是 TCP/IP)连接,尤其要指出的是所有的 Windows Sockets 实现都支持流 Socket 和数据报 Socket。流 Socket 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 Socket 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 Socket 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 Socket 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 Socket 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。应用程序调用 Windows Sockets 的 API 实现相互之间的通讯。Windows Sockets 又利用下层的网络通讯协议功能和操作系统调用实现实际的通讯工作。

Socket 实际上代表的是网络通信的一个端点,借助于它,用户所开发的 Socket 应用程序可以通过网络与其他 Socket 应用程序通信。这与电话机之间的通话十分相似。

※ WinSock API 主要函数的使用

在 Windows(包括 Windows 9x、Windows NT 和 Windows 2000)中进行 WinSock 应用程序开发时,可供使用的编程语言很多,如 Visual C++, Java, Delphi, Visual Basic 等。其中 Delphi 对原来的 Windows Sockets API 进行了一系列封装,使得 Socket 应用程序的开发变得更为容易,这一点将在后面介绍。但对于一个 WinSock 编程的初学者,那么了解一些最基本的 WinSock API 函数对将来的编程会大有好处,因为它可以大大加深对 WinSock 的理解。

下面对 WinSock API 中主要函数的使用方法做一个简要说明：

1. 连接 WinSock 库

```
int WSAStartup (
  WORD wVersionRequested,
  LPWSADATA lpWSAData
);
```

应用程序在使用 WinSock API 之前，必须调用该函数，只有该函数成功返回(表示应用程序与 WinSock 库成功地建立起连接)，应用程序才可以调用其他 Windows Sockets DLL 中的函数。

2. 断开 WinSock 库

```
int WSACleanup (void);
```

WSACleanup 函数可以结束 Windows Sockets DLL 的使用。当应用程序不再需要使用 Windows Sockets DLL 时，必须调用此函数来注销，以便释放其占用的资源。

3. 套接字建立函数

```
SOCKET Socket (
  int af,
  int type,
  int protocol
);
```

使用该函数可以建立使用特定协议的网络套接字。例如对于 UDP 协议，可以采用如下写法：

```
SOCKRET s;
s = Socket(AF_INET,SOCK_DGRAM,0);
```

或

```
s = Socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP);
```

4. 套接字绑定函数

```
int bind (
  SOCKET s,
  const struct sockaddr FAR* name,
  int namelen
);
```

s 是使用 Socket 函数创建好的套接字；name 指向描述通讯对象的结构体的指针；namelen 是该结构体的长度。该结构体中的分量包括：

- IP 地址：对应 name.sin_addr.s_addr。
- 端口号：对应 name.sin_port。端口号用于表示同一台计算机上的不同进程(即应用程序)，其分配方法有两种：

  第一种分配方法是，进程让系统为套接字自动分配一端口号，这只要在调用 bind 前

将端口号指定为 0 即可。由系统自动分配的端口号位于 1024 ~ 5000 之间，而 1 ~ 1023 之间的任一 TCP 或 UDP 端口都是保留的，系统不允许任一进程使用保留端口，除非其有效用户 ID 是 0(即超级用户)。

第二种分配方法是，进程为套接字指定一特定端口，指定范围在 1024 ~ 65536 之间。这对于需要给套接字分配一众所周知的端口的服务器是很有用的。

- 地址类型：对应 name.sin_family，一般都赋值为 AF_INET，表示是 Internet 地址(即 IP 地址)。IP 地址通常使用点分表示法表示，但它实际上是一个 32 位的长整数。这两者之间可以通过 inet_addr()函数转换。

5. 套接字监视函数

```
int listen (
  SOCKET s,
  int backlog
);
```

listen 函数用来设定 Socket 为监视状态，这种状态表明 Socket 准备被连接。注意，此函数一般在服务程序上使用，其中的 backlog 参数用于设定等待连接的客户数(目前最大值为 5，最小值为 1)。

6. 接受连接请求

```
SOCKET accept (
  SOCKET s,
  struct sockaddr FAR* addr,
  int FAR* addrlen
);
```

服务端应用程序调用此函数来接受客户端 Socket 连接请求。accept() 函数的返回值为一新的 Socket，新 Socket 可用来完成服务端和客户端之间的信息传递，而原来的 Socket 仍然可以接收其他客户端的连接要求。

7. 接收信息

```
int recv (
  SOCKET s,
  char FAR* buf,
  int len,
  int flags
);
```

此函数被用于面向连接的 Socket 中，以便可以从连接的另一端接收信息。

8. 发送信息

```
int send (
  SOCKET s,
  const char FAR * buf,
  int len,
  int flags
);
```

此函数被用于面向连接的 Socket 中，以便可以向连接的另一端发送信息。

了解了这些基本的函数之后，我们还需要明确另一个重要的问题，即服务端 Socket 与客户端 Socket 是如何交互的。目前最为常用的方式是：服务程序在一个众所周知的地址（其中包括端口的信息）监视对服务的请求，也就是说，服务进程一直处于休眠状态，直到一个客户对这个服务的地址提出了连接请求。在这个时刻，服务程序被唤醒并且对客户的请求作出适当的反应。注意，服务器与客户机之间的交互既可以是面向连接的（即基于流套接字），也可以是无连接的（即基于数据包套接字）。其编程模型如图 1-2 所示。

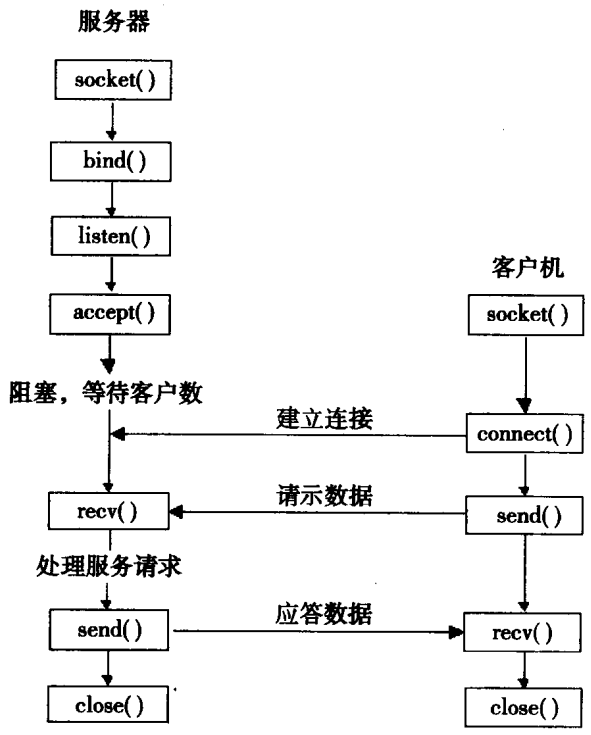


图 1-2　Socket 交互的编程模型

※　ScktComp 单元对 WinSock API 的封装

前面说过，Delphi 对原来的 Windows Sockets API 进行了一系列封装。这些封装都是在 Delphi 的 SckComp 单元中完成的。而在 WinSock 单元中则将原来的 WinSock API 转换成了符合 Pascal 语法的说明；SckComp 单元引用并扩展了 WinSock 单元。

在 SckComp 单元中，Delphi 5 创建了几个十分有用的组件，包括 TClientSocket 组件、TServerSocket 组件以及它们的祖先类。这些组件之间构成了一个层状的继承关系，如下所示：

- TServerSocket 的直接上级是 TcustomServerSocket。
- TCustomServerSocket 的上级是 TCustomSocket。
- TCustomSocket 的上级是 TAbstractSocket。

- TAbstractSocket 的上级是 TComponent。
- TClientSocket 组件的直接上级是 TCustomSocket。

其中 TClientSocket 组件和 TServerSocket 组件分别用来操作客户端 Socket 与服务器端 Socket 的连接和通信。要说明的是，这两个组件用于管理客户端和服务器端的连接，本身并不是 Socket 对象。操作 Socket 对象的是 TCustomWinSocket 及其派生类，如 TClientWinSocket、TServerWinSocket 和 TServerClientWinSocket 等。

要在 Delphi 程序中建立服务器端 Socket，只需把一个 TServerSocket 组件放到 Form 或数据模块上，应用程序就变成了一个 TCP/IP 服务器。当服务器处于监视状态时，就可用 TServerWinSocket 来操作服务器端的 Socket 对象了。

当客户提出连接请求，服务器接受了请求并建立起连接之后，就可用 TServerClientWinSocket 来操作服务器端 Socket 与客户端 Socket 的连接。

要使服务器能监视客户的连接请求，首先要设置 TServerSocket 的 Port 属性，以指定端口号。如果服务器提供的是标准服务，如 FTP、HTTP、FINGER 或 TIME，则只要用 TServerSocket 的 Service 属性指定服务类型就可以了，因为这些标准服务的端口号是事先约定的，此时端口号已经隐含确定了。如果同时设置了 Port 属性和 Service 属性，Port 属性将被忽略。

指定了端口号或由 Service 属性隐含指定了端口号后，就可以调用 TServerSocket 组件的 Open 开始监视。如果在设计时把 Active 属性设为 True，服务器程序启动的时候就会自动进入监视状态。也就是说，把 Active 属性设为 True 与调用 Open 是等价的。

一旦进入了监视状态，就可以通过 TServerSocket 组件的 Socket 属性返回服务器端的 Socket 对象(TServerWinSocket)，并由此对象获得有关连接的信息。例如，Socket 对象的 SockeHandle 属性可以获得 Socket 的 Windows 句柄，在直接调用 WinSock 的 API 时需要用到这个句柄。Socket 对象的 Handle 属性可以访问从 Socket 连接中检索消息的窗口。

服务器进入监视状态后，当客户提出连接请求时，服务器就自动接受请求，然后建立连接并触发 OnClientConnect 事件。此时，服务器端 Socket 与客户端 Socket 的连接由 TServerClientWinSocket 来操作。TServerWinSocket 的 Connections 属性可以返回一个列表，该列表由服务器当前的活动连接(TServeClientWinSocket)组成。

如果要断开连接，只要在服务器端调用 Close 或者把 Active 属性设为 False 即可。注意，这将导致所有的客户连接都被断开，并退出监视状态。如果在客户端调用 Close，将只断开该客户与服务器的连接，不影响其他客户的连接。

要创建客户端 Socket，则只需把一个 TClientSocket 组件放到 Form 或数据模块上，就可以用 TClientWinSocket 来操作客户端的 Socket 对象了。

需要注意的是，一个应用程序可以兼有 TCP/IP 服务器和客户两种角色，这只需把 TServerSocket 组件和 TClientSocket 组件放到同一个 Form 或数据模块上即可。

TClientSocket 组件的 Host 属性用于指定服务器的主机名，而 Address 属性则用来指定 IP 地址，这两种方式的效果是等价的，但如果同时设置了 Host 属性和 Address 属性，Address 属性将被忽略。

要连接服务器，还要指定服务器端所使用的端口号。这里也有两种方式：一是直接设置 Port 属性来指定端口号；二是设置 Service 属性隐含确定的端口号。如果同时设置了 Port