# 实 验 报 告

## 实验名称： Socket通信编程实验

## 课程名称： 计算机网络实验

院系： 人工智能学院

组员： 胡钧淇、梁宇阳、刘若飞、黄一鸣

组长： 胡钧淇

日期： 2024.10.26

**中山大学**

1. 实验内容
2. TCP套接字编程：一个客户端与一个服务器之间能进行简单的文字通信
3. 基于TCP的聊天程序设计：
4. 实现一个具有一定功能的聊天程序
5. 可视化聊天程序
6. 实验分工

胡钧淇：TCP套接字编程、多人聊天、文字消息发送，实验报告编写

梁宇阳：TCP套接字编程、多人聊天、文字消息发送，实验报告编写

刘若飞：可视化聊天程序、文件传输、多人聊天、客户登陆、文字消息发送

黄一鸣：文件传输、文字消息发送

1. 实验环境

一台 PC 机（使用 Windows 10 系统，能够访问网络）。

在PC 中配置 C++、MinGW、Cmake、QT，在这基础上完成一个具有一定功能的可视化聊天程序

1. 实验步骤
2. TCP套接字编程：实现客户端与服务器之间的基本通信功能

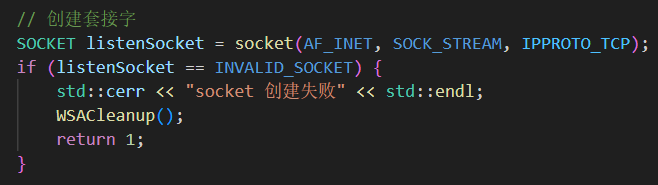
服务端：

1. 初始化winsock：



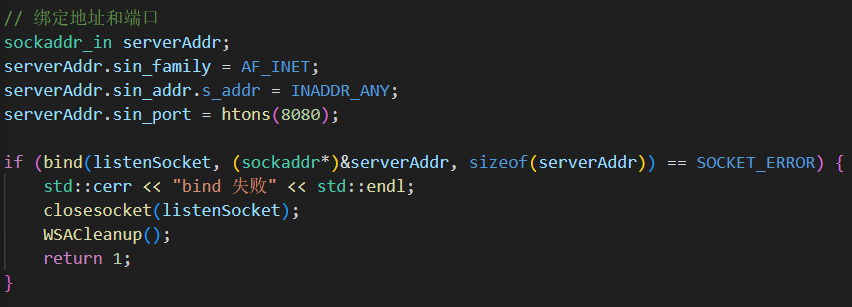
MAKEWORD(2,2)表示使用WINSOCK2版本.wsaData用来存储系统传回的关于WINSOCK的资料加载WINSOCK库

1. 创建服务器套接字：



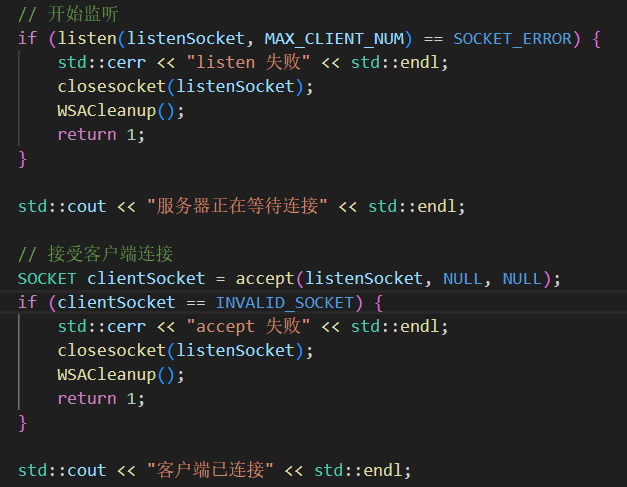
通过socket()函数创建一个套接字实体，AF\_INET表示使用IPv4 地址族，SOCK\_STREAM表示选择TCP通信方式。

接着用bind()函数将套接字与服务端的地址与端口进行绑定：



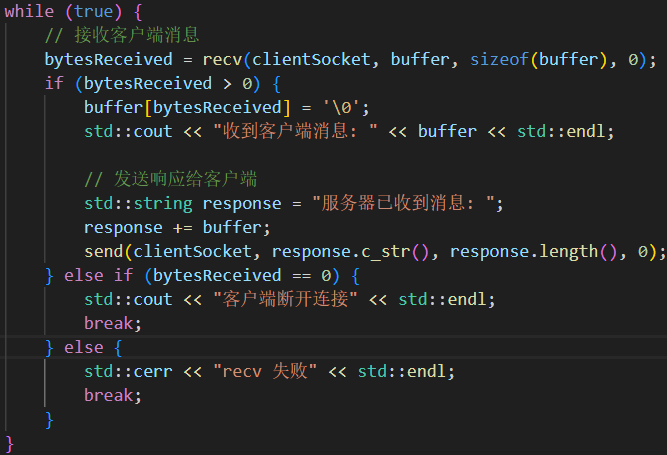
其中INADDR\_ANY表示IP地址可随意配置

1. 监听与接收客户端连接：



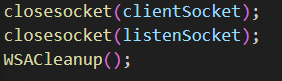
使用listen()函数指定服务端套接字进行被动连接，指定最大连接数目MAX\_CLIENT\_NUM当有客户端请求连接，使用accept()函数接受，并返回一个用于与该客户端通信的套接字。

1. 与客户端通信：



不断使用rev()函数接受客户端传来的信息，显示在终端，再使用send()函数给客户端发送信息告诉客户端收到该消息。

1. 关闭程序

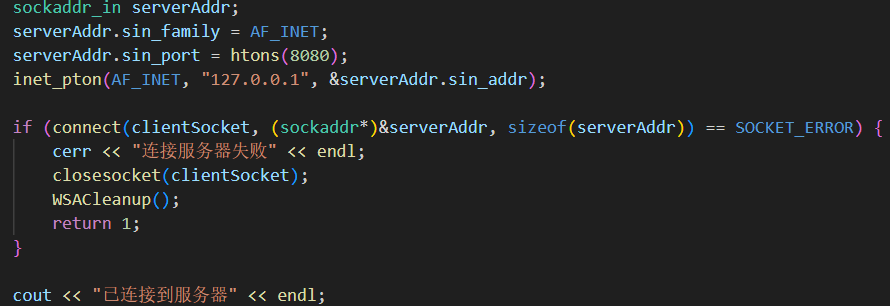


将建立的套接字关闭并解除与winsock动态链接库的绑定，释放资源。

客户端：

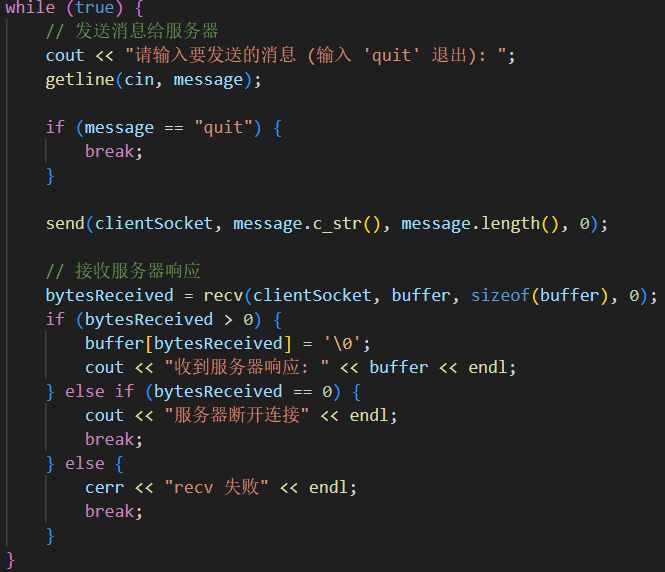
客户端的winsock初始化，套接字创建，套接字关闭与服务端相同，以下是不同的部分

1. 连接服务端



套接字创建好后，使用connect()函数输入客户端套接字，指定服务端ip向服务端请求连接，连接成功后使用该套接字与服务端进行通讯。

2．与服务器通信



不断使用send()函数将getline在终端读取到的输入发送给服务端，并使用recv()接受服务端的响应。

遇到的困难与解决方法：

1. 可能找不到lws2\_32库

解决： 在命令行中加入“-lws2\_32”即可， 或在配置文件中参数部分加入“-lws2\_32”

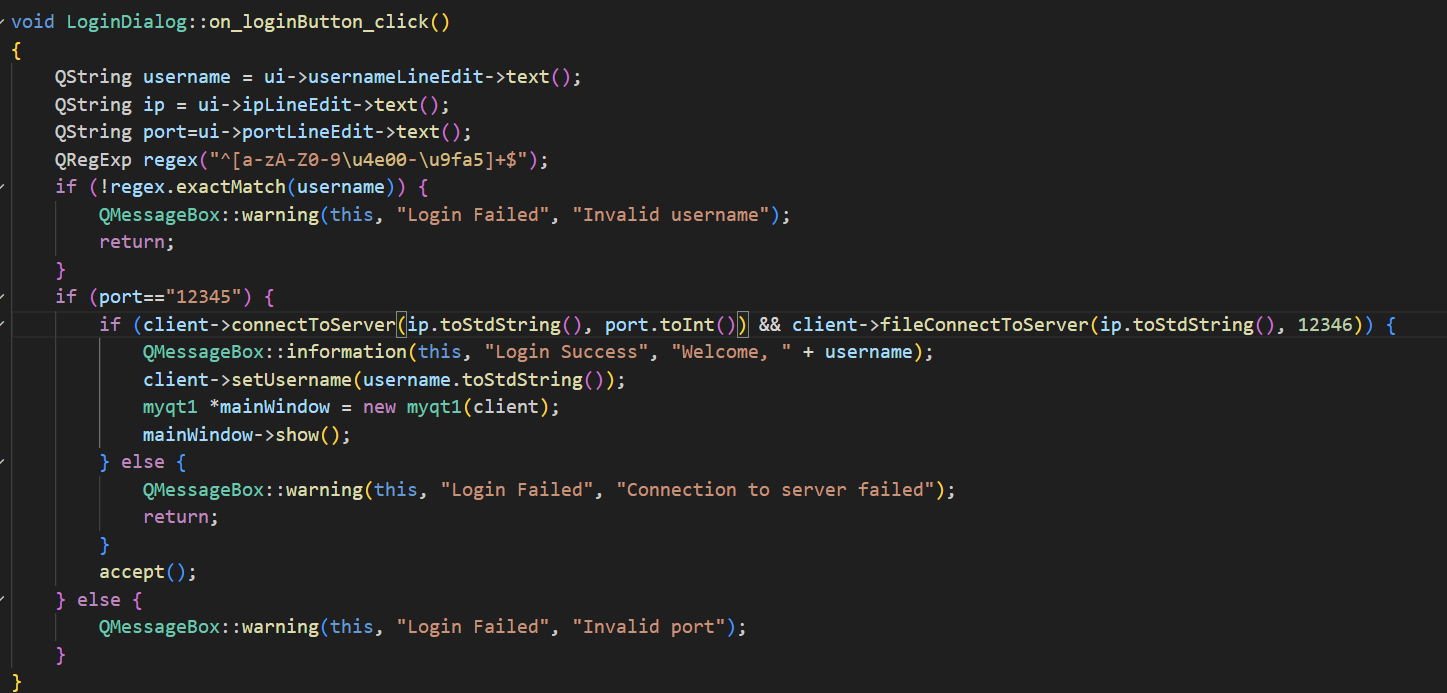
2. 通讯使用中文可能出现乱码

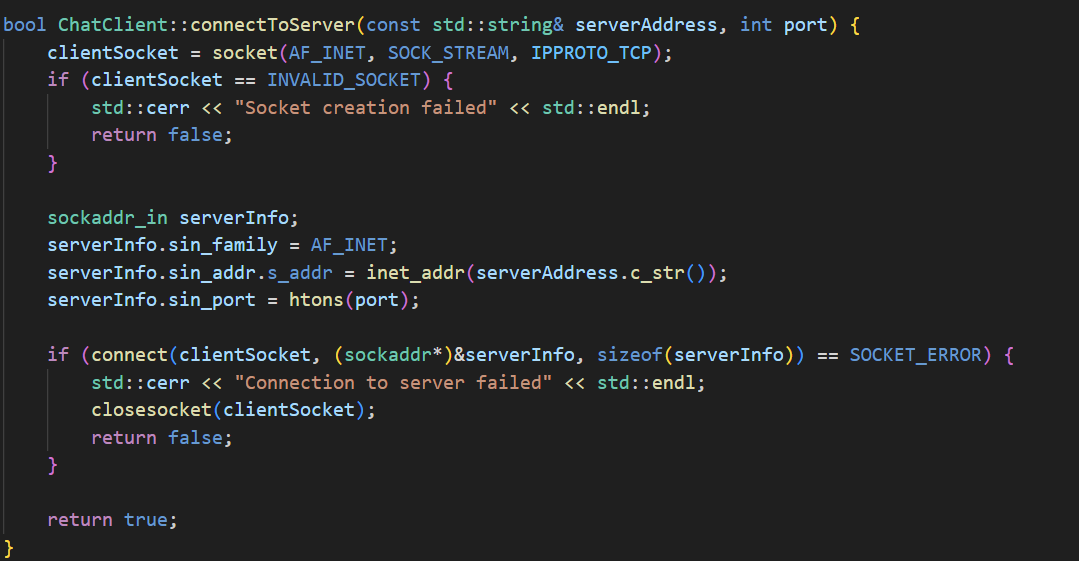
解决： 在客户端，服务端代码中使用SetConsoleCP(936)和SetConsoleOutputCP(936)将控制台编码统一为GBK即可。或在客户端，服务端终端运行前使输入chcp = 936统一编码即可。

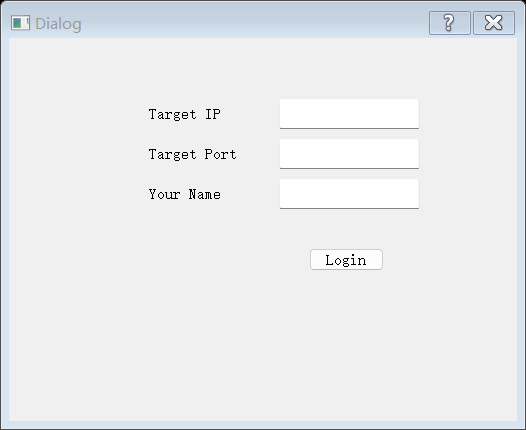
二、基于TCP的聊天程序设计：实现一个具有一定功能的聊天程序并可视化

1.客户登陆：我们利用qt设计了一个登入窗口让用户键入要连接的服务器IP地址和端口，以及自己登入服务器的用户名，先判断是否输入正确的IP地址与端口并与服务器建立，然后将用户名发送给服务器：

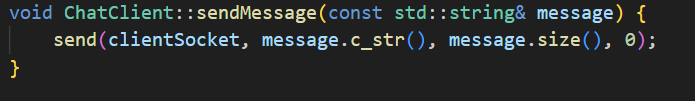
客户端：



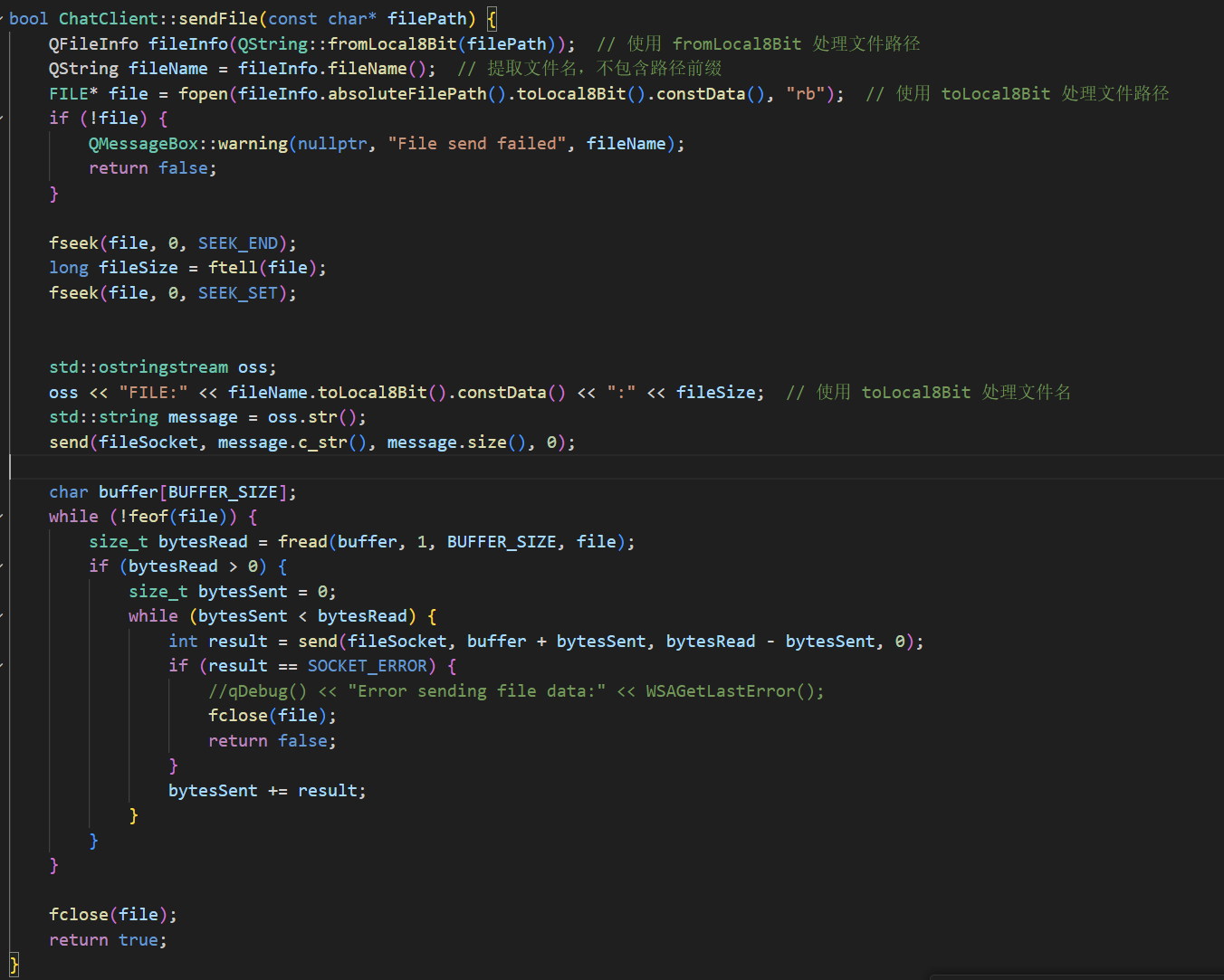




2.文字消息发送：在上个部分的实验中已经实现，通过send函数将文字消息发给服务器



为了完成下面部分，我们还添加了文件发送：

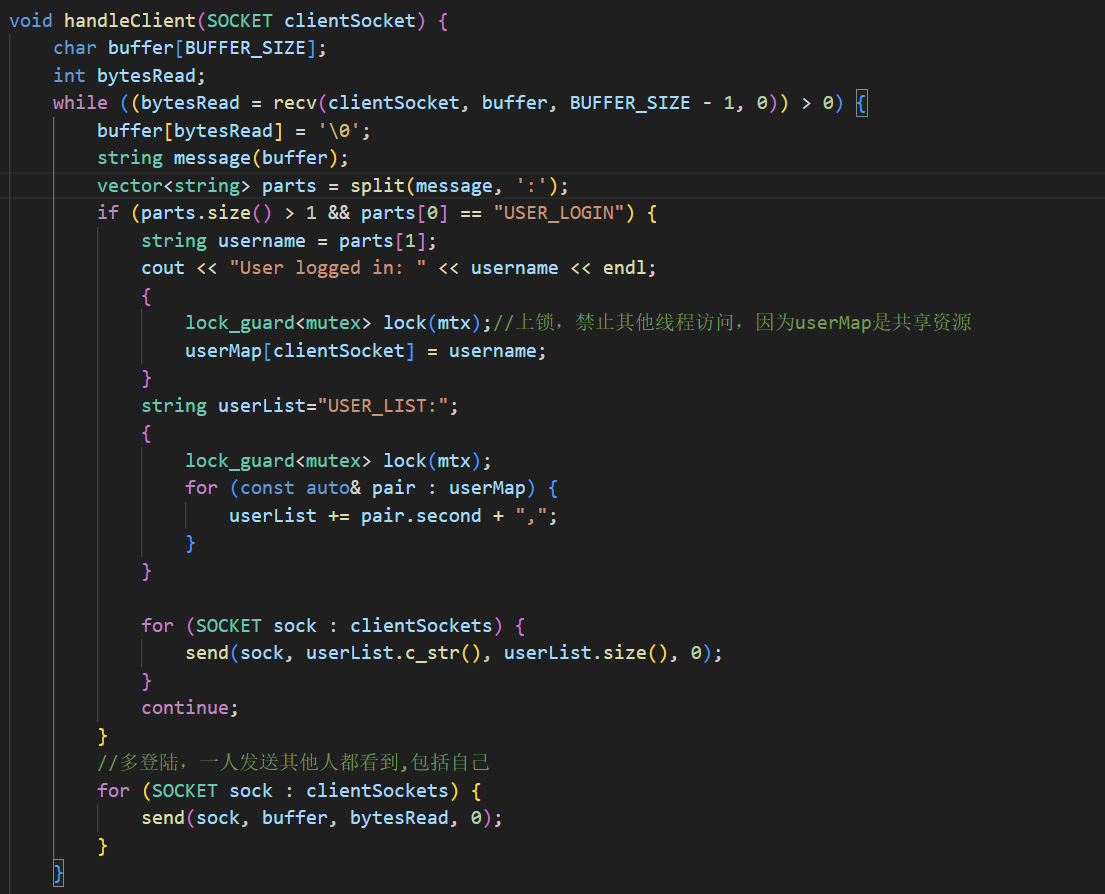


3.多人聊天：我们利用多线程来实现多人聊天，首先我们会用一个数组来存放当前服务器在线用户，而每当有新用户进入服务器，我们会给新用户分配一个线程来实现服务器与该用户之间的通信，而且通过这个线程，我们实时更新每个在线用户聊天窗口中的用户列表，并通过服务器分配的线程对每个在线用户广播该用户所发送的消息，这样在每个在线用户的聊天窗口中就可以同时看到彼此之间发送的消息与在线用户列表了。

服务器接受新用户：



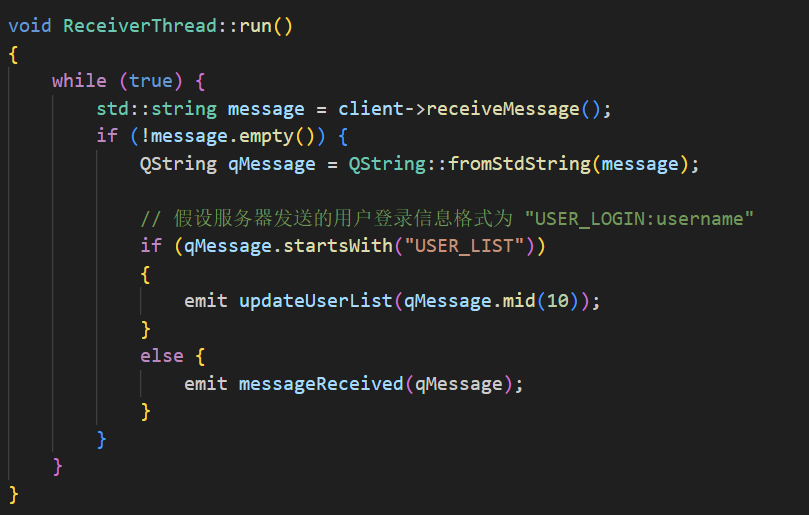
服务器与用户间通信的线程函数handleClient：



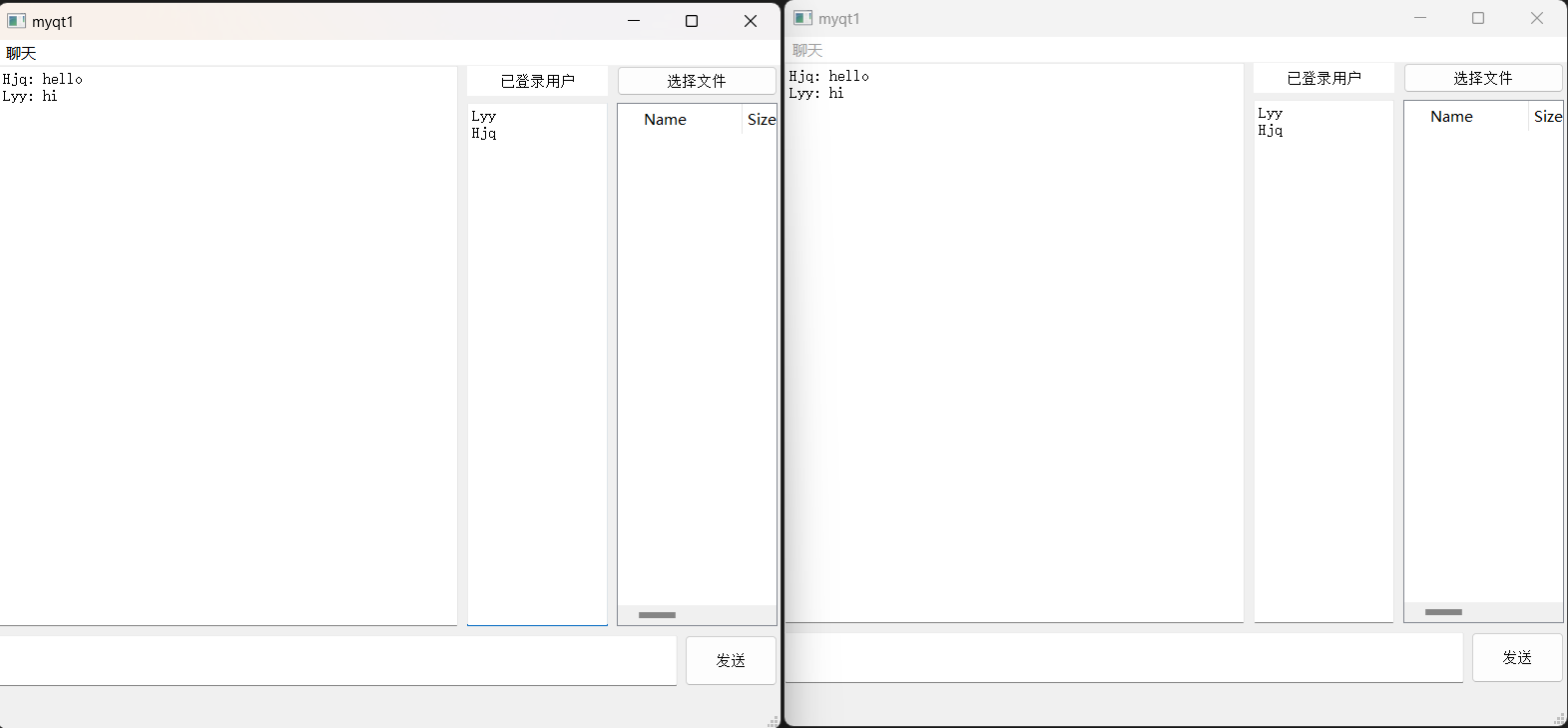
不仅如此，我们还设置了用户登出后的客户端更新(同样在handleClient函数下)，每当有用户登出服务器时，该线程会更新在线用户列表，并关闭与该用户的通信连接：



同时为了保证在线用户们能同时完成消息的发送与接收，对于每个用户的接收我们同样也用线程去实现：

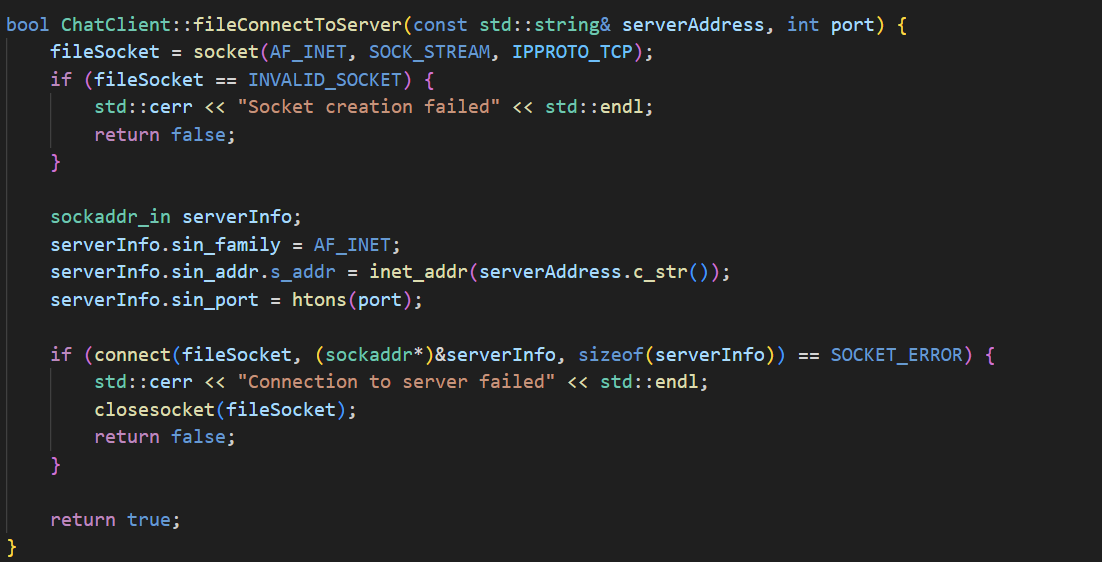


结果(同样我们用Qt可视化了一个聊天窗口，里面包含在线用户列表以及聊天区)：



4. 文件传输：同样的，每个新用户进入服务器时，服务器与用户端之间也会建立起一个文件通信连接，类似于(1)中的用户连接connectToServer，我们还建立了一个用户的文件通信连接fileConnectToServer，通过该连接，每个用户都可以在用户端上传本地的文件给服务器，而服务器会保存上传的文件在自己的主机中，并把已上传的文件广播给每个在线用户，这样在线用户们就可以通过服务器进行下载，将已上传的文件通过服务器保存到自己的本地中。

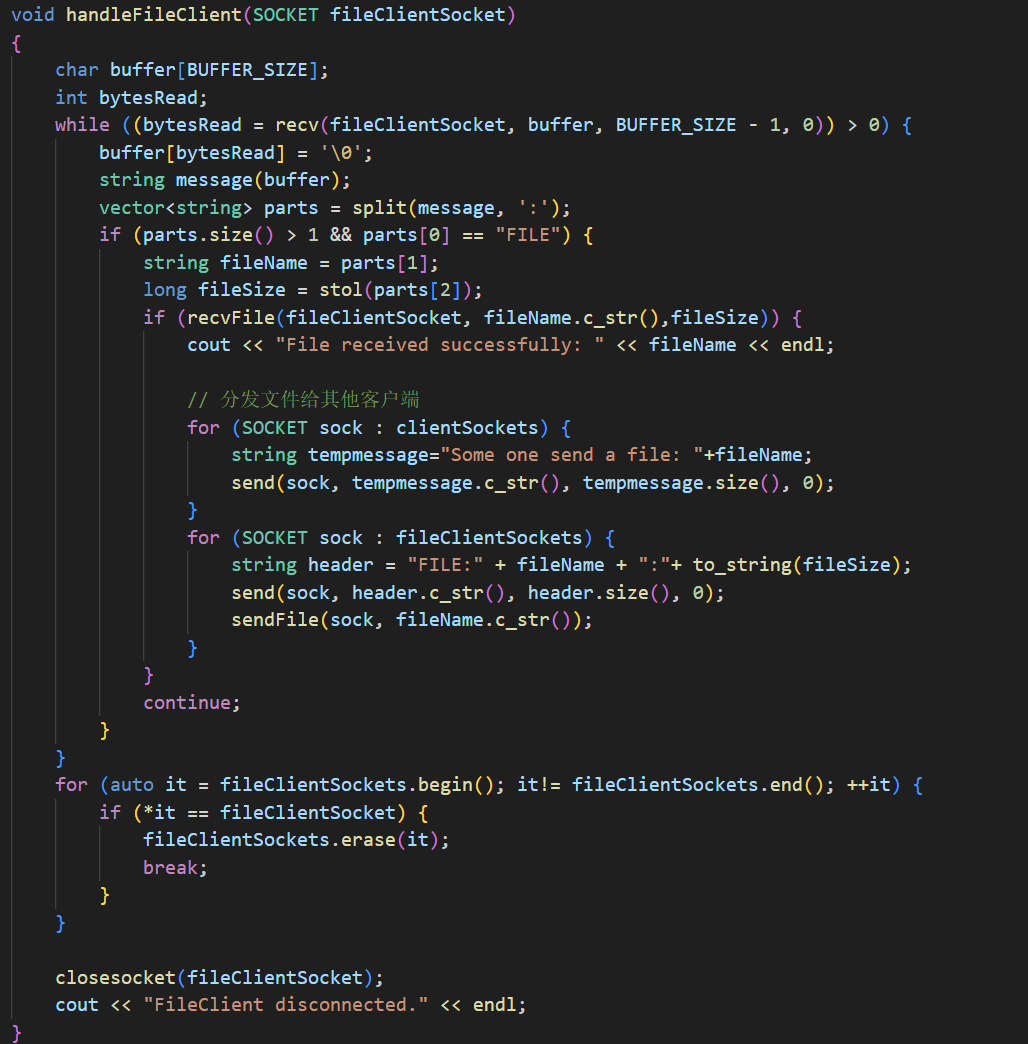
文件通信连接：



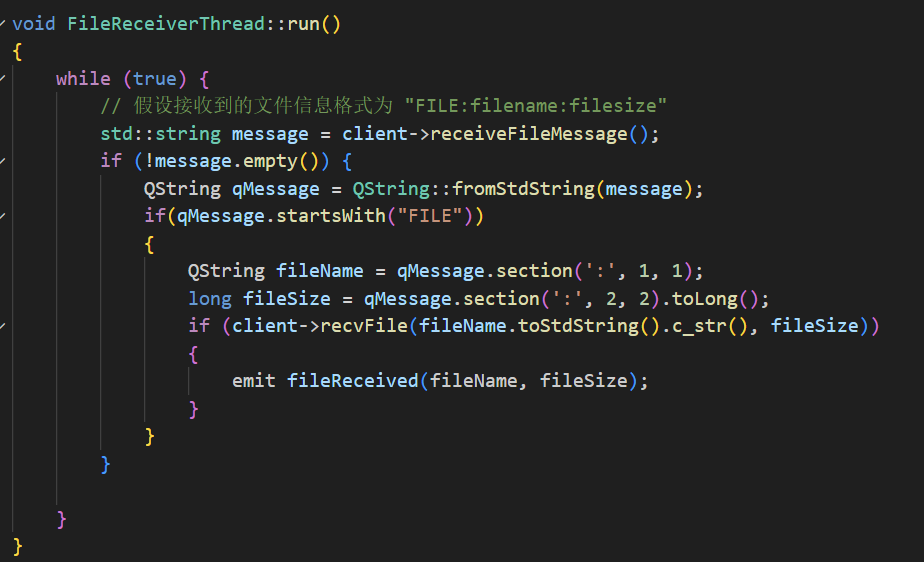
服务器接受文件通信连接：



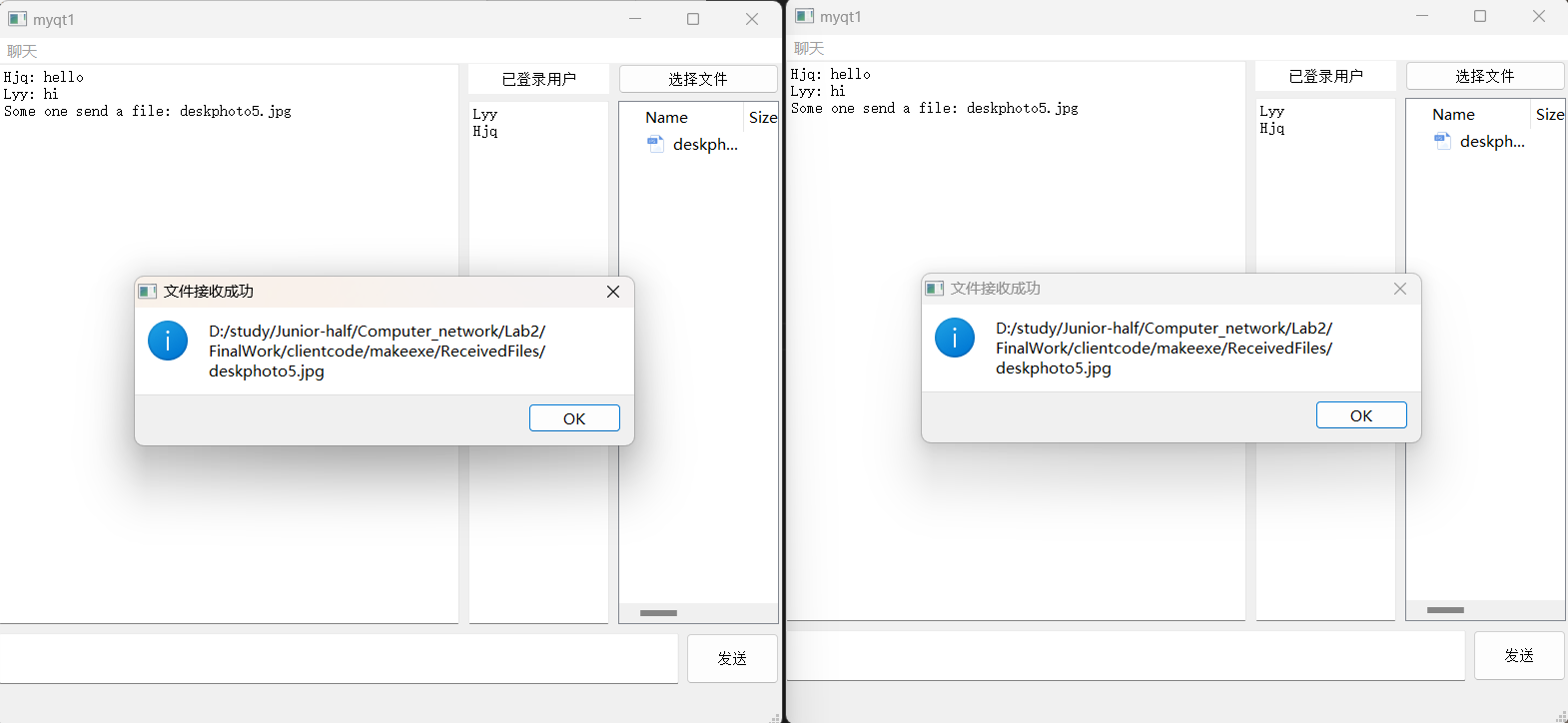
服务器与用户间通信的线程函数handleFileClient：



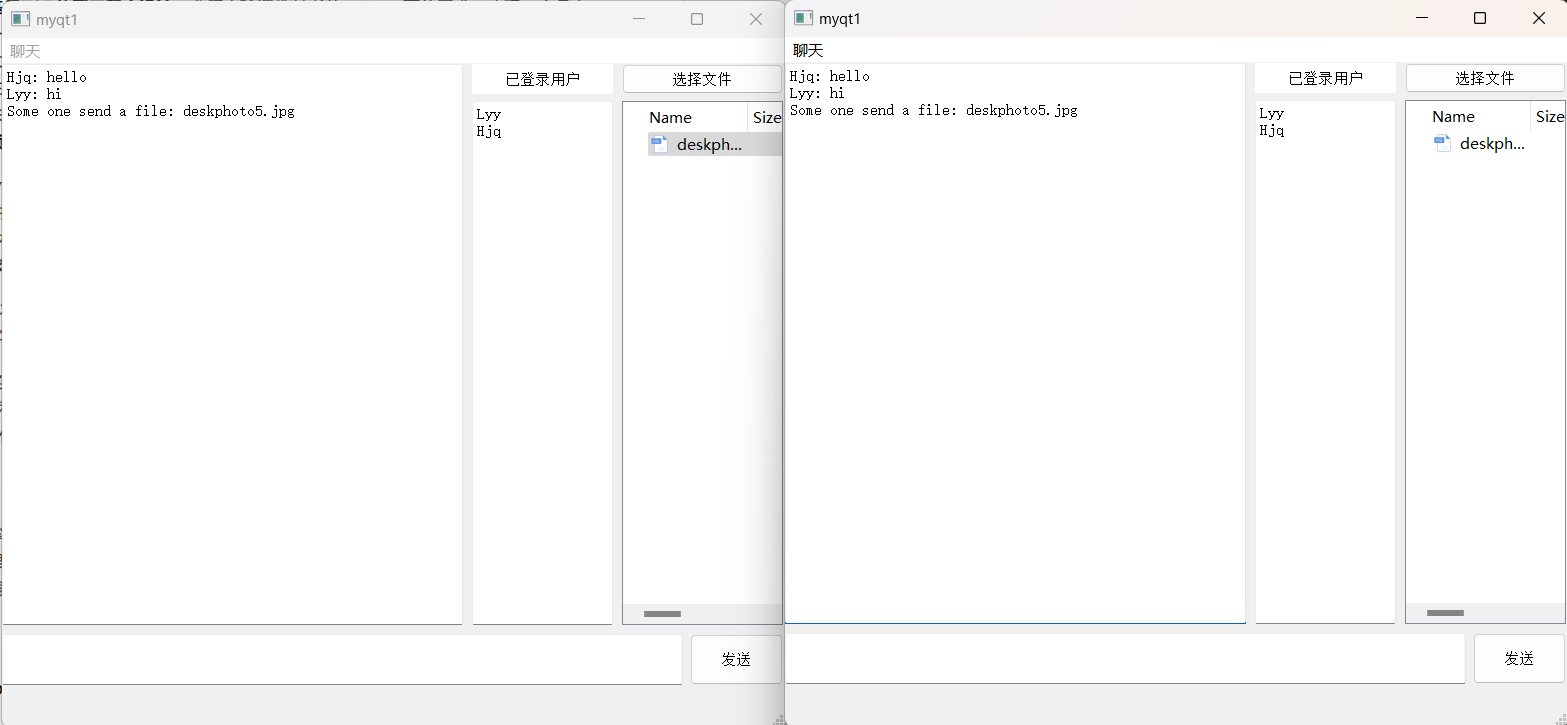
同样，为了保证在线用户们能同时完成文件的发送与接收，对于每个用户的文件接收我们也用线程去实现：



结果：



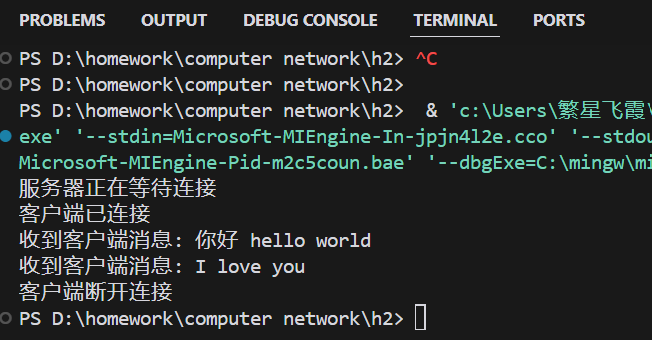
可以看到在聊天窗口的右侧已经保存了我们用户端本地上传的文件：



四、 实验结果与分析

1. 我们在自己电脑上配置了服务器和用户端，并进行了相互之前的基本通讯

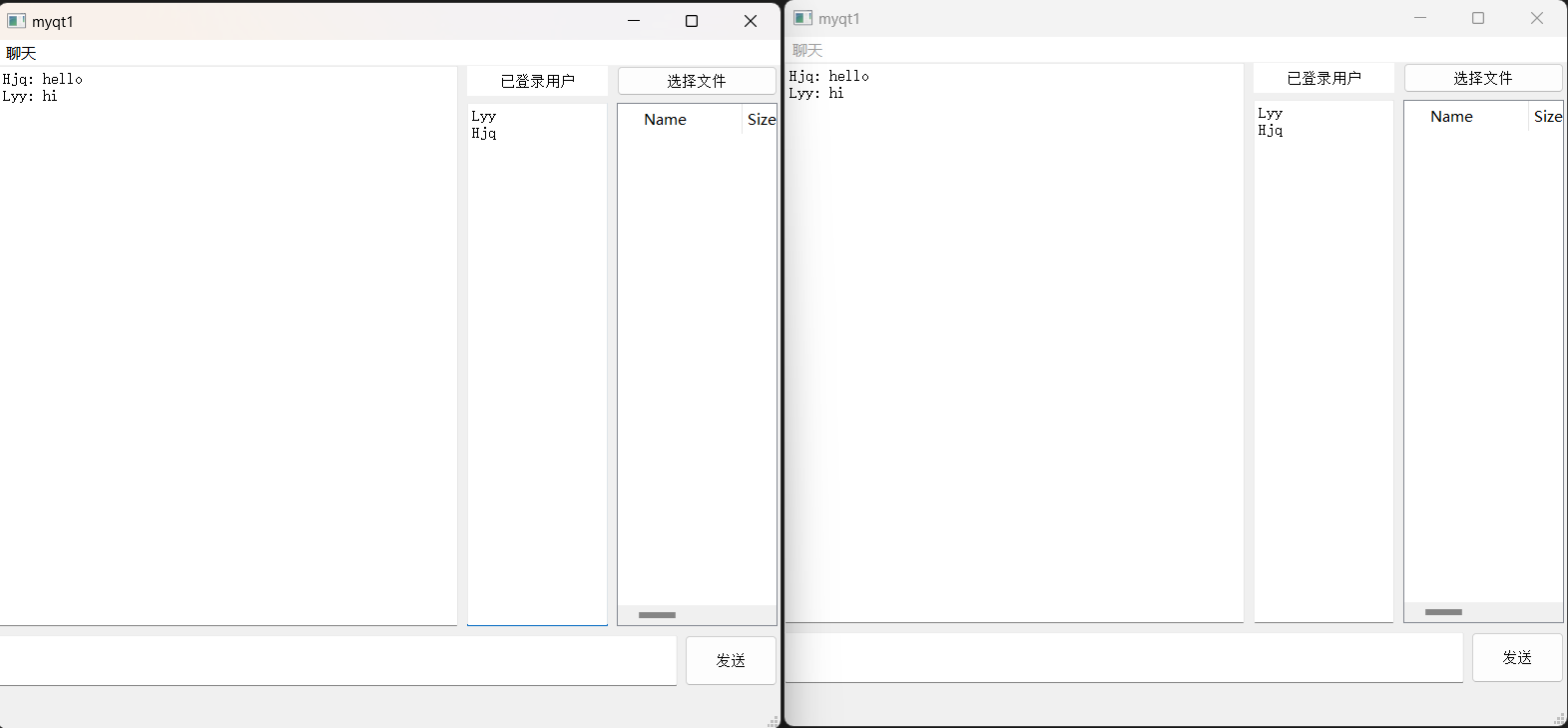
服务器通信记录：



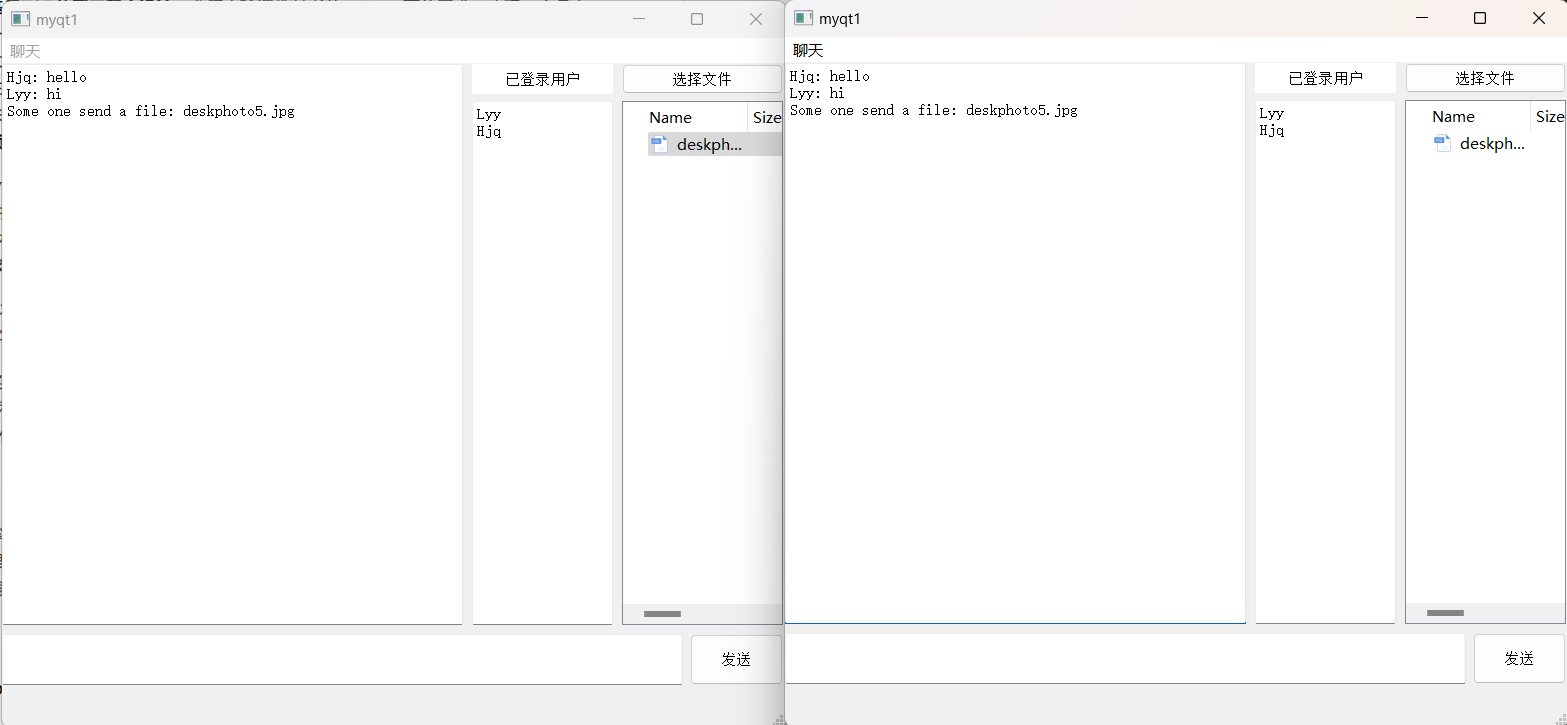
客户端通信记录：

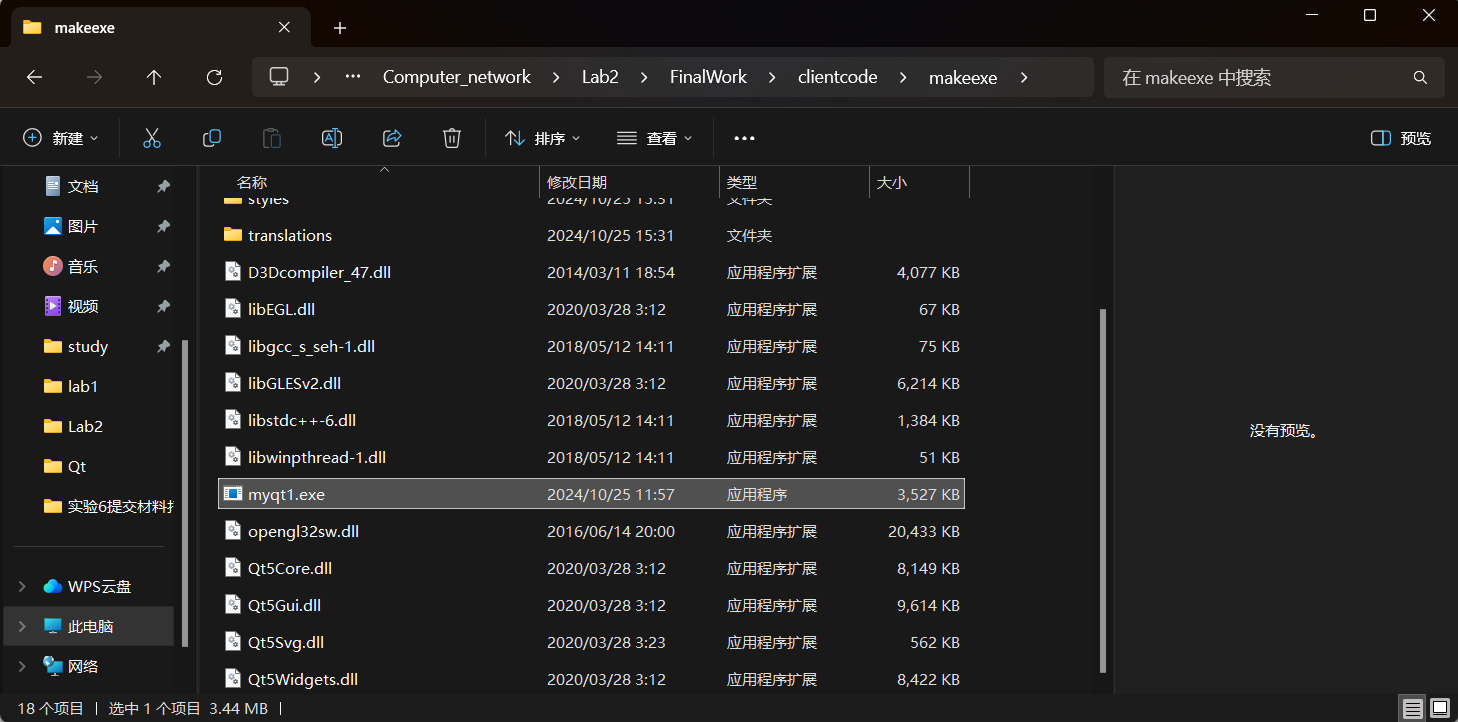


1. 在此基础上设计自己的多人同时在线聊天程序，并成功在本地访问



1. 然后我们可视化该聊天过程，设计出一个可用户登陆、多人聊天、文件传输、文字消息发送的聊天窗口，并生成了一个可运行的.exe





1. 我们将服务器程序配置在了具有公网IP的Win10系统上，任何Windows用户可以通过运行我们的exe文件，并输入相应IP地址以及端口，进行服务器的访问与互联，并实现上述多人聊天以及文件传输的功能。
2. 实验总结
3. 通过这次实验，我充分了解了TCP协议下客服端与服务器之间是如何进行通信的。
4. 而且了解了TCP协议下报文的发送和接收，对TCP协议的通信结构有了更充分的认知。
5. 不仅如此，还尝试利用Qt库对我们的多人聊天程序进行可视化，并且实现了多种聊天功能。