南开大学
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

计算机网络实验报告

利用 Socket 编写一个聊天程序

李潇逸 2111454

年级：2021 级

专业：信息安全、法学

指导教师：张建忠

2023 年 10 月 21 日

目录

一、 实验要求

（1）给出你聊天协议的完整说明。

（2）利用 C 或 C++ 语言，使用基本的 Socket 函数完成程序。不允许使用 CSocket 等封装后的类编写程序。

（3）使用流式套接字、采用多线程（或多进程）方式完成程序。

（4）程序应有基本的对话界面，但可以不是图形界面。程序应有正常的退出方式。

（5）完成的程序应能支持多人聊天，支持英文和中文聊天。

（6）编写的程序应该结构清晰，具有较好的可读性。

（7）在实验中观察是否有数据的丢失，提交源码和实验报告。

二、 Socket 概述

(一) 简介

Socket 是应用层与 TCP/IP 协议族通信的中间软件抽象层，它是一组接口。在设计模式中，Socket 其实就是一个门面模式，它把复杂的 TCP/IP 协议族隐藏在 Socket 接口后面，对用户来说，一组简单的接口就是全部。

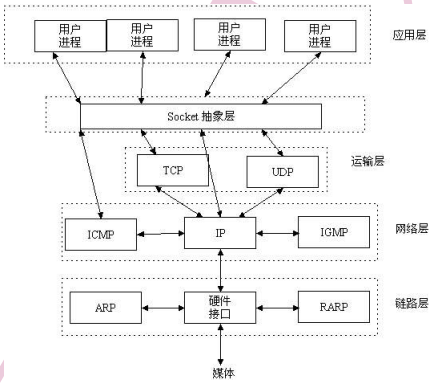

图 1: socket 概述

socket 起源于 Unix，而 Unix/Linux 基本哲学之一就是“一切皆文件”，都可以用“打开 open –> 读写 write/read –> 关闭 close”模式来操作。Socket 就是该模式的一个实现，socket 即是一种特殊的文件，一些 socket 函数就是对其进行的操作（读/写 IO、打开、关闭）

想给另一台计算机发消息，知道他的 IP 地址，他的机器上同时运行着 qq、迅雷、word、浏览器等程序，你想给他的 qq 发消息，那想一下，你现在只能通过 ip 找到他的机器，但如果让这台机器知道把消息发给 qq 程序呢？答案就是通过 port，一个机器上可以有 0-65535 个端口，你的程序想从网络上收发数据，就必须绑定一个端口，这样，远程发到这个端口上的数据，就全会转给这个程序

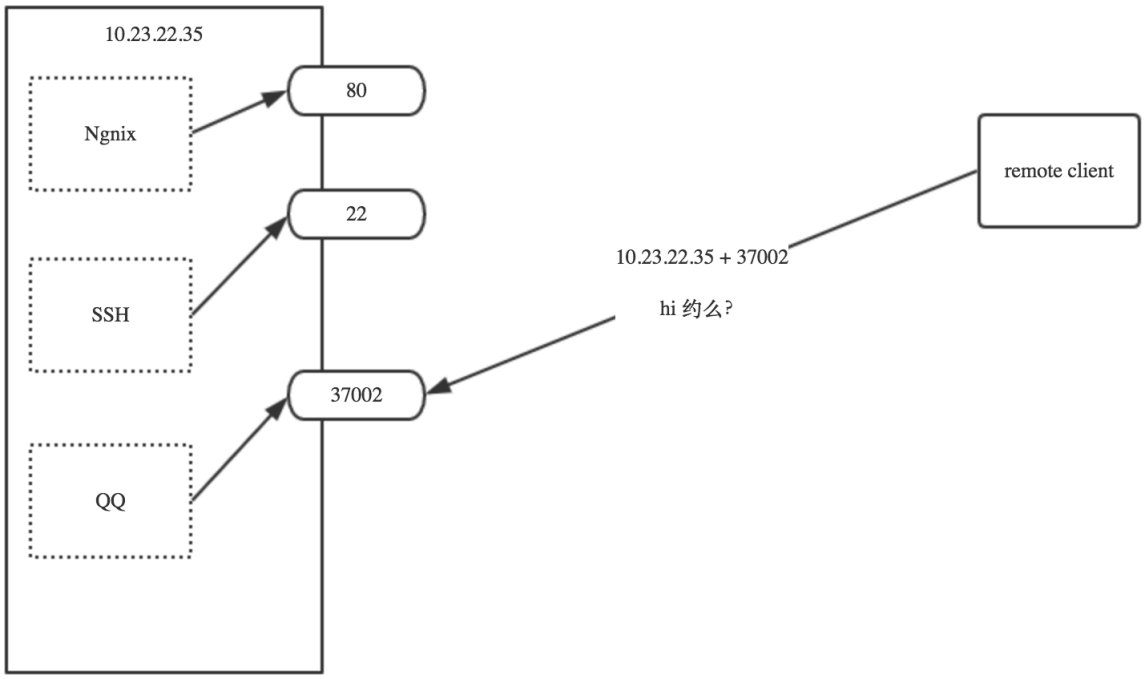

图 2: socket 概述

三、 运行结果展示

当运行程序后客户端会悬停在如下界面等待输入聊天室地址

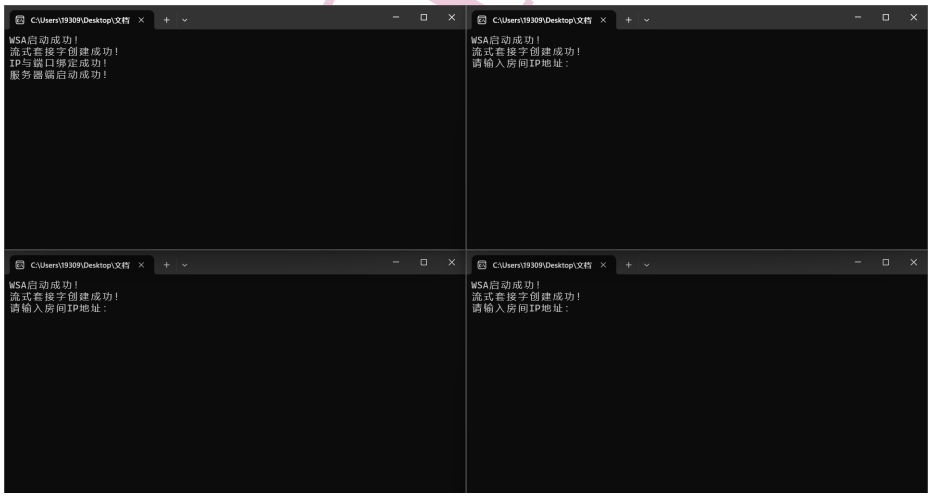

图 3: 登录

输入聊天室地址后（这里使用的是 127.0.0.1）可以进入聊天室，并收到系统自动分配的端口号及欢迎

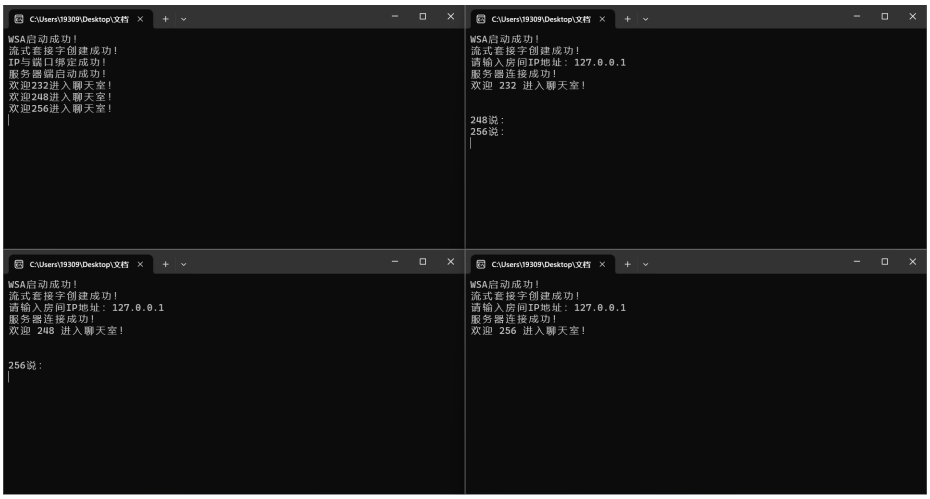

图 4: 进入聊天室

在聊天室内直接发送消息所有在聊天室内的用户都将接受到相关消息

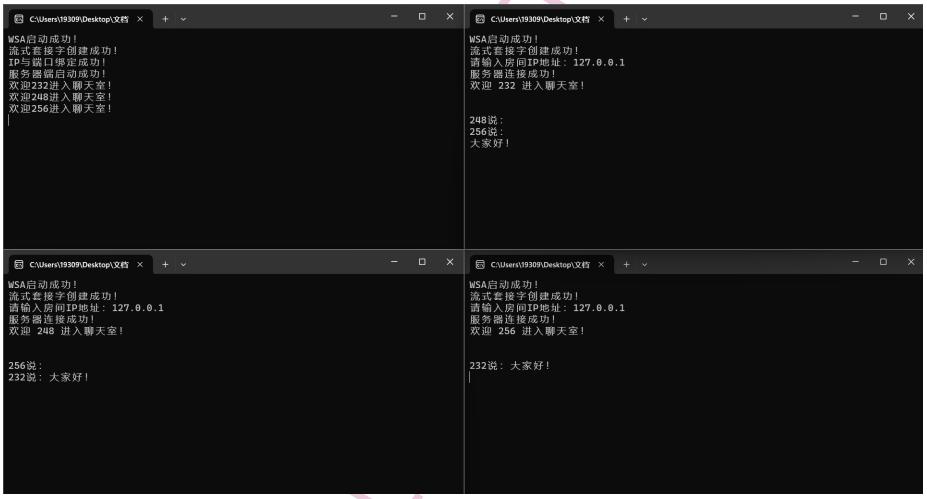

图 5: 公聊

为了用户可以了解该聊天室内都有哪些用户，同时为私聊做准备，因此设计了”users“函数。当用户输入”users“关键字时，将会且仅会在其终端显示在聊天室内的所有用户信息。

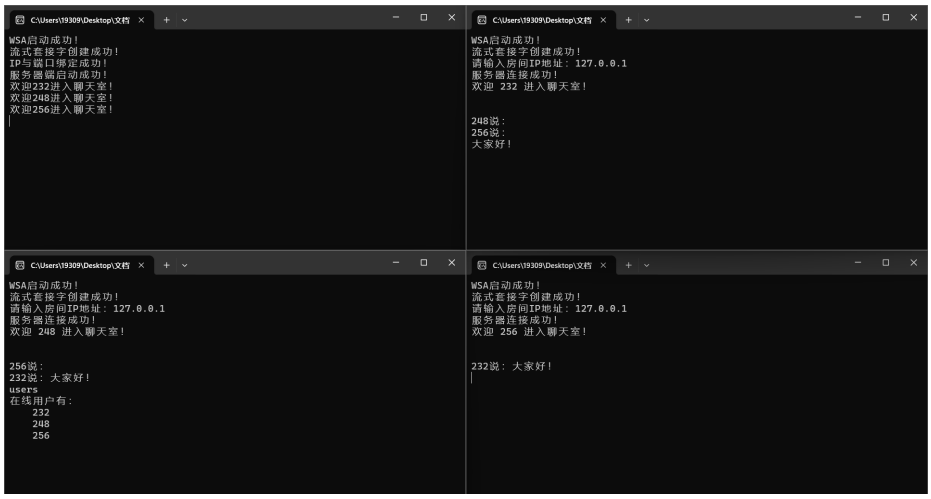

图 6: 显示聊天室中的用户

我们还为用户设置了私聊选项。在聊天室的界面输入关键字”to “，后接想要私聊的用户的端口号，之后发送信息即可实现私聊。在图中我们可以看到当用户 256 向用户 232 发起私聊后，仅 232 接收到了私聊信息。

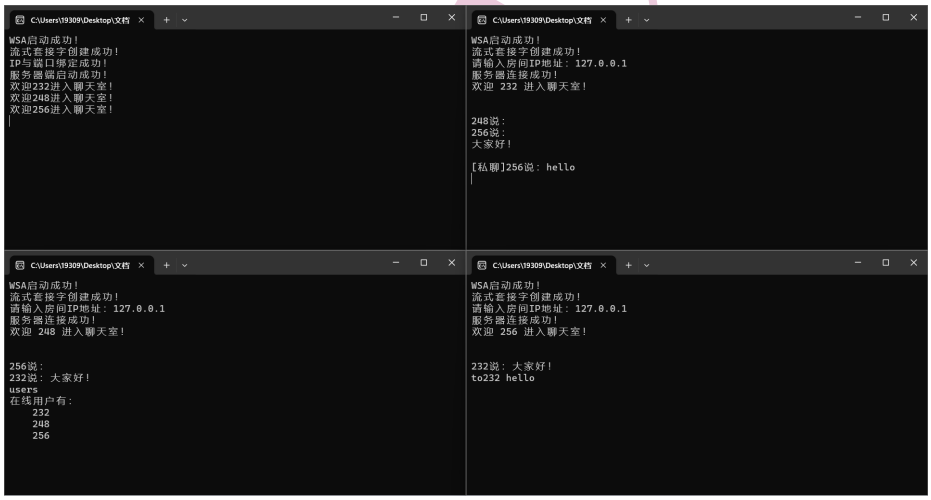

图 7: 私聊

当用户输入关键字” quit “后，用户即可退出系统，其余用户将会得知该用户进行了退出。同时在列表中也无法发现这一用户。

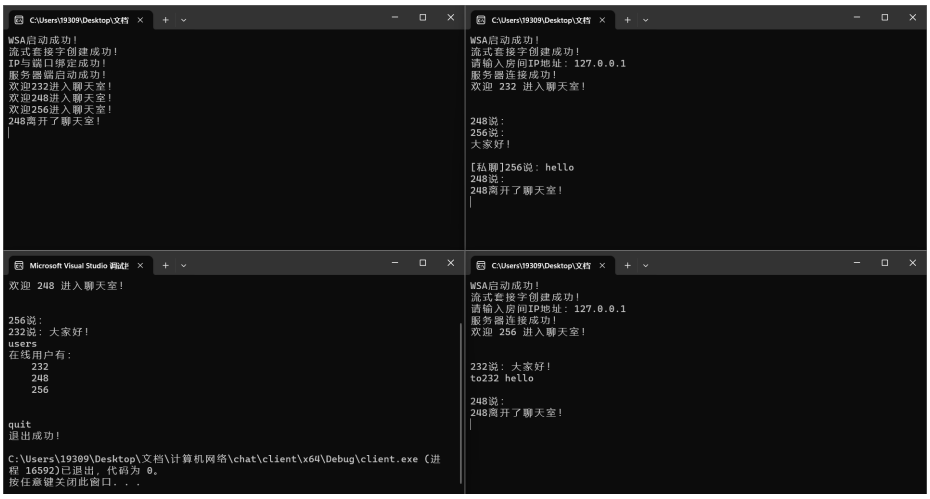

图 8: 退出

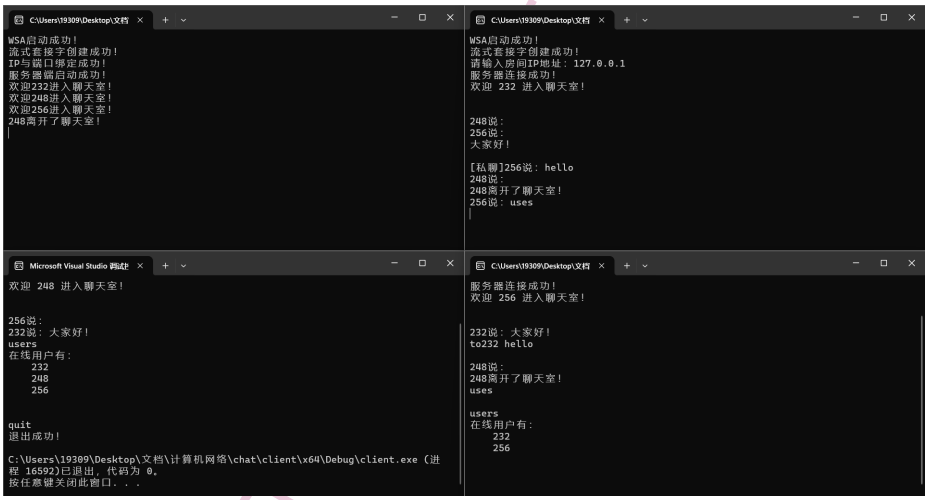

图 9: 退出后的用户列表

四、 代码讲解

(一) 大体思路

本次实验的实现主要使用了 winSocket，编写了客户端和服务器端两个主要代码。客户端链接到服务器端，将信息通过服务器端进行转发；服务器端向各个客户端提供信息服务。

(二) 多线程

在本次实验的实现过程中，为了实现客户端和服务端即时接受和发送信息，这就需要多线程技术。在本程序中，客户端开启两个线程，一个用于接收信息，一个用于发送信息；服务器端则是每一个链接都开启一个线程。

(三) 套接字

socket 使用套接字进行双方的通信，具体使用以下函数：

- send(): 发送数据
- recv(): 接收数据

五、 源码

(一) 服务器端

服务器端

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <string>
#include <winsock2.h>
#include <Windows.h>
#pragma comment(lib,"ws2_32.lib")

using namespace std;

#define MAXNUM 8
SOCKET client_fds[MAXNUM];

// stdcall的线程处理函数
DWORD WINAPI ThreadFun(LPVOID lpThreadParameter);

int main()
{
    WSADATA wd;
    if (WSAStartup(MAKEWORD(2, 2), &wd) != 0)
    {
        cout << "WSAStartup Error:" << WSAGetLastError() << endl;
        return 0;
    }
    else
    {
        cout << "WSA启动成功! " << endl;
    }

    // 创建流式套接字
    SOCKET s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (s == INVALID_SOCKET)
    {
        cout << "socket error:" << WSAGetLastError() << endl;
        return 0;
    }
    else
    {
        cout << "流式套接字创建成功! " << endl;
```

```
40      }
41
42      // 绑定端口和ip
43      sockaddr_in addr;
44      memset(&addr, 0, sizeof(sockaddr_in));
45      addr.sin_family = AF_INET;
46      addr.sin_port = htons(8000);
47      addr.sin_addr.s_addr = htonl(INADDR_ANY);
48
49      int len = sizeof(sockaddr_in);
50      if (bind(s, (SOCKADDR*)&addr, len) == SOCKET_ERROR)
51      {
52          cout << "bind Error:" << WSAGetLastError() << endl;
53          return 0;
54      }
55      else
56      {
57          cout << "IP与端口绑定成功! " << endl;
58      }
59
60      // 监听
61      listen(s, 10);
62      cout << "服务器端启动成功! " << endl;
63      // 主线程循环接收客户端的连接
64      while (true)
65      {
66          sockaddr_in addrClient;
67          len = sizeof(sockaddr_in);
68          // 接受成功返回与client通讯的Socket
69          SOCKET c = accept(s, (SOCKADDR*)&addrClient, &len);
70          if (c != INVALID_SOCKET)
71          {
72              int index = -1;
73              for (int i = 0; i < MAXNUM; i++)
74              {
75                  if (client_fds[i] == 0)
76                  {
77                      index = i;
78                      client_fds[i] = c;
79                      break;
80                  }
81              }
82              // 创建线程, 并且传入与client通讯的套接字
83              HANDLE hThread = CreateThread(NULL, 0, ThreadFun, (LPVOID)c, 0,
                    NULL);
84              CloseHandle(hThread); // 关闭对线程的引用
85          }
86
```

```
87      }
88
89      // 关闭监听套接字
90      closesocket(s);
91
92      // 清理winsock2的环境
93      WSACleanup();
94
95      return 0;
96  }
97
98  DWORD WINAPI ThreadFun(LPVOID lpThreadParameter)
99  {
100     // 与客户端通讯，发送或者接受数据
101     SOCKET c = (SOCKET)lpThreadParameter;
102
103     cout << "欢迎" << c << "进入聊天室！" << endl;
104
105     // 发送数据
106     char buf[100] = { 0 };
107     sprintf(buf, "欢迎␣%d␣进入聊天室！", c);
108     send(c, buf, 100, 0);
109
110     // 循环接收客户端数据
111     int ret = 0;
112     do
113     {
114         char buf2[100] = { 0 };
115         send(c, buf2, 100, 0);
116         ret = recv(c, buf2, 100, 0);
117         // 当发现输入的字符串开头为“to”时进入私聊模式
118         if (buf2[0] == 't' && buf2[1] == 'o')
119         {
120             string s = to_string(c);
121             char* temp = const_cast<char*>(s.c_str());
122             char temp2[] = "说：";
123             char temp3[] = "[私聊]";
124             char buf3[100] = "";
125             strcpy(buf3, temp3);
126             strcat(buf3, temp);
127             strcat(buf3, temp2);
128             // 查找要私聊的用户端口
129             int id = 0;
130             int index = 2;
131             for (index = 2;(buf2[index] <= '9' && buf2[index] >= '0');index
                    ++)
132             {
133                 id *= 10;
```

```
134                 id += (buf2[index] - '0');
135             }
136             index++;
137             char buf4[100] = { 0 };
138             for (int i = index;i < 100;i++)
139             {
140                 buf4[i - index] = buf2[i];
141             }
142             strcat(buf3, buf4);
143             for (int i = 0;i < MAXNUM;i++)
144             {
145                 if (client_fds[i] != 0 && client_fds[i] == id)
146                 {
147                     send(client_fds[i], buf3, 100, 0);
148                 }
149             }
150         }
151         // 列举所有在聊天室的用户
152         else if (buf2[0] == 'u' && buf2[1] == 's' && buf2[2] == 'e' && buf2
            [3] == 'r' && buf2[4] == 's')
153         {
154             char temp[] = "在线用户有: \n";
155             char temp1[] = "\n";
156             char temp3[] = "␣␣␣␣";
157             char buf3[100] = "";
158             strcpy(buf3, temp);
159             for (int i = 0;i < MAXNUM;i++)
160             {
161                 if (client_fds[i] != 0)
162                 {
163                     string s = to_string(client_fds[i]);
164                     char* temp2 = const_cast<char*>(s.c_str());
165                     strcat(buf3, temp3);
166                     strcat(buf3, temp2);
167                     strcat(buf3, temp1);
168                 }
169             }
170             send(c, buf3, 100, 0);
171         }
172         // 公聊
173         else
174         {
175             string s = to_string(c);
176             char* temp = const_cast<char*>(s.c_str());
177             char temp2[] = "说: ";
178             char buf3[100] = "";
179             strcpy(buf3, temp);
180             strcat(buf3, temp2);
```

```
            strcat(buf3, buf2);
            for (int i = 0;i < MAXNUM;i++)
            {
                if (client_fds[i] != 0 && client_fds[i] != c)
                {
                    send(client_fds[i], buf3, 100, 0);
                }
            }
        }

    } while (ret != SOCKET_ERROR && ret != 0);

    char buf2[100] = { 0 };
    send(c, buf2, 100, 0);
    string s = to_string(c);
    char* temp = const_cast<char*>(s.c_str());
    char temp2[] = "离开了聊天室! ";
    char buf3[100] = "";
    strcpy(buf3, temp);
    strcat(buf3, temp2);
    cout << buf3 << endl;
    for (int i = 0;i < MAXNUM;i++)
    {
        if (client_fds[i] != 0 && client_fds[i] != c)
        {
            send(client_fds[i], buf3, 100, 0);
        }
        if (client_fds[i] != 0 && client_fds[i] == c)
        {
            client_fds[i] = 0;
        }
    }

    return 0;
}
```

(二) 客户端

客户端

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <string>
#include <winsock2.h>
#include <Windows.h>
#pragma comment(lib,"ws2_32.lib")

```

```
using  namespace std;

// stdcall的线程处理函数
DWORD WINAPI ThreadGET(LPVOID lpThreadParameter);

int  main()
{
    //加载winsock2的环境
    WSADATA  wd;
    if (WSAStartup(MAKEWORD(2, 2), &wd) != 0)
    {
        cout << "WSAStartup  error: " << GetLastError() << endl;
        return 0;
    }
    else
    {
        cout << "WSA启动成功! " << endl;
    }

    // 创建流式套接字
    SOCKET  s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (s == INVALID_SOCKET)
    {
        cout << "socket  error: " << GetLastError() << endl;
        return 0;
    }
    else
    {
        cout << "流式套接字创建成功! " << endl;
    }
    cout << "请输入房间IP地址: ";
    char addr_ip[50];
    cin >> addr_ip;
    // 链接服务器
    sockaddr_in   addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(8000);
    addr.sin_addr.s_addr = inet_addr(addr_ip);

    int len = sizeof(sockaddr_in);
    if (connect(s, (SOCKADDR*)&addr, len) == SOCKET_ERROR)
    {
        cout << "connect  error: " << GetLastError() << endl;
        return 0;
    }
    else
    {
        cout << "服务器连接成功! " << endl;
```

```
57      }
58
59      bool flag = true;
60
61      while (flag)
62      {
63          // 接收服务端的消息
64          HANDLE hThreadGET = CreateThread(NULL, 0, ThreadGET, (LPVOID)s, 0,
                NULL);
65          // 随时给服务端发消息
66          int  ret = 0;
67          do
68          {
69              char buf[100] = { 0 };
70              scanf("%[^\n]", &buf);
71              getchar();
72              // 发现输入为“quit”时退出
73              if (buf[0] == 'q' && buf[1] == 'u' && buf[2] == 'i' && buf[3] ==
                    't' && strlen(buf) == 4)
74              {
75                  cout << "退出成功! ";
76                  flag = false;
77                  CloseHandle(hThreadGET);
78                  break;
79              }
80              ret = send(s, buf, 100, 0);
81          } while (ret != SOCKET_ERROR && ret != 0);
82      }
83      // 关闭监听套接字
84      closesocket(s);
85
86      // 清理winsock2的环境
87      WSACleanup();
88
89
90      return 0;
91  }
92
93  // 数据接收线程
94  DWORD WINAPI ThreadGET(LPVOID lpThreadParameter)
95  {
96      // 与服务器端通讯, 接受数据
97      SOCKET c = (SOCKET)lpThreadParameter;
98
99      // 循环接收服务器端数据
100     int ret = 0;
101     do
102     {
```

```
103         char buf[100] = { 0 };
104         ret = recv(c, buf, 100, 0);
105         cout << buf << endl;
106 
107     } while (ret != SOCKET_ERROR && ret != 0);
108 
109     return 0;
110 }
```

六、 总结

本次实验编写了一个简单的聊天客户端，深入理解了相应原理，对计算机网络有了更深的了解。同时有以下问题：

- 可能会造成冲突
- 设计的关键字导致某些信息无法发出

这将在日后进行改进。