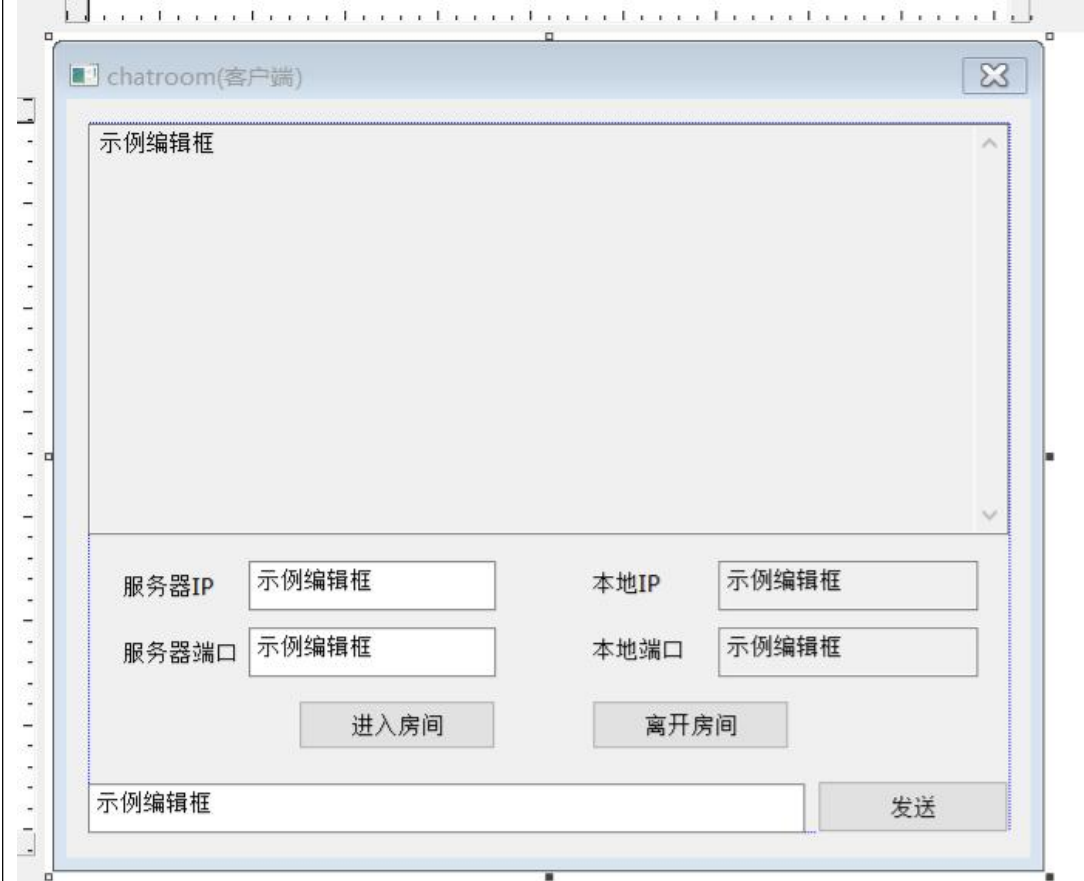


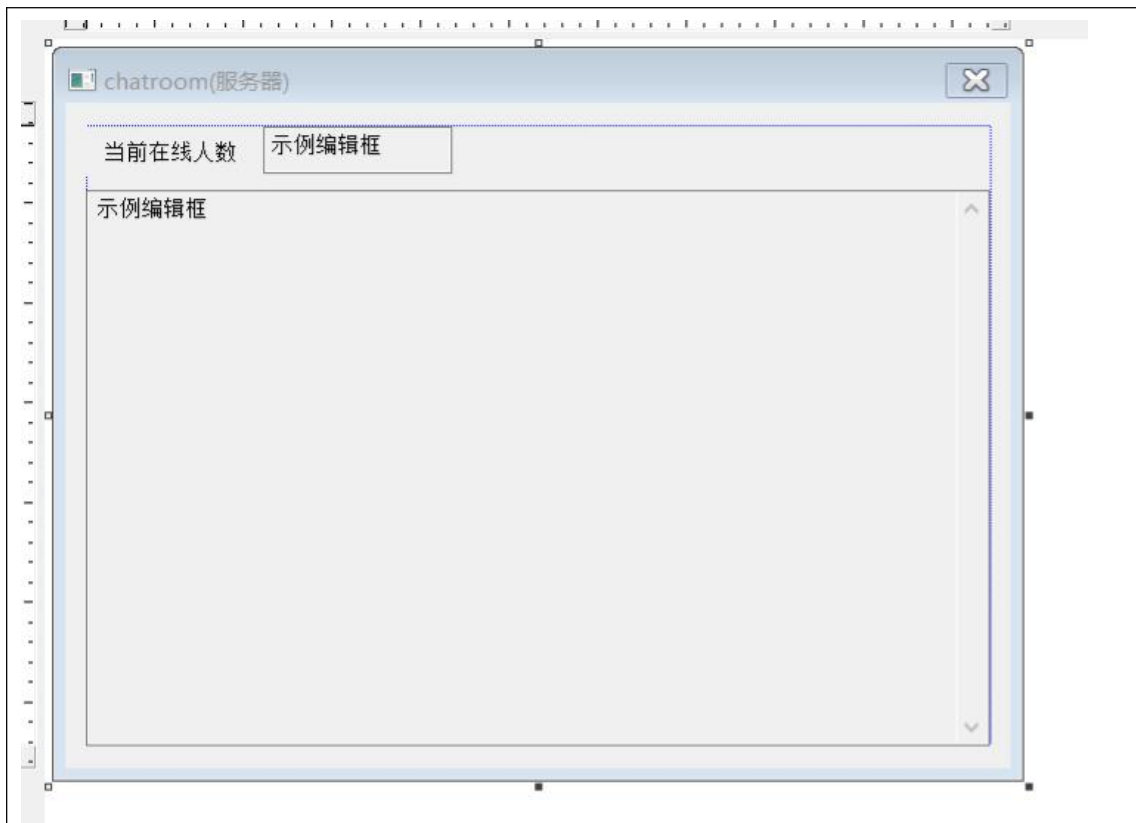
计算机学院 计算机网络 课程实验报告

| | | |
|----------------|-------|--|
| 实验题目：TCP 多人聊天室 | | 学号、姓名： 202200130048 陈静雯 202200130281 姚思彤 202200130250 阳享芝 |
| 日期：2024/6/7 | 班级： 6 | |

实验内容：
编写类似微信群功能，允许多人聊天
(1) 必须实现功能（自由选择 TCP 或 UDP，编程语言）：
1 服务器端：允许用户联网接入，接收用户键入内容，把内容传输给所有在群里的用户（客户端）。
2 客户端：加入服务器，向服务器发送用户键入的内容，接收并显示服务器发送的内容。
(2) 扩展内容：自由构思

实验方法介绍：
本实验使用 MFC 进行对聊天室的相关操作开发。
(设计界面自动生成的代码不再展示)
1. 界面设计





2. CConnectSocket

1) .h 文件

继承自 CSocket 类的一个派生类，重写 OnReceive 函数，并添加构造和析构函数以及一个 mp 指针指向客户端窗口。

```
▼ CConnectSocket

#pragma once
#include <afxsock.h>

class CchatclientDlg;

class CConnectSocket :
{
public:
    CConnectSocket(CchatclientDlg* pdlg);
    virtual ~CConnectSocket();

private:
    CchatclientDlg* m_pMainDlg;

public:
    virtual void OnReceive(int nErrorCode);
};
```

2) .cpp 实现文件

简写构造和析构,而 Receive 的重写具体实现放到主程序的 processpendingread 函数里

```
1  #include "pch.h"
2  #include "ConnectSocket.h"
3  #include "chatclientDlg.h"
4
5
6  CConnectSocket::CConnectSocket(CchatclientDlg* pDlg) { //构造
7      m_pMainDlg = pDlg;
8  }
9  CConnectSocket::~CConnectSocket() { //析构
10
11  }
12
13  void CConnectSocket::OnReceive(int nErrorCode)
14  {
15      // TODO: 在此添加专用代码和/或调用基类
16
17      CSocket::OnReceive(nErrorCode);
18      m_pMainDlg->ProcessPendingRead();
19  }
```

3. ListenSocket

1) .h 头文件

也是继承自 CSocket 的一个派生类,除了自己的构造和析构函数,还需重写一个 OnAccept 函数

```
#pragma once
#include <afxsock.h>

class CchatserverDlg;

class CListenSocket :
public CSocket
{
public:
    CListenSocket(CchatserverDlg* pDlg);
    virtual ~CListenSocket();
private:
    CchatserverDlg* m_pMainDlg;
public:
    virtual void OnAccept(int nErrorCode);
};
```

2) .cpp 实现文件

同样将 OnAccept 的重写具体实现放到主程序的 processpendingaccept 函数实现

```
#include "pch.h"
#include "ListenSocket.h"
#include "chatserverDlg.h"

class CchatserverDlg;

CListenSocket::CListenSocket(CchatserverDlg* pDlg) {
    m_pMainDlg = pDlg;
}

CListenSocket::~CListenSocket() {
}

void CListenSocket::OnAccept(int nErrorCode)
{
    // TODO: 在此添加专用代码和/或调用基类

    CSocket::OnAccept(nErrorCode);
    m_pMainDlg->ProcessPendingAccept();
}
```

4. chatserver

1) .h 头文件

需要添加的是 public 成员, 包括两个实现函数, 监听套接字、连接套接字、nroom 显示此时聊天室中有多少成员。以及两个 list 装房间成员的套接字和 address。

```
public:
    void ProcessPendingRead(CConnectSocket* connectSocket);
    void ProcessPendingAccept();
    CConnectSocket* m_pConnectSocket;
    CListenSocket* m_pListenSocket;
    int nRoom;
    CList<CConnectSocket*>m_pConnectsocketlist;
    CList<ClientAddr>m_pClientAddrlist;
```

2) .cpp 实现函数

首先初始化, 创建套接字并开始监听

```

// TODO: 在此添加额外的初始化代码
m_pListenSocket = new CListenSocket(this);
m_pListenSocket->Create(8080);

if (!m_pListenSocket->Listen()) {
    MessageBox(_T("监听失败"));
    return FALSE;
}

```

然后是 accept 的重写

```

void CchatserverDlg::ProcessPendingAccept() {
    CConnectSocket* pConnectSocket = new CConnectSocket(this);
    if (m_pListenSocket->Accept(*pConnectSocket)) {
        m_pConnectsocketlist.AddTail(pConnectSocket);
    }
    else {
        delete pConnectSocket;
    }
}

```

最后是服务器接收数据的函数，接收数据，获取发送消息的客户端的 ip 和端口号，然后解析数据，判断发送的是进入房间还是离开房间还是发送了一条聊天消息，分别进行不同的操作，最后对房间里的所有成员将消息转发到他们的客户端窗口。

```

void CchatserverDlg::ProcessPendingRead(CConnectSocket* connectSocket) {
    TCHAR buff[4096];
    ClientAddr clientaddr;

    //接收数据
    int nread = connectSocket->Receive(buff, 4096);
    if (nread == SOCKET_ERROR) {
        return;
    }

    //获取连接的客户端的ip、端口
    connectSocket->GetPeerName(clientaddr.strip, clientaddr.uiport);

    buff[nread] = _T('\0');
    CString strtemp(buff);
    POSITION pos = m_pConnectsocketlist.GetHeadPosition();
    CString temp;
}

```

```

//解析数据（进入房间、离开房间、消息）
if (strtemp.CompareNoCase(_T("enter")) == 0) {
    //有人进入房间
    nRoom++;
    SetDlgItemInt(IDC_EDIT1, nRoom);
    m_pClientAddrlist.AddTail(clientaddr);

    temp.Format(L"系统消息: %ls(%u)进入了房间!", clientaddr.strip.GetString(), clientaddr.uiport);
    while(pos!=NULL) {
        CConnectSocket* p = m_pConnectsocketlist.GetNext(pos);
        p->Send(temp, temp.GetLength() + 100);
    }

    CString allMsg;
    GetDlgItemText(IDC_EDIT2, allMsg);
    SetDlgItemText(IDC_EDIT2, allMsg + _T("\r\n") + temp); //追加消息
}

else if (strtemp.CompareNoCase(_T("leave")) == 0) {
    //有人离开房间
    nRoom--;
    SetDlgItemInt(IDC_EDIT1, nRoom);

    temp.Format(L"系统消息: %ls(%u)离开了房间!", clientaddr.strip.GetString(), clientaddr.uiport);
    while (pos != NULL) {
        CConnectSocket* p = m_pConnectsocketlist.GetNext(pos);
        p->Send(temp, temp.GetLength() + 100);
    }

    //删除该套接字和addr
    pos = m_pConnectsocketlist.GetHeadPosition();
    while (pos != NULL) {
        CConnectSocket* p = m_pConnectsocketlist.GetAt(pos);
        if (*p == *connectSocket) {
            m_pConnectsocketlist.RemoveAt(pos);
            break;
        }
        m_pConnectsocketlist.GetNext(pos);
    }

    pos = m_pClientAddrlist.GetHeadPosition();
    while (pos != NULL) {
        ClientAddr p = m_pClientAddrlist.GetAt(pos);
        if (p == clientaddr) {
            m_pClientAddrlist.RemoveAt(pos);
            break;
        }
        m_pClientAddrlist.GetNext(pos);
    }

    CString allMsg;
    GetDlgItemText(IDC_EDIT2, allMsg);
    SetDlgItemText(IDC_EDIT2, allMsg + _T("\r\n") + temp); //追加消息
}

```



```

else {
    temp.Format(L"%ls(%u): %ls", clientaddr.strip.GetString(), clientaddr.uiport, strtemp.GetString());
    while (pos != NULL) {
        CConnectSocket* p = m_pConnectSocketList.GetNext(pos);
        p->Send(temp, temp.GetLength() + 100);
    }

    CString allMsg;
    GetDlgItemText(IDC_EDIT2, allMsg);
    SetDlgItemText(IDC_EDIT2, allMsg + _T("\r\n") + temp); //追加消息
}
}

```

5. chatclientDlg

1) .h 头文件

头文件除了自动生成的 button 等函数，还有添加的 connectsocket 指针，一个标记，以及一个主要实现函数 processpendingread ()。

```

public:
    void ProcessPendingRead();
    CConnectSocket* m_pConnectSocket;
    BOOL m_bEnterRoom;
    afx_msg void OnBnClickedButtonEnter();
    afx_msg void OnBnClickedButtonLeave();
    afx_msg void OnBnClickedButtonSend();
    afx_msg void OnEnChangeEditMessage();
    virtual BOOL DestroyWindow();
};

```

2) .cpp 实现文件

首先是初始化，最开始加载窗口时，设置服务器端口和 ip 地址，此时只有加入房间按钮可用。

```

// TODO: 在此添加额外的初始化代码
SetDlgItemText(IDC_EDIT_SERVERIP, _T("127.0.0.1"));
SetDlgItemInt(IDC_EDIT_SERVERPORT, 8080);
GetDlgItem(IDC_BUTTON_ENTER)->EnableWindow(TRUE); //初始化只有加入房间可用
GetDlgItem(IDC_BUTTON_LEAVE)->EnableWindow(FALSE);
GetDlgItem(IDC_BUTTON_SEND)->EnableWindow(FALSE);
((CEdit*)GetDlgItem(IDC_EDIT_SENDMESSAGE))->SetReadOnly(TRUE);

```

加入房间，创建套接字、获取本地 ip 和端口号，连接到服务器，连接成功后，加入房间按钮变灰，发送消息和离开房间按钮可用

```

void CchatclientDlg::OnBnClickedButtonEnter()
{
    // TODO: 在此添加控件通知处理程序代码

    //创建TCP套接字并绑定一个系统分配的端口号
    m_pConnectSocket = new CConnectSocket(this);
    m_pConnectSocket->Create();

    //获取本地ip、端口号
    CString strlocalip;
    UINT uilocalport;
    m_pConnectSocket->GetSockName(strlocalip, uilocalport);
    SetDlgItemText(IDC_EDIT_LOCALIP, strlocalip); //显示到窗口
    SetDlgItemInt(IDC_EDIT_LOCALPORT, uilocalport);

    //调用connect函数，连接到服务器
    CString strserverip;
    UINT uiserverport;
    GetDlgItemText(IDC_EDIT_SERVERIP, strserverip);
    uiserverport = GetDlgItemInt(IDC_EDIT_SERVERPORT);

    if (!m_pConnectSocket->Connect(strserverip, uiserverport)) {
        MessageBox(_T("无法连接服务器"));
        return ;
    }

    CString strEnterMsg("enter");
    m_pConnectSocket->Send(strEnterMsg, strEnterMsg.GetLength() + 100);

    GetDlgItem(IDC_BUTTON_ENTER)->EnableWindow(FALSE); //进入房间后
    GetDlgItem(IDC_BUTTON_LEAVE)->EnableWindow(TRUE);
    GetDlgItem(IDC_BUTTON_SEND)->EnableWindow(TRUE);
    ((CEdit*)GetDlgItem(IDC_EDIT_SERVERIP))->SetReadOnly(TRUE);
    ((CEdit*)GetDlgItem(IDC_EDIT_SERVERPORT))->SetReadOnly(TRUE);
    ((CEdit*)GetDlgItem(IDC_EDIT_SENDDMESSAGE))->SetReadOnly(FALSE);

    m_bEnterRoom = TRUE;
}

```

接收消息并显示在窗口上


```

void CchatclientDlg::ProcessPendingRead() {
    TCHAR buff[4096];
    int nread = m_pConnectSocket->Receive(buff, 4096);
    if (nread == SOCKET_ERROR) {
        return;
    }

    buff[nread] = _T('\0');
    CString chatMsg(buff);

    CString allMsg;
    GetDlgItemText(IDC_EDIT_MESSAGE, allMsg);
    SetDlgItemText(IDC_EDIT_MESSAGE, allMsg + _T("\r\n") + chatMsg); //追加消息
}

```

发送消息

```

void CchatclientDlg::OnBnClickedButtonSend()
{
    // TODO: 在此添加控件通知处理程序代码
    CString strMsg;
    GetDlgItemText(IDC_EDIT_SENDMESSAGE, strMsg);
    m_pConnectSocket->Send(strMsg, strMsg.GetLength() + 100);
    SetDlgItemText(IDC_EDIT_SENDMESSAGE, _T(""));
}

```

离开房间

```

void CchatclientDlg::OnBnClickedButtonLeave()
{
    // TODO: 在此添加控件通知处理程序代码
    CString strLeaveMsg("leave");
    m_pConnectSocket->Send(strLeaveMsg, strLeaveMsg.GetLength() + 100);

    GetDlgItem(IDC_BUTTON_ENTER)->EnableWindow(TRUE); //离开房间后重回初始化状态
    GetDlgItem(IDC_BUTTON_LEAVE)->EnableWindow(FALSE);
    GetDlgItem(IDC_BUTTON_SEND)->EnableWindow(FALSE);
    ((CEdit*)GetDlgItem(IDC_EDIT_SERVERIP))->SetReadOnly(FALSE);
    ((CEdit*)GetDlgItem(IDC_EDIT_SERVERPORT))->SetReadOnly(FALSE);
    ((CEdit*)GetDlgItem(IDC_EDIT_SENDMESSAGE))->SetReadOnly(TRUE);

    m_bEnterRoom = FALSE;
}

```

若在房间时关闭窗口，也调用一次离开房间函数，向服务器发送离开消息

```

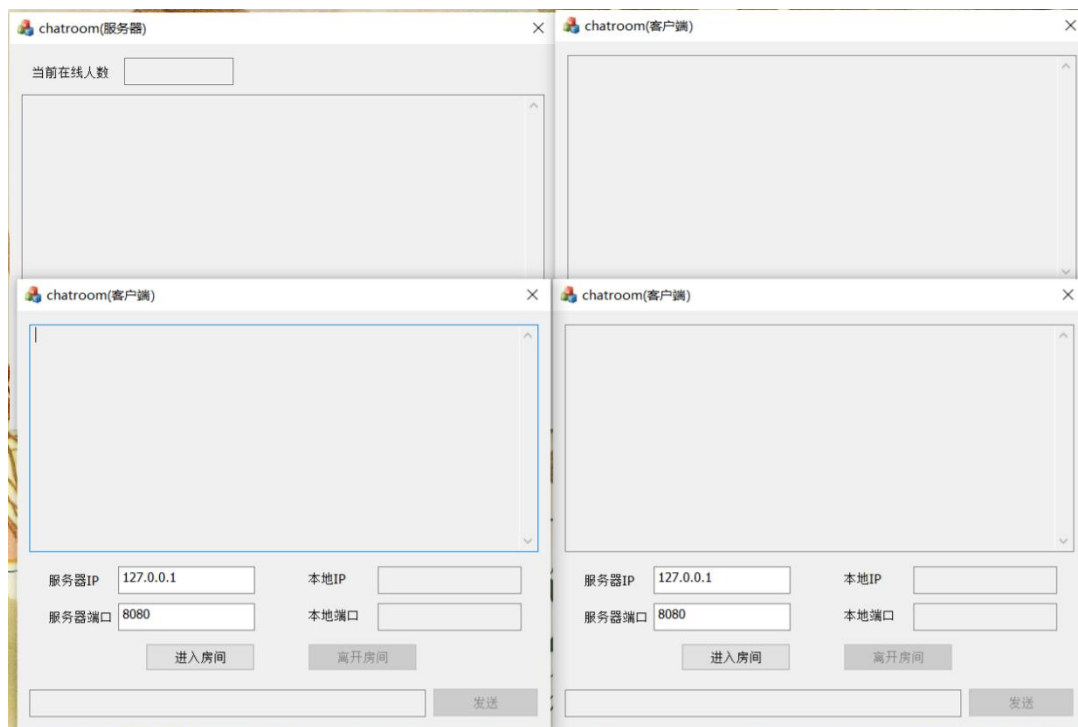
BOOL CchatclientDlg::DestroyWindow()
{
    // TODO: 在此添加专用代码和/或调用基类
    if (m_bEnterRoom) {
        OnBnClickedButtonLeave();
    }
    return CDialogEx::DestroyWindow();
}

```

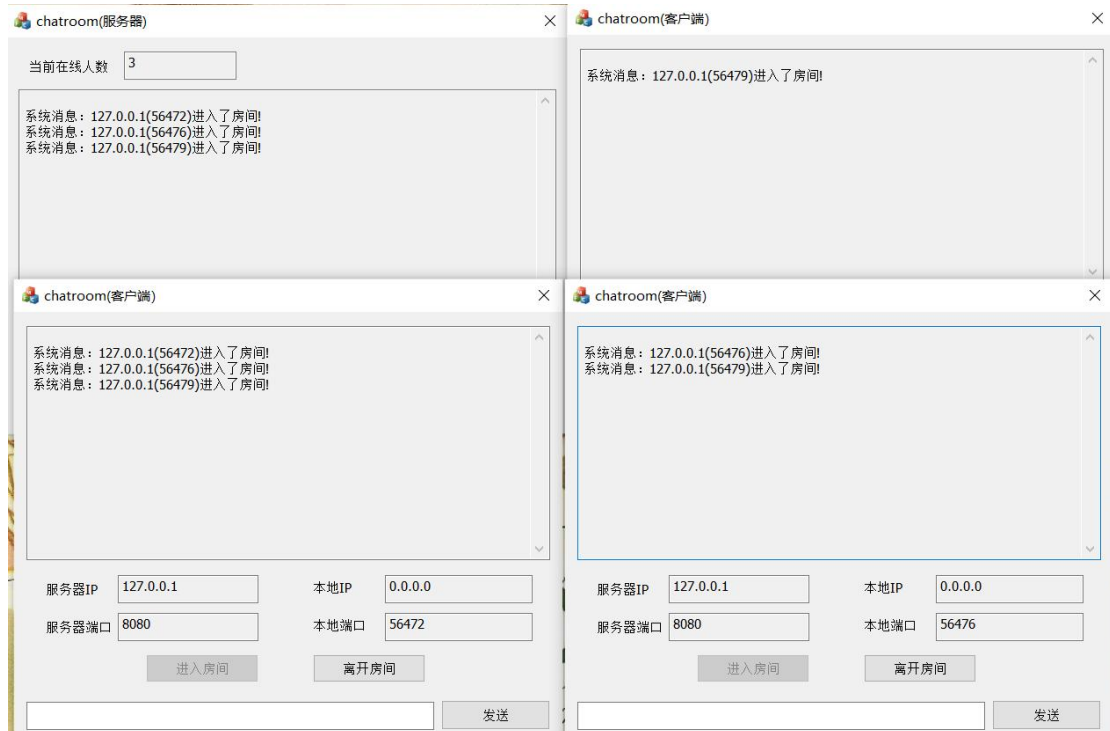
结论分析：

运行结果展示，以三人群聊为例：

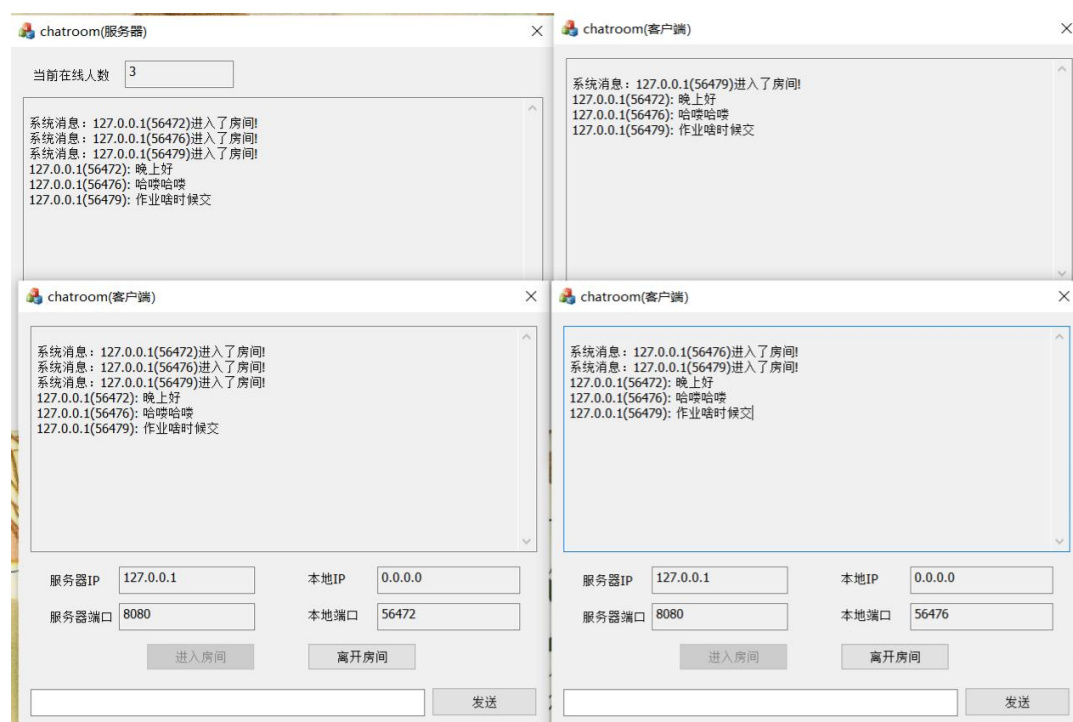
初始状态：可以看到客户端只有进入房间可点击



进入房间，分配本地端口：群聊不能显示某人进入之间的聊天消息，服务器显示的是自服务器启动的所有消息。进入房间后，可以离开房间或发送消息。服务器显示当前群聊人数

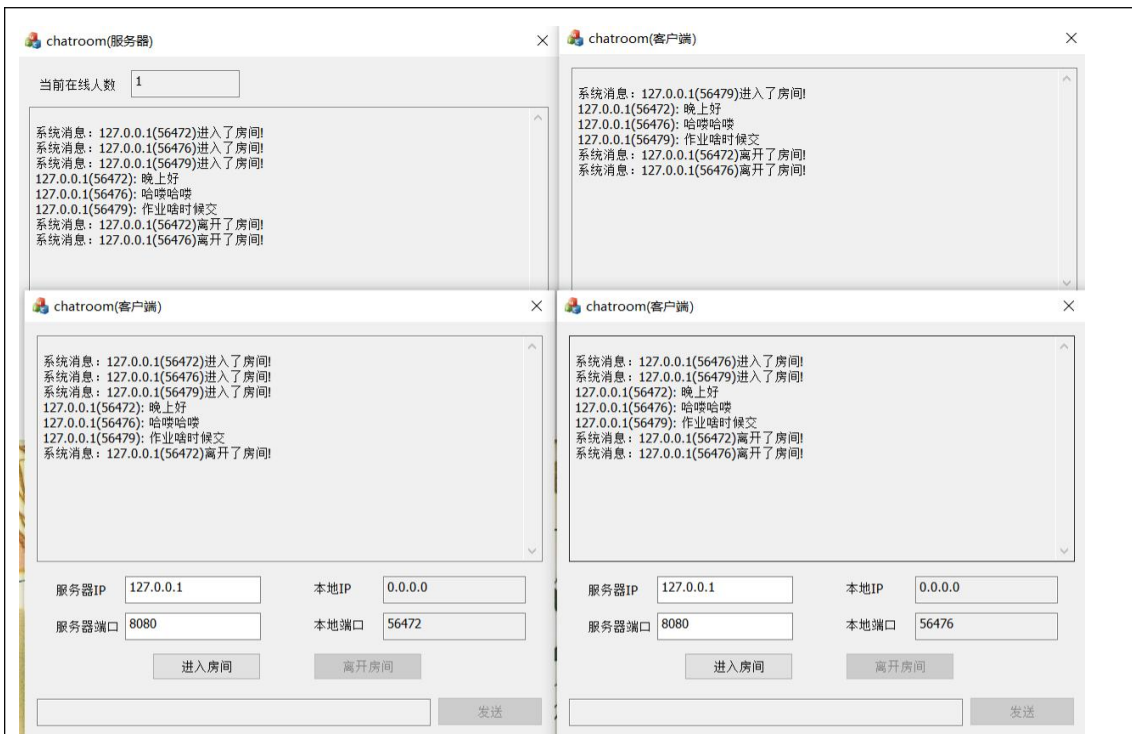


聊天展示

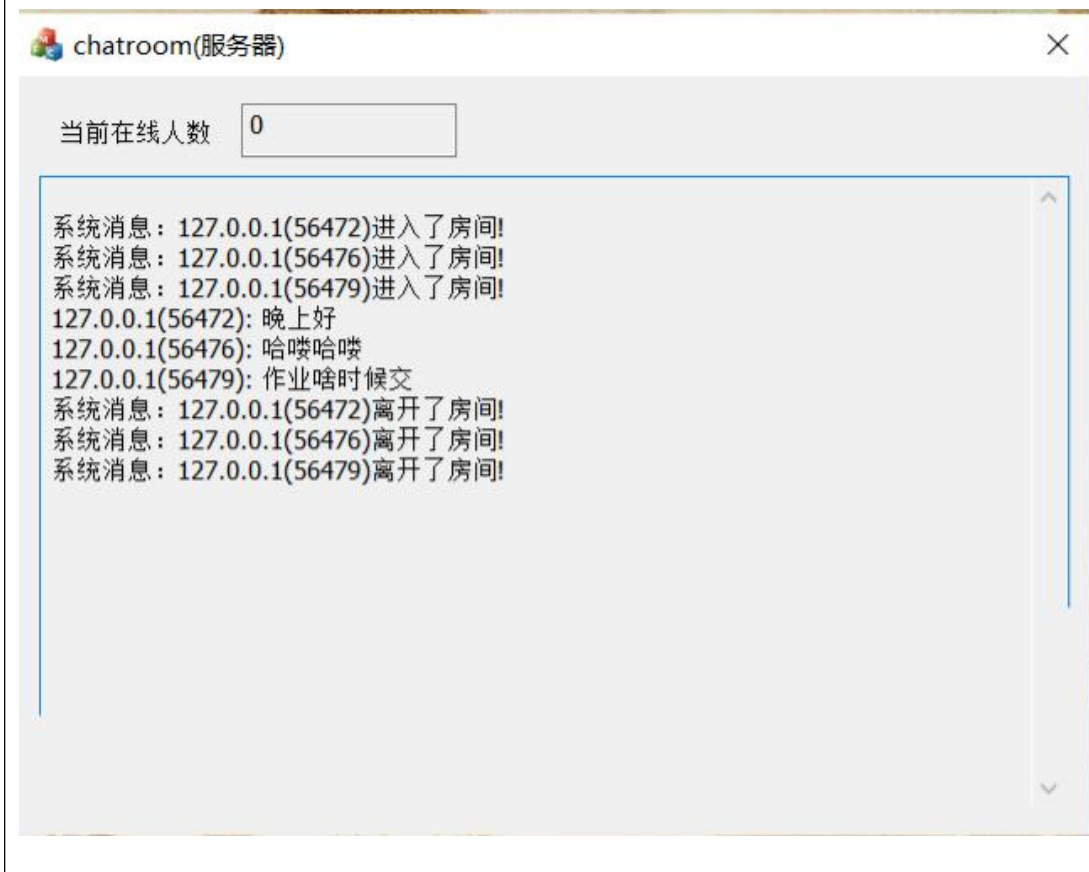


依次离开房间后，在线人数改变





最后一个没有点击离开房间，而是直接关闭窗口，效果也同离开房间



总结与改进：

主要使用了 MFC 框架进行 TCP 聊天室的开发。在模拟 TCP 传输过程中，通能够实现数据的可靠传输，减少数据丢失和乱序的问题。MFC 框架提供了丰富的界面设计工具，可以帮助我们快速搭建出美观、易用的网络聊天室界面，可以根据需求在框架内填写相应操作即可实现。

改进：实现登录功能，这样聊天的时候可以显示昵称而不是 ip+端口号，添加在线成员展示，可以对聊天室成员进行私聊。