**实验6 基于消息选择模型的聊天软件**

【实验要求】

1、定义套接字网络事件对应的用户消息

2、如果不存在窗口，则创建窗口和窗口例程支持函数

3、调用 WSAAsyncSelect0函数为套接字设置网络事件、用户消息和消息接收窗口之间的关系。

4、增加消息循环的具体功能，或者添加消息与消息处理函数的映射关系。

5、添加消息处理框架的具体功能，判断是哪个套接字上发生了网络事件，使用 WSAGETSELECTEVENT宏了解所发生的网络事件，从而进行相应的处理。

界面功能：

设计用户界面，能够判断用户输入的合法性，接收用户输入的服务器IP地址和端口信息，获取用户的的聊天内容，并将聊天内容显示在用户界面中。

网络功能：

提供服务器和客户端两种角色的网络功能。对服务器，设计若干函数来完成服务器的主体功能；对客户端，设计若干函数完成客户端的主体功能。

异步I/O处理功能：

基于 WSAAsyncSelectt模型实现异步/O,构建程序中合理的消息处理框架，其中包括自定义消息声明，消息接收窗口的创建与注册，消息的判断与处理等。

【实验过程】

// MFCApplication1Dlg.h: 头文件

#pragma once

#define MAXLINE 200

#define SERVER 1

#define CLIENT 0

#define WM\_READ WM\_USER+101

// CMFCApplication1Dlg 对话框

class CMFCApplication1Dlg : public CDialogEx

{

// 构造

public:

CMFCApplication1Dlg(CWnd\* pParent = nullptr); // 标准构造函数

// 对话框数据

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_MFCAPPLICATION1\_DIALOG };

#endif

protected:

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

// 实现

protected:

HICON m\_hIcon;

// 生成的消息映射函数

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnPaint();

afx\_msg HCURSOR OnQueryDragIcon();

DECLARE\_MESSAGE\_MAP()

public:

CButton m\_nServerType;

CIPAddressCtrl m\_ulIP;

CEdit m\_usPort;

CEdit m\_SendText;

CListBox m\_List;

SOCKET m\_Socket = INVALID\_SOCKET;

struct sockaddr\_in m\_peer, m\_local;

afx\_msg void OnBnClickedBegin();

afx\_msg void OnBnClickedSend();

afx\_msg void OnBnClickedServer();

afx\_msg void OnBnClickedClient();

afx\_msg LRESULT OnRecvFrom(WPARAM wParam, LPARAM lParam);

};

// MFCApplication1Dlg.cpp: 实现文件

//

#include "pch.h"

#include "framework.h"

#include "MFCApplication1.h"

#include "MFCApplication1Dlg.h"

#include "afxdialogex.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// 用于应用程序“关于”菜单项的 CAboutDlg 对话框

class CAboutDlg : public CDialogEx

{

public:

CAboutDlg();

// 对话框数据

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_ABOUTBOX };

#endif

protected:

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

// 实现

protected:

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialogEx(IDD\_ABOUTBOX)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

END\_MESSAGE\_MAP()

// CMFCApplication1Dlg 对话框

CMFCApplication1Dlg::CMFCApplication1Dlg(CWnd\* pParent /\*=nullptr\*/)

: CDialogEx(IDD\_MFCAPPLICATION1\_DIALOG, pParent)

{

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

}

void CMFCApplication1Dlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_SERVER, m\_nServerType);

DDX\_Control(pDX, IDC\_IPADDRESS, m\_ulIP);

DDX\_Control(pDX, IDC\_PORT, m\_usPort);

DDX\_Control(pDX, IDC\_SENDTEXT, m\_SendText);

DDX\_Control(pDX, IDC\_LIST1, m\_List);

}

BEGIN\_MESSAGE\_MAP(CMFCApplication1Dlg, CDialogEx)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(IDC\_BEGIN, &CMFCApplication1Dlg::OnBnClickedBegin)

ON\_BN\_CLICKED(IDC\_SEND, &CMFCApplication1Dlg::OnBnClickedSend)

ON\_BN\_CLICKED(IDC\_SERVER, &CMFCApplication1Dlg::OnBnClickedServer)

ON\_BN\_CLICKED(IDC\_CLIENT, &CMFCApplication1Dlg::OnBnClickedClient)

ON\_MESSAGE(WM\_READ, &CMFCApplication1Dlg::OnRecvFrom)

END\_MESSAGE\_MAP()

// CMFCApplication1Dlg 消息处理程序

BOOL CMFCApplication1Dlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

// 将“关于...”菜单项添加到系统菜单中。

// IDM\_ABOUTBOX 必须在系统命令范围内。

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

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != nullptr)

{

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

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

}

}

// 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

// TODO: 在此添加额外的初始化代码

return TRUE; // 除非将焦点设置到控件，否则返回 TRUE

}

void CMFCApplication1Dlg::OnSysCommand(UINT nID, LPARAM lParam)

{

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

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialogEx::OnSysCommand(nID, lParam);

}

}

// 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 CMFCApplication1Dlg::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 CMFCApplication1Dlg::OnQueryDragIcon()

{

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

}

void CMFCApplication1Dlg::OnBnClickedBegin()

{

int iResult = 0;

WSADATA wsaData;

UpdateData(TRUE);

//Windows Sockets Dll初始化

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

if (iResult != 0)

{

m\_List.InsertString(0, "WSAStartup函数出错.");

return;

}

//当程序作为服务器

if (m\_nServerType.GetState() == SERVER)

{

//为服务器的本地地址m\_local设置用户输入的IP和端口号

memset(&m\_local, 0, sizeof(m\_local));

m\_local.sin\_family = AF\_INET;

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

m\_local.sin\_port = htons(GetDlgItemInt(IDC\_PORT));

//创建服务器端的数据报套接字并在指定端口号上监听

m\_Socket = socket(AF\_INET,SOCK\_DGRAM,0);

if (m\_Socket == INVALID\_SOCKET)

{

m\_List.InsertString(0, "socket函数出错!");

WSACleanup();

return;

}

iResult = bind(m\_Socket, (const struct sockaddr\*)&m\_local, sizeof(SOCKADDR\_IN));

if (iResult == SOCKET\_ERROR) {

m\_List.InsertString(0, "bind函数出错!");

closesocket(m\_Socket);

WSACleanup();

return;

}

m\_List.InsertString(0, \_T("服务器已启动"));

}

else

{

//创建客户端的数据报套接字并向服务器请求建立连接

UpdateData(TRUE);

if (m\_ulIP.IsBlank() || GetDlgItemInt(IDC\_PORT) == NULL)

{

m\_List.InsertString(0, "IP 地址和端口号不能为空");

return;

}

// 指明服务器的地址m\_peer为用户输入的IP和端口号

DWORD ipaddr;

m\_ulIP.GetAddress(ipaddr);

memset(&m\_peer, 0, sizeof(m\_peer));

m\_peer.sin\_family = AF\_INET;

m\_peer.sin\_addr.S\_un.S\_addr = htonl(ipaddr);

m\_peer.sin\_port = htons(GetDlgItemInt(IDC\_PORT));

// 创建客户端的数据报套接字

m\_Socket = socket(AF\_INET, SOCK\_DGRAM,0 );

if (m\_Socket == INVALID\_SOCKET)

{

m\_List.InsertString(0, "socket函数出错!");

WSACleanup();

return;

}

m\_List.InsertString(0, \_T("客户端已启动"));

}

iResult = WSAAsyncSelect(m\_Socket, m\_hWnd, WM\_READ, FD\_READ);

if (iResult == SOCKET\_ERROR)

{

m\_List.InsertString(0, "WSAAsyncSelect设定失败!");

return;

}

//更新对话框控件的可操作性

GetDlgItem(IDC\_SENDTEXT)->EnableWindow(TRUE);

GetDlgItem(IDC\_SEND)->EnableWindow(TRUE);

GetDlgItem(IDC\_IPADDRESS)->EnableWindow(FALSE);

GetDlgItem(IDC\_SERVER)->EnableWindow(FALSE);

GetDlgItem(IDC\_CLIENT)->EnableWindow(FALSE);

GetDlgItem(IDC\_PORT)->EnableWindow(FALSE);

GetDlgItem(IDC\_BEGIN)->EnableWindow(FALSE);

}

void CMFCApplication1Dlg::OnBnClickedSend()

{

time\_t ticks;

char buff[MAXLINE], info[MAXLINE];

int iResult = -1;

// 更新对话框里的数据

UpdateData(TRUE);

if (m\_SendText.GetWindowTextLengthA() > MAXLINE)

{

m\_List.InsertString(0, "发送内容超长！");

return;

}

if (m\_peer.sin\_addr.S\_un.S\_addr == 0)

{

m\_List.InsertString(0, "目标地址为空，请确定正确的目标地址！");

return;

}

// 获取当前时间

ticks = time(NULL);

memset(buff, 0, sizeof(buff));

sprintf\_s(buff, MAXLINE, "%.24s\r\n", ctime(&ticks));

if (m\_nServerType.GetState() == SERVER)

{

sprintf\_s(info, MAXLINE, "%s服务器向%s发送数据:", buff, inet\_ntoa(m\_peer.sin\_addr));

}

else

{

sprintf\_s(info, MAXLINE, "%s客户端向%s发送数据:", buff, inet\_ntoa(m\_peer.sin\_addr));

}

// 发送数据

char Content[MAXLINE] = { 0 };

m\_SendText.GetWindowText((LPTSTR)&Content, MAXLINE);

iResult = sendto(m\_Socket,Content, m\_SendText.GetWindowTextLength(),0, (SOCKADDR\*)&m\_peer, sizeof(m\_peer));

if (iResult == SOCKET\_ERROR)

{

sprintf\_s(info, MAXLINE, "sendto函数调用错误，错误号：%d", WSAGetLastError());

m\_List.InsertString(0, info);

closesocket(m\_Socket);

}

else

{

m\_List.InsertString(0, strcat(info,Content));

}

}

void CMFCApplication1Dlg::OnBnClickedServer()

{

// TODO: 在此添加控件通知处理程序代码

m\_nServerType.SetCheck(SERVER);

UpdateData(TRUE);

}

void CMFCApplication1Dlg::OnBnClickedClient()

{

// TODO: 在此添加控件通知处理程序代码

m\_nServerType.SetCheck(CLIENT);

UpdateData(TRUE);

}

LRESULT CMFCApplication1Dlg::OnRecvFrom(WPARAM wParam, LPARAM lParam)

{

int iResult;

time\_t ticks;

char buff[MAXLINE], info[MAXLINE];

char recvbuf[MAXLINE];

struct sockaddr\_in cliaddr = {};

int addrlen = sizeof(sockaddr\_in);

CString str;

memset(recvbuf, 0, MAXLINE);

// 获取当前时间

ticks = time(NULL);

memset(buff, 0, sizeof(buff));

sprintf\_s(buff, MAXLINE, "%.24s\r\n", ctime(&ticks));

// 判断网络事件

if (WSAGETSELECTERROR(lParam))

m\_List.InsertString(0, "套接字错误。");

else

{

switch (WSAGETSELECTEVENT(lParam))

{

case FD\_READ:

// 接收数据

iResult = recvfrom(m\_Socket, recvbuf, MAXLINE, 0, (SOCKADDR\*)&cliaddr, &addrlen);

if (iResult == SOCKET\_ERROR)

{

sprintf\_s(info, MAXLINE, "%s接收数据发生错误，错误号：%d", buff, WSAGetLastError());

str = info;

m\_List.InsertString(0, str);

return 0;

}

sprintf\_s(info, MAXLINE, "%s接收到来自%s的数据：", buff, inet\_ntoa(cliaddr.sin\_addr));

str = info;

m\_List.InsertString(0, str + recvbuf);

// 更新通信对方地址

m\_peer = cliaddr;

break;

default:

break;

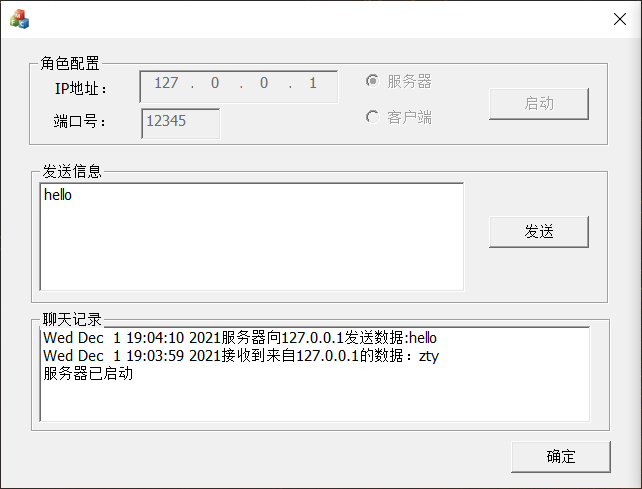
}

}

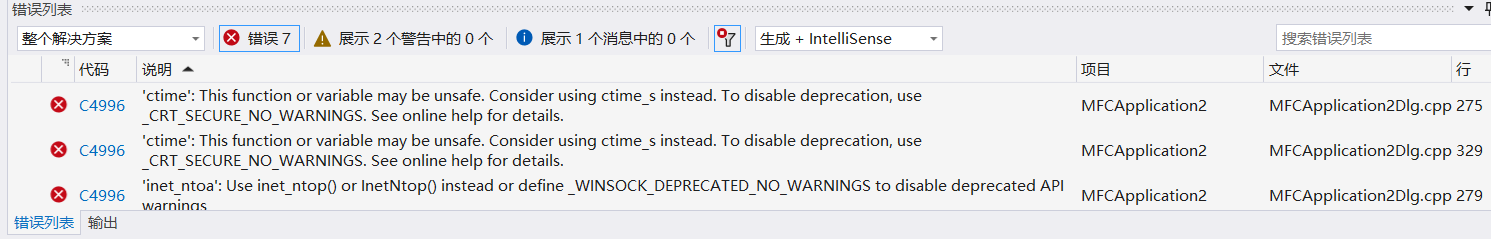
return 0;

}

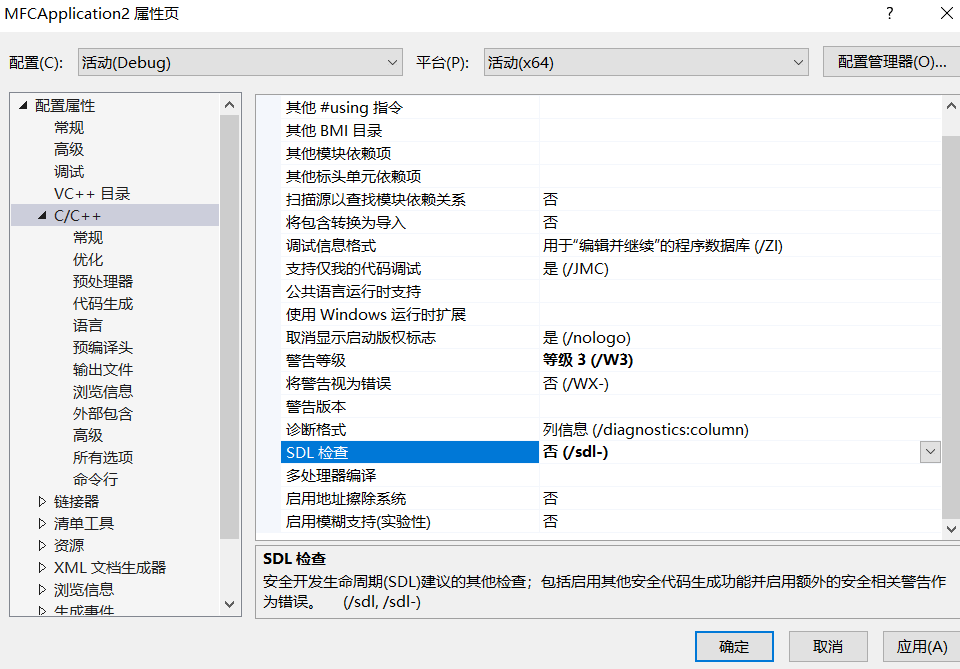
截图：



编译出现问题：



通过将SDL检查由是改为否解决：



【实验总结】

MFC框架中“消息映射”是通过巧妙的宏定义形成一张消息映射表格来完成的。消息一旦发生，框架就可以根据消息映射表格来进行消息映射和命令传递。