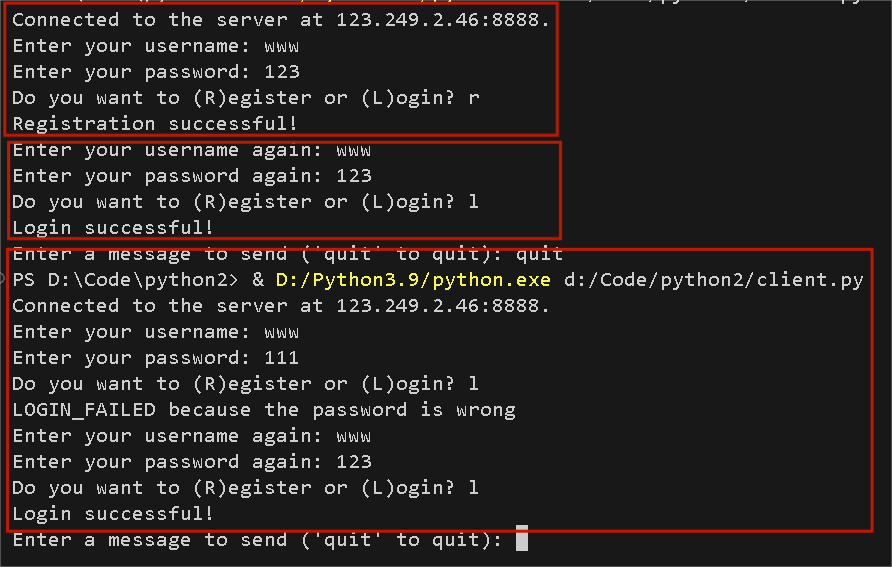
8.2.2 实现一个简单的聊天程序

一、说明完成的基本功能的情况，并充分测试，贴图说明。

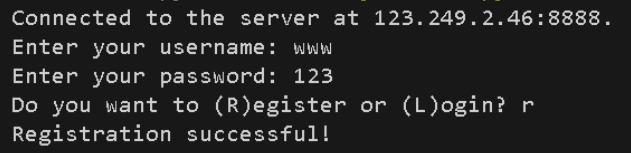
**基本功能测试：**

1. 验证用户登录；—— 只验证用户名、密码，正确的通过验证，不正确的不能通过验证。不要求有用户注册功能（不要贴图展示注册功能）。

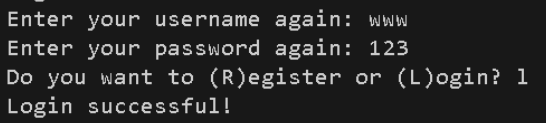
这是一个总体的新客户端连接服务器，退出然后重新登录的过程



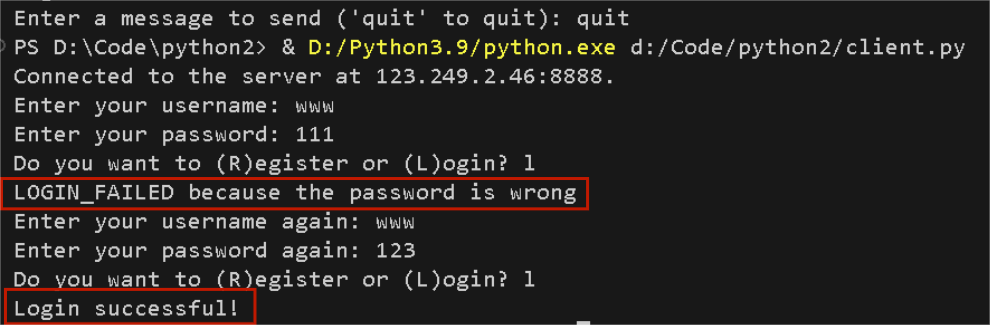
过程分为三个小步骤，首先新用户连接服务器需要注册账号



这里注册新用户www，密码123，注册完成后需要重新登录

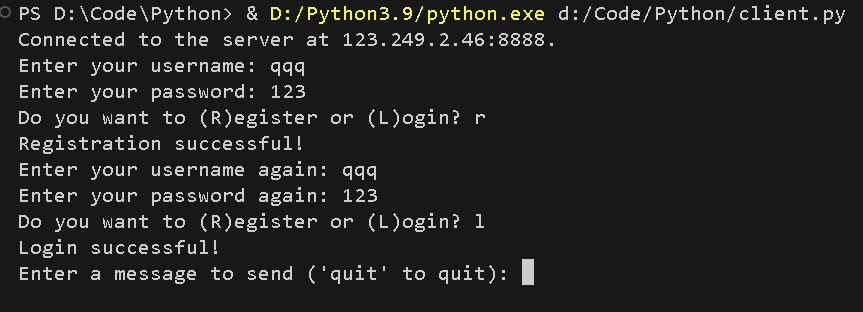


重新登录这里密码正确，登录成功。然后退出账号重新登录



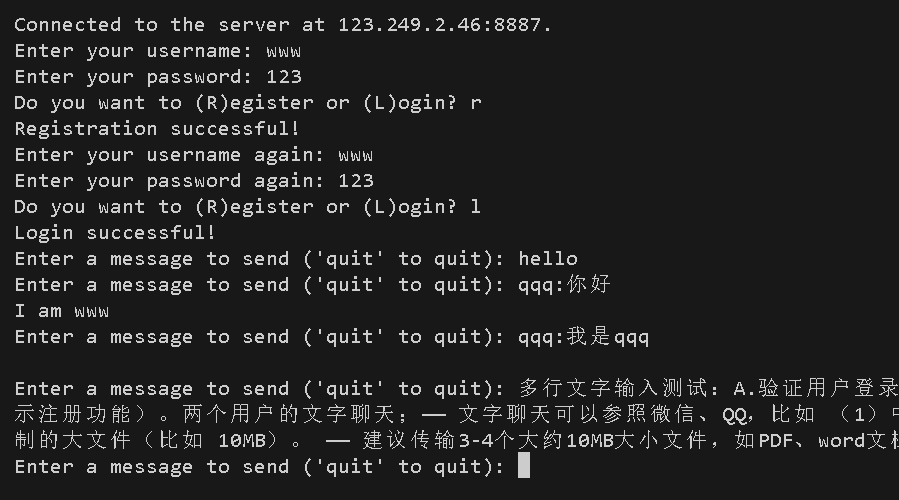
这里输入了错误的密码111，所以登录失败，然后再用正确的密码123登录，登录成功。

下面是另一个客户端新用户qqq登录过程

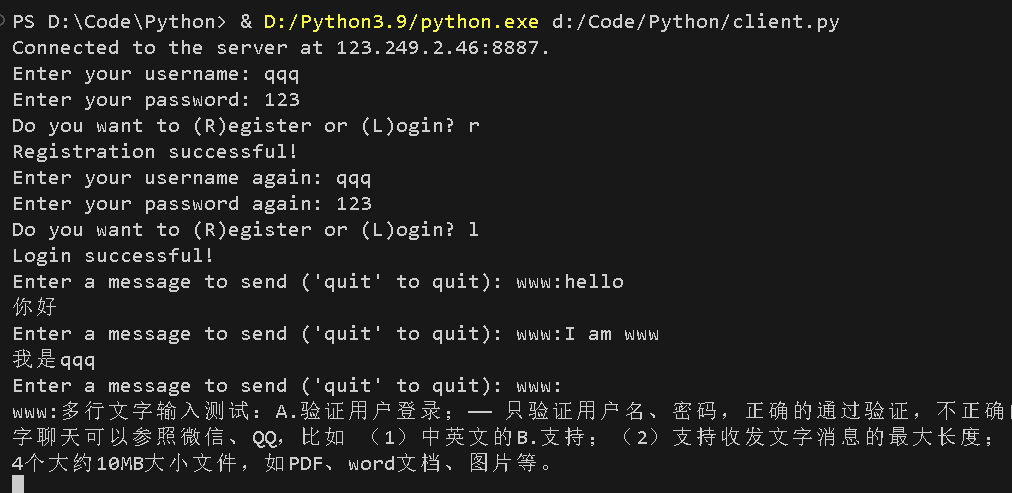


1. 两个用户的文字聊天；—— 文字聊天可以参照微信、QQ，比如 （1）中英文的支持；（2）支持收发文字消息的最大长度； （3）支持多行文字吗？

用户www界面

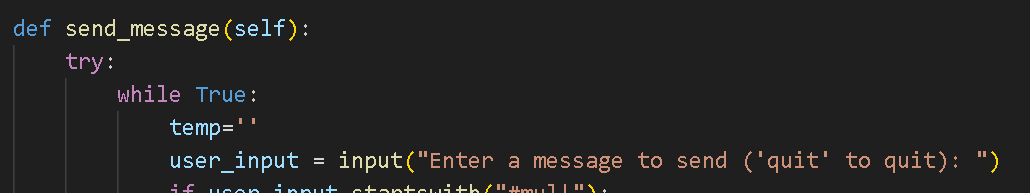


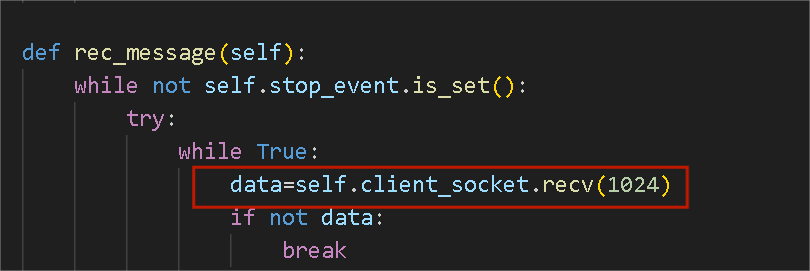
另一个用户qqq界面



通过两个客户端的聊天界面可以看到该聊天程序两边互相能接收到消息，消息支持中英文输入，而且支持多行文字输入。

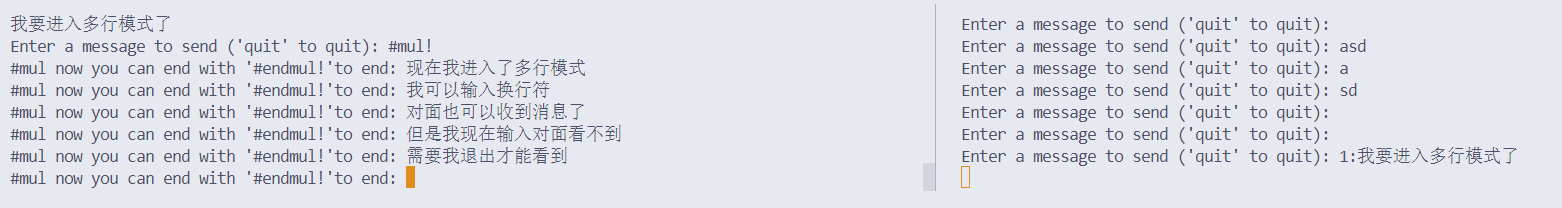
至于收发消息最大长度，发送消息没有长度限制，接收消息限制最大长度为1024字节，这点从代码中分析体现。所以虽然发送方可以无限输入消息，但由于接收方接收窗口为1024字节，其实际发送出的消息也就只有1024字节。



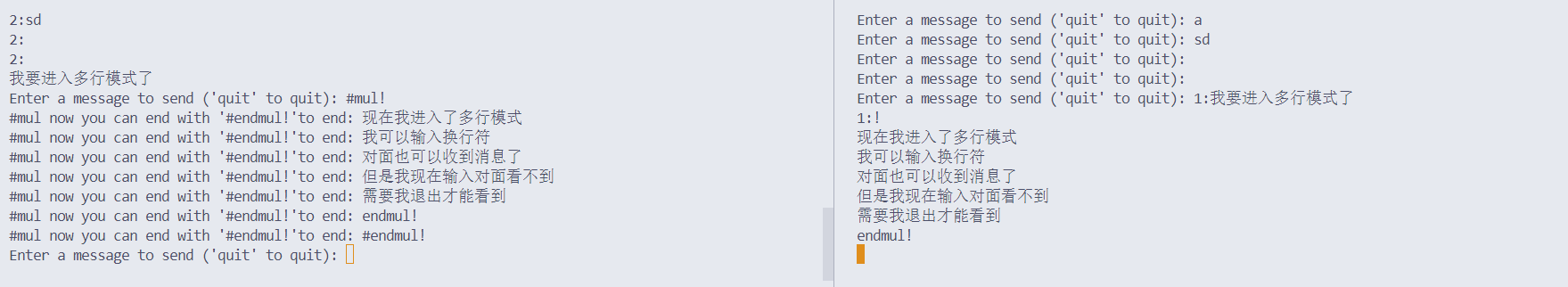


1. 多行文字输入展示：

可以看到发送端输入了换行消息，且对方没有收到



直到发送方输入设计好的保留字#endmul!，才会退出多行模式并发送：

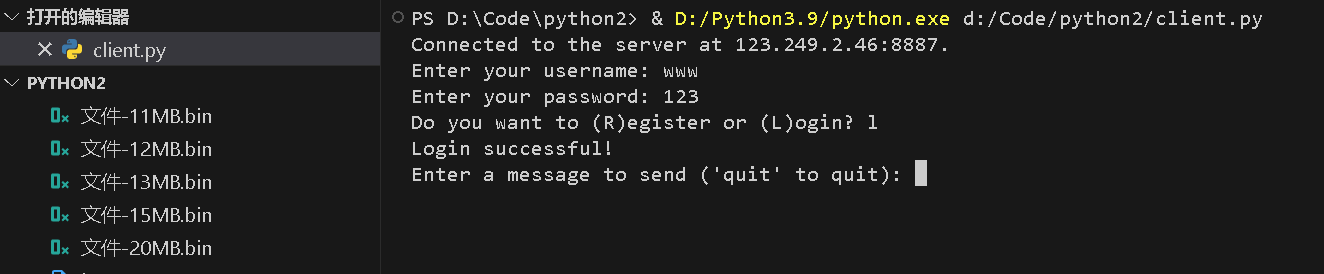


接收方直接接受大段文字。

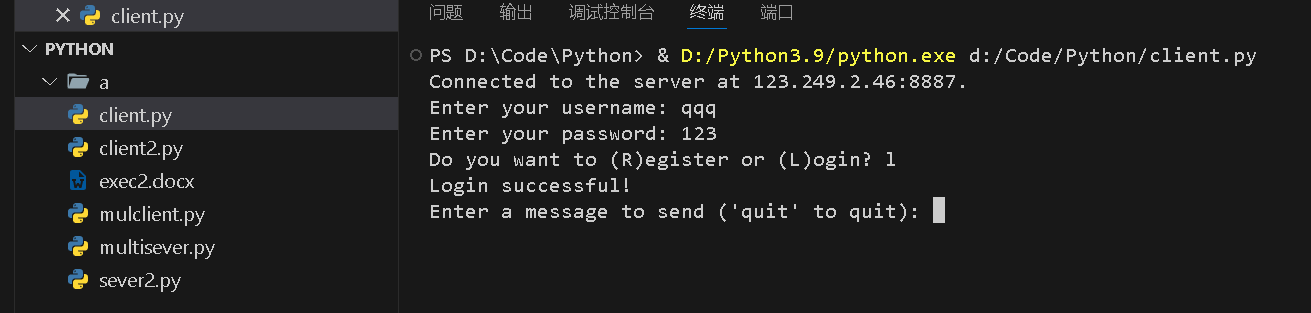
1. 用户之间传输文件，包括二进制的大文件（比如 10MB）。 —— 建议传输3-4个大约10MB大小文件，如PDF、word文档、图片等。

测试过程，从用户www发送4个大于10MB的二进制文件到另一用户qqq，以下是详细过程：

（1）首先这是用户www的界面，可以看到在当前目录下有4个大于10MB的二进制文件

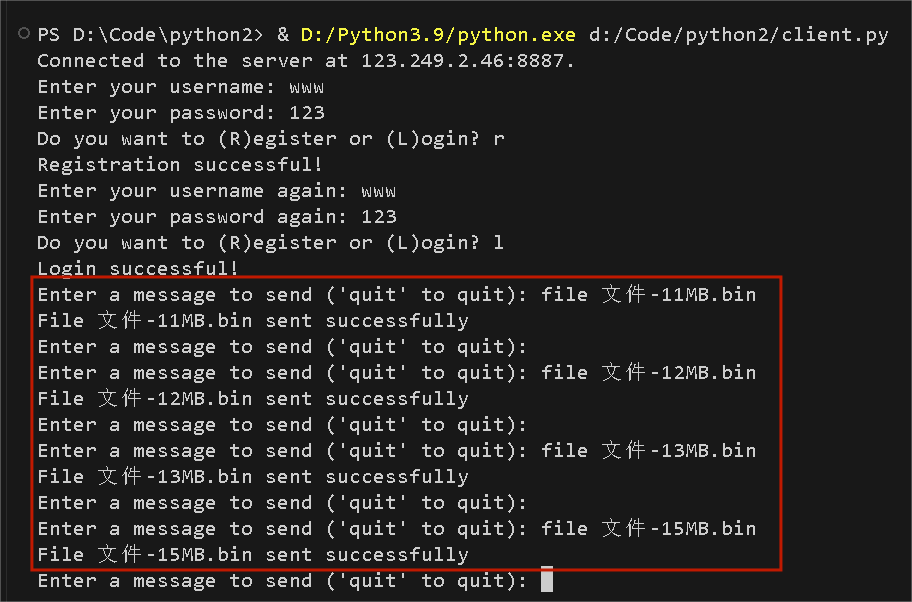


（2）这是用户qqq的界面，可以看到在当前目录下没有与www相同的文件



1. 接下来是传输过程

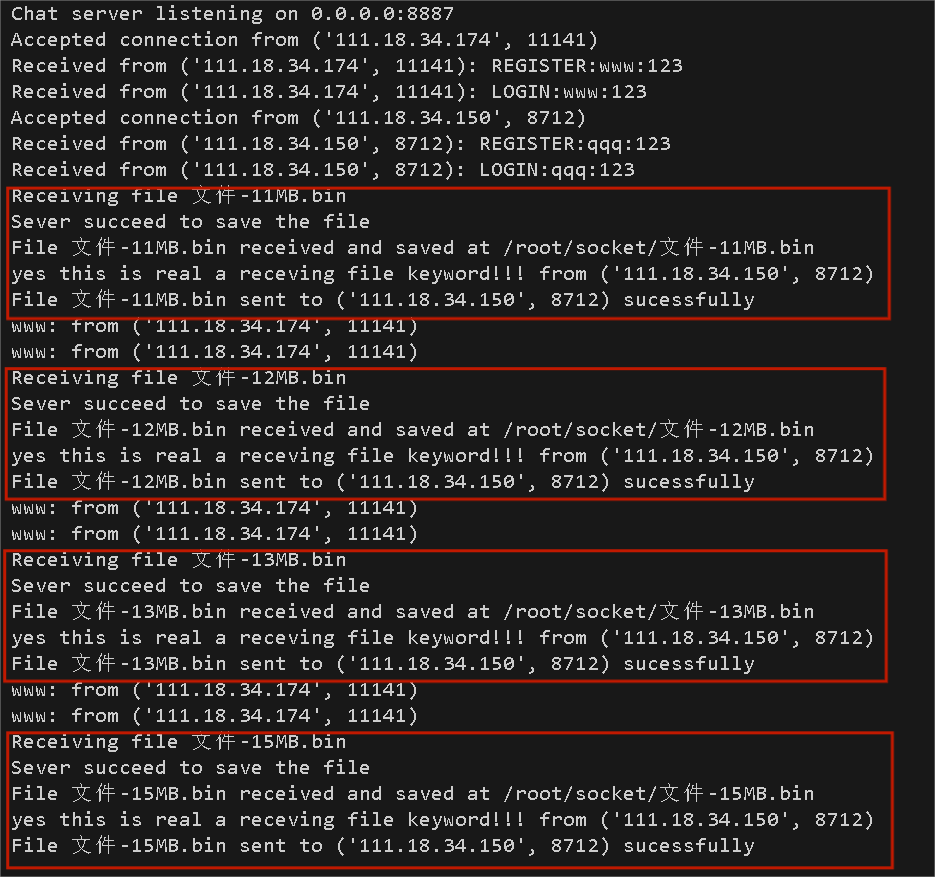
用户www发送文件界面



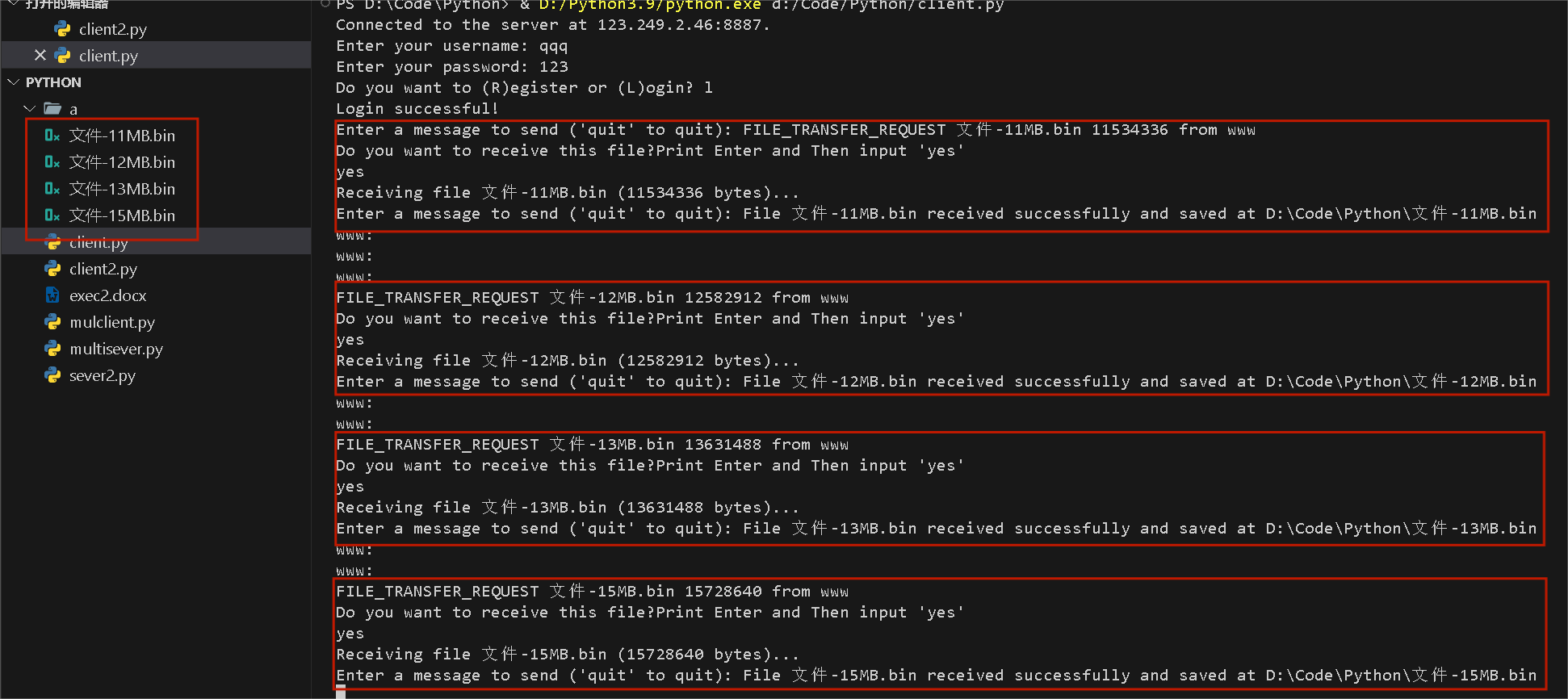
用户www通过输入指令file <file\_path>来发送指定路径下的文件，这里文件在当前目录下直接输入文件名，然后将文件发送到服务器，经服务器下载再广播给其他用户。如图www进行了4次不同大小文件发送，并且都显示发送成功。

服务器转载文件过程如下：首先服务器收到客户端发送过来的文件，然后服务器将文件下载到自己当前目录下，然后再将下载好的文件广播给其他用户。

下面是文件传输过程的服务器界面，可以看到服务器首先接收文件，然后下载文件，然后将文件发送出去，进行了4次同样的过程，分别对应4次大小不同的文件传输。



这是用户qqq接收文件的界面



可以看到界面左侧相比于之前多了4个从用户www传过来的大于10MB的二进制文件。

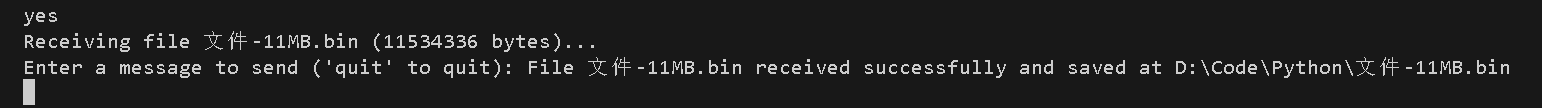
具体文件接收过程如下：当服务器将文件广播出去时，除了发送文件的客户端之外的其他客户端收到一条消息显示“FILE\_TRANSFER\_REQUEST <文件名> <文件大小> from <用户名>”，



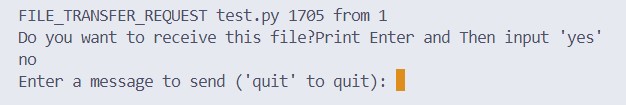
然后选择是否接收该文件



如果输入yes，就开始下载文件到默认当前目录下



如果输入no，不接受文件，继续正常聊天：



综上所述，通过测试可以验证聊天程序文件传输功能可以正常运行。

成功接受对方发送的视频文件：

1. 所有的测试必须在广域网上进行（建议服务器程序运行在云服务器上，客户端程序运行在本地的PC电脑上）
2. 服务器：我们使用华为云服务器（123.249.2.46），然后通过本地vscode软件ssh远程连接到云服务器上，然后在主机上运行云服务器程序。



1. 客户端：客户端程序在本地PC电脑vscode开多个窗口运行。

四、自测报告中简要介绍设计思路：

（1）说明具体的通信模式，如C/S模式，或混合 C/S 和 P2P 模式;

**该聊天程序采用了 C/S 模式。服务器通过 socket 创建监听套接字并绑定到指定的主机和端口，然后使用 accept 方法接受客户端的连接请求。一旦客户端连接成功，服务器将为该客户端创建一个新的线程来处理通信。**

**客户端通过 socket 创建连接套接字并连接到指定的服务器地址和端口。客户端与服务器之间通过 TCP 协议进行通信，以实现可靠的数据传输。**

让我们详细分析这一点：

服务器端：

在服务器端的 ChatServer 类中，通过 socket 库创建了一个 TCP 套接字（self.server\_socket），并使用 bind 绑定到指定的主机和端口。

通过调用 listen 方法来监听连接，等待客户端连接请求。一旦有客户端连接请求到来，服务器调用 accept 方法接受连接，并返回与客户端通信的套接字。

对于每个客户端连接，服务器会创建一个新的线程（client\_thread），并将其作为参数传递给 handle\_client 方法来处理与客户端的通信。

在 handle\_client 方法中，服务器通过接收客户端的消息来处理登录、注册、聊天和文件传输等功能，并通过广播消息来实现多人聊天。

客户端：

在客户端的 Client 类中，首先创建了一个 TCP 套接字（self.client\_socket），并通过调用 connect 方法连接到指定的服务器地址和端口。

客户端通过输入用户名和密码来进行登录或注册，然后根据用户选择进行相应的操作（登录或注册）。

客户端通过 input 函数接收用户输入的消息，并通过 send 方法将消息发送给服务器。

客户端通过调用 recv 方法接收服务器发送的消息，并根据消息类型进行相应的处理，比如显示聊天消息或处理文件传输。

（2）说明使用的具体协议。

**在这个聊天程序中，在传输层上使用 TCP（Transmission Control Protocol）作为传输层协议，确保数据的可靠传输。在应用层并没有显式定义特定的通信协议，而是通过自定义的消息格式来进行通信。主要使用了自定义的简单文本协议来实现客户端和服务器之间的通信。让我们详细分析一下使用的具体协议：**

**登录协议**：客户端发送格式为 LOGIN:username:password 的消息给服务器进行登录请求。服务器收到后解析消息，验证用户名和密码是否匹配，然后返回相应的登录结果给客户端，格式为 LOGIN\_SUCCESS 或 LOGIN\_FAILED，如果登录失败，还会返回失败原因。

**注册协议：**客户端发送格式为 REGISTER:username:password 的消息给服务器进行注册请求。服务器收到后解析消息，检查用户名是否已存在，如果不存在则将用户名和密码存入数据库，并返回 REGISTER\_SUCCESS 给客户端，表示注册成功；如果用户名已存在，则返回 REGISTER\_FAILED 给客户端，表示注册失败。

**聊天协议：**聊天消息协议：客户端发送格式为 username:message 的消息给服务器进行聊天。服务器收到消息后，会将其转发给所有其他在线客户端，实现多人聊天功能。

**文件传输请求协议：**客户端输入命令file <文件路径>，然后发送格式为 FILE\_TRANSFER filename filesize from sender\_username 的消息给服务器进行文件传输请求。服务器收到请求后，将根据消息中的文件名和大小等信息准备接收文件，然后通知其他在线客户端有文件传输请求并将文件广播给其他在线客户端。

**文件接收确认协议：**服务器发送格式为 FILE\_TRANSFER\_REQUEST filename filesize from sender\_username 的消息给其他在线客户端，表示有文件传输请求。其他客户端收到消息后，会询问用户是否接收文件，并将用户的回复发送给服务器，如果用户同意接收文件就会将文件下载到客户端当前目录下。

**文件传输数据协议：**客户端在确认接收文件后，服务器开始发送文件数据。客户端根据接收的文件大小进行数据接收，直到接收完整个文件。文件传输数据是通过 TCP 连接进行的。

**语音请求协议：**客户端发送格式为 AUDIO!!! to target\_username from sender\_username 的消息给服务器进行语音请求。服务器收到请求后，会将请求转发给目标用户，询问其是否接受语音通话。

**语音接受确认协议：**目标用户收到语音请求后，可以选择接受或拒绝语音通话。如果接受，则向服务器发送 AUDIOyes!! 消息，服务器将通知请求方开始语音通话；如果拒绝，则向服务器发送 AUDIOno!! 消息，服务器将通知请求方语音通话被拒绝。

通过这些自定义的简单文本协议，客户端和服务器能够在不同的功能场景下进行通信和交互，实现了聊天、文件传输和语音通话等功能。

（3）说明服务器支持几个并发的用户同时在线和聊天、传输文件等。【1个、2个或多个】

服务器支持多个并发的用户同时在线和聊天，使用了多线程的机制来处理并发连接。每当有新的客户端连接到服务器时，服务器将为该客户端创建一个新的线程来处理通信。这样可以确保服务器能够同时处理多个客户端的连接和通信，并提供并发的服务。

（4）说明服务器是否支持用户Alice和Bob在聊天的同时，传输文件；或传输文件的同时，文字聊天。

我们所设计的服务器并不支持两个用户聊天和传输文件同时进行，仅仅是可以在发完消息后可以传输文件，或传输文件结束后能正常聊天。这是该聊天程序一个缺陷，在设计的时候忽略了这一点。

在自测之后进行了反思，得出不能支持两个用户的原因：

（5）如果服务器能够支持多个并发TCP连接（或UDP会话），请说明其实现的机制（如多进程、多线程、基于事件驱动、异步模式等）。

我们设计的服务器可以支持多个并发TCP连接，具体实现机制是通过多线程来实现的。

服务器在 start方法中执行以下步骤：

1.使用 server\_socket.accept() 方法等待客户端连接请求。

2.一旦有新的连接请求到达，accept 方法会返回一个新的客户端套接字对象 (client\_socket) 和客户端地址 (client\_address)。

3.服务器为新的客户端连接创建一个新的线程，调用 handle\_client 方法来处理客户端的消息。这样，每个客户端连接都有一个独立的线程来处理其消息，不会被其他客户端的操作阻塞。

4.handle\_client 方法中通过 client\_socket.recv() 方法接收客户端发送的消息，并根据消息内容执行相应的操作。

在 handle\_client 方法中，服务器处理客户端的消息并执行相应的操作，如登录、注册、文字聊天、文件传输等。这些操作在各自的线程中独立执行，不会相互干扰。同时，使用互斥锁 self.Lock 来保护共享资源，如客户端列表 self.clients 和用户数据库 self.user\_db，确保多个线程之间的并发安全。

综上所述，服务器通过多线程的方式实现了支持多个并发TCP连接。每个客户端连接都会被分配一个独立的线程来处理其消息，从而实现了并发处理多个客户端的连接和消息。

（6）说明应用层的协议设计，包括（但不限于）登录消息的协议设计、聊天的文字消息的协议设计、文件传输的协议设计等。

应用层使用的自设计协议如下：

**登录协议**：客户端发送格式为 LOGIN:username:password 的消息给服务器进行登录请求。服务器收到后解析消息，验证用户名和密码是否匹配，然后返回相应的登录结果给客户端，格式为 LOGIN\_SUCCESS 或 LOGIN\_FAILED，如果登录失败，还会返回失败原因。

**注册协议：**客户端发送格式为 REGISTER:username:password 的消息给服务器进行注册请求。服务器收到后解析消息，检查用户名是否已存在，如果不存在则将用户名和密码存入数据库，并返回 REGISTER\_SUCCESS 给客户端，表示注册成功；如果用户名已存在，则返回 REGISTER\_FAILED 给客户端，表示注册失败。

**聊天协议：**聊天消息协议：客户端发送格式为 username:message 的消息给服务器进行聊天。服务器收到消息后，会将其转发给所有其他在线客户端，实现多人聊天功能。

**文件传输请求协议：**客户端输入命令file <文件路径>，然后发送格式为 FILE\_TRANSFER filename filesize from sender\_username 的消息给服务器进行文件传输请求。服务器收到请求后，将根据消息中的文件名和大小等信息准备接收文件，然后通知其他在线客户端有文件传输请求并将文件广播给其他在线客户端。

**文件接收确认协议：**服务器发送格式为 FILE\_TRANSFER\_REQUEST filename filesize from sender\_username 的消息给其他在线客户端，表示有文件传输请求。其他客户端收到消息后，会询问用户是否接收文件，并将用户的回复发送给服务器，如果用户同意接收文件就会将文件下载到客户端当前目录下。

**文件传输数据协议：**客户端在确认接收文件后，服务器开始发送文件数据。客户端根据接收的文件大小进行数据接收，直到接收完整个文件。文件传输数据是通过 TCP 连接进行的。

**语音请求协议：**客户端发送格式为 AUDIO!!! to target\_username from sender\_username 的消息给服务器进行语音请求。服务器收到请求后，会将请求转发给目标用户，询问其是否接受语音通话。

**语音接受确认协议：**目标用户收到语音请求后，可以选择接受或拒绝语音通话。如果接受，则向服务器发送 AUDIOyes!! 消息，服务器将通知请求方开始语音通话；如果拒绝，则向服务器发送 AUDIOno!! 消息，服务器将通知请求方语音通话被拒绝。

通过这些自定义的简单文本协议，客户端和服务器能够在不同的功能场景下进行通信和交互，实现了聊天、文件传输和语音通话等功能。

1. 如果登录有安全性方面的设计，请特别说明。

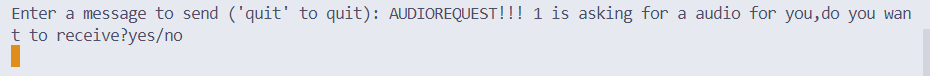
**密码加密传输：**用户在登录时，密码是以明文形式发送给服务器的。为了增加安全性，可以在客户端对密码进行加密处理，然后再发送给服务器，服务器端再对密码进行相应的解密验证。这样可以避免密码在网络传输过程中被窃取。

**登录结果处理：**服务器会对用户提供的用户名和密码进行验证，验证成功后返回登录成功的消息，否则返回相应的登录失败消息。这样可以保护用户的隐私信息，不会泄露具体的登录失败原因。

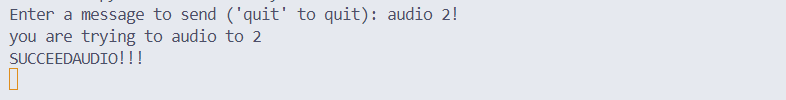
**注册结果处理**：服务器在用户注册时，会返回注册成功或失败的消息。如果用户名已经被注册，服务器会返回注册失败的消息，这样可以防止同一用户名被多次注册。

拓展语音功能：由于前文使用了聊天室里广播发送文件，所有用户都能收到文件的机制。对于语音，我们选择语音发起方能够选择指定的用户进行一对一通信：

  
 发起方通过关键字 audio [username]!，来向指定用户发起语音请求

对方用户会收到如下信息：  
 

对方用户会收到发起语音通话的请求，打印发起用户的用户名，输入yes进入通话，输入no拒绝通话，输入yes展示：

双方进入语音的显示：  
 发起方：

接收方：  


双方采用键入ctrl+c断开语音，并输入AUDIOEXIT关键字退出语音  
之后正常聊天：

