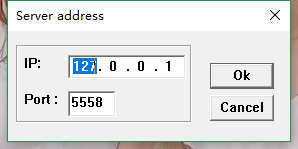
# Computer network project

# B0529001

# 馮少迪

操作說明：

先打開 sever.exe



設定好IP與port,點擊OK

再打開client.exe

輸入地址與端口，再輸入用戶名與密碼（密碼只能是數字）點登陸

如果沒有賬號，則一樣直接輸入，會直接新建一個賬號并登陸。

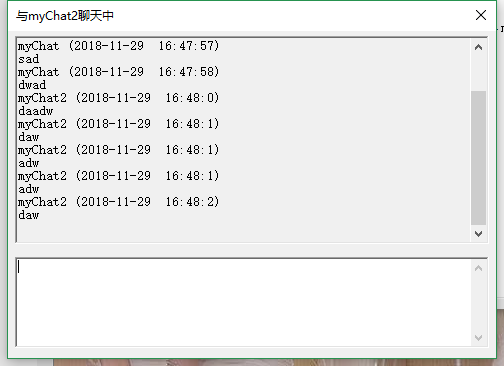


進入聊天名單



點擊一個使用者

輸入消息聊天



聊天消息會儲存在chatlog.txt.

程式說明

Sever

Class”

CServerDig:建立伺服器主視窗，管理使用者資訊鏈錶，管理線上使用者鏈錶，管理離線訊息鏈錶，讀取儲存使用者資訊，讀取儲存離線訊息。

class CServerDlg : public CDialog

{

// Construction

public:

CServerDlg(CWnd\* pParent = NULL); //构造函数

virtual ~CServerDlg(); //析构函数

// Dialog Data

//{{AFX\_DATA(CServerDlg)

enum { IDD = IDD\_SERVER\_MAIN\_DIALOG };

CListCtrl m\_ctlUserList;

//}}AFX\_DATA

// ClassWizard generated virtual function overrides

//{{AFX\_VIRTUAL(CServerDlg)

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

//}}AFX\_VIRTUAL

// Implementation

public:

BOOL Init(void); //初始化

BOOL StartService(void); //启动服务

BOOL ProcessPendingRead(CArchive \*pArchiveIn,

CArchive \*pArchiveOut,CClientSocket \*pClientSocket);//接收数据

void ProcessPendingAccept(void); //接受客户端连接请求

void ReleaseChatter(CClientSocket \*pClientSocket); //当用户离线时删除用户

void DeleteChatter(CClientSocket \*pClientSocket); //用户登录密码错误或者重复登录时删除用户

void DeleteAllChatter(void); //删除所有在线用户

protected:

void InitListCtrlSetting(void); //定义列表控件

void InitUserList(void); //初始化用户列表

void SaveUserList(CObList &obList); //保存用户信息

void LoadUserList(CObList &obList); //读取用户信息

void SaveOfflineMsg(CObList &obList); //保存离线消息

void LoadOfflineMsg(CObList &obList); //读取离线消息

void SendUserList(void); //发送用户列表

void CopyUserList(CObList &obList); //复制用链表

void DeleteTempUserList(CObList &obList); //删除临时链表

BOOL UpdateUserList(const CUserInfo &userInfo, CClientSocket \*pClientSocket);//更新用户链表

void TransmitMsg(const CChatPacket &packet, CClientSocket \*pClientSocket); //转发离线消息

void UpdateServerListCtl(const CObList &obList); //更新服务器界面

protected:

HICON m\_hIcon;

// Generated message map functions

//{{AFX\_MSG(CServerDlg)

virtual BOOL OnInitDialog();

afx\_msg void OnPaint();

afx\_msg HCURSOR OnQueryDragIcon();

afx\_msg void OnDestroy();

afx\_msg void OnRclickClientList(NMHDR\* pNMHDR, LRESULT\* pResult);

afx\_msg void OnServerListDeleteMenuitem();

//}}AFX\_MSG

DECLARE\_MESSAGE\_MAP()

CListenSocket \*m\_pListenSocket; //CListenSocket指针

CImageList \*m\_pImageList; //图标列表指针

CObList m\_UserList; //用户链表

CObList m\_OfflineMsgList; //离线信息链表

CObList m\_ChatterList; //在线用户链表

CCriticalSection m\_csUserList; //保护用户链表临界区对象

CCriticalSection m\_csOfflineList;//保护离线信息链表临界区对象

CCriticalSection m\_csChatterList;//保护在线用户链表临界区对象

};

CListenSocket:繼承CSocket,監聽客戶端連線請求。

CClistenSocket:繼承Csocket接收客戶端資料，向客戶端傳送資料

CSeverAddressDig:登錄的對話方塊，設置伺服器的IP和PORT

class CServerAddressDlg : public CDialog

{

// Construction

public:

CServerAddressDlg(CWnd\* pParent = NULL); // standard constructor

// Dialog Data

//{{AFX\_DATA(CServerAddressDlg)

enum { IDD = IDD\_SERVER\_ADDRESS\_DLG };

CIPAddressCtrl m\_ctlServIP;

CEdit m\_ctlServPort;

//}}AFX\_DATA

// Overrides

// ClassWizard generated virtual function overrides

//{{AFX\_VIRTUAL(CServerAddressDlg)

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

//}}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

//{{AFX\_MSG(CServerAddressDlg)

virtual void OnOK();

virtual BOOL OnInitDialog();

//}}AFX\_MSG

DECLARE\_MESSAGE\_MAP()

public:

DWORD m\_strServIP; //服务器IP地址

SHORT m\_shServPort; //服务器端口

};

CChatPacket:資料封包類別

class CChatPacket : public CObject

{

public:

CChatPacket();

virtual ~CChatPacket();

DECLARE\_SERIAL( CChatPacket )

enum PACKETTYPE //包的类型

{

MESSAGE, //聊天消息

USERLIST, //用户列表

SERVERMSG, //服务器发送给用户的消息

UNKNOWN //未知类型

};

public:

void Init(void); //初始化成员变量

virtual void Serialize(CArchive& ar); //序列化

public:

PACKETTYPE m\_type; //包类型

CUserInfo m\_UserInfo; //用户信息

CString m\_strMsg; //消息

CObList \*m\_pUserList; //用户链表

CUserInfo m\_OfflineUserInfo; //离线用户信息

CTime m\_time; //日期和时间

};

CUserInfo:使用者資訊類別。

class CUserInfo : public CObject

{

public:

CUserInfo();

CUserInfo(const CUserInfo& userInfo);

virtual ~CUserInfo();

DECLARE\_SERIAL(CUserInfo)

enum USERSTATUE{

ONLINE, //在线

OFFLINE, //离线

LOGIN, //登录

UNKNOWN //未知

};

public:

void Init(void); //初始化

virtual void Serialize(CArchive& ar); //序列化

CUserInfo &operator = (const CUserInfo &userInfo); //赋值函数

public:

CString m\_strName; //名称

CString m\_strPassword; //密码

USERSTATUE m\_eStatus; //状态

DWORD m\_lIP; //IP地址

SHORT m\_nPort; //端口

CTime m\_time; //日期和时间

};

CseverApp:應用程序實體類別

class CServerApp : public CWinApp

{

public:

CServerApp();

// Overrides

// ClassWizard generated virtual function overrides

//{{AFX\_VIRTUAL(CServerApp)

public:

virtual BOOL InitInstance();

//}}AFX\_VIRTUAL

// Implementation

//{{AFX\_MSG(CServerApp)

// NOTE - the ClassWizard will add and remove member functions here.

// DO NOT EDIT what you see in these blocks of generated code !

//}}AFX\_MSG

DECLARE\_MESSAGE\_MAP()

};

在CSeverApp類別的InitInstance()中，宣告一個CSeverDig的實體，然後呼叫StartSever()

如果成功，則呼叫CDialog中的DoModal()的涵式建立伺服器主界面。如果失敗，則退出。

在該涵式中宣告CSeverAddressDig 實體，建立伺服器位址和port對話方塊，當按下OK，則呼叫該類別中的OnOk（）涵式。

在startSever（）中，首先建立一個CListenSocket,然後呼叫CSocket()類別中的Creat()完成Socket建立和綁定。呼叫create()的時候，建立一個不顯示的視窗，并呼叫WSAAsyncSelect()涵式註冊網路事件。在程式中使用預設參數呼叫Create()涵式，建立串流式Socket,註冊全部網路事件。

在severDig中的init()中，initUserList涵式實作初始化使用者列表的功能

首先呼叫LoadUserList()涵式從“userlist.ini”檔案中讀取使用者資訊。

在initUserList中使用COblist類別GetHeaderPosition()取得節點頭，然後不斷呼叫CObList類別中的GetNext()取得下一個節點。當走訪到最後則結束。

InserItem()插入資料項目，設置使用者為離線狀態，然後呼叫CListCtrl類別的SetItemText()為每個資料項目插入子項目。

LoadOfflineMsg()涵式從“offlinemesglist.ini”中讀取使用者離線訊息

在CListenSocket中的OnAceept()中，呼叫CSeverDig類別ProcessPendingAccept()涵式接受客戶端的連線請求。其中呼叫了CSocket的PumpMessages（）用於阻塞，CClientSocket類別的init()，建立一個CSocketFile物件，為CArchive物件接受和傳送資料做準備。

CsocketWind中的OnSocketNotify()涵式被呼叫時，會首先呼叫CSocket類別的AuxQueueAdd(),將網路事件訊息加入Socket執行狀態變數的鏈錶。然後呼叫CSocket的ProcessAuxQueue（），將網路事件訊息取出來。

在CClientSocket的OnReceive中，建立讀取寫入檔案物件，然後呼叫CSeverDig的ProcessPendingRead(),如果傳送資料的使用者密碼錯誤或者重新登錄，則刪除使用者。

DoCallBack()呼叫衍生類別的OnrReceive（）的虛擬涵式。OnReceive呼叫CSeverDig類別ProcessPendingRead(),最終呼叫到CChatPacket類別Serilalize()。Serialize中的檔案物件呼叫CSockeFile類別的Read(),最終呼叫CSocket類別的Receive()接受客戶端資料。

在CSeverDig類別中的ProcessPendingRead()中，對接收的資料做如下處理：儲存離線訊息，儲存使用者資訊，更行使用者列表，傳送使用者列表，轉發離線訊息和更新伺服器界面。

CSeverDig 類別的UpdateUserList()涵式作為對伺服器使用者的列表的更新。

使用者成功登陸伺服器，伺服器向所有的線上使用者傳送使用者；列表，通知他們更新使用者列表。

SendUserList()涵式實作向線上使用者傳送使用者列表的功能。

TransmitMsg()涵式轉送離線訊息。

SendUserMsg()將離線訊息傳送給使用者。

UppdateSeverListCtl()實作伺服器界面的更新。

OnDestory()呼叫SaveUserList()儲存使用者資訊，SaveOfficeMsg涵式儲存離線資訊，DeleteAllChatter刪除線上使用者。并刪除CListenSocket和CImageList物件。

SaveUserList()實作儲存使用者資訊的功能。伺服器啟動時，在InitUserList中呼叫LoadUserList().

SaveOfflineMsg()涵式實作儲存離線訊息的功能。

**Client**

CClistenDig:建立客戶端主視窗，建立聊天視窗，接受和傳送離線訊息。

CChatDig:編輯和顯示信息。

CListenSocket:監聽其他客戶連線請求，

CChatSocket:接受資料封包。

CCListenLoginDig:登錄伺服器對話方塊。

CChatPacket:資料封包的CLASS。

CUserInfo:使用者資訊。

CClistenApp:應用程式class

ConnectToSeve()連線伺服器，請求使用者列表，初始化客戶端。CreateChatDig()用來建立聊天視窗。

BeginingListen用於監聽其他客戶連線請求。

ConnectToPeer()用於向另外一個使用者發起聊天請求。

首先InitListenCtrlSetting實作初始化列表控制項的功能。

然後BegingListen建立Socket，呼叫Listen()開始監聽。在呼叫GetSocketName()取得port.

連接伺服器之前先建立CChatSocket物件，再建立登錄對話方塊，取得伺服器位置，port，name和password。按下登陸后呼叫Connect()連接伺服器。

連接成功后，呼叫GetSoketName()涵式取得IP位址，然後定義資料封包，再呼叫sendPacket（）涵式講資料包傳送給伺服器。

CChatSocket的OnReceive()涵式被呼叫，在該涵式中呼叫CClistenDig類比的ProcessPendingRead()涵式，其中呼叫ReadPacket()，接受伺服器資料。當接受資料是離線訊息時，呼叫ShowOfflineMsg()涵式顯示訊息，當接受使用者列表時，呼叫CreateUserList()建立使用者列表，然後UpdateClistenList()。如果登錄失敗，則呼叫EndDialog退出。

ShowOfflineMsg()呼叫時，會呼叫CChatDig建構離線訊息視窗。

DisplayOfflineMessage()實作離線訊息的功能。

CClistenDig類別的OnDblclkClientList()涵式被回應。在該涵式內呼叫CListCtrl類比的GetItemTest()涵式取得選中資料項目的名稱，呼叫CreateChatDig()涵式建立聊天視窗，并判斷使用者身份。

CChatDig中的ConnecgtToPeer涵式建立CChatSocket物件，并發出連線請求。當傳送訊息后，PerTranslateMessage()涵式中攔截該鍵盤訊息進行處理。

DisplaySentMessage()中利用CTime類別中的Format()格式化日期和時間。

當要接受訊息時，CChatSocket中的OnReceive()被呼叫，該涵式呼叫CChatDig類別的ProcessPendingRead()涵式，并接受訊息，同時呼叫DisplayRecvMessage()顯示訊息。

# Code

**Sever:**

**ChatPacket.h**

#if !defined(AFX\_CHATPACKET\_H\_\_7F5E67FB\_273B\_4679\_8A59\_23575978DDCC\_\_INCLUDED\_)

#define AFX\_CHATPACKET\_H\_\_7F5E67FB\_273B\_4679\_8A59\_23575978DDCC\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

#include "UserInfo.h"

class CUserInfo;

class CChatPacket : public CObject

{

public:

CChatPacket();

virtual ~CChatPacket();

DECLARE\_SERIAL( CChatPacket )

enum PACKETTYPE //包的类型

{

MESSAGE, //聊天消息

USERLIST, //用户列表

SERVERMSG, //服务器发送给用户的消息

UNKNOWN //未知类型

};

public:

void Init(void); //初始化成员变量

virtual void Serialize(CArchive& ar); //序列化

public:

PACKETTYPE m\_type; //包类型

CUserInfo m\_UserInfo; //用户信息

CString m\_strMsg; //消息

CObList \*m\_pUserList; //用户链表

CUserInfo m\_OfflineUserInfo; //离线用户信息

CTime m\_time; //日期和时间

};

**ChatPacket.cpp**

#include "stdafx.h"

#include "ChatPacket.h"

#ifdef \_DEBUG

#undef THIS\_FILE

static char THIS\_FILE[]=\_\_FILE\_\_;

#define new DEBUG\_NEW

#endif

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

//IMPLEMENT\_DYNCREATE(CChatPacket, CObject)

IMPLEMENT\_SERIAL(CChatPacket, CObject, 1 )

CChatPacket::CChatPacket()

{

Init();

}

CChatPacket::~CChatPacket()

{

}

/\*

\* 初始化

\*/

void CChatPacket::Init(void)

{

m\_type = UNKNOWN; //为知类型

m\_strMsg = \_T(""); //清空

m\_pUserList = NULL; //清空

m\_time = CTime::GetCurrentTime();

}

/\*

\* 序列化

\*/

void CChatPacket::Serialize(CArchive& ar)

{

CObject::Serialize(ar);//调用基类的序列化函数

if (ar.IsStoring())//发送数据

{

BYTE byType = m\_type;

ar << byType; //包的类型

ar << m\_strMsg; //消息

long lTime = m\_time.GetTime(); //日期和时间

ar << lTime;

}else//接收数据

{

BYTE byType;

ar >> byType;

m\_type = (PACKETTYPE)byType; //包的类型

ar >> m\_strMsg; //消息

long lTime;

ar >> lTime; //日期和时间

m\_time = CTime((time\_t)lTime);

}

m\_UserInfo.Serialize(ar); //序列化用户信息

m\_OfflineUserInfo.Serialize(ar); //序列化离线用户信息

if (m\_type == USERLIST && NULL != m\_pUserList)//序列化用户列表

{

m\_pUserList->Serialize(ar);

}

}

**ClientSocket.cpp**

// ClientSocket.cpp : implementation file

//

#include "stdafx.h"

#include "Server.h"

#include "ClientSocket.h"

#include "ServerDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CClientSocket

CClientSocket::CClientSocket(CServerDlg \*pServDlg)

{

m\_pServDlg = pServDlg;

}

CClientSocket::~CClientSocket()

{

m\_pServDlg = NULL;

if (m\_pFile != NULL)

delete m\_pFile;

}

// Do not edit the following lines, which are needed by ClassWizard.

#if 0

BEGIN\_MESSAGE\_MAP(CClientSocket, CSocket)

//{{AFX\_MSG\_MAP(CClientSocket)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

#endif // 0

/////////////////////////////////////////////////////////////////////////////

// CClientSocket member functions

/\*

\* 初始化

\*/

void CClientSocket::Init(void)

{

//创建CSocketFile对象

m\_pFile = new CSocketFile(this);

}

/\*

\* 接收数据

\*/

void CClientSocket::OnReceive(int nErrorCode)

{

CArchive \*pArchiveIn = new CArchive(m\_pFile,CArchive::load); //创建读入文档对象

CArchive \*pArchiveOut = new CArchive(m\_pFile,CArchive::store); //创建写入文档对象

BOOL reVal = m\_pServDlg->ProcessPendingRead(pArchiveIn, pArchiveOut, this); //接收数据

delete pArchiveIn; //删除读入文档对象

pArchiveIn = NULL;

delete pArchiveOut; //删除写入文档对象

pArchiveOut = NULL;

if (FALSE == reVal) //用户密码错误，或者重复登录

{

delete this; //删除自身

}

}

/\*

\* 关闭套接字

\*/

void CClientSocket::OnClose(int nErrorCode)

{

//通知用户不再发送数据

ShutDown(sends);

//更新用户状态

m\_pServDlg->ReleaseChatter(this);

delete this;

}

/\*

\* 发送用户列表

\*/

void CClientSocket::SendUserList(CChatPacket \*pPacket)

{

CArchive \*pArchiveOut = new CArchive(m\_pFile,CArchive::store); //初始化写入流

pPacket->Serialize(\*pArchiveOut);

pArchiveOut->Flush();

delete pArchiveOut;

pArchiveOut = NULL;

}

/\*

\* 发送消息

\*/

void CClientSocket::SendUserMsg(CChatPacket \*pPacket)

{

CArchive \*pArchiveOut = new CArchive(m\_pFile,CArchive::store);//创建写入流

pPacket->Serialize(\*pArchiveOut); //发送数据

pArchiveOut->Flush();

delete pArchiveOut; //删除写入流

pArchiveOut = NULL;

}

/\*

\* 保存用户地址

\*/

void CClientSocket::SaveUserInfo(SOCKADDR\_IN clientAddr)

{

m\_userInfo.m\_lIP = clientAddr.sin\_addr.S\_un.S\_addr;

m\_userInfo.m\_nPort = clientAddr.sin\_port;

}

/\*

\* 保存用户状态信息

\*/

void CClientSocket::SaveUserInfo(const CUserInfo &userInfo)

{

m\_userInfo.m\_eStatus = userInfo.m\_eStatus;

m\_userInfo.m\_strName = userInfo.m\_strName;

m\_userInfo.m\_strPassword = userInfo.m\_strPassword;

m\_userInfo.m\_time = userInfo.m\_time;

}

/\*

\* 保存用户信息

\*/

CUserInfo CClientSocket::GetUserInfo(void)

{

return m\_userInfo;

}

**ListenSocket.cpp**

// ListenSocket.cpp : implementation file

//

#include "stdafx.h"

#include "Server.h"

#include "ListenSocket.h"

#include "ServerDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CListenSocket

CListenSocket::CListenSocket(CServerDlg\* pServerDlg)

{

m\_pServerDlg = pServerDlg;

}

CListenSocket::~CListenSocket()

{

m\_pServerDlg = NULL;

// Close();

}

// Do not edit the following lines, which are needed by ClassWizard.

#if 0

BEGIN\_MESSAGE\_MAP(CListenSocket, CSocket)

//{{AFX\_MSG\_MAP(CListenSocket)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

#endif // 0

/////////////////////////////////////////////////////////////////////////////

// CListenSocket member functions

/\*

\* 接受客户端的链接请求

\*/

void CListenSocket::OnAccept(int nErrorCode)

{

m\_pServerDlg->ProcessPendingAccept();

}

**Server.CPP**

// Server.cpp : Defines the class behaviors for the application.

//

#include "stdafx.h"

#include "Server.h"

#include "ServerDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CServerApp

BEGIN\_MESSAGE\_MAP(CServerApp, CWinApp)

//{{AFX\_MSG\_MAP(CServerApp)

// NOTE - the ClassWizard will add and remove mapping macros here.

// DO NOT EDIT what you see in these blocks of generated code!

//}}AFX\_MSG

ON\_COMMAND(ID\_HELP, CWinApp::OnHelp)

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CServerApp construction

CServerApp::CServerApp()

{

// TODO: add construction code here,

// Place all significant initialization in InitInstance

}

/////////////////////////////////////////////////////////////////////////////

// The one and only CServerApp object

CServerApp theApp;

/////////////////////////////////////////////////////////////////////////////

// CServerApp initialization

BOOL CServerApp::InitInstance()

{

if (!AfxSocketInit())

{

AfxMessageBox(IDP\_SOCKETS\_INIT\_FAILED);

return FALSE;

}

AfxEnableControlContainer();

// Standard initialization

// If you are not using these features and wish to reduce the size

// of your final executable, you should remove from the following

// the specific initialization routines you do not need.

#ifdef \_AFXDLL

Enable3dControls(); // Call this when using MFC in a shared DLL

#else

Enable3dControlsStatic(); // Call this when linking to MFC statically

#endif

CServerDlg dlg;

if (!dlg.StartService())

{

AfxMessageBox(\_T("启动服务器失败！"),MB\_OK, -1);

return FALSE;

}

m\_pMainWnd = &dlg;

int nResponse = dlg.DoModal();

if (nResponse == IDOK)

{

// TODO: Place code here to handle when the dialog is

// dismissed with OK

}

else if (nResponse == IDCANCEL)

{

// TODO: Place code here to handle when the dialog is

// dismissed with Cancel

}

// Since the dialog has been closed, return FALSE so that we exit the

// application, rather than start the application's message pump.

return FALSE;

}

**ServerAddressDlg.cpp**

// ServerAddressDlg.cpp : implementation file

//

#include "stdafx.h"

#include "Server.h"

#include "ServerAddressDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CServerAddressDlg dialog

CServerAddressDlg::CServerAddressDlg(CWnd\* pParent /\*=NULL\*/)

: CDialog(CServerAddressDlg::IDD, pParent)

{

//{{AFX\_DATA\_INIT(CServerAddressDlg)

//}}AFX\_DATA\_INIT

}

void CServerAddressDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

//{{AFX\_DATA\_MAP(CServerAddressDlg)

DDX\_Control(pDX, IDC\_SERVER\_IPADDRESS, m\_ctlServIP);

DDX\_Control(pDX, IDC\_SERVER\_PORT\_EDIT, m\_ctlServPort);

//}}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP(CServerAddressDlg, CDialog)

//{{AFX\_MSG\_MAP(CServerAddressDlg)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CServerAddressDlg message handlers

void CServerAddressDlg::OnOK()

{

//获取服务器IP和端口

m\_ctlServIP.GetAddress(m\_strServIP);

CString strServPort;

m\_ctlServPort.GetWindowText(strServPort);

m\_shServPort = atoi(strServPort); //将字符串转换为SHORT

CDialog::OnOK();

}

BOOL CServerAddressDlg::OnInitDialog()

{

CDialog::OnInitDialog();

m\_ctlServIP.SetAddress(127, 0, 0, 1);

CString strServIP;

strServIP.Format("%d",SERVERPORT);

m\_ctlServPort.SetWindowText(strServIP);

// TODO: Add extra initialization here

return TRUE; // return TRUE unless you set the focus to a control

// EXCEPTION: OCX Property Pages should return FALSE

}

**ServerDlg.cpp**

**// ServerDlg.cpp : implementation file**

**//**

**#include "stdafx.h"**

**#include "Server.h"**

**#include "ServerDlg.h"**

**#include "ClientSocket.h"**

**#include "ListenSocket.h"**

**#include "ServerAddressDlg.h"**

**#ifdef \_DEBUG**

**#define new DEBUG\_NEW**

**#undef THIS\_FILE**

**static char THIS\_FILE[] = \_\_FILE\_\_;**

**#endif**

**/////////////////////////////////////////////////////////////////////////////**

**// CServerDlg dialog**

**CServerDlg::CServerDlg(CWnd\* pParent /\*=NULL\*/)**

**: CDialog(CServerDlg::IDD, pParent)**

**{**

**//{{AFX\_DATA\_INIT(CServerDlg)**

**// NOTE: the ClassWizard will add member initialization here**

**//}}AFX\_DATA\_INIT**

**// Note that LoadIcon does not require a subsequent DestroyIcon in Win32**

**m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);**

**m\_pListenSocket = NULL; //监听套接字**

**m\_pImageList = NULL; //图标列表**

**}**

**void CServerDlg::DoDataExchange(CDataExchange\* pDX)**

**{**

**CDialog::DoDataExchange(pDX);**

**//{{AFX\_DATA\_MAP(CServerDlg)**

**DDX\_Control(pDX, IDC\_CLIENT\_LIST, m\_ctlUserList);**

**//}}AFX\_DATA\_MAP**

**}**

**BEGIN\_MESSAGE\_MAP(CServerDlg, CDialog)**

**//{{AFX\_MSG\_MAP(CServerDlg)**

**ON\_WM\_PAINT()**

**ON\_WM\_QUERYDRAGICON()**

**ON\_WM\_DESTROY()**

**ON\_NOTIFY(NM\_RCLICK, IDC\_CLIENT\_LIST, OnRclickClientList)**

**ON\_COMMAND(ID\_SERVER\_LIST\_DELETE\_MENUITEM, OnServerListDeleteMenuitem)**

**//}}AFX\_MSG\_MAP**

**END\_MESSAGE\_MAP()**

**/////////////////////////////////////////////////////////////////////////////**

**// CServerDlg message handlers**

**BOOL CServerDlg::OnInitDialog()**

**{**

**CDialog::OnInitDialog();**

**// Set the icon for this dialog. The framework does this automatically**

**// when the application's main window is not a dialog**

**SetIcon(m\_hIcon, TRUE); // Set big icon**

**SetIcon(m\_hIcon, FALSE); // Set small icon**

**//初始化**

**Init();**

**return TRUE;**

**}**

**// If you add a minimize button to your dialog, you will need the code below**

**// to draw the icon. For MFC applications using the document/view model,**

**// this is automatically done for you by the framework.**

**void CServerDlg::OnPaint()**

**{**

**if (IsIconic())**

**{**

**CPaintDC dc(this); // device context for painting**

**SendMessage(WM\_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);**

**// Center icon in client rectangle**

**int cxIcon = GetSystemMetrics(SM\_CXICON);**

**int cyIcon = GetSystemMetrics(SM\_CYICON);**

**CRect rect;**

**GetClientRect(&rect);**

**int x = (rect.Width() - cxIcon + 1) / 2;**

**int y = (rect.Height() - cyIcon + 1) / 2;**

**// Draw the icon**

**dc.DrawIcon(x, y, m\_hIcon);**

**}**

**else**

**{**

**CDialog::OnPaint();**

**}**

**}**

**// The system calls this to obtain the cursor to display while the user drags**

**// the minimized window.**

**HCURSOR CServerDlg::OnQueryDragIcon()**

**{**

**return (HCURSOR) m\_hIcon;**

**}**

**CServerDlg::~CServerDlg()**

**{**

**}**

**/\***

**\* 接受客户端连接请求**

**\*/**

**void CServerDlg::ProcessPendingAccept(void)**

**{**

**CClientSocket \*pClient = new CClientSocket(this);//创建实例**

**SOCKADDR\_IN clientAddr;**

**int socketLen = sizeof(SOCKADDR\_IN);**

**m\_pListenSocket->Accept(\*pClient,(SOCKADDR\*)&clientAddr, &socketLen);//接受客户端的连接**

**pClient->Init(); //初始化**

**pClient->SaveUserInfo(clientAddr); //保存用户的地址**

**m\_csChatterList.Lock();**

**m\_ChatterList.AddTail(pClient); //加入在线用户链表**

**m\_csChatterList.Unlock();**

**}**

**/\***

**\* 初始化**

**\*/**

**BOOL CServerDlg::Init(void)**

**{**

**InitListCtrlSetting(); //定义列表控件**

**InitUserList(); //初始化用户列表**

**LoadOfflineMsg(m\_OfflineMsgList); //读取离线消息**

**return TRUE;**

**}**

**/\***

**\* 启动服务**

**\*/**

**BOOL CServerDlg::StartService(void)**

**{**

**CServerAddressDlg servAddrDlg; //服务器地址和端口对话框**

**if (IDOK != servAddrDlg.DoModal())**

**{**

**return FALSE;**

**}**

**//获取服务器地址**

**in\_addr servAdd;**

**servAdd.S\_un.S\_addr = htonl(servAddrDlg.m\_strServIP); //主机字节转换为网络字节**

**CString strServIP = inet\_ntoa(servAdd); //转换为点格式**

**//创建CListenSocket对象**

**m\_pListenSocket = new CListenSocket(this);**

**//创建套接字**

**if (m\_pListenSocket->Create(servAddrDlg.m\_shServPort, SOCK\_STREAM,strServIP))**

**{**

**//开始监听**

**if (!m\_pListenSocket->Listen())**

**{**

**delete m\_pListenSocket;**

**m\_pListenSocket = NULL;**

**AfxMessageBox(\_T("创建套接字失败！"));**

**return FALSE;**

**}**

**}else**

**{**

**delete m\_pListenSocket;**

**m\_pListenSocket = NULL;**

**AfxMessageBox(\_T("创建套接字失败！"));**

**return FALSE;**

**}**

**return TRUE;**

**}**

**/\***

**\* 定义列表控件**

**\*/**

**void CServerDlg::InitListCtrlSetting(void)**

**{**

**m\_pImageList = new CImageList();**

**ASSERT(m\_pImageList != NULL);**

**//32\*32，8位，初始化为2个图标，每次增长2个图标**

**m\_pImageList->Create(32, 32, ILC\_COLOR8|ILC\_MASK, 2, 2);**

**CWinApp\* pApp = AfxGetApp(); //获得应用程序指针**

**HICON hIconOnline = pApp->LoadIcon(IDI\_CHATTER\_ONLINE); //在线图标**

**HICON hIconOffline = pApp->LoadIcon(IDI\_CHATTER\_OFFLINE); //离线图标**

**m\_pImageList->Add(hIconOnline); //加入图标**

**m\_pImageList->Add(hIconOffline); //加入图标**

**//将图标列表赋值给列表控件**

**m\_ctlUserList.SetImageList(m\_pImageList, LVSIL\_SMALL);**

**//设置列表头**

**m\_ctlUserList.InsertColumn(0, \_T("名称"), LVCFMT\_IMAGE|LVCFMT\_LEFT);**

**m\_ctlUserList.InsertColumn(1, \_T("密码"), LVCFMT\_CENTER);**

**m\_ctlUserList.InsertColumn(2, \_T("地址"), LVCFMT\_CENTER);**

**m\_ctlUserList.InsertColumn(3, \_T("端口"), LVCFMT\_CENTER);**

**m\_ctlUserList.InsertColumn(4, \_T("时间"), LVCFMT\_CENTER);**

**//设置各列的宽度**

**CRect rect;**

**m\_ctlUserList.GetClientRect(&rect);//获取列表视图控件的宽度**

**int nWidth = rect.Width();**

**m\_ctlUserList.SetColumnWidth(0, nWidth \* 25 / 100);//名称占25%**

**m\_ctlUserList.SetColumnWidth(1, nWidth \* 13 / 100);//密码占13%**

**m\_ctlUserList.SetColumnWidth(2, nWidth \* 25 / 100);//地址占25%**

**m\_ctlUserList.SetColumnWidth(3, nWidth \* 12 / 100);//端口占12%**

**m\_ctlUserList.SetColumnWidth(4, nWidth \* 25 / 100);//时间占25%**

**}**

**/\***

**\* 初始化用户列表**

**\*/**

**void CServerDlg::InitUserList(void)**

**{**

**LoadUserList(m\_UserList);//读取用户信息**

**//遍历链表的每个节点，读取数据，初始化用户列表**

**POSITION pos;**

**int nLine = 0;**

**for (pos = m\_UserList.GetHeadPosition(); NULL != pos;)**

**{**

**CUserInfo \*pUserInfo = (CUserInfo\*)m\_UserList.GetNext(pos);**

**if (NULL != pUserInfo)**

**{**

**m\_ctlUserList.InsertItem(nLine,pUserInfo->m\_strName,1); //名称，用户初始化为离线状态**

**m\_ctlUserList.SetItemText(nLine,1,pUserInfo->m\_strPassword); //密码**

**in\_addr userAddr;**

**userAddr.S\_un.S\_addr = (long)pUserInfo->m\_lIP;**

**CString strUserAddr(inet\_ntoa(userAddr)); //IP**

**m\_ctlUserList.SetItemText(nLine,2,strUserAddr);**

**CString strUserPort; //端口**

**strUserPort.Format("%d",ntohs(pUserInfo->m\_nPort));**

**m\_ctlUserList.SetItemText(nLine,3,strUserPort);**

**CString strUserTime = pUserInfo->m\_time.Format("%c"); //端口**

**m\_ctlUserList.SetItemText(nLine,4,strUserTime);**

**nLine++;**

**}**

**}**

**}**

**/\***

**\* 保存用户信息**

**\*/**

**void CServerDlg::SaveUserList(CObList &obList)**

**{**

**char appPath[256];**

**GetCurrentDirectory(256,appPath); //取得应用程序当前路径**

**CString filePath; //保存Ini文件名**

**filePath.Format("%s",appPath);**

**filePath +="\\";**

**filePath += USERLISTINI;**

**//删除原来的文件**

**DeleteFile(filePath);**

**POSITION pos;**

**int nIndex = 0;**

**//遍历整个用户链表**

**for(pos = obList.GetHeadPosition(); pos != NULL; )**

**{**

**//获取数据**

**CUserInfo \*pUserInfo = (CUserInfo\*)obList.GetNext(pos);//用户信息**

**if (NULL == pUserInfo)**

**{**

**break;**

**}**

**CString strSection("section"); //section**

**CString strIndex;**

**strIndex.Format("%d",nIndex);**

**strSection += strIndex;**

**in\_addr userAddr;**

**userAddr.S\_un.S\_addr = (long)pUserInfo->m\_lIP;**

**CString strUserAddr(inet\_ntoa(userAddr)); //用户IP**

**CString strUserPort; //用户端口**

**strUserPort.Format("%d",pUserInfo->m\_nPort);**

**CString strUserTime; //用户登录线时间**

**strUserTime.Format("%ld", (long)pUserInfo->m\_time.GetTime());**

**//写用户名字**

**WritePrivateProfileString(strSection, \_T("Name"), pUserInfo->m\_strName, filePath);**

**//写用户密码**

**WritePrivateProfileString(strSection, \_T("Password"), pUserInfo->m\_strPassword, filePath);**

**//写用户地址**

**WritePrivateProfileString(strSection, \_T("Address"), strUserAddr,filePath);**

**//写用户端口**

**WritePrivateProfileString(strSection, \_T("Port"), strUserPort,filePath);**

**//写用时间**

**WritePrivateProfileString(strSection, \_T("Time"), strUserTime,filePath);**

**delete pUserInfo;//删除对象**

**pUserInfo = NULL;**

**nIndex++;**

**}**

**obList.RemoveAll();//删除链表所有节点**

**}**

**/\***

**\* 读取用户信息**

**\*/**

**void CServerDlg::LoadUserList(CObList &obList)**

**{**

**#define TEMP\_BUF\_SIZE 32//缓冲区长度**

**char appPath[256];**

**GetCurrentDirectory(256,appPath); //取得应用程序当前路径**

**CString filePath; //保存Ini文件名**

**filePath.Format("%s",appPath);**

**filePath += "\\";**

**filePath += USERLISTINI;**

**int nIndex = 0;**

**while (TRUE)**

**{**

**CString strSection("section");**

**CString strIndex;**

**strIndex.Format("%d",nIndex);**

**strSection += strIndex;**

**//用户名称**

**CString strSectionKey = "Name";**

**CString strValue = \_T("");**

**char cBuf[TEMP\_BUF\_SIZE];**

**memset(cBuf, 0, TEMP\_BUF\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**CUserInfo \*pUserInfo = new CUserInfo();//新建CUserInfo对象**

**pUserInfo->m\_strName = cBuf;**

**//读取用户密码**

**strSectionKey = "Password";**

**memset(cBuf, 0, TEMP\_BUF\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**pUserInfo->m\_strPassword = cBuf;**

**//读取用户地址**

**strSectionKey = "Address";**

**memset(cBuf, 0, TEMP\_BUF\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**u\_long clinetAddr = inet\_addr(cBuf);**

**pUserInfo->m\_lIP = (DWORD)clinetAddr;**

**//读取用户端口**

**strSectionKey = "Port";**

**memset(cBuf, 0, TEMP\_BUF\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**pUserInfo->m\_nPort = atoi(cBuf);**

**//读取用户时间**

**strSectionKey = "Time";**

**memset(cBuf, 0, TEMP\_BUF\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**time\_t time = atol(cBuf);**

**pUserInfo->m\_time = time;**

**obList.AddTail(pUserInfo); //加入链表**

**nIndex++;**

**}**

**}**

**/\***

**\* 保存离线消息**

**\*/**

**void CServerDlg::SaveOfflineMsg(CObList &obList)**

**{**

**char appPath[256];**

**GetCurrentDirectory(256,appPath); //取得应用程序当前路径**

**CString filePath; //保存Ini文件名**

**filePath.Format("%s",appPath);**

**filePath +="\\";**

**filePath += OFFLINEMSGLIST;**

**//删除原来的文件**

**DeleteFile(filePath);**

**POSITION pos;**

**int nIndex = 0;**

**for(pos = obList.GetHeadPosition(); pos != NULL; )**

**{**

**CChatPacket \*pPacket = (CChatPacket\*)obList.GetNext(pos);//用户信息**

**if (NULL == pPacket)**

**{**

**break;**

**}**

**CString strSection("section"); //section**

**CString strIndex;**

**strIndex.Format("%d",nIndex);**

**strSection += strIndex;**

**CString strTime;**

**strTime.Format("%ld",(long)pPacket->m\_time.GetTime());**

**pPacket->m\_strMsg.Replace("\r\n", "$");//取代回车符为$**

**//写发送消息的用户名称**

**WritePrivateProfileString(strSection, \_T("SendMsgUser"), pPacket->m\_UserInfo.m\_strName, filePath); //发送消息用户**

**//写接收消息的用户名称**

**WritePrivateProfileString(strSection, \_T("RecvMsgUser"), pPacket->m\_OfflineUserInfo.m\_strName, filePath); //接收消息用户**

**//写发送消息 的时间**

**WritePrivateProfileString(strSection, \_T("Time"), strTime,filePath); //发送消息时间**

**//写发送的消息**

**WritePrivateProfileString(strSection, \_T("Message"), pPacket->m\_strMsg, filePath); //发送的消息**

**delete pPacket;**

**pPacket = NULL;**

**nIndex++;**

**}**

**obList.RemoveAll();**

**}**

**/\***

**\* 读取离线消息**

**\*/**

**void CServerDlg::LoadOfflineMsg(CObList &obList)**

**{**

**m\_csOfflineList.Lock();**

**char appPath[256];**

**GetCurrentDirectory(256,appPath); //取得应用程序当前路径**

**CString filePath; //保存Ini文件名**

**filePath.Format("%s",appPath);**

**filePath += "\\";**

**filePath += OFFLINEMSGLIST;**

**int nIndex = 0;**

**while (TRUE)**

**{**

**CString strSection("section"); //section**

**CString strIndex;**

**strIndex.Format("%d",nIndex);**

**strSection += strIndex;**

**//读取发送消息的用户名称**

**CString strSectionKey = "SendMsgUser";**

**CString strValue = \_T("");**

**char cBuf[MAX\_MSG\_SIZE];**

**memset(cBuf, 0, MAX\_MSG\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**CChatPacket \*pPacket = new CChatPacket(); //新建CChatPacket对象**

**pPacket->m\_type = CChatPacket::MESSAGE;**

**pPacket->m\_UserInfo.m\_strName = cBuf; //发送消息的用户名**

**//读取接收消息的用户名称**

**strSectionKey = "RecvMsgUser";**

**memset(cBuf, 0, MAX\_MSG\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**pPacket->m\_OfflineUserInfo.m\_strName = cBuf;**

**//读取发送消息时间**

**strSectionKey = "Time";**

**memset(cBuf, 0, MAX\_MSG\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**time\_t time = atol(cBuf);**

**pPacket->m\_time = time;**

**//读取发送消息**

**strSectionKey = "Message";**

**memset(cBuf, 0, MAX\_MSG\_SIZE);**

**if(0 == GetPrivateProfileString (strSection, strSectionKey, NULL, cBuf, TEMP\_BUF\_SIZE, filePath))**

**{**

**break;**

**}**

**////取代$为回车符**

**CString strMsg(cBuf);**

**strMsg.Replace( "$","\r\n");**

**pPacket->m\_strMsg = strMsg;**

**obList.AddTail(pPacket);//加入链表**

**nIndex++;**

**}**

**m\_csOfflineList.Unlock();**

**}**

**/\***

**\* 处理数据**

**\*/**

**BOOL CServerDlg::ProcessPendingRead(CArchive \*pArchiveIn, CArchive \*pArchiveOut, CClientSocket \*pClientSocket)**

**{**

**do**

**{**

**CChatPacket chatPact;//接收数据包**

**chatPact.Serialize(\*pArchiveIn);**

**if (CChatPacket::MESSAGE == chatPact.m\_type)//发给离线用户的消息**

**{**

**m\_csOfflineList.Lock();**

**//保存离线用户消息**

**CChatPacket \*pChatPacket = new CChatPacket();**

**//包类型**

**pChatPacket->m\_type = CChatPacket::MESSAGE;**

**//发送消息用户**

**pChatPacket->m\_UserInfo.m\_strName = chatPact.m\_UserInfo.m\_strName;**

**//接收消息用户**

**pChatPacket->m\_OfflineUserInfo.m\_strName = chatPact.m\_OfflineUserInfo.m\_strName;**

**//发送的消息**

**pChatPacket->m\_strMsg = chatPact.m\_strMsg;**

**//发包时间**

**pChatPacket->m\_time = chatPact.m\_time;**

**//加入离线消息链表**

**m\_OfflineMsgList.AddTail(pChatPacket);**

**m\_csOfflineList.Unlock();**

**}else if (CChatPacket::USERLIST == chatPact.m\_type)//请求用户列表--用户刚登录**

**{**

**//保存用户信息**

**pClientSocket->SaveUserInfo(chatPact.m\_UserInfo);**

**//户更新信息**

**if (!UpdateUserList(chatPact.m\_UserInfo, /\*pArchiveOut,\*/ pClientSocket))**

**return FALSE;**

**//向所有的在线用户发送用户列表**

**SendUserList();**

**//转发离线消息**

**TransmitMsg(chatPact, /\*pArchiveOut\*/pClientSocket);**

**//更新服务器界面**

**UpdateServerListCtl(m\_UserList);**

**}**

**} while(!pArchiveIn->IsBufferEmpty());**

**return TRUE;**

**}**

**/\***

**\* 发送用户列表**

**\*/**

**void CServerDlg::SendUserList()**

**{**

**//复制用户链表**

**CObList tempChatterlist;**

**CopyUserList(tempChatterlist);**

**CChatPacket packet;**

**packet.m\_type = CChatPacket::USERLIST; //包的类型**

**packet.m\_pUserList = &tempChatterlist; //用户链表**

**m\_csChatterList.Lock();//上锁**

**//向所有在线用户发送列表**

**POSITION pos = NULL;**

**for (pos = m\_ChatterList.GetHeadPosition(); NULL!=pos;)**

**{**

**CClientSocket \*pClientSocket = (CClientSocket\*)m\_ChatterList.GetNext(pos);**

**pClientSocket->SendUserList(&packet);//发送数据包**

**}**

**m\_csChatterList.Unlock();//解锁**

**DeleteTempUserList(tempChatterlist);//删除临时链表**

**}**

**/\***

**\* 复制用户链表**

**\*/**

**void CServerDlg::CopyUserList(CObList &obList)**

**{**

**if (m\_UserList.IsEmpty())**

**{**

**return ;**

**}**

**POSITION pos = NULL;**

**CUserInfo \*pUserInfo = NULL;**

**CUserInfo \*pTempUserInfo = NULL;**

**for (pos = m\_UserList.GetHeadPosition(); pos != NULL;)**

**{**

**pUserInfo = (CUserInfo\*)m\_UserList.GetNext(pos);**

**//不要复制用户的密码**

**pTempUserInfo = new CUserInfo();**

**pTempUserInfo->m\_strName = pUserInfo->m\_strName; //名称**

**pTempUserInfo->m\_eStatus = pUserInfo->m\_eStatus; //状态**

**pTempUserInfo->m\_lIP = pUserInfo->m\_lIP; //IP**

**pTempUserInfo->m\_nPort = pUserInfo->m\_nPort; //端口**

**obList.AddTail(pTempUserInfo);**

**}**

**}**

**/\***

**\* 删除临时链表**

**\*/**

**void CServerDlg::DeleteTempUserList(CObList &obList)**

**{**

**if (obList.IsEmpty())**

**{**

**return;**

**}**

**POSITION pos = NULL;**

**CUserInfo \*pUserInfo = NULL;**

**for (pos = obList.GetHeadPosition(); NULL != pos;)**

**{**

**pUserInfo = (CUserInfo\*)obList.GetNext(pos);**

**delete pUserInfo;**

**pUserInfo = NULL;**

**}**

**obList.RemoveAll();**

**}**

**/\***

**\* 更新用户链表**

**\*/**

**BOOL CServerDlg::UpdateUserList(const CUserInfo &userInfo, CClientSocket \*pClientSocket)**

**{**

**BOOL retVal = TRUE; //返回值**

**m\_csUserList.Lock(); //上锁**

**POSITION pos; //位置变量**

**CUserInfo \*pUserInfo = NULL; //用户对象指针**

**BOOL bFind = FALSE; //是否找到该用户**

**//该用户是否存在于链表中**

**for (pos = m\_UserList.GetHeadPosition(); NULL != pos;)**

**{**

**pUserInfo = (CUserInfo\*)m\_UserList.GetNext(pos);//节点下移**

**if (pUserInfo->m\_strName == userInfo.m\_strName )//名字相同**

**{**

**if (pUserInfo->m\_strPassword == userInfo.m\_strPassword) //在册用户**

**{**

**if(CUserInfo::ONLINE == pUserInfo->m\_eStatus) //用户已经登录**

**{**

**//从在线链表中删除该节点**

**DeleteChatter(pClientSocket);**

**//发送“用户已经登录”消息**

**CChatPacket packet;**

**packet.m\_type = CChatPacket::SERVERMSG; //数据包类型**

**packet.m\_strMsg = \_T(" 用户已经登录! ");//消息**

**packet.m\_time = CTime::GetCurrentTime(); //时间**

**pClientSocket->SendUserMsg(&packet); //发送数据**

**m\_csUserList.Unlock(); //解锁**

**return FALSE;**

**}else//用户登录**

**{**

**pUserInfo->m\_eStatus = CUserInfo::ONLINE; //修改用户状态**

**pUserInfo->m\_lIP = userInfo.m\_lIP; //IP地址**

**pUserInfo->m\_nPort = userInfo.m\_nPort; //端口**

**bFind = TRUE;**

**break;**

**}**

**}else//密码错误**

**{**

**//从在线链表中删除该节点**

**DeleteChatter(pClientSocket);**

**//发送“密码错误”消息**

**CChatPacket packet;**

**packet.m\_type = CChatPacket::SERVERMSG; //数据包类型**

**packet.m\_strMsg = \_T(" 密码错误\t\n 请重新登录! "); //消息**

**packet.m\_time = CTime::GetCurrentTime(); //时间**

**pClientSocket->SendUserMsg(&packet);//发送数据**

**m\_csUserList.Unlock();//解锁**

**return FALSE;**

**}**

**}**

**}**

**if (FALSE == bFind)//注册新用户**

**{**

**CUserInfo \*pUserInfo = new CUserInfo();**

**pUserInfo->m\_strName = userInfo.m\_strName; //姓名**

**pUserInfo->m\_strPassword = userInfo.m\_strPassword; //密码**

**pUserInfo->m\_eStatus = CUserInfo::ONLINE; //状态**

**pUserInfo->m\_lIP = userInfo.m\_lIP; //地址**

**pUserInfo->m\_nPort = userInfo.m\_nPort; //端口**

**pUserInfo->m\_time = userInfo.m\_time; //时间**

**m\_UserList.AddTail(pUserInfo); //加入链表**

**}**

**m\_csUserList.Unlock();//解锁**

**return retVal;**

**}**

**/\***

**\* 转发离线消息**

**\*/**

**void CServerDlg::TransmitMsg(const CChatPacket &packet, CClientSocket \*pClientSocket)**

**{**

**m\_csOfflineList.Lock();**

**POSITION pos1 = NULL; //位置变量**

**POSITION pos2 = NULL; //位置变量**

**CChatPacket \*pPacket = NULL;//数据包**

**//遍历整个离线消息链表**

**for (pos1 = m\_OfflineMsgList.GetHeadPosition(); (pos2 = pos1) != NULL;)**

**{**

**//取出数据包**

**pPacket = (CChatPacket\*)m\_OfflineMsgList.GetNext(pos1);**

**//登录用户的名字和接收离线消息的用户名字相同**

**if (pPacket->m\_OfflineUserInfo.m\_strName == packet.m\_UserInfo.m\_strName)**

**{**

**pClientSocket->SendUserMsg(pPacket);//发送离线消息**

**m\_OfflineMsgList.RemoveAt(pos2); //删除链表节点**

**delete pPacket; //删除对象**

**pPacket = NULL;**

**}**

**}**

**m\_csOfflineList.Unlock();**

**}**

**/\***

**\* 更新服务器界面**

**\*/**

**void CServerDlg::UpdateServerListCtl(const CObList &obList)**

**{**

**m\_csUserList.Lock();//上锁**

**//删除原来数据项**

**m\_ctlUserList.DeleteAllItems();**

**POSITION pos = NULL; //位置变量**

**int nIndex = 0; //序号**

**for (pos = m\_UserList.GetHeadPosition(); NULL != pos;)**

**{**

**//取出用户信息**

**CUserInfo \*pUserInfo = (CUserInfo\*)m\_UserList.GetNext(pos);**

**if (CUserInfo::ONLINE == pUserInfo->m\_eStatus)//在线状态**

**{**

**//插入数据项**

**m\_ctlUserList.InsertItem(nIndex, pUserInfo->m\_strName,CUserInfo::ONLINE);**

**}else//离线状态**

**{**

**//插入数据项**

**m\_ctlUserList.InsertItem(nIndex, pUserInfo->m\_strName,CUserInfo::OFFLINE);**

**}**

**m\_ctlUserList.SetItemText(nIndex,1,pUserInfo->m\_strPassword); //用户密码**

**in\_addr userAddr;**

**userAddr.S\_un.S\_addr = (long)pUserInfo->m\_lIP;**

**CString strUserAddr(inet\_ntoa(userAddr)); //用户IP**

**m\_ctlUserList.SetItemText(nIndex,2,strUserAddr);**

**CString strUserPort; //用户端口**

**strUserPort.Format("%d",ntohs(pUserInfo->m\_nPort));**

**m\_ctlUserList.SetItemText(nIndex,3,strUserPort);**

**CString strUserTime = pUserInfo->m\_time.Format("%c"); //登录时间**

**m\_ctlUserList.SetItemText(nIndex,4,strUserTime);**

**nIndex++;**

**}**

**m\_csUserList.Unlock();//解锁**

**}**

**/\***

**\* 当用户离线时删除用户**

**\*/**

**void CServerDlg::ReleaseChatter(CClientSocket \*pClientSocket)**

**{**

**m\_csChatterList.Lock();**

**POSITION pos = NULL;**

**//找到该用户**

**if ((pos = m\_ChatterList.Find(pClientSocket)) != NULL)**

**{**

**ASSERT((CClientSocket\*)m\_ChatterList.GetAt(pos) == pClientSocket);**

**//获取该用户信息**

**CUserInfo userInfo = pClientSocket->GetUserInfo();**

**//更新该用户信息**

**POSITION posUser = NULL;**

**for (posUser = m\_UserList.GetHeadPosition(); posUser != NULL;)**

**{**

**CUserInfo \*pUserInfo = (CUserInfo\*)m\_UserList.GetNext(posUser);**

**if (pUserInfo->m\_strName == userInfo.m\_strName)**

**{**

**pUserInfo->m\_eStatus = CUserInfo::OFFLINE; //修改状态为离线**

**pUserInfo->m\_lIP = userInfo.m\_lIP; //修改用户地址**

**pUserInfo->m\_nPort = userInfo.m\_nPort; //修改用户端口**

**pUserInfo->m\_time = CTime::GetCurrentTime();//设置离线时间**

**break;**

**}**

**}**

**//删除该在线用户**

**m\_ChatterList.RemoveAt(pos);**

**//通知所有在线用户更新用户列表**

**SendUserList();**

**//更新服务器界面**

**UpdateServerListCtl(m\_UserList);**

**}**

**m\_csChatterList.Unlock();**

**}**

**/\***

**\* 登录失败处理**

**\*/**

**void CServerDlg::DeleteChatter(CClientSocket \*pClientSocket)**

**{**

**m\_csChatterList.Lock();**

**POSITION pos = NULL;**

**if ((pos = m\_ChatterList.Find(pClientSocket)) != NULL)**

**{**

**ASSERT((CClientSocket\*)m\_ChatterList.GetAt(pos) == pClientSocket);**

**//删除该在线用户**

**m\_ChatterList.RemoveAt(pos);**

**}**

**m\_csChatterList.Unlock();**

**}**

**/\***

**\* WM\_DESTROY消息相应函数**

**\*/**

**void CServerDlg::OnDestroy()**

**{**

**SaveUserList(m\_UserList); //保存用户信息**

**SaveOfflineMsg(m\_OfflineMsgList); //保存离线信息**

**DeleteAllChatter(); //删除在线用户**

**if (NULL != m\_pListenSocket) //删除CListenSocket类对象**

**{**

**delete m\_pListenSocket;**

**m\_pListenSocket = NULL;**

**}**

**if (NULL != m\_pImageList) //删除CImageList类对象**

**{**

**delete m\_pImageList;**

**m\_pImageList = NULL;**

**}**

**}**

**/\***

**\* 右击列表控件响应函数**

**\*/**

**void CServerDlg::OnRclickClientList(NMHDR\* pNMHDR, LRESULT\* pResult)**

**{**

**NM\_LISTVIEW\* pNMListView = (NM\_LISTVIEW\*)pNMHDR;**

**if(pNMListView->iItem != -1)**

**{**

**DWORD dwPos = GetMessagePos(); //鼠标获取位置**

**CPoint point( LOWORD(dwPos), HIWORD(dwPos) );**

**CMenu menu;**

**VERIFY( menu.LoadMenu(IDR\_SERVER\_LIST\_MENU)); //装载菜单资源**

**CMenu\* popup = menu.GetSubMenu(0); //获取菜单项**

**ASSERT( popup != NULL );**

**//弹出快捷菜单**

**popup->TrackPopupMenu(TPM\_LEFTALIGN | TPM\_RIGHTBUTTON, point.x, point.y, this );**

**}**

**\*pResult = 0;**

**}**

**/\***

**\* 处理菜单消息**

**\*/**

**void CServerDlg::OnServerListDeleteMenuitem()**

**{**

**//获得被选中数据项的位置**

**POSITION pos = m\_ctlUserList.GetFirstSelectedItemPosition();**

**int nSelectedItem = m\_ctlUserList.GetNextSelectedItem(pos);**

**//获取用户名称**

**CString strPeerName = m\_ctlUserList.GetItemText(nSelectedItem, 0);**

**POSITION pos1 = NULL; //位置变量**

**POSITION pos2 = NULL; //位置变量**

**BOOL bDeleteUser = FALSE; //是否找到该用户**

**//在用户链表中查找**

**for (pos1 = m\_UserList.GetHeadPosition();(pos2=pos1)!= NULL;)**

**{**

**//取出数据**

**CUserInfo\* pUserInfo = (CUserInfo\*)m\_UserList.GetNext(pos1);**

**if (pUserInfo->m\_strName == strPeerName) //名称相同**

**{**

**if (pUserInfo->m\_eStatus == CUserInfo::ONLINE) //不能删除在线用户**

**{**

**AfxMessageBox("不能删除在线用户！",MB\_OK, -1);**

**return ;**

**}else//删除不在线用户**

**{**

**CUserInfo \*pUserInfo = (CUserInfo\*)m\_UserList.GetAt(pos2);**

**m\_UserList.RemoveAt(pos2); //删除指针**

**delete pUserInfo; //删除对象**

**pUserInfo = NULL;**

**bDeleteUser = TRUE;**

**//更新服务器界面**

**UpdateServerListCtl(m\_UserList);**

**}**

**}**

**}**

**if (TRUE == bDeleteUser)**

**{**

**//通知客户端更新成员列表**

**SendUserList();**

**}**

**}**

**/\***

**\* 删除所有在线用户**

**\*/**

**void CServerDlg::DeleteAllChatter( void )**

**{**

**POSITION pos;**

**CClientSocket \*pClient = NULL;**

**for (pos = m\_ChatterList.GetHeadPosition(); NULL != pos; )**

**{**

**pClient = (CClientSocket\*)m\_ChatterList.GetNext(pos);**

**delete pClient;**

**pClient = NULL;**

**}**

**m\_ChatterList.RemoveAll();**

**}**

**UserInfo.cpp**

// UserInfo.cpp: implementation of the CUserInfo class.

//

//////////////////////////////////////////////////////////////////////

#include "stdafx.h"

#include "UserInfo.h"

#ifdef \_DEBUG

#undef THIS\_FILE

static char THIS\_FILE[]=\_\_FILE\_\_;

#define new DEBUG\_NEW

#endif

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

//IMPLEMENT\_DYNCREATE(CUserInfo, CObject)

IMPLEMENT\_SERIAL(CUserInfo, CObject, 1)//new一个对象然后调用默认构造函数->init

CUserInfo::CUserInfo()

{

Init();

}

CUserInfo::~CUserInfo()

{

}

/\*

\* 初始化

\*/

void CUserInfo::Init(void)

{

m\_strName = \_T("");

m\_strPassword = \_T("");

m\_eStatus = UNKNOWN;

m\_lIP = 0;

m\_nPort = 0;

m\_time = CTime::GetCurrentTime();

}

/\*

\* 序列化

\*/

void CUserInfo::Serialize(CArchive& ar)

{

CObject::Serialize(ar);//调用基类的序列化函数

if (ar.IsStoring())//发送数据

{

ar << m\_strName; //名字

ar << m\_strPassword; //密码

WORD byStatus = 0;

byStatus = (WORD)m\_eStatus;

ar << byStatus; //状态

ar << m\_lIP; //地址

ar << m\_nPort; //端口

long lTime = m\_time.GetTime(); //日期和时间

ar << lTime;

}else

{

ar >> m\_strName; //名字

ar >> m\_strPassword; //密码

WORD byStatus = 0;

ar >> byStatus;

m\_eStatus = (USERSTATUE)byStatus;//状态

ar >> m\_lIP; //地址

ar >> m\_nPort; //端口

long lTime;

ar >> lTime; //日期和时间

m\_time = CTime((time\_t)lTime);

}

}

/\*

\* 赋值构造函数

\*/

CUserInfo &CUserInfo::operator = (const CUserInfo &userInfo)

{

m\_strName = userInfo.m\_strName;

m\_strPassword = userInfo.m\_strPassword;

m\_eStatus = userInfo.m\_eStatus;

m\_lIP = userInfo.m\_lIP;

m\_nPort = userInfo.m\_nPort;

m\_time = userInfo.m\_time;

return \*this;

}

/\*

\* 赋值构造函数

\*/

CUserInfo::CUserInfo(const CUserInfo& userInfo)

{

m\_strName = userInfo.m\_strName;

m\_strPassword = userInfo.m\_strPassword;

m\_eStatus = userInfo.m\_eStatus;

m\_lIP = userInfo.m\_lIP;

m\_nPort = userInfo.m\_nPort;

m\_time = userInfo.m\_time;

}

# Client

**ChatDlg.cpp**

// ChatDlg.cpp : implementation file

//

#include "stdafx.h"

#include "resource.h"

#include "Client.h"

#include "ChatDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CChatDlg dialog

CChatDlg::CChatDlg(CWnd\* pParent /\*=NULL\*/)

: CDialog(CChatDlg::IDD, pParent),m\_strRec("")

{

//{{AFX\_DATA\_INIT(CChatDlg)

// NOTE: the ClassWizard will add member initialization here

//}}AFX\_DATA\_INIT

m\_pSocket = NULL;

m\_pFile = NULL;

m\_pArchiveIn = NULL;

m\_pArchiveOut = NULL;

m\_bOfflineWnd = FALSE;

m\_pOfflinePacket = NULL;

}

/\*

\* 在线/离线窗口

\*/

CChatDlg::CChatDlg(CWnd\* pParent, const CUserInfo &peerInfo, const CUserInfo &userInfor)

:CDialog(CChatDlg::IDD, pParent),m\_strRec("")

{

m\_PeerInfo = peerInfo; //对方用户信息

m\_UserInfo = userInfor; //用户信息

m\_pSocket = NULL; //CChatSocket指针

m\_pFile = NULL; //文件对象

m\_pArchiveIn = NULL; //读入文档对象

m\_pArchiveOut = NULL; //写入文档对象

m\_pOfflinePacket = NULL;//离线数据包

m\_bOfflineWnd = FALSE; //接收离线窗口

}

/\*

\* 显示离线消息窗口

\*/

CChatDlg::CChatDlg(CWnd\* pParent, CChatPacket \*pPacket)

:CDialog(CChatDlg::IDD, pParent),m\_strRec("")

{

m\_pOfflinePacket = pPacket; //离线数据包

m\_bOfflineWnd = TRUE; //接收离线窗口

m\_pSocket = NULL; //CChatSocket指针

m\_pFile = NULL; //文件对象

m\_pArchiveIn = NULL; //读入文档对象

m\_pArchiveOut = NULL; //写入文档对象

}

void CChatDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

//{{AFX\_DATA\_MAP(CChatDlg)

DDX\_Control(pDX, IDC\_CLIENT\_CHAT\_OUTPUT\_EDIT, m\_ctlOutput);

DDX\_Control(pDX, IDC\_CLIENT\_CHAT\_IPUT\_EDIT, m\_ctlInput);

//}}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP(CChatDlg, CDialog)

//{{AFX\_MSG\_MAP(CChatDlg)

ON\_WM\_DESTROY()

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CChatDlg message handlers

/\*

\* 接收消息

\*/

void CChatDlg::ProcessPendingRead(void)

{

do

{

CChatPacket chatPact;

chatPact.Serialize(\*m\_pArchiveIn);

if (CChatPacket::MESSAGE == chatPact.m\_type)//消息

{

DisplayRecvMessage(&chatPact);

}

} while(!m\_pArchiveIn->IsBufferEmpty());

}

/\*

\* WM\_INITDIALOG消息响应函数

\*/

BOOL CChatDlg::OnInitDialog()

{

//调用基类函数

CDialog::OnInitDialog();

if (NULL != m\_pOfflinePacket && TRUE == m\_bOfflineWnd) //显示离线消息

{

DisplayOfflineMessage(m\_pOfflinePacket);

}else if (m\_PeerInfo.m\_eStatus == CUserInfo::ONLINE) //在线聊天

{

SetWindowText(\_T("与") + m\_PeerInfo.m\_strName + \_T("聊天中"));

}else //发送离线消息

{

SetWindowText(m\_PeerInfo.m\_strName + \_T("不在线哦"));

}

return TRUE;

}

/\*

\* 连接对方

\*/

BOOL CChatDlg::ConnectToPeer(void)

{

//创建CChatSocket对象

if(!InitSocket())

{

return FALSE;

}

//创建套接字，绑定

if (!m\_pSocket->Create())

{

delete m\_pSocket;

m\_pSocket = NULL;

AfxMessageBox(\_T("创建套接字失败！"));

return FALSE;

}

m\_pFile = new CSocketFile(m\_pSocket); //创建文件对象

m\_pArchiveIn = new CArchive(m\_pFile,CArchive::load);//创建读入文档对象

//连接对方

SOCKADDR\_IN peerAddr;

peerAddr.sin\_family = AF\_INET;

peerAddr.sin\_addr.S\_un.S\_addr = m\_PeerInfo.m\_lIP;

peerAddr.sin\_port = m\_PeerInfo.m\_nPort;

while (!m\_pSocket->Connect((SOCKADDR\*)&peerAddr, sizeof(SOCKADDR\_IN)))

{

if (AfxMessageBox(\_T("连接对方失败，是否再次尝试连接。"),MB\_YESNO) == IDNO)

{

return FALSE;

}

}

return TRUE;

}

BOOL CChatDlg::InitSocket(void)

{

//创建套接字

m\_pSocket = new CChatSocket(this);

return TRUE;

}

/\*

\* 显示主动聊天接收到的消息

\*/

void CChatDlg::DisplayRecvMessage(CChatPacket \*pPacket)

{

CString chatHeader = pPacket->m\_UserInfo.m\_strName;//发送消息的用户名称

//格式化日期和时间

CTime timeChatting(pPacket->m\_time);

int year = timeChatting.GetYear();

int month = timeChatting.GetMonth();

int day = timeChatting.GetDay();

int hour = timeChatting.GetHour();

int minute = timeChatting.GetMinute();

int second = timeChatting.GetSecond();

CString chatTime;

chatTime.Format(" (%d-%d-%d %d:%d:%d)", year, month, day, hour, minute, second);

//显示消息头

chatHeader += chatTime;

CString strItem = chatHeader + "\r\n"+ pPacket->m\_strMsg + "\r\n";

//显示消息

int len = m\_ctlOutput.GetWindowTextLength();

m\_ctlOutput.SetSel(len,len);

m\_ctlOutput.ReplaceSel(strItem);

}

/\*

\* 处理键盘消息

\*/

BOOL CChatDlg::PreTranslateMessage(MSG\* pMsg)

{

if(pMsg->hwnd == m\_ctlInput.GetSafeHwnd()) //输入消息窗口

{

if (pMsg->wParam == VK\_RETURN) //return键

{

if (TRUE == m\_bOfflineWnd ) //显示离线消息窗口

{

return TRUE;

}else //在线/离线聊天窗口

{

SendPeerMessage(); //发送消息

m\_ctlInput.SetWindowText(""); //清空输入消息窗口内容

}

return TRUE;

}

}

return CDialog::PreTranslateMessage(pMsg); //其他消息

}

/\*

\* 发送消息

\*/

void CChatDlg::SendPeerMessage( void )

{

//输入消息窗口内容为空

if (0 == m\_ctlInput.GetWindowTextLength())

{

return;

}

//对方在线

if(m\_PeerInfo.m\_eStatus == CUserInfo::ONLINE) //发送在线消息

{

CChatPacket packet;

packet.m\_UserInfo = this->m\_UserInfo; //发送消息用户名称

packet.m\_type = CChatPacket::MESSAGE; //包的类型

m\_ctlInput.GetWindowText(packet.m\_strMsg); //消息

packet.m\_time = CTime::GetCurrentTime(); //发包时间

//发送消息

m\_pArchiveOut = new CArchive(m\_pFile,CArchive::store);//创建写入文档对象

packet.Serialize(\*m\_pArchiveOut);

m\_pArchiveOut->Flush();

delete m\_pArchiveOut;

m\_pArchiveOut = NULL;

//显示发送的消息

DisplaySentMessage(&packet);

}else//对方离线

{

//显示发送的离线消息

DisplaySentMessage();

}

}

/\*

\* 显示发送的消息

\*/

void CChatDlg::DisplaySentMessage(CChatPacket \*pPacket)

{

CString chatHeader = pPacket->m\_UserInfo.m\_strName; //发送消息用户名称

//格式化发送消息的时间

CTime timeChatting = pPacket->m\_time;

int year = timeChatting.GetYear(); //年

int month = timeChatting.GetMonth(); //月

int day = timeChatting.GetDay(); //日

int hour = timeChatting.GetHour(); //时

int minute = timeChatting.GetMinute(); //分

int second = timeChatting.GetSecond(); //秒

CString chatTime;

chatTime.Format(" (%d-%d-%d %d:%d:%d)", year, month, day, hour, minute, second);

chatHeader += chatTime;//消息头

//获取输入消息

CString strMsg;

m\_ctlInput.GetWindowText(strMsg);

//消息头与消息分行显示

CString strItem = chatHeader + "\r\n"+ pPacket->m\_strMsg + "\r\n";

//在当前消息行下面显示消息

int len = m\_ctlOutput.GetWindowTextLength();//当前文本长度

m\_ctlOutput.SetSel(len,len); //移动插入符到当前文本最后

m\_ctlOutput.ReplaceSel(strItem); //在当前文本后插入消息

}

/\*

\* 显示发送的消息

\*/

void CChatDlg::DisplaySentMessage(void)

{

CString chatHeader = m\_UserInfo.m\_strName;

CTime timeChatting = CTime::GetCurrentTime();

int year = timeChatting.GetYear(); //年

int month = timeChatting.GetMonth(); //月

int day = timeChatting.GetDay(); //日

int hour = timeChatting.GetHour(); //时

int minute = timeChatting.GetMinute(); //分

int second = timeChatting.GetSecond(); //秒

CString chatTime;

chatTime.Format(" (%d-%d-%d %d:%d:%d)", year, month, day, hour, minute, second);

chatHeader += chatTime;//消息头

CString strMsg;

m\_ctlInput.GetWindowText(strMsg);

CString strItem = chatHeader + "\r\n"+ strMsg + "\r\n";

//显示消息

int len = m\_ctlOutput.GetWindowTextLength();

m\_ctlOutput.SetSel(len,len);

m\_ctlOutput.ReplaceSel(strItem);

}

void CChatDlg::InitArchive(void)

{

//初始化输入和输出流

m\_pFile = new CSocketFile(m\_pSocket); //初始化CSocketFile

m\_pArchiveIn = new CArchive(m\_pFile,CArchive::load); //初始化读入流

}

void CChatDlg::SetPeerInfor( CUserInfo &peerInfo )

{

m\_PeerInfo = peerInfo;

}

void CChatDlg::SetUserInfor( CUserInfo &userInfo )

{

m\_UserInfo = userInfo;

}

void CChatDlg::OnDestroy()

{

m\_ctlOutput.GetWindowText(m\_strRec);//聊天记录

CDialog::OnDestroy();

}

CChatDlg::~CChatDlg()

{

if(FALSE == m\_bOfflineWnd)

{

if (NULL != m\_pArchiveIn)

{

delete m\_pArchiveIn;

m\_pArchiveIn = NULL;

}

if (NULL != m\_pFile)

{

delete m\_pFile;

m\_pFile = NULL;

}

if (NULL != m\_pSocket)

{

delete m\_pSocket;

m\_pSocket = NULL;

}

}

}

CChatSocket \*CChatDlg::GetChatSocket( void )

{

return m\_pSocket;

}

/\*

\* 显示离线消息

\*/

void CChatDlg::DisplayOfflineMessage( CChatPacket \*pPacket )

{

CRect inputWndRect;

m\_ctlInput.GetWindowRect(&inputWndRect);//获取输入消息窗口区域

m\_ctlInput.MoveWindow(0, 0, 0, 0); //设置该窗口大小为0

CRect chatDlgRect;

GetClientRect(&chatDlgRect); //获取对话框客户区大小

MoveWindow(0, 0, chatDlgRect.Width(), //修改对话框的高度

chatDlgRect.Height() - inputWndRect.Height());

GetClientRect(&chatDlgRect); //获取对话框客户区大小

m\_ctlOutput.MoveWindow(5, 5, //修改显示消息窗口高度

chatDlgRect.Width() -10, chatDlgRect.Height() -10);

m\_PeerInfo = pPacket->m\_UserInfo; //发送消息用户名称

SetWindowText(m\_PeerInfo.m\_strName + \_T("发送给您的离线消息")); //设置标题

m\_ctlOutput.SetWindowText(pPacket->m\_strMsg ); //设置消息

}

**ChatPacket.cpp**

// ChatPacket.cpp: implementation of the CChatPacket class.

//

//////////////////////////////////////////////////////////////////////

#include "stdafx.h"

#ifdef \_DEBUG

#undef THIS\_FILE

static char THIS\_FILE[]=\_\_FILE\_\_;

#define new DEBUG\_NEW

#endif

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

//IMPLEMENT\_DYNCREATE(CChatPacket, CObject)

IMPLEMENT\_SERIAL(CChatPacket, CObject, 1 )

CChatPacket::CChatPacket()

{

Init();

}

CChatPacket::~CChatPacket()

{

}

/\*

\* 初始化

\*/

void CChatPacket::Init(void)

{

m\_type = UNKNOWN; //为知类型

m\_strMsg = \_T(""); //清空

m\_pUserList = NULL; //清空

m\_time = CTime::GetCurrentTime();

}

/\*

\* 序列化

\*/

void CChatPacket::Serialize(CArchive& ar)

{

CObject::Serialize(ar);//调用基类的序列化函数

if (ar.IsStoring())//发送数据

{

BYTE byType = m\_type;

ar << byType; //包的类型

ar << m\_strMsg; //消息

long lTime = m\_time.GetTime(); //日期和时间

ar << lTime;

}else//接收数据

{

BYTE byType;

ar >> byType;

m\_type = (PACKETTYPE)byType; //包的类型

ar >> m\_strMsg; //消息

long lTime;

ar >> lTime; //日期和时间

m\_time = CTime((time\_t)lTime);

}

m\_UserInfo.Serialize(ar); //序列化用户信息

m\_OfflineUserInfo.Serialize(ar); //序列化离线用户信息

if (m\_type == USERLIST && NULL != m\_pUserList)//序列化用户列表

{

m\_pUserList->Serialize(ar);

}

}

**ChatSocket.cpp**

// ChatSocket.cpp : implementation file

//

#include "stdafx.h"

#include "Client.h"

#include "ChatSocket.h"

#include "ClientDlg.h"

#include "ChatDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CChatSocket

CChatSocket::CChatSocket(CClientDlg \*pClientDlg)

{

m\_pClientDlg = pClientDlg;

m\_pChatDlg = NULL;

}

CChatSocket::CChatSocket(CChatDlg \*pChatDlg )

{

m\_pChatDlg = pChatDlg;

m\_pClientDlg = NULL;

}

CChatSocket::~CChatSocket()

{

m\_pClientDlg = NULL;

m\_pChatDlg = NULL;

}

// Do not edit the following lines, which are needed by ClassWizard.

#if 0

BEGIN\_MESSAGE\_MAP(CChatSocket, CSocket)

//{{AFX\_MSG\_MAP(CChatSocket)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

#endif // 0

/////////////////////////////////////////////////////////////////////////////

// CChatSocket member functions

void CChatSocket::OnReceive(int nErrorCode)

{

if (NULL != m\_pClientDlg)

{

m\_pClientDlg->ProcessPendingRead();

}else if (NULL != m\_pChatDlg)

{

m\_pChatDlg->ProcessPendingRead();

}

}

**Client.cpp**

// Client.cpp : Defines the class behaviors for the application.

//

#include "stdafx.h"

#include "Client.h"

#include "ClientDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CClientApp

BEGIN\_MESSAGE\_MAP(CClientApp, CWinApp)

//{{AFX\_MSG\_MAP(CClientApp)

// NOTE - the ClassWizard will add and remove mapping macros here.

// DO NOT EDIT what you see in these blocks of generated code!

//}}AFX\_MSG

ON\_COMMAND(ID\_HELP, CWinApp::OnHelp)

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CClientApp construction

CClientApp::CClientApp()

{

// TODO: add construction code here,

// Place all significant initialization in InitInstance

}

/////////////////////////////////////////////////////////////////////////////

// The one and only CClientApp object

CClientApp theApp;

/////////////////////////////////////////////////////////////////////////////

// CClientApp initialization

BOOL CClientApp::InitInstance()

{

if (!AfxSocketInit())

{

AfxMessageBox(\_T("初始化套接字失败！"));

return FALSE;

}

AfxEnableControlContainer();

// Standard initialization

// If you are not using these features and wish to reduce the size

// of your final executable, you should remove from the following

// the specific initialization routines you do not need.

#ifdef \_AFXDLL

Enable3dControls(); // Call this when using MFC in a shared DLL

#else

Enable3dControlsStatic(); // Call this when linking to MFC statically

#endif

CClientDlg dlg;

m\_pMainWnd = &dlg;

int nResponse = dlg.DoModal();

if (nResponse == IDOK)

{

// TODO: Place code here to handle when the dialog is

// dismissed with OK

}

else if (nResponse == IDCANCEL)

{

// TODO: Place code here to handle when the dialog is

// dismissed with Cancel

}

// Since the dialog has been closed, return FALSE so that we exit the

// application, rather than start the application's message pump.

return FALSE;

}

**ClientDlg.cpp**

// ClientDlg.cpp : implementation file

//

#include "stdafx.h"

#include "Client.h"

#include "ClientDlg.h"

#include "ClientLogin.h"

#include "ChatDlg.h"

#include "ChatSocket.h"

#include "ListenSocket.h"

#include "PeerSocket.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CClientDlg dialog

CClientDlg::CClientDlg(CWnd\* pParent /\*=NULL\*/)

: CDialog(CClientDlg::IDD, pParent)

{

//{{AFX\_DATA\_INIT(CClientDlg)

// NOTE: the ClassWizard will add member initialization here

//}}AFX\_DATA\_INIT

// Note that LoadIcon does not require a subsequent DestroyIcon in Win32

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

m\_pImageList = NULL;

m\_pArchiveIn = NULL;

m\_pArchiveOut = NULL;

m\_pFile = NULL;

m\_pSocket = NULL;

memset(&m\_hostAddr, 0, sizeof(SOCKADDR\_IN));

}

CClientDlg::~CClientDlg()

{

if (NULL != m\_pImageList)

{

delete m\_pImageList;

m\_pImageList = NULL;

}

if (NULL != m\_pArchiveIn)

{

delete m\_pArchiveIn;

m\_pArchiveIn = NULL;

}

if (NULL != m\_pArchiveOut)

{

delete m\_pArchiveOut;

m\_pArchiveOut = NULL;

}

if (NULL != m\_pFile)

{

delete m\_pFile;

m\_pFile = NULL;

}

if (m\_pSocket != NULL)

{

BYTE Buffer[50];

m\_pSocket->ShutDown();

while(m\_pSocket->Receive(Buffer,50) > 0);//????

delete m\_pSocket;

m\_pSocket = NULL;

}

}

void CClientDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

//{{AFX\_DATA\_MAP(CClientDlg)

DDX\_Control(pDX, IDC\_CLIENT\_LIST, m\_ctlUserList);

//}}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP(CClientDlg, CDialog)

//{{AFX\_MSG\_MAP(CClientDlg)

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_WM\_DESTROY()

ON\_NOTIFY(NM\_DBLCLK, IDC\_CLIENT\_LIST, OnDblclkClientList)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CClientDlg message handlers

BOOL CClientDlg::OnInitDialog()

{

CDialog::OnInitDialog();

// Set the icon for this dialog. The framework does this automatically

// when the application's main window is not a dialog

SetIcon(m\_hIcon, TRUE); // Set big icon

SetIcon(m\_hIcon, FALSE); // Set small icon

// TODO: Add extra initialization here

if (!Init())

EndDialog(IDCANCEL);

return FALSE;

//#define WM\_USER 0x0400

// CUserInfo userInfo;

// userInfo.m\_time = CTime::GetCurrentTime();

// userInfo.m\_strName = "sunhm";

// CString str("我吃饭去了。");

// ShowOfflineMsg(userInfo, str);

return TRUE; // return TRUE unless you set the focus to a control

}

// If you add a minimize button to your dialog, you will need the code below

// to draw the icon. For MFC applications using the document/view model,

// this is automatically done for you by the framework.

void CClientDlg::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this); // device context for painting

SendMessage(WM\_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

// Center icon in client rectangle

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// Draw the icon

dc.DrawIcon(x, y, m\_hIcon);

}

else

{

CDialog::OnPaint();

}

}

// The system calls this to obtain the cursor to display while the user drags

// the minimized window.

HCURSOR CClientDlg::OnQueryDragIcon()

{

return (HCURSOR) m\_hIcon;

}

/\*

\* 初始化

\*/

BOOL CClientDlg::Init(void)

{

InitListCtrlSetting(); //初始化列表视图控件

BegingListen(); //开始监听

if(!ConnectToServer()) //连接服务器

return FALSE;

return TRUE;

}

/\*

\* 连接服务器

\*/

BOOL CClientDlg::ConnectToServer(void)

{

//创建CChatSocket对象

m\_pSocket = new CChatSocket(this);

if (!m\_pSocket->Create())//创建套接字，绑定，注册网络事件

{

delete m\_pSocket;

m\_pSocket = NULL;

AfxMessageBox(\_T("创建套接字失败！"));

return FALSE;

}

m\_pFile = new CSocketFile(m\_pSocket); //创建CSocketFile对象

m\_pArchiveIn = new CArchive(m\_pFile,CArchive::load); //创建读入文档对象

CClientLoginDlg loginDlg(this);//登录对话框

if (IDOK == loginDlg.DoModal())

{

//服务器端口

SHORT shServPort = (SHORT)atoi((LPCTSTR)loginDlg.m\_strServerPort);

m\_servAddr.sin\_family = AF\_INET; //地址家族

m\_servAddr.sin\_addr.S\_un.S\_addr = htonl((u\_long)loginDlg.m\_dwIP); //地址

m\_servAddr.sin\_port = htons(shServPort); //端口

//连接服务器

while (!m\_pSocket->Connect((SOCKADDR\*)&m\_servAddr, sizeof(m\_servAddr)))

{

if (AfxMessageBox(\_T("连接服务器失败，是否再次尝试连接。"),MB\_YESNO) == IDNO)

{

return FALSE;

}

}

//获取本机套接字地址

SOCKADDR\_IN hostAddr;

int nSockaddLen = sizeof(SOCKADDR\_IN);

m\_pSocket->GetSockName((SOCKADDR\*)&hostAddr, &nSockaddLen);

m\_hostAddr.sin\_addr.S\_un.S\_addr = hostAddr.sin\_addr.S\_un.S\_addr;//主机地址

CChatPacket packet; //数据包

packet.m\_type = CChatPacket::USERLIST; //类型

packet.m\_UserInfo.m\_strName = loginDlg.m\_strName; //名称

packet.m\_UserInfo.m\_strPassword = loginDlg.m\_strPassword; //密码

packet.m\_UserInfo.m\_eStatus = CUserInfo::LOGIN; //用户状态

packet.m\_UserInfo.m\_lIP = hostAddr.sin\_addr.S\_un.S\_addr; //IP

packet.m\_UserInfo.m\_nPort = m\_hostAddr.sin\_port; //端口

packet.m\_UserInfo.m\_time = CTime::GetCurrentTime(); //登录时间

m\_UserInfo = packet.m\_UserInfo; //保存用户信息

//请求用户列表

SendPacket(packet);

return TRUE;

}else

{

return FALSE;

}

}

/\*

\* 等待接收消息

\*/

void CClientDlg::ProcessPendingRead(void)

{

do

{

ReadPacket();

if (m\_pSocket == NULL)

return;

}

while(!m\_pArchiveIn->IsBufferEmpty());//将缓冲区数据全部读入

}

/\*

\* 接收服务器数据

\*/

void CClientDlg::ReadPacket(void)

{

CObList obList;//临时链表

CChatPacket packet;//数据包

packet.m\_pUserList = &obList;

TRY

{

packet.Serialize(\*m\_pArchiveIn); //接收数据

if (packet.m\_type == CChatPacket::MESSAGE) //离线消息

{

ShowOfflineMsg(&packet);

}else if (packet.m\_type == CChatPacket::USERLIST) //用户列表

{

CreateUserList(&obList); //创建用户链表

UpdateClientListCtl(); //更新用户界面

m\_UserInfo.m\_eStatus = CUserInfo::ONLINE; //修改用户状态

}else if (packet.m\_type == CChatPacket::SERVERMSG) //登录失败消息

{

MessageBox(packet.m\_strMsg, "Server Messages", MB\_OK);

EndDialog(IDCANCEL); //退出

}

}

CATCH (CFileException, e)

{

MessageBox(\_T("读入chatter链表错误"));

}

END\_CATCH

}

/\*

\* 更新用户列表

\*/

void CClientDlg::UpdateClientListCtl(void)

{

//删除原来用户链表

m\_ctlUserList.DeleteAllItems();

POSITION pos;

int nIndex = 0;

for(pos = m\_UserList.GetHeadPosition(); pos != NULL;)//遍历整个链表

{

//获取每个用户信息

CUserInfo userInfo;

userInfo =\*(CUserInfo\*)m\_UserList.GetNext(pos);

LVITEM lvItem;

lvItem.mask = LVIF\_TEXT |LVIF\_IMAGE |LVIF\_PARAM;

lvItem.iItem = nIndex;

lvItem.iSubItem = 0;

lvItem.pszText = userInfo.m\_strName.GetBuffer(10);

if (userInfo.m\_eStatus == CUserInfo::ONLINE) //在线状态

{

lvItem.iImage = CUserInfo::ONLINE;

}else

{

lvItem.iImage = CUserInfo::OFFLINE; //离线状态

}

lvItem.lParam = (LPARAM)userInfo.m\_strName.GetBuffer(10);

m\_ctlUserList.InsertItem(&lvItem); //添加数据项

userInfo.m\_strName.ReleaseBuffer();

nIndex++;

}

}

/\*

\* 显示离线消息

\*/

void CClientDlg::ShowOfflineMsg(CChatPacket \*pPacket)

{

CChatDlg chatDlg(this, pPacket);//创建显示离线消息对话框

chatDlg.DoModal();

SaveChatLog(&chatDlg); //保存聊天记录

}

/\*

\* 发送数据

\*/

void CClientDlg::SendPacket(CChatPacket &packet)

{

m\_pArchiveOut = new CArchive(m\_pFile,CArchive::store); //初始化写入流

if (NULL != m\_pArchiveOut)

{

TRY

{

packet.Serialize(\*m\_pArchiveOut);

m\_pArchiveOut->Flush();

}

CATCH(CFileException, e)

{

m\_pArchiveOut->Abort();

delete m\_pArchiveOut;

m\_pArchiveOut = NULL;

MessageBox(\_T("发送数据包错误"));

}

END\_CATCH

}

if (NULL != m\_pArchiveOut)

{

delete m\_pArchiveOut;

m\_pArchiveOut = NULL;

}

}

void CClientDlg::CreateUserList(CObList \*pObList)

{

POSITION pos = NULL;

CUserInfo \*pUserInfo = NULL;

//删除原来链表

for (pos = m\_UserList.GetHeadPosition(); NULL != pos;)

{

pUserInfo = (CUserInfo\*)m\_UserList.GetNext(pos);

delete pUserInfo;

pUserInfo = NULL;

}

m\_UserList.RemoveAll();

//复制链表

CUserInfo \*pNewUserInfo = NULL;

for (pos = pObList->GetHeadPosition(); NULL != pos;)

{

pUserInfo = (CUserInfo\*)pObList->GetNext(pos);

pNewUserInfo= new CUserInfo();

pNewUserInfo->m\_strName = pUserInfo->m\_strName; //名称

pNewUserInfo->m\_eStatus = pUserInfo->m\_eStatus; //状态

pNewUserInfo->m\_lIP = pUserInfo->m\_lIP; //地址

pNewUserInfo->m\_nPort = pUserInfo->m\_nPort; //端口

m\_UserList.AddTail(pNewUserInfo); //加入链表

delete pUserInfo;

pUserInfo = NULL;

}

pObList->RemoveAll();

}

/\*

\* 双击列表视图控件消息消息响应函数

\*/

void CClientDlg::OnDblclkClientList(NMHDR\* pNMHDR, LRESULT\* pResult)

{

NM\_LISTVIEW\* pNMListView = (NM\_LISTVIEW\*)pNMHDR;

if(pNMListView->iItem != -1)

{

int nSelectItem = pNMListView->iItem;

CString strPeerName = m\_ctlUserList.GetItemText(nSelectItem, 0);

CreateChatDlg(strPeerName);

}

\*pResult = 0;

}

/\*

\* 创建聊天窗口

\*/

void CClientDlg::CreateChatDlg( CString strPeerName )

{

//不给自己发送消息

if (strPeerName == m\_UserInfo.m\_strName)

return;

//在用户链表中找到该用户

CUserInfo \*pPeerInfo = NULL;

POSITION pos = NULL;

BOOL bFinder = FALSE;

for (pos = m\_UserList.GetHeadPosition(); NULL != pos; )

{

pPeerInfo = (CUserInfo\*)m\_UserList.GetNext(pos);

if (pPeerInfo->m\_strName == strPeerName)

{

bFinder = TRUE;

break;

}

}

ASSERT(TRUE == bFinder);//找到该用户

if (pPeerInfo->m\_eStatus == CUserInfo::ONLINE) //用户在线

{

CChatDlg chatDlg(this, \*pPeerInfo, m\_UserInfo); //定义对话框

if (chatDlg.ConnectToPeer()) //连接对方

{

chatDlg.DoModal();//显示聊天窗口

//保存聊天记录

SaveChatLog(&chatDlg);

}

}else//发送离线消息

{

CChatDlg chatDlg(this, \*pPeerInfo, m\_UserInfo);//定义对话框

chatDlg.DoModal();//显示聊天窗口

//离线数据包

CChatPacket packet;

packet.m\_type = CChatPacket::MESSAGE; //类型

packet.m\_UserInfo = m\_UserInfo; //发送离线消息的用户信息

packet.m\_OfflineUserInfo = \*pPeerInfo; //接收离线消息的用户信息

packet.m\_strMsg = chatDlg.m\_strRec; //离线消息

packet.m\_time = CTime::GetCurrentTime();//发送消息的时间

//发送离线数据包

SendPacket(packet);

//保存聊天记录

SaveChatLog(&chatDlg);

}

}

/\*

\* 开始监听

\*/

BOOL CClientDlg::BegingListen( void )

{

m\_pListenSocket = new CListenSocket(this); //创建监听对象

if (m\_pListenSocket->Create()) //创建绑定套接字

{

if (!m\_pListenSocket->Listen()) //开始监听

{

delete m\_pListenSocket;

m\_pListenSocket = NULL;

AfxMessageBox(\_T("创建套接字失败！"));

return FALSE;

}

}

//获取地址信息

SOCKADDR\_IN hostAddr;

int hostAddrLen = sizeof(SOCKADDR\_IN);

m\_pListenSocket->GetSockName((SOCKADDR\*)&hostAddr,&hostAddrLen);

m\_hostAddr.sin\_port = hostAddr.sin\_port;//保存监听端口

return TRUE;

}

/\*

\*接受对方的连接

\*/

void CClientDlg::ProcessPendingAccept(void)

{

CChatDlg chatDlg(this); //聊天对话框

chatDlg.InitSocket(); //创建CChatSocket对象

//接受对方连接

CChatSocket \*pChatSocket = chatDlg.GetChatSocket();

SOCKADDR\_IN peerAddr;

int socketLen = sizeof(SOCKADDR\_IN);

m\_pListenSocket->Accept(\*pChatSocket,(SOCKADDR\*)&peerAddr, &socketLen);

//在用户链表中查找该用户

POSITION pos;

CUserInfo \*pPeerInfo = NULL;

BOOL bFinder = FALSE;

for (pos = m\_UserList.GetHeadPosition(); NULL != pos;)

{

pPeerInfo = (CUserInfo\*)m\_UserList.GetNext(pos);

if (pPeerInfo->m\_lIP == peerAddr.sin\_addr.S\_un.S\_addr &&//找到该用户

pPeerInfo->m\_eStatus == CUserInfo::ONLINE)

{

bFinder = TRUE;

break;

}

}

ASSERT(TRUE == bFinder);

//获取对方信息

CUserInfo peerInfo;

peerInfo.m\_strName = pPeerInfo->m\_strName; //名字

peerInfo.m\_eStatus = pPeerInfo->m\_eStatus; //状态

peerInfo.m\_lIP = peerAddr.sin\_addr.S\_un.S\_addr; //地址

peerInfo.m\_nPort = peerAddr.sin\_port; //端口

chatDlg.SetPeerInfor(peerInfo); //设置对方信息

chatDlg.SetUserInfor(m\_UserInfo); //设置用户信息

chatDlg.InitArchive(); //设置套接字读入文档

//打开聊天对话框

chatDlg.DoModal();

//保存聊天记录

SaveChatLog(&chatDlg);

}

/\*

\* 初始化列表视图控件

\*/

BOOL CClientDlg::InitListCtrlSetting( void )

{

//创建图标列表

m\_pImageList = new CImageList();

//32\*32，8位，初始化为2个图标，每次增长2个图标

m\_pImageList->Create(32, 32, ILC\_COLOR8|ILC\_MASK, 2, 2);

CWinApp\* pApp = AfxGetApp(); //获得应用程序指针

HICON hIconOnline = pApp->LoadIcon(IDI\_CHATTER\_ONLINE); //在线图标

HICON hIconOffline = pApp->LoadIcon(IDI\_CHATTER\_OFFLINE); //离线图标

m\_pImageList->Add(hIconOnline); //加入图标

m\_pImageList->Add(hIconOffline); //加入图标

//将图标列表赋值给列表控件

m\_ctlUserList.SetImageList(m\_pImageList, LVSIL\_SMALL);

//设置列表头

m\_ctlUserList.InsertColumn(0,\_T("Chatter"),LVCFMT\_IMAGE|LVCFMT\_LEFT);

//设置第一列的宽度为这个列表控件的宽度

CRect rect;

m\_ctlUserList.GetClientRect(&rect);

m\_ctlUserList.SetColumnWidth(0,rect.Width());

return TRUE;

}

/\*

\* 保存聊天记录

\*/

void CClientDlg::SaveChatLog(CChatDlg \*pChatDlg)

{

//没有聊天消息

if (pChatDlg->m\_strRec.IsEmpty())

return;

//保存聊天记录

char appPath[256];

GetCurrentDirectory(256,appPath); //取得应用程序当前路径

CString filePath; //文件名

filePath.Format("%s",appPath);

filePath +="\\";

filePath += CHATLOG;

TRY

{

CFile file;

if (!file.Open(filePath, CFile::modeReadWrite)) //文件已经存在

{

file.Open(filePath, CFile::modeCreate|CFile::modeReadWrite);//新建文件

}else

{

file.SeekToEnd();//移动文件指针到文件尾部

}

int nRecLen = pChatDlg->m\_strRec.GetLength(); //聊天记录长度

file.Write(pChatDlg->m\_strRec, nRecLen); //写入数据

file.Flush();

file.Close(); //关闭文件

}

CATCH(CFileException, e)

{

MessageBox(\_T("保存聊天记录异常"),MB\_OK, -1);

}

END\_CATCH

}

**ClientLogin.cpp**

// ClientLogin.cpp : implementation file

//

#include "stdafx.h"

#include "Client.h"

#include "ClientLogin.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CClientLogin dialog

CClientLoginDlg::CClientLoginDlg(CWnd\* pParent /\*=NULL\*/)

: CDialog(CClientLoginDlg::IDD, pParent)

{

//{{AFX\_DATA\_INIT(CClientLogin)

m\_strServerPort.Format("%d",SERVERPORT);

m\_strName = \_T("myChat");

m\_strPassword = \_T("");

//}}AFX\_DATA\_INIT

}

void CClientLoginDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

//{{AFX\_DATA\_MAP(CClientLogin)

DDX\_Control(pDX, IDC\_CLIENT\_IPADDRESS, m\_serverIP);

DDX\_Text(pDX, IDC\_CLIENT\_PASSWORD\_EDIT, m\_strPassword);

DDV\_MaxChars(pDX, m\_strPassword, 10);

DDX\_Text(pDX, IDC\_CLIENT\_NAME\_EDIT, m\_strName);

DDV\_MaxChars(pDX, m\_strName, 10);

DDX\_Text(pDX, IDC\_CLIENT\_PORT\_EDIT, m\_strServerPort);

DDV\_MaxChars(pDX, m\_strServerPort, 5);

//}}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP(CClientLoginDlg, CDialog)

//{{AFX\_MSG\_MAP(CClientLogin)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CClientLogin message handlers

BOOL CClientLoginDlg::OnInitDialog()

{

CDialog::OnInitDialog();

m\_serverIP.SetAddress(127, 0, 0, 1);

return TRUE;

}

void CClientLoginDlg::OnOK(void)

{

UpdateData(TRUE);

m\_serverIP.GetAddress(m\_dwIP);

CDialog::OnOK();

}

**ListenSocket.cpp**

// ListenSocket.cpp : implementation file

//

#include "stdafx.h"

#include "client.h"

#include "ListenSocket.h"

#include "ClientDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CListenSocket

CListenSocket:: CListenSocket(CClientDlg \*pClientDlg)

{

m\_pClientDlg = pClientDlg;

}

CListenSocket::~CListenSocket()

{

}

// Do not edit the following lines, which are needed by ClassWizard.

#if 0

BEGIN\_MESSAGE\_MAP(CListenSocket, CSocket)

//{{AFX\_MSG\_MAP(CListenSocket)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

#endif // 0

/////////////////////////////////////////////////////////////////////////////

// CListenSocket member functions

void CListenSocket::OnAccept(int nErrorCode)

{

m\_pClientDlg->ProcessPendingAccept();

}

**UserInfo.cpp**

// UserInfo.cpp: implementation of the CUserInfo class.

//

//////////////////////////////////////////////////////////////////////

#include "stdafx.h"

#include "UserInfo.h"

#ifdef \_DEBUG

#undef THIS\_FILE

static char THIS\_FILE[]=\_\_FILE\_\_;

#define new DEBUG\_NEW

#endif

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

//IMPLEMENT\_DYNCREATE(CUserInfo, CObject)

IMPLEMENT\_SERIAL(CUserInfo, CObject, 1)//new一个对象然后调用默认构造函数->init

CUserInfo::CUserInfo()

{

Init();

}

CUserInfo::~CUserInfo()

{

}

/\*

\* 初始化

\*/

void CUserInfo::Init(void)

{

m\_strName = \_T("");

m\_strPassword = \_T("");

m\_eStatus = UNKNOWN;

m\_lIP = 0;

m\_nPort = 0;

m\_time = CTime::GetCurrentTime();

}

/\*

\* 序列化

\*/

void CUserInfo::Serialize(CArchive& ar)

{

CObject::Serialize(ar);//调用基类的序列化函数

if (ar.IsStoring())//发送数据

{

ar << m\_strName; //名字

ar << m\_strPassword; //密码

WORD byStatus = 0;

byStatus = (WORD)m\_eStatus;

ar << byStatus; //状态

ar << m\_lIP; //地址

ar << m\_nPort; //端口

long lTime = m\_time.GetTime(); //日期和时间

ar << lTime;

}else

{

ar >> m\_strName; //名字

ar >> m\_strPassword; //密码

WORD byStatus = 0;

ar >> byStatus;

m\_eStatus = (USERSTATUE)byStatus;//状态

ar >> m\_lIP; //地址

ar >> m\_nPort; //端口

long lTime;

ar >> lTime; //日期和时间

m\_time = CTime((time\_t)lTime);

}

}

/\*

\* 赋值构造函数

\*/

CUserInfo &CUserInfo::operator = (const CUserInfo &userInfo)

{

m\_strName = userInfo.m\_strName;

m\_strPassword = userInfo.m\_strPassword;

m\_eStatus = userInfo.m\_eStatus;

m\_lIP = userInfo.m\_lIP;

m\_nPort = userInfo.m\_nPort;

m\_time = userInfo.m\_time;

return \*this;

}

/\*

\* 赋值构造函数

\*/

CUserInfo::CUserInfo(const CUserInfo& userInfo)

{

m\_strName = userInfo.m\_strName;

m\_strPassword = userInfo.m\_strPassword;

m\_eStatus = userInfo.m\_eStatus;

m\_lIP = userInfo.m\_lIP;

m\_nPort = userInfo.m\_nPort;

m\_time = userInfo.m\_time;

}

# 參考資料

《WindowsSocket網路程式設計教學》