#include "stdafx.h"

#include "Server.h"

#include "ServerDlg.h"

#include "afxdialogex.h"

#include "ClientSocket.h"

#define PORT 4980

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

#define port 4979

// CServerDlg 对话框

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

: CDialogEx(CServerDlg::IDD, pParent)

, m\_strFilePath(\_T(""))

{

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

}

void CServerDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

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

DDX\_Text(pDX, IDC\_FILE\_PATH, m\_strFilePath);

}

BEGIN\_MESSAGE\_MAP(CServerDlg, CDialogEx)

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_EN\_CHANGE(IDC\_FILE\_PATH, &CServerDlg::OnEnChangeEdit1)

ON\_BN\_CLICKED(IDC\_BTN\_BROWSER, &CServerDlg::OnBnClickedBtnBrowser)

ON\_BN\_CLICKED(IDC\_BTN\_SEND, &CServerDlg::OnBnClickedBtnSend)

END\_MESSAGE\_MAP()

// CServerDlg 消息处理程序

BOOL CServerDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

// 设置此对话框的图标。 当应用程序主窗口不是对话框时，框架将自动

// 执行此操作

SetIcon(m\_hIcon, TRUE); // 设置大图标

SetIcon(m\_hIcon, FALSE); // 设置小图标

m\_pClientSocket = new CClientSocket;

//创建套接字

CString strMsg;

if (FALSE == m\_pClientSocket->Create(port, SOCK\_STREAM))

{

strMsg.Format(\_T("创建套接字失败.错误编号:%d"), GetLastError());

MessageBox(strMsg, \_T("温馨提示"), MB\_OK);

delete m\_pClientSocket;

m\_pClientSocket = NULL;

//关闭对话框

EndDialog(IDOK);

return TRUE;

}

if (FALSE == m\_pClientSocket->Connect(\_T(" 10.21.186.29"), port))

{

strMsg.Format(\_T("连接服务器失败.错误编号:%d"), GetLastError());

MessageBox(strMsg, \_T("温馨提示"), MB\_OK);

delete m\_pClientSocket;

m\_pClientSocket = NULL;

//关闭对话框

EndDialog(IDOK);

return TRUE;

}

m\_list.SetExtendedStyle(LVS\_EX\_FULLROWSELECT);

m\_list.InsertColumn(0, \_T("IP地址"), LVCFMT\_LEFT, 200);

m\_list.InsertColumn(1, \_T("端口"), LVCFMT\_LEFT, 100);

//创建对象

m\_pListenSocket = new CListenSocket;

//创建套接字

CString strMsg;

if (FALSE == m\_pListenSocket->Create(PORT, SOCK\_STREAM))

{

strMsg.Format(\_T("创建套接字失败.错误编号:%d"), GetLastError());

MessageBox(strMsg, \_T("温馨提示"), MB\_OK);

delete m\_pListenSocket;

m\_pListenSocket = NULL;

//关闭对话框

EndDialog(IDOK);

return TRUE;

}

//将套接字设置为监听模式

if (FALSE == m\_pListenSocket->Listen())

{

strMsg.Format(\_T("监听失败.错误编号:%d"), GetLastError());

MessageBox(strMsg, \_T("温馨提示"), MB\_OK);

delete m\_pListenSocket;

m\_pListenSocket = NULL;

EndDialog(IDOK);

return TRUE;

}

return TRUE; // 除非将焦点设置到控件，否则返回 TRUE

}

// 如果向对话框添加最小化按钮，则需要下面的代码

// 来绘制该图标。 对于使用文档/视图模型的 MFC 应用程序，

// 这将由框架自动完成。

void CServerDlg::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this); // 用于绘制的设备上下文

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// 使图标在工作区矩形中居中

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;

// 绘制图标

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

}

else

{

CDialogEx::OnPaint();

}

}

//当用户拖动最小化窗口时系统调用此函数取得光标

//显示。

HCURSOR CServerDlg::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

void CServerDlg::OnEnChangeEdit1()

{

// TODO: 如果该控件是 RICHEDIT 控件，它将不

// 发送此通知，除非重写 CDialogEx::OnInitDialog()

// 函数并调用 CRichEditCtrl().SetEventMask()，

// 同时将 ENM\_CHANGE 标志“或”运算到掩码中。

// TODO: 在此添加控件通知处理程序代码

}

void CServerDlg::OnBnClickedBtnBrowser()

{

CFileDialog dlg(TRUE);

if (IDCANCEL == dlg.DoModal())

return;

m\_strFilePath = dlg.GetPathName();

UpdateData(FALSE);

}

//发送

void CServerDlg::OnBnClickedBtnSend()

{

UpdateData(TRUE);

if (m\_strFilePath.IsEmpty())

{

MessageBox(\_T("请您先选择需要发送的文件."), \_T("温馨提示"), MB\_OK);

return;

}

int nSel = m\_list.GetSelectionMark();

if (nSel < 0)

{

MessageBox(\_T("请您先选择需要发送的人."), \_T("温馨提示"), MB\_OK);

return;

}

//获取套接字

CClientSocket\*pSocket = (CClientSocket\*)m\_list.GetItemData(nSel);

//获取文件属性

WIN32\_FIND\_DATA wfd;

HANDLE hFinder = FindFirstFile(m\_strFilePath, &wfd);

FindClose(hFinder);

//先发文件属性

pSocket->Send(&wfd, sizeof(wfd));

//发送文件内容

CFile file;

if (FALSE == file.Open(m\_strFilePath, CFile::modeRead|CFile::typeBinary))

{

MessageBox(\_T("打开文件失败."), \_T("温馨提示"), MB\_OK);

return;

}

//循环发送

DWORD dwReadCount = 0;

while (dwReadCount < wfd.nFileSizeLow)

{

char \* buffer = new char[1025];

int nRead = file.Read(buffer, 1024);

buffer[nRead] = '\0';

//发送

pSocket->Send(buffer, nRead);

dwReadCount += nRead;

delete[] buffer;

}

//关闭文件

file.Close();

MessageBox(\_T("发送数据完毕"), \_T("温馨提示"), MB\_OK);

}