服务端代码：

ServerDlg.cpp

#include "stdafx.h"

#include "Server.h"

#include "ServerDlg.h"

#include "ListenSocket.h"

#include "CLientSocket.h"

#include "../Content/UserType.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

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

#endif

class CAboutDlg : public CDialog

{

public:

CAboutDlg();

enum { IDD = IDD\_ABOUTBOX };

protected:

virtual void DoDataExchange(CDataExchange\* pDX);

protected:

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialog)

END\_MESSAGE\_MAP()

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

: CDialog(CServerDlg::IDD, pParent)

{

m\_uPort = GetPrivateProfileInt(\_T("NET"), \_T("SERVERPORT"), 8811, \_T("./Server.ini"));

GetPrivateProfileString(\_T("NET"), \_T("SERVERIP"),

\_T("127.0.0.1"),m\_strIP.GetBuffer(15), 15, \_T("./Server.ini"));

m\_hIcon = AfxGetApp()->LoadIcon(IDI\_ICON\_Logo);

}

void CServerDlg::DoDataExchange(CDataExchange\* pDX)

{

this->SetDefID(IDC\_BUTTON\_StartServer);

CDialog::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_LIST\_ServerOnLineList, m\_ServerOnLine);

DDX\_Text(pDX, IDC\_EDIT\_SerPort, m\_uPort);

DDX\_Text(pDX, IDC\_EDIT\_IP, m\_strIP);

}

BEGIN\_MESSAGE\_MAP(CServerDlg, CDialog)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(IDC\_BUTTON\_StartServer, OnBUTTONStartServer)

ON\_BN\_CLICKED(IDC\_BUTTON\_PauseServer, OnBUTTONPauseServer)

ON\_WM\_CLOSE()

//}}AFX\_MSG\_MAP

ON\_MESSAGE(UD\_NETPACK, OnNetRecv)

END\_MESSAGE\_MAP()

BOOL CServerDlg::OnInitDialog()

{

CDialog::OnInitDialog();

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL)

{

CString strAboutMenu;

strAboutMenu.LoadString(IDS\_ABOUTBOX);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

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

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

GetDlgItem(IDC\_BUTTON\_PauseServer)->EnableWindow(FALSE);

SetDlgItemText(IDC\_IPADDRESS\_IP, m\_strIP);

m\_ServerOnLine.SetExtendedStyle(LVS\_EX\_FULLROWSELECT|LVS\_EX\_GRIDLINES|LVS\_EX\_HEADERDRAGDROP );

m\_ServerOnLine.InsertColumn(0, \_T("序号"), LVCFMT\_LEFT, 50, -1);

m\_ServerOnLine.InsertColumn(1, \_T("用户名"), LVCFMT\_LEFT, 80, -1);

m\_ServerOnLine.InsertColumn(2, \_T("时间"), LVCFMT\_LEFT, 160, -1);

m\_ServerOnLine.InsertColumn(3, \_T("事件"), LVCFMT\_LEFT, 190, -1);

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

}

void CServerDlg::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialog::OnSysCommand(nID, lParam);

}

}

void CServerDlg::OnPaint()

{

if (IsIconic())

{

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

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

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;

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

}

else

{

CDialog::OnPaint();

}

}

HCURSOR CServerDlg::OnQueryDragIcon()

{

return (HCURSOR) m\_hIcon;

}

void CServerDlg::OnBUTTONStartServer()

{

UpdateData(TRUE);

if (m\_Socket.m\_hSocket != 0xffffffff)

{

return;

}

int res = m\_Socket.Create(m\_uPort, SOCK\_STREAM, m\_strIP);

if (!res)

{

MessageBox(\_T("创建套接字失败"), \_T("提示"), MB\_ICONINFORMATION);

return;

}

if (!m\_Socket.Listen())

{

MessageBox(\_T("监听失败"), \_T("提示"), MB\_ICONINFORMATION);

m\_Socket.Close(); //监听失败关闭Socket

return;

}

GetDlgItem(IDC\_BUTTON\_StartServer)->EnableWindow(FALSE);

InsertLog(\_T("服务器"), \_T("启动服务器"));

GetDlgItem(IDC\_BUTTON\_PauseServer)->EnableWindow(TRUE);

UpdateData(FALSE);

}

void CServerDlg::OnBUTTONPauseServer()

{

GetDlgItem(IDC\_BUTTON\_PauseServer)->EnableWindow(FALSE);

NET\_PACK pPack;

memset(&pPack, 0, sizeof(NET\_PACK));

pPack.fun = SRVSTOP;

lstrcpy(pPack.Content, \_T("服务器维护中..."));

list<CCLientSocket \*>::iterator itr;

for (itr=CCLientSocket::m\_plClient.begin(); itr!=CCLientSocket::m\_plClient.end(); itr++)

{

(\*itr)->Send(&pPack, sizeof(NET\_PACK));

delete (\*itr);

}

CCLientSocket::m\_plClient.clear();

if (m\_Socket.m\_hSocket != 0xffffffff)

{

m\_Socket.Close();

InsertLog(\_T("服务器"), \_T("关闭服务器"));

GetDlgItem(IDC\_BUTTON\_StartServer)->EnableWindow(TRUE);

}

}

void CServerDlg::InsertLog(CString strName, CString strEvent)

{

int iCount = m\_ServerOnLine.GetItemCount();

m\_iNumber.Format("%d", iCount+1);

m\_ServerOnLine.InsertItem(iCount, m\_iNumber);

m\_ServerOnLine.SetItemText(iCount, 1, strName);

//获取服务器启动的时间

CTime StartTime = CTime::GetCurrentTime();

CString SerStartTime = StartTime.Format(\_T("%Y/%m/%d %H:%M:%S"));

m\_ServerOnLine.SetItemText(iCount, 2, SerStartTime);

m\_ServerOnLine.SetItemText(iCount, 3, strEvent);

}

LRESULT CServerDlg::OnNetRecv(WPARAM wParam, LPARAM lParam)

{

NET\_PACK \*pPack = (NET\_PACK \*)wParam;

CCLientSocket \*pClient = (CCLientSocket \*)lParam;

switch(pPack->fun)

{

case LOGIN:

DealLogin(pPack->SenderID, pClient);

break;

case LOGOUT:

DealLogout(pPack->SenderID);

break;

case TRANSMIT:

DealTransmit(pPack);

break;

case SENDUSERLIST:

SendToClientOLUserList(pPack->ReceiveID);

break;

}

delete pPack;

pPack = NULL;

return 0;

}

void CServerDlg::DealLogin(CString strName, CCLientSocket \*pClient)

{

NET\_PACK pack;

memset(&pack, 0, sizeof(NET\_PACK));

CCLientSocket \*pSocket = FindClientSocket(strName);

if (pSocket == NULL)

{

//未登录

pack.fun = LOGOK;

/\* CString UserList;

UserList.Format(\_T("%s 在线"), strName);

lstrcpy(pack.Content, UserList);

pClient->Send(&pack, sizeof(NET\_PACK));\*/

pClient->m\_strName = strName;

CCLientSocket::m\_plClient.push\_back(pClient);

InsertLog(strName, strName+\_T(" 登陆成功"));

}

else

{

//用户已登录

pack.fun = LOGERR;

lstrcpy(pack.Content, \_T("该用户已登录"));

pClient->Send(&pack, sizeof(NET\_PACK));

pClient->Close();

}

SendToClientOLUserList(strName);

}

CCLientSocket \* CServerDlg::FindClientSocket(CString strName, BOOL bRemove)

{

CCLientSocket\* pClient = NULL;

list<CCLientSocket\*>::iterator itr;

for (itr=CCLientSocket::m\_plClient.begin(); itr!=CCLientSocket::m\_plClient.end(); itr++)

{

if ((\*itr)->m\_strName == strName)

{

pClient = \*itr;

if (bRemove)

{

CCLientSocket::m\_plClient.erase(itr);

}

break;

}

}

return pClient;

}

void CServerDlg::DealLogout(CString strName)

{

CCLientSocket\* pSocket = FindClientSocket(strName, TRUE);

if (pSocket)

{

delete pSocket;

pSocket = NULL;

InsertLog(strName, strName+\_T(" 注销成功"));

}

}

void CServerDlg::DealTransmit(NET\_PACK \*pPack)

{

CCLientSocket \*pSocket = FindClientSocket(pPack->ReceiveID);

if (pSocket)

{

pSocket->Send(pPack, sizeof(NET\_PACK));

}

}

void CServerDlg::OnClose()

{

OnBUTTONPauseServer();

CDialog::OnClose();

}

BOOL CServerDlg::PreTranslateMessage(MSG\* pMsg)

{

if (pMsg->message == WM\_KEYDOWN)

{

switch(pMsg->wParam)

{

case VK\_ESCAPE:

return TRUE;

}

}

return CDialog::PreTranslateMessage(pMsg);

}

void CServerDlg::SendToClientOLUserList(CString strName)

{

NET\_PACK pack;

memset(&pack, 0, sizeof(NET\_PACK));

pack.fun = SENDUSERLIST;

CString UserList;

UserList.Format(\_T("%s (在线)"), strName);

lstrcpy(pack.Content, UserList);

list<CCLientSocket \*>::iterator itr;

for (itr=CCLientSocket::m\_plClient.begin(); itr!=CCLientSocket::m\_plClient.end(); itr++)

{

(\*itr)->Send(&pack, sizeof(NET\_PACK));

//delete (\*itr);

}

}

ListenSocket.cpp

#include "stdafx.h"

#include "Server.h"

#include "ListenSocket.h"

#include "CLientSocket.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

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

#endif

CListenSocket::CListenSocket()

{

}

CListenSocket::~CListenSocket()

{

}

#if 0

BEGIN\_MESSAGE\_MAP(CListenSocket, CSocket)

END\_MESSAGE\_MAP()

#endif // 0

void CListenSocket::OnAccept(int nErrorCode)

{

CCLientSocket \*pClient = new CCLientSocket;

Accept(\*pClient);

CSocket::OnAccept(nErrorCode);

}

CLientSocket.cpp

#include "stdafx.h"

#include "Server.h"

#include "CLientSocket.h"

#include "../Content/UserType.h"

#include "ServerDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

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

#endif

list<CCLientSocket \*> CCLientSocket::m\_plClient;

CCLientSocket::CCLientSocket()

{

m\_strName = \_T("");

}

CCLientSocket::~CCLientSocket()

{

}

#if 0

BEGIN\_MESSAGE\_MAP(CCLientSocket, CSocket)

END\_MESSAGE\_MAP()

#endif // 0

void CCLientSocket::OnReceive(int nErrorCode)

{

NET\_PACK \*pPack = new NET\_PACK;

Receive(pPack, sizeof(NET\_PACK));

CServerDlg \*pDlg = (CServerDlg \*)AfxGetApp()->m\_pMainWnd;

pDlg->PostMessage(UD\_NETPACK, (WPARAM)pPack, (LPARAM)this);

CSocket::OnReceive(nErrorCode);

}

#ifndef \_USER\_TYPE\_

#define \_USER\_TYPE\_

typedef enum{LOGIN = 1, LOGOUT, LOGERR, LOGOK, TRANSMIT, SRVSTOP, SENDUSERLIST}FUN;

typedef struct

{

FUN fun;

char SenderID[32];

char ReceiveID[32];

char Content[1024];

}NET\_PACK;

#endif

客户端代码：

ClientDlg.cpp

#include "stdafx.h"

#include "Client.h"

#include "ClientDlg.h"

#include "../Content/UserType.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

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

#endif

class CAboutDlg : public CDialog

{

public:

CAboutDlg();

enum { IDD = IDD\_ABOUTBOX };

protected:

virtual void DoDataExchange(CDataExchange\* pDX);

protected:

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialog)

END\_MESSAGE\_MAP()

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

: CDialog(CClientDlg::IDD, pParent)

{

color = RGB(0, 0, 255);

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

m\_Name = \_T("");

m\_Input = \_T("");

m\_Receiver = \_T("");

m\_MsgList = \_T("");

m\_strAddUserName = \_T("");

m\_uPort = GetPrivateProfileInt(\_T("NET"), \_T("CLIENTPORT"), 8000, \_T("./Client.ini"));

GetPrivateProfileString(\_T("NET"), \_T("CLIENTIP"),

\_T("127.0.0.1"), m\_strIP.GetBuffer(15), 15, \_T("./Client.ini"));

m\_hIcon = AfxGetApp()->LoadIcon(IDI\_ICON\_Client);

}

void CClientDlg::DoDataExchange(CDataExchange\* pDX)

{

this->SetDefID(IDC\_BUTTON\_Send);

GetDlgItem(IDC\_EDIT\_Input)->SetFocus();

CDialog::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_LIST\_UserList, m\_UserList);

DDX\_Text(pDX, IDC\_EDIT\_Name, m\_Name);

DDX\_Text(pDX, IDC\_EDIT\_Input, m\_Input);

DDX\_Text(pDX, IDC\_EDIT\_Recver, m\_Receiver);

DDX\_Text(pDX, IDC\_EDIT\_MsgList, m\_MsgList);

DDX\_Text(pDX, IDC\_EDIT\_Addfri, m\_strAddUserName);

}

BEGIN\_MESSAGE\_MAP(CClientDlg, CDialog)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(IDC\_BUTTON\_Login, OnBUTTONLogin)

ON\_BN\_CLICKED(IDC\_BUTTON\_Logout, OnBUTTONLogout)

ON\_BN\_CLICKED(IDC\_BUTTON\_Send, OnBUTTONSend)

ON\_WM\_CLOSE()

ON\_BN\_CLICKED(IDC\_BUTTON\_Exit, OnBUTTONExit)

ON\_BN\_CLICKED(IDC\_BUTTON\_AddFriend, OnBUTTONAddFriend)

ON\_NOTIFY(NM\_CLICK, IDC\_LIST\_UserList, OnClickLISTUserList)

ON\_MESSAGE(UD\_NETPACK, OnNetRecv)

END\_MESSAGE\_MAP()

BOOL CClientDlg::OnInitDialog()

{

CDialog::OnInitDialog();

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL)

{

CString strAboutMenu;

strAboutMenu.LoadString(IDS\_ABOUTBOX);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

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

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

m\_UserList.SetExtendedStyle(LVS\_EX\_FULLROWSELECT|LVS\_EX\_GRIDLINES|LVS\_EX\_HEADERDRAGDROP);

m\_UserList.InsertColumn(0, \_T("序号"), LVCFMT\_LEFT, 48, -1);

m\_UserList.InsertColumn(1, \_T("用户名"), LVCFMT\_LEFT, 150, -1);

InitBtnStatus();

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

}

void CClientDlg::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialog::OnSysCommand(nID, lParam);

}

}

void CClientDlg::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this);

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

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;

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

}

else

{

CDialog::OnPaint();

}

}

HCURSOR CClientDlg::OnQueryDragIcon()

{

return (HCURSOR) m\_hIcon;

}

void CClientDlg::OnBUTTONLogin()

{

UpdateData(TRUE);

if (m\_Socket.m\_hSocket != 0xffffffff)

{

return;

}

if (m\_Name.IsEmpty())

{

MessageBox(\_T("账号不能为空"), \_T("提示"), MB\_ICONINFORMATION);

return;

}

if (!m\_Socket.Create())

{

MessageBox(\_T("创建套接字失败"), \_T("提示"), MB\_ICONINFORMATION);

return;

}

if (!m\_Socket.Connect(m\_strIP, m\_uPort))

{

MessageBox(\_T("连接服务器失败"), \_T("提示"), MB\_ICONINFORMATION);

m\_Socket.Close(); //关闭Socket

return;

}

LogOkBtnStatus();

SetUsetListColor(RGB(0, 0, 255));

UpdateData(FALSE);

NET\_PACK pack;

memset(&pack, 0, sizeof(NET\_PACK));

pack.fun = LOGIN;

lstrcpy(pack.SenderID, m\_Name);

m\_Socket.Send(&pack, sizeof(NET\_PACK));

}

LRESULT CClientDlg::OnNetRecv(WPARAM wParam, LPARAM lParam)

{

NET\_PACK \*pPack = (NET\_PACK \*)wParam;

switch (pPack->fun)

{

case LOGERR://登录失败

DealLogErr(pPack->Content);

break;

case LOGOK://登陆成功

DealLogOk(pPack->Content);

break;

case TRANSMIT://转发消息

DealTransmit(pPack);

break;

case SRVSTOP://服务器停止

DealSrvStop(pPack->Content);

break;

case SENDUSERLIST:

RecvUserList(pPack->Content);

break;

}

delete pPack;

pPack = NULL;

return 0;

}

void CClientDlg::DealLogOk(CString strContent)

{

}

void CClientDlg::DealLogErr(CString strContent)

{

LogOutBtnStatus();

m\_Socket.Close(); //关闭Socket

MessageBox(strContent, \_T("提示"), MB\_ICONINFORMATION);

}

void CClientDlg::DealTransmit(NET\_PACK \*pPack)

{

UpdateData(TRUE);

CString strTime, strInfo;

CTime tm = CTime::GetCurrentTime();

strTime = tm.Format("%Y-%m-%d %H:%M:%S");

strInfo.Format(\_T(" %s\r\n【%s】：%s\r\n"), strTime, pPack->SenderID, pPack->Content);

m\_MsgList += strInfo;

UpdateData(FALSE);

}

void CClientDlg::DealSrvStop(CString strContent)

{

m\_Socket.Close(); //关闭Socket

m\_UserList.DeleteAllItems();

LogOutBtnStatus();

MessageBox(strContent, \_T("提示"));

}

void CClientDlg::OnClose()

{

OnBUTTONLogout();

CDialog::OnClose();

}

void CClientDlg::OnBUTTONExit()

{

if (IDOK == MessageBox(\_T("确定关闭并退出吗？"), \_T("温馨提示"), MB\_OKCANCEL|MB\_ICONQUESTION))

{

OnBUTTONLogout();

::PostMessage(AfxGetMainWnd()->GetSafeHwnd(), WM\_CLOSE, 0, 0);

}

}

void CClientDlg::InsertUser(CString strUserName)

{

int iCount = m\_UserList.GetItemCount();

m\_iItem.Format("%d", iCount+1);

m\_UserList.InsertItem(iCount, m\_iItem);

m\_UserList.SetItemText(iCount, 1, strUserName);

}

BOOL CClientDlg::PreTranslateMessage(MSG\* pMsg)

{

if (pMsg->message == WM\_KEYDOWN)

{

switch(pMsg->wParam)

{

case VK\_ESCAPE:

return TRUE;

}

}

return CDialog::PreTranslateMessage(pMsg);

}

void CClientDlg::RecvUserList(CString strContent)

{

InsertUser(strContent);

}

void CClientDlg::SetUsetListColor(COLORREF color)

{

m\_UserList.SetTextColor(color);

}

void CClientDlg::OnBUTTONAddFriend()

{

UpdateData(TRUE);

if (m\_strAddUserName.IsEmpty())

{

MessageBox(\_T("对方账号不能为空"), \_T("温馨提示"), MB\_ICONINFORMATION);

return;

}

CString m\_iNumber, str;

int iCount = m\_UserList.GetItemCount();

for (int i=0; i<iCount; i++)

{

str = m\_UserList.GetItemText(i++, 1);

if (m\_strAddUserName == str.Left(str.GetLength()-7))

{

MessageBox(\_T("该用户已经是你的好友，不能重复添加。"));

return;

}

}

InsertUser(m\_strAddUserName);

m\_strAddUserName = \_T("");

UpdateData(FALSE);

}

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

{

UpdateData(TRUE);

int nItem = -1;

LPNMITEMACTIVATE lpNMItemActivate = (LPNMITEMACTIVATE)pNMHDR;

if(lpNMItemActivate != NULL)

{

nItem = lpNMItemActivate->iItem;

}

CString strName, Extra;

strName.Format("%d",nItem);

strName = m\_UserList.GetItemText(nItem,1);

if (strName.GetLength() > 7)

{

m\_Receiver = strName.Left(strName.GetLength()-7);

}

else

{

m\_Receiver = strName;

}

UpdateData(FALSE);

\*pResult = 0;

}

void CClientDlg::InitBtnStatus()

{

GetDlgItem(IDC\_BUTTON\_Logout)->EnableWindow(FALSE);

GetDlgItem(IDC\_BUTTON\_Send)->EnableWindow(FALSE);

GetDlgItem(IDC\_BUTTON\_AddFriend)->EnableWindow(FALSE);

GetDlgItem(IDC\_EDIT\_Addfri)->EnableWindow(FALSE);

GetDlgItem(IDC\_EDIT\_Input)->EnableWindow(FALSE);

GetDlgItem(IDC\_EDIT\_Recver)->EnableWindow(FALSE);

GetDlgItem(IDC\_LIST\_UserList)->EnableWindow(FALSE);

}

ClientSocket.cpp

#include "stdafx.h"

#include "Client.h"

#include "ClientSocket.h"

#include "ClientDlg.h"

#include "../Content/UserType.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

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

#endif

CClientSocket::CClientSocket()

{

}

CClientSocket::~CClientSocket()

{

}

#if 0

BEGIN\_MESSAGE\_MAP(CClientSocket, CSocket)

END\_MESSAGE\_MAP()

#endif // 0

void CClientSocket::OnReceive(int nErrorCode)

{

NET\_PACK \*pPack = new NET\_PACK;

Receive(pPack, sizeof(NET\_PACK));

CClientDlg \*pDlg = (CClientDlg \*)AfxGetApp()->m\_pMainWnd;

pDlg->PostMessage(UD\_NETPACK, (WPARAM)pPack, NULL);

CSocket::OnReceive(nErrorCode);

}

#ifndef \_USER\_TYPE\_

#define \_USER\_TYPE\_

typedef enum{LOGIN = 1, LOGOUT, LOGERR, LOGOK, TRANSMIT, SRVSTOP, SENDUSERLIST}FUN;

typedef struct

{

FUN fun;

char SenderID[32];

char ReceiveID[32];

char Content[1024];

}NET\_PACK;