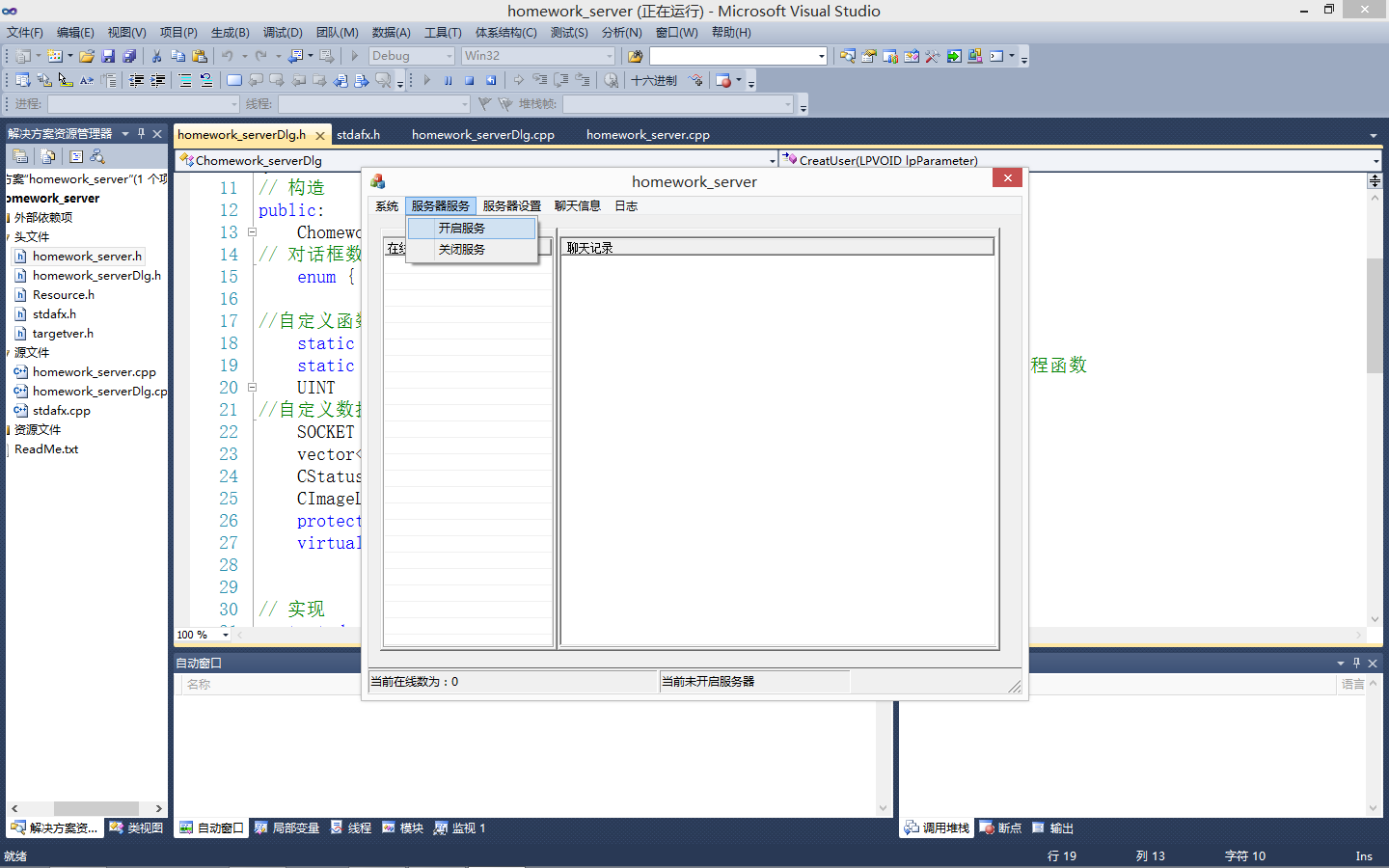
**设计报告**

**电信学部 信通学院 刘诚 21509099**

**作业题目：聊天系统**

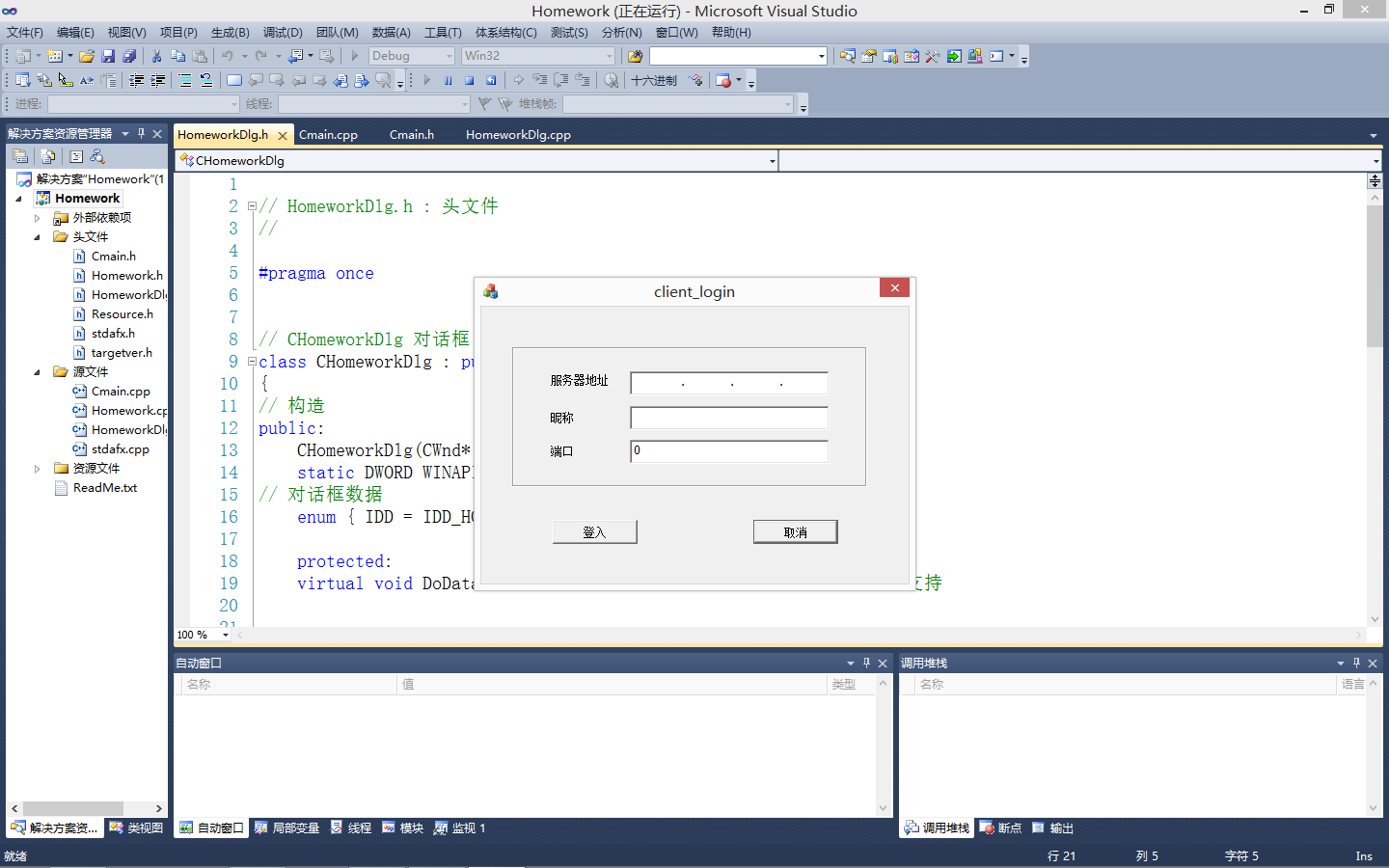
**使用说明：**

1. 首先打开服务器端(homework\_server)，出现界面如下：



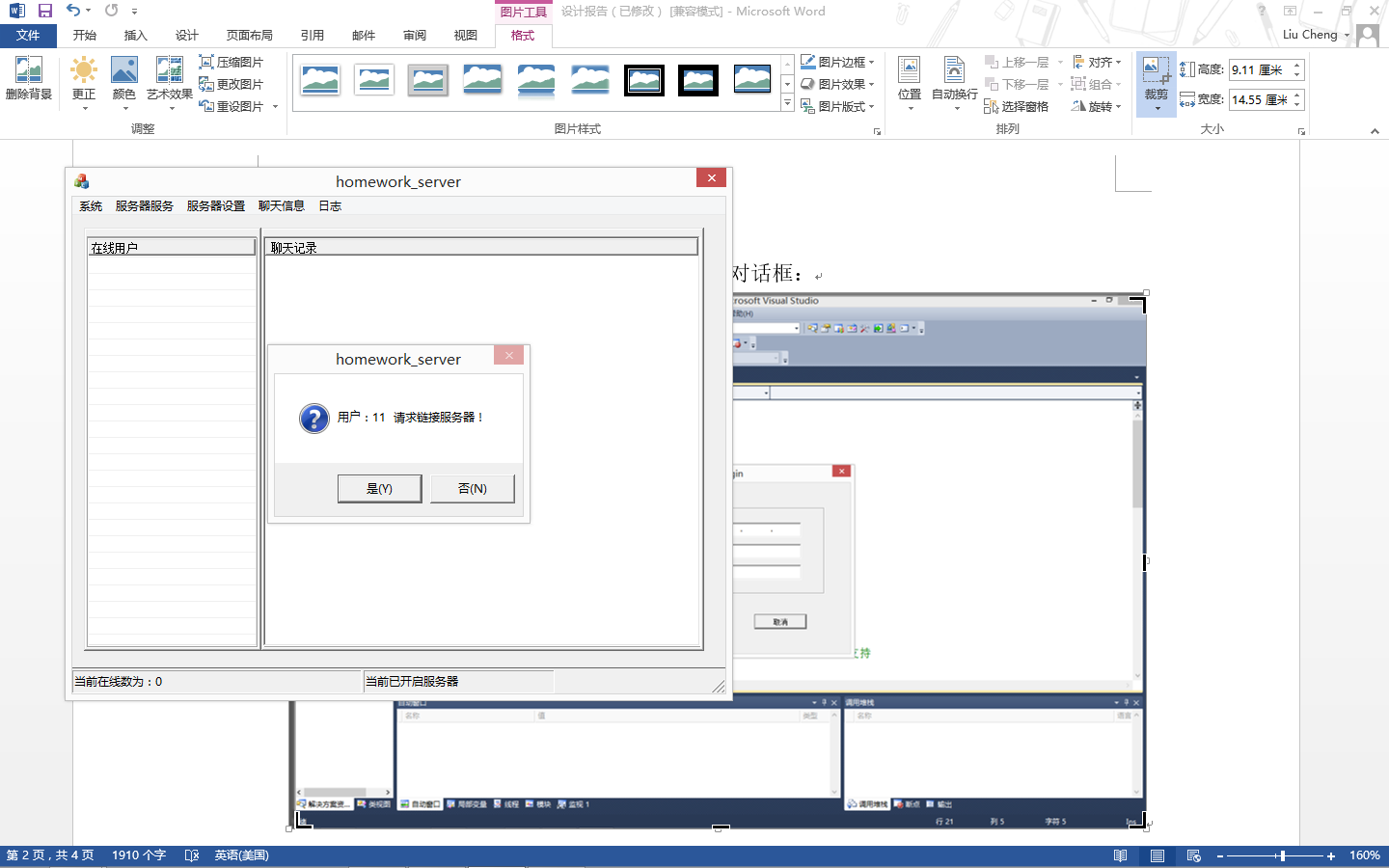
在菜单栏选择服务器服务选项，点击开启服务，底部状态栏则显示当前已开始服务，代表监听开始，等待客户端的连接。

1. 然后打开客户端(homework)，首先出现的界面如下：



服务器地址为127.0.0.1，昵称自定，端口号为6000。

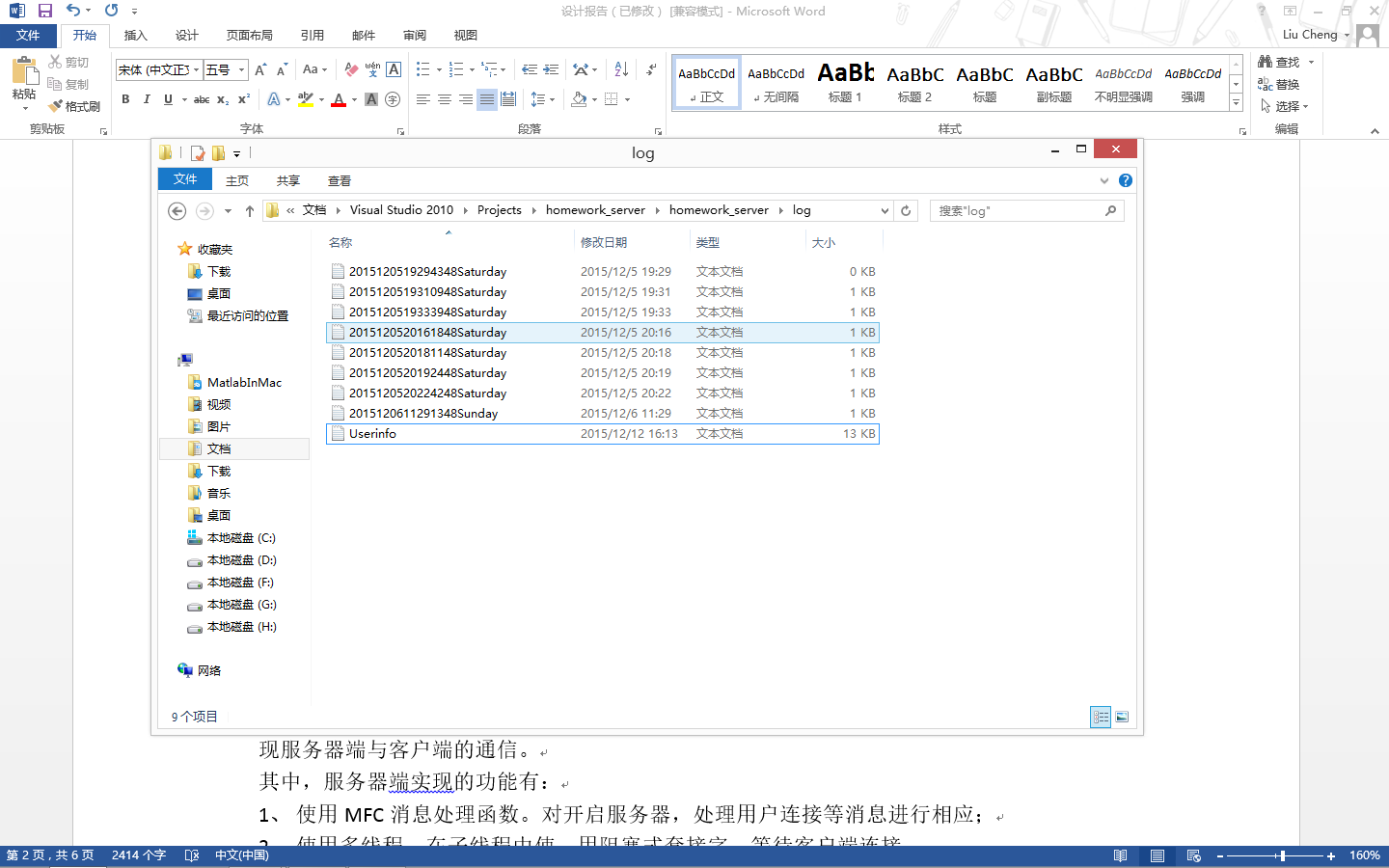
1. 填好信息后点击登录，服务端会弹出确认登录对话框：



选择是，则用户可以进入聊天室，开始对话。

查看日志：

在homework\_server/log/文件中存有用户登录退出的日志文件：



**注：**由于时间比较紧，菜单栏部分按钮的功能没有实现。

传输文件部分还有bug未解决。

**问题分析：**

本项目实现一个网络聊天室，需要搭建服务器端与客户端，利用Socket网络编程，实现服务器端与客户端的通信。

其中，服务器端实现的功能有：

1. 使用MFC消息处理函数。对开启服务器，处理用户连接等消息进行相应；
2. 使用多线程，在子线程中使。用阻塞式套接字，等待客户端连接；
3. 在开启服务器功能后，创建一个新线程，在新线程中进行套接字创建，端口绑定，开始监听，不影响GUI线程；
4. 在收到用户连接请求以后，服务器端为连接的用户创建新线程，在新线程中为客户创建套接字，使用recv()函数接受客户端发来的消息；
5. 接受用户端发来的消息以后，将消息广播出去；
6. 刷新在线用户用户列表。
7. 将用户登录退出，聊天记录等信息保存入日志文件，方便查看。

客户端实现的功能有：

1. 创建套接字，通过IP地址和端口号连接服务器；
2. 创建子线程负责接收服务器数据；
3. 刷新在线用户列表；
4. 导出聊天记录；
5. 传输文件。

**算法选择：**

1. TCP/IP协议：

为了保证通信的可靠性，选择面向连接的TCP/IP协议。在客户端向服务器发送连接请求以后，服务器需要通过认证才能建立连接，保证通信安全。

1. 多线程技术：

在服务器端使用多线程技术，创建子线程实现监听，与客户端通信等功能，可以在不影响GUI进程的前提下，进行通信，提高效率。

1. 消息处理函数:

当监听线程收到链接请求或者信息之后会向主线程发送消息，该消息为链接申请和聊天消息，主线程的消息处理函数负责处理此消息。

**方案设计:**

**服务器端：**

服务器端开启，启动服务器，监听线程启动(默认端口为6000)。在收到客户端的链接申请之后，监听线程发送WM\_USERRECV信号同时将用户套接字放在用户套接字vector里，调用主线程WM\_USERRECV信号绑定的OnRecvData函数。这个函数将接收到的数据包发送给主线程，主线程拆包获得用户信息（用户名，用户IP地址，用户客户端端口），使用MessageBox提醒客户端管理员有新用户申请连接服务器，管理员可以接受或者拒绝，如果拒绝，系统将其从用户套接字vector中删除。管理员在接受连接之后，创建一个新的用户信息对象将其放在vector中，然后创建子线程接收用户端的聊天信息并将用户ID传递给该线程。子线程负责和客户端进行数据传输，收到信息之后将信息发给主线程。对话框线程负责处理该信息。该信息有USERLIST（客户端初始化）、MSG（聊天信息）、EXIT（用户退出）三个信息。这三个信息分别对应WM\_USER\_LIST，WM\_USER\_SENDMSG，WM\_USER\_EXIT这三个消息。WM\_USER\_LIST消息绑定的函数为OnUserList，其主要功能为发送在线用户列表和聊天记录。WM\_USER\_SENGMSG绑定的函数为OnUserEcho。该函数主要的功能为将收到的聊天信息重新封包发送给每个用户。WM\_USER\_EXIT消息绑定的函数为OnUserExit，该函数的作用为将内存中的用户信息删掉，将用户套接字关闭，并将用户列表更新。服务器还具有聊天记录保存功能和用户信息日志功能。聊天信息保存是将聊天信息以时间为文件名保存在log文件夹中。用户信息保存是把用户信息包括用户名字、IP地址、登录时间、离线时间和总在线时间保存在log文件夹中的userinfo.txt文件中。服务器还具有修改监听端口功能。

**客户端：**

客户端程序首先启动登录对话框，在输入IP、端口和用户名之后，系统创建套接字，封包之后将其发送给服务器并提出链接申请，在得到允许之后，创建主对话框并把主要的用户名和套接字传递给主对话框。主对话框中初始化函数中创建子线程去接收服务器信息和设置一个定时为1S的定时器。定时器的主要功能是为了产生延时，在保证所有子线程初始化完成的情况下，进行聊天记录的初始化和用户列表的初始化。初始化列表的过程为首先向服务器发送USERLIST进行初始化申请，服务器会发送聊天记录和在线用户列表，在接受这些信息之后，解析数据包，将数据包信息添加到list中，这样初始化就完成了。完成初始化后，可以在下面输入聊天信息并发送。使用套接字发送聊天内容。数据发给服务器之后，服务器负责把信息发送给每个客户端。这样就实现了聊天功能。接收线程收到信息之后将数据包发送WM\_RECVDATA消息，主线程解析数据包，解包之后判断其命令，命令字包括MSG（信息），LOGIN（用户列表添加）和LOGOUT（用户列表删除）三类其设计和服务器类似，这里不再赘述。退出时调用析构函数关闭套接字、释放堆指针等。该客户端还具有保存聊天记录功能

**注：**无论客户端还是服务器端只要不正常关闭程序即程序关闭时都会关闭套接字，程序都会检测到。如果客户端关闭，服务器会默认该用户退出登录。如果服务器不正常服务，客户端会强制关闭，并通知用户服务器已经关闭。

**编程实现：**

**创建套接字代码：**

DWORD WINAPI Chomework\_serverDlg::RecvProc(LPVOID lpParameter) //静态类成员函数(实现)

{

SOCKET sock=((RECVPARAM\*)lpParameter)->sock;

HWND hwnd=((RECVPARAM\*)lpParameter)->hwnd;

char tempBuf[300];

//1.创建套接字

sock = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if(INVALID\_SOCKET == WSAGetLastError())

AfxMessageBox("Create Socket failed!");

//2.绑定套接字到一个端口号

SOCKADDR\_IN addrSrv;

memset(&addrSrv, 0, sizeof(addrSrv));

addrSrv.sin\_addr.S\_un.S\_addr = htonl(INADDR\_ANY);

addrSrv.sin\_family = AF\_INET;

addrSrv.sin\_port = htons(6000);

bind(sock, (SOCKADDR\*)&addrSrv, sizeof(SOCKADDR));

//3.设置套接字为监听模式

listen(sock,5);

//4.等待用户的连接到来

SOCKADDR\_IN addrClient;

int sockLen = sizeof(SOCKADDR);

while(1)

{

SOCKET sockConn = accept(sock, (SOCKADDR\*)&addrClient, &sockLen);

if(INVALID\_SOCKET == WSAGetLastError())

AfxMessageBox("Connect client failed!");

//5.用返回的套接字和用户通信，发送数据或是接受数据

char recvBuf[128];

recv(sockConn, recvBuf, 128, 0);

if(SOCKET\_ERROR == WSAGetLastError())

AfxMessageBox("Server recieve data failed!");

::PostMessage(hwnd,WM\_RECVDATA,0,(LPARAM)recvBuf);

UserSocket.push\_back(sockConn);

}

return 0;

}

**处理连接请求：**

if(vecString[0] == "Login")

{

CString str;

str.Format("用户：%s 请求链接服务器！",vecString[1]);

if(IDYES==AfxMessageBox(str,MB\_YESNO|MB\_ICONQUESTION))

{

m\_userNum++;

MessageBox("已允许链接");

//创建新的用户信息表

UserInfo newUserInfo;

newUserInfo.UserId = m\_userNum;

newUserInfo.time = CTime::GetCurrentTime();

newUserInfo.UserName = vecString[1];

newUserInfo.UserIp = vecString[2];

newUserInfo.port = atoi(vecString[3]);

newUserInfo.m\_socket = UserSocket[m\_userNum-1];

Exint \* tempInt = new Exint;

tempInt->number = m\_userNum;

tempInt->hwnd = m\_hWnd;

newUserInfo.ThreadId = CreateThread(NULL,0,CreatUser,(LPVOID)(tempInt),0,NULL);//创建子线程进行通信

user\_info.push\_back(newUserInfo);

//以下是设置控件内容

//设置状态栏信息

CString temp;

temp.Format("当前在线数为：%d",m\_userNum);

m\_StatBar->SetText(temp,0,0);

//以下是对listcontrol的更新

int nIndex = m\_list.GetItemCount();

LV\_ITEM lvitemAdd = {0};

lvitemAdd.mask = LVIF\_TEXT|LVIF\_IMAGE;

lvitemAdd.iItem = nIndex ;

lvitemAdd.iSubItem = 0;

lvitemAdd.pszText = newUserInfo.UserName.GetBuffer();

newUserInfo.UserName.ReleaseBuffer();

lvitemAdd.iImage = 0; //(某个窗口的窗口句柄)

m\_list.InsertItem(&lvitemAdd);

/\*

int nIndex1 = m\_list\_ChatContext.GetItemCount();

LV\_ITEM lvitemAdd1 = {0};

lvitemAdd1.mask = LVIF\_TEXT;

lvitemAdd1.iItem = nIndex1 ;

lvitemAdd1.iSubItem = 0;

temp.Format("%s:加入聊天室",newUserInfo.UserName);

lvitemAdd1.pszText = temp.GetBuffer();

temp.ReleaseBuffer();

m\_list\_ChatContext.InsertItem(&lvitemAdd1);

\*/

//发送用户信息

char sendBuf[] = "y";

send(UserSocket[m\_userNum-1], sendBuf, 100, 0);

Sleep(1000);

CString strList = "LOGIN," + newUserInfo.UserName;

SendMessage\_r(strList);

}

**客户端发送信息：**

void Cmain::OnBnClickedButtonSend()

{

// TODO: 在此添加控件通知处理程序代码

UpdateData(true);

if(m\_SendMessage == "")

{

AfxMessageBox("要发送的内容不能为空！");

}

else

{

CString str = m\_name + ":" + m\_SendMessage;

char sendBuf[128];

char \* temp = (LPSTR)(LPCSTR)str;

strcpy(sendBuf,temp);

send(m\_sock, sendBuf,128, 0);

m\_SendMessage = "";

UpdateData(false);

}

}

**将用户登录情况写入日志:**

CString Fpatch = ".\\log\\Userinfo.txt";

CTime tempTime = CTime::GetCurrentTime();

CTimeSpan span = tempTime - user\_info[ii].time;

CString str1 = "用户:"+ user\_info[ii].UserName + "\n";

CString str2 = "IP地址为: " + user\_info[ii].UserIp + "\n";

CString str3 = "登录时间为：" + user\_info[ii].time.Format("%Y-%m-%d %H:%M:%S %W-%A") + "\n";

CString str4 = "退出时间为: " + tempTime.Format("%Y-%m-%d %H:%M:%S %W-%A") + "\n";

CString str5 = "在线时间为：" + span.Format("%D天-%H小时-%M分钟-%S秒") + "\n" + "\n";

CString WriteStr = str1 + str2 + str3 + str4 + str5;