**实 验 报 告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 组号： | 1-6和4-5 |  |  |  |  |
| 姓名： | 向胤兴 | 学号： | 2215012469 | 班级： | 计算机2102 |
| 姓名： | 曾水靖 | 学号： | 2214611804 | 班级： | 计算机2102 |
|  |  |  |  |  |  |

1. 实验名称

Socket 网络编程实验，实现一个简单的聊天程序

1. 实验原理

使用 Python语言进行编写，采用了 Client/Server 模式的工作原理。

1. 验证用户登录时，客户端发送用户输入的账户和密码，服务器端对账户的密码进行比较，若密码正确则服务器允许该客户进行登录。

2. 在发送文字消息时，客户端向服务器端发送包含用户消息的报文，由服务器端将报文广播给聊天室内其他用户。

3. 发送在线文件时，客户端将文件发送给服务器，服务器将文件下载到目录下，然后再将文件广播给其他聊天室内的客户端，被广播文件的客户端会收到文件提示，可以选择接收或拒绝文件，若选择接收则将文件下载到当前目录下

4. 进行语音通话时，两个客户端通过服务器建立连接，进行语音输入时将音频输入到音频输入流中转换成音频数据，然后将音频数据发送给另一方，另一方接收到音频数据后将其写入音频输出流中并播放出来。

5.当实现NAT穿透时，客户端通过STUN协议向STUN服务器获得自己的公网ip和关联端口号，然后将其发送给服务器，服务器发送给指定的客户端，被指定的客户端收到后根据ip和端口号与发起客户端建立连接，进行点对点通信，不再需要通过服务器转发消息。

1. 实验目的

1. 通过 socket 编程，进一步理解计算机网络的工作原理，掌握基本的 socket 程序的编写方法，尝试设计和实现自己的应用层协议。

2. 掌握多线程编程的基本方法，理解多线程程序的工作原理和调试方法。

1. 实验内容

1.基本功能

1) **验证用户登录**

用户通过之前注册过的账户以及密码进行登录操作，若密码错误服务器不允许该用户登录。

2) **用户之间的文字聊天**

用户之间可以进行文字聊天，且实时性较好，支持中英文输入和多行文字输入，消息输入和接收长度有限制

1. 用**户之间传输文件**，包括二进制的大文件（比如 100MB）

2.高级功能

**1)语音聊天**

用户之间进行质量较好的语音聊天，聊天语音清晰，延迟较小，

4) 支持用户先把文件上传至服务器，其他用户再从服务器上下载，但不支持文件断点传送

1. 实验实现
2. 人员分工
3. 向胤兴：聊天程序结构基本架构设计和编程，用户注册登录功能文字聊天功能、文件传输功能、语音聊天功能的设计和编程；参与自测报告和最终实验报告部分编写工作
4. 曾水靖：参与文件传输功能设计与编程，协助代码设计与编程，参与代码debug和测试验证；负责自测报告和最终实验报告的主要编写工作
5. 实验设计
6. 协议

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。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：

1）**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。

2）**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。

3）**聊天协议**：客户端时刻监听键盘输入input(”Enter a message to send (‘quit’ to quit): ”)，当用户通过键盘进行输入后，发送格式为 username:message 的消息给服务器进行聊天，如果键盘输入“quit”就会断开与服务器的连接退出聊天。服务器收到消息后，会将其广播转发给所有其他在线客户端，实现多人聊天功能。客户端在接收消息方面时刻监听服务器端，当有消息从f服务器传来就将其打印。

4）**多行输入协议**：当客户端需要在消息栏进行多行输入时，输入多行输入命令#mul，这时再输入换行符和回车不会将消息发送出去，会进行换行输入，当输入结束后想将消息发送出去，输入多行输入结束命令#endmul，然后再回车就会将先前输入的多行消息全部发送出去。

5）**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> <file\_size> 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。

6）**文件接收确认协议**：服务器发送格式为 FILE\_TRANSFER\_REQUEST <filename> <file\_size> 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，会打印一条询问消息：“Do you want to receive this file? Print Enter and Then input ‘yes’”,询问用户是否接收文件，并将用户的回复发送给服务器，如果用户输入’yes’同意接收文件，服务器在收到’yes’后将文件内容发送给该客户端，客户端接收文件内容后结合前面接收到的文件名和文件大小将文件保存到客户端当前目录下。

7）**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<file\_size>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。

8）**语音请求协议**：用户在命令行输入”aulio <target\_name>!”来向指定用户发起语音聊天请求，然后客户端发送格式为 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。

8）**语音接受确认协议**：目标用户收到语音请求 “AUDIOREQUEST!!! <sender\_name> is asking for a audio for you,do you want to receive?yes/no”后，可以选择接受或拒绝语音通话。如果用户输入yes接受，客户端则向服务器发送 AUDIOyes!! 消息，服务器将通知请求方开始语音通话；如果输入no拒绝，客户端则向服务器发送 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. UI设计

本实验采用命令行风格，无图形化界面设计

1. 框架结构

【说清楚客户端程序和服务器端程序的框架结构。特别是服务器程序：说明支持多个用户（连接）的方法，如多进程，多线程，多路复用等；如果存在进程或线程之间共享数据的，还要注意保护加锁等。说明错误处理的方法。】

服务器程序框架：服务器采用python面向对象风格，构造了一个服务器类。初始化时创建一个服务器socket对象，负责监听客户端发起的连接请求。



示例化一个chatsever对象后，对象start开始对客户端socket连接进行监听并负责在接收到请求后处理连接。我们使用**多线程**机制来支持多个用户的连接。每当self.sever\_socket.accpet()接收到一个客户端的连接后，创建一个新的子线程，使用handle\_client成员函数来进行和客户端之间的通信。



使用多线程，不同线程之间可以很方便的使用共享变量，但同时也会引入同时访问临界资源带来的错误。为了消除这一影响，必须使用线程锁机制：





当某个线程在访问临界时，必须先获取对应相关共享变量的锁才能进行操作，并在操作完成后归还锁：



如上，当验证用户注册账号或登录账号时，需要根据共享变量self.login\_db,self.user\_db来确定账号是否已被注册，登录密码是否正确等。所以在进行这些临界资源的访问之前，先获取到管理相关临界资源的互斥锁，确定不再使用时候归还锁，这样在多个程序同时想要对临界资源进行访问时，就不会发生错误了。

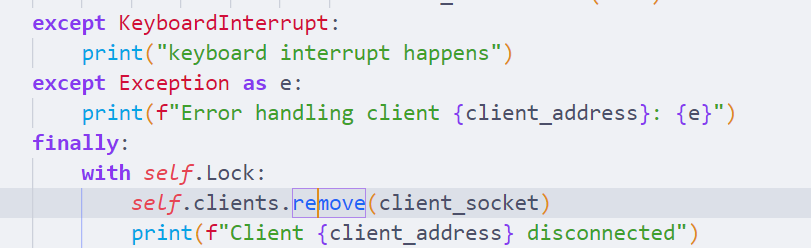
**错误处理：**

代码框架中包含的错误处理机制如下：

客户端进行用户注册时如果用户名已被注册、登录时账号正在被使用、密码与账号不匹配，服务器会告知客户端，客户端正常进行下一次登录。

整个用户处理线程的程序中，使用try-except语句来捕获可能发生的异常。





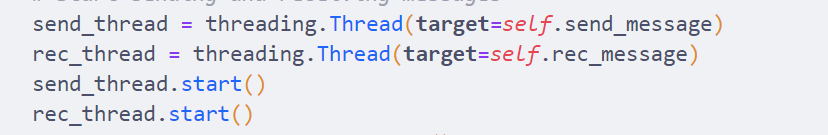
使用Exception捕获所有异常，打印输出错误原因。并在finally中的self.clients列表中删除该客户端socket对象，保证用来记录已连接客户端socket对象列表的self.clients列表内容正确。

客户端程序框架：

客户端同样使用python面向对象编程风格。声明了一个用来和服务器进行通信的client类。



Client对象需要同时能够发送消息和接收消息。我们同样使用了多线程机制来实现这一功能。



创建了一个发送信息的子线程，一个接收信息的子线程。

虽然采用多线程，但读写线程之间各有独立的缓冲区，不存在对临界资源的竞争。

主要需要解决的时线程间通信问题，由于采用命令行风格，并且只使用两个线程，一些确认或者拒绝消息需要通过发送信息的线程以信息的格式把命令发送出去。此时线程间的先后顺序就显得尤其重要。

例如，在我们的设计中，当一个用户向另一个用户发起语音请求后，对方是否接受请求，需要在对方做出回复后通过接收消息线程来进行确定。所以接收消息线程必须要能够区分出此时收到的信息是关于对方是否同意语音通话的回复，而不是当作发送来的信息打印出来。并且最后需要告知发送信息进程对方是否接受，以判断是否进入音频流通信模式。

由此可见线程间的通信在我们的架构中十分重要。我们使用了threading.event来保证线程间的时序关系，使用共有的bool类型变量来进行信息的通信（对方是否同意语音连接请求？同意将某个bool变量置为True，否则置为False）。

1. 语音功能的具体实现方法

客户端初始化时，创建python库中的audio。当一个客户端发起连接并且对方同意时，服务器将两个客户端socket对象置入一个元组列表中。当检测到发送消息的客户端是元组列表中某个元组的一个元素时，从向所有用户广播信息改为只向元组中的另一个socket对象发送信息。客户端方面。发送信息线程发送的信息从发送键盘输入内容改为从音频输入流发送，接受信息线程接受的到的信息的处理方式从打印到屏幕上改为从音频输出流输出。

1. 文件直接从服务器下载：

服务器在进行用户间的文件转发时，会在服务器本地保存文件。用户可以通过指定文件名直接向服务器发起文件下载请求，但是不支持断点传送。

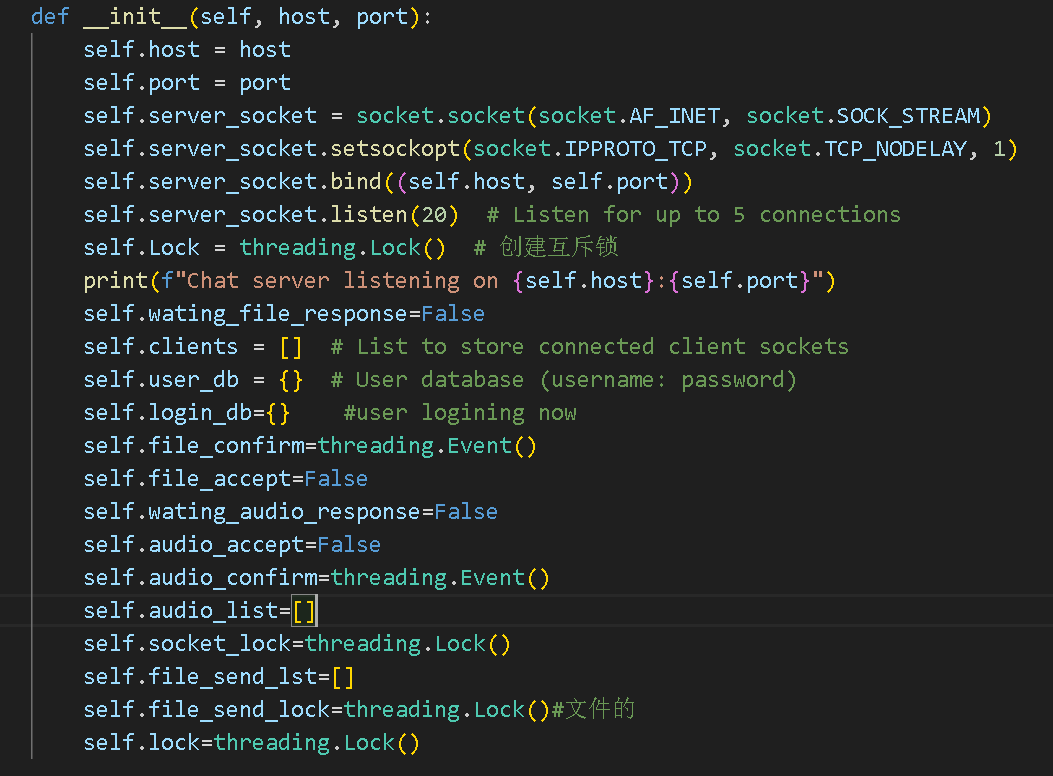
1. 关键代码的描述

【主要代码源文件名及功能描述，是借鉴还是自主编写，自主编写需注明编写人。】

**本次实验的代码均由小组成员通过Python语言进行编写**

1. 关键代码1：服务器端初始化方法。

**自主编写：向胤兴**

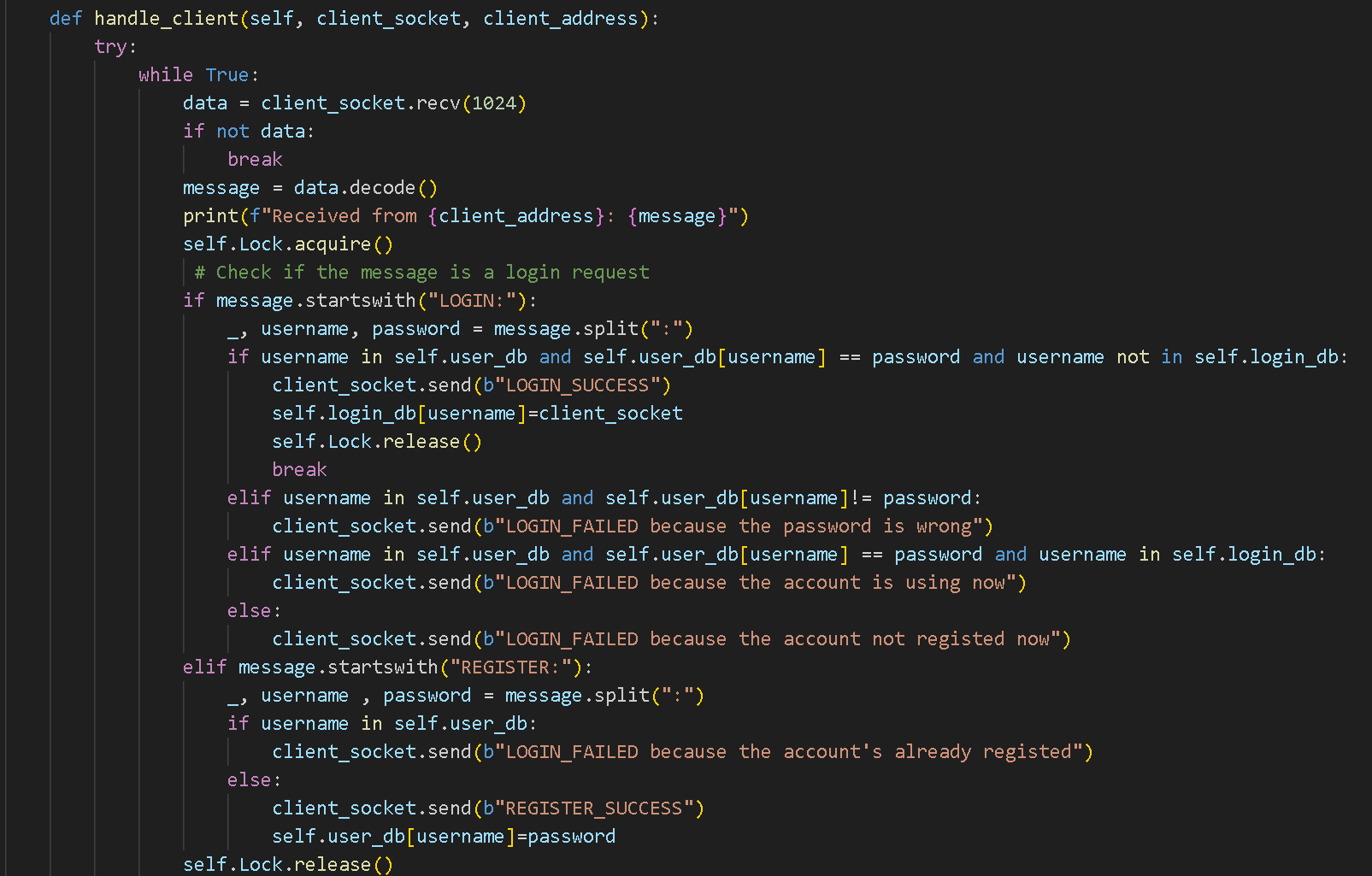


该方法主要功能是通过socket模块建立TCP服务器套接字，通过bind方法将套接字绑定到指定地址和端口，这里是云服务器的地址和端口，然后通过listen方法监听来自客户端的连接。剩下的就是对下面要用到的全局变量进行声明和初始化。

1. 关键代码2：服务器端客户端的线程处理方法。

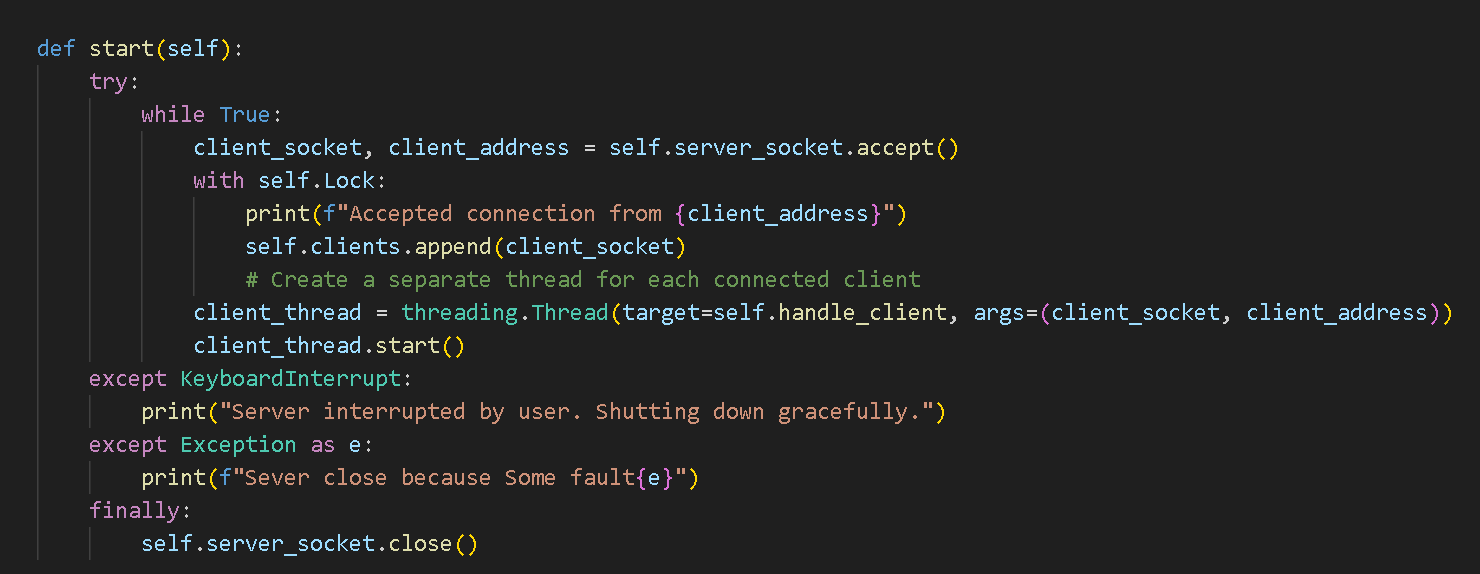
**自主编写：向胤兴**

handle\_client方法过长只截部分代码

该方法的主要功能是对连接上服务器的客户端线程进行处理，通过recv方法监听并接收客户端发送的消息，根据客户端发送的消息判断客户端要进行的操作比如用户注册和登录、文字聊天、文件传输和语音聊天。如果用户进行注册和登录时，需要对用户输入的用户名和密码进行记录，当用户注册和登录后给予反馈；当用户之间进行文字聊天时，对用户之间发送的消息进行广播转发；当用户之间进行文件传输时需要接收发送方发送的文件，然后再将文件转发给指定用户。总的来说，服务器通过该方法担任一个中间角色，为多个客户端提供间接连接，处理用户需求。

1. 关键代码3：服务器启动方法

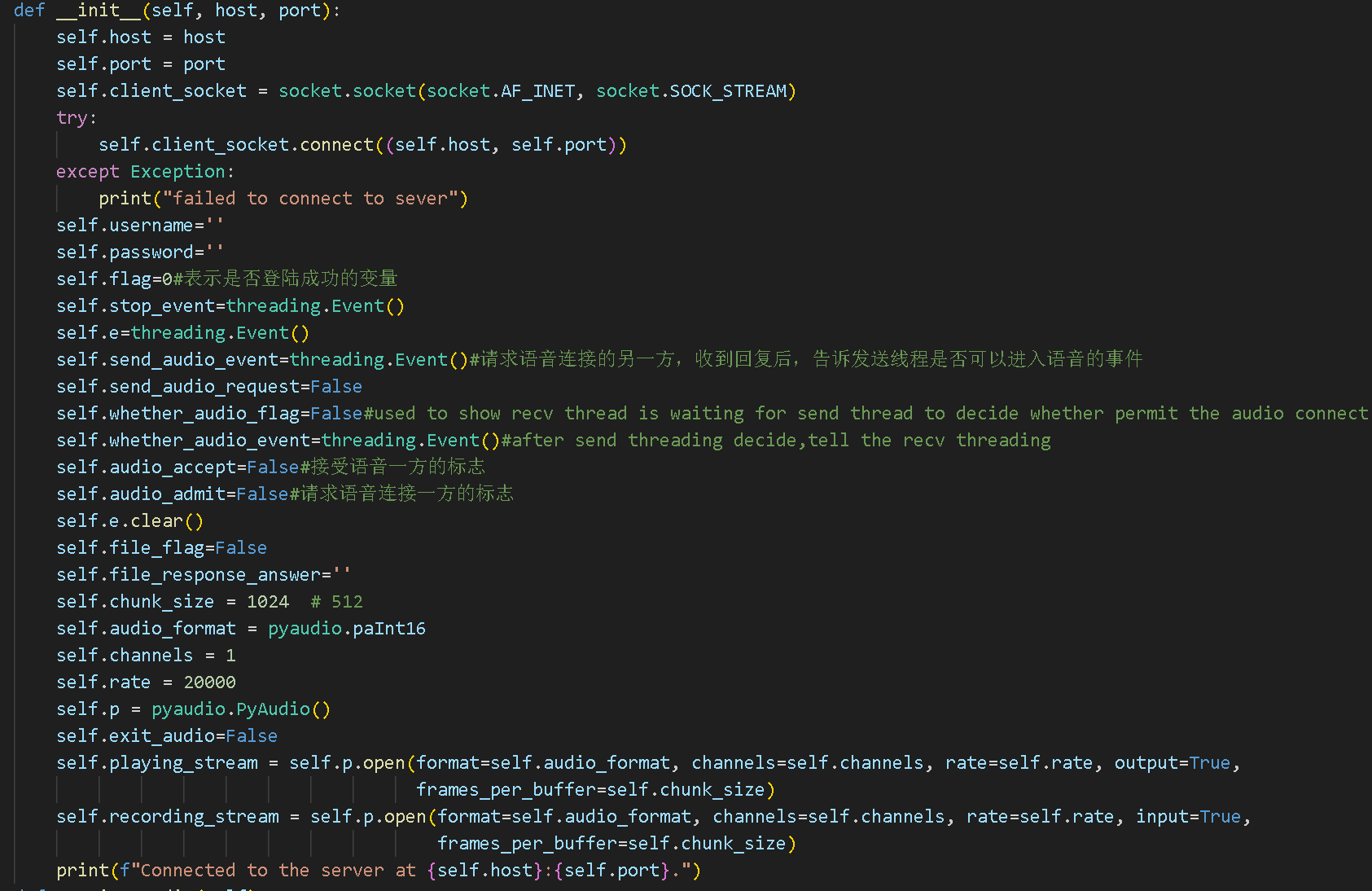
**自主编写：向胤兴**



该方法的功能是，启动服务器对客户端的监听，通过accept方法阻塞监听客户端的连接，当有客户端连接该服务器时，accept方法返回新客户端的套接字对象和地址，然后服务器给新客户端创建一个新的线程并启动。

（四）关键代码4：客户端初始化

**自主编写：向胤兴**



该方法的主要功能是创建客户端TCP套接字对象，然后通过connect方法试图连接上指定服务器的地址和端口，当服务器在线时就可以连接上服务器。然后下面都是对后续要实现的功能所声明和初始化的全局变量。由于支持语音功能，初始化时就创建了py.audio对象，并绑定好了输入输出流。

（五）关键代码5：客户端启动方法

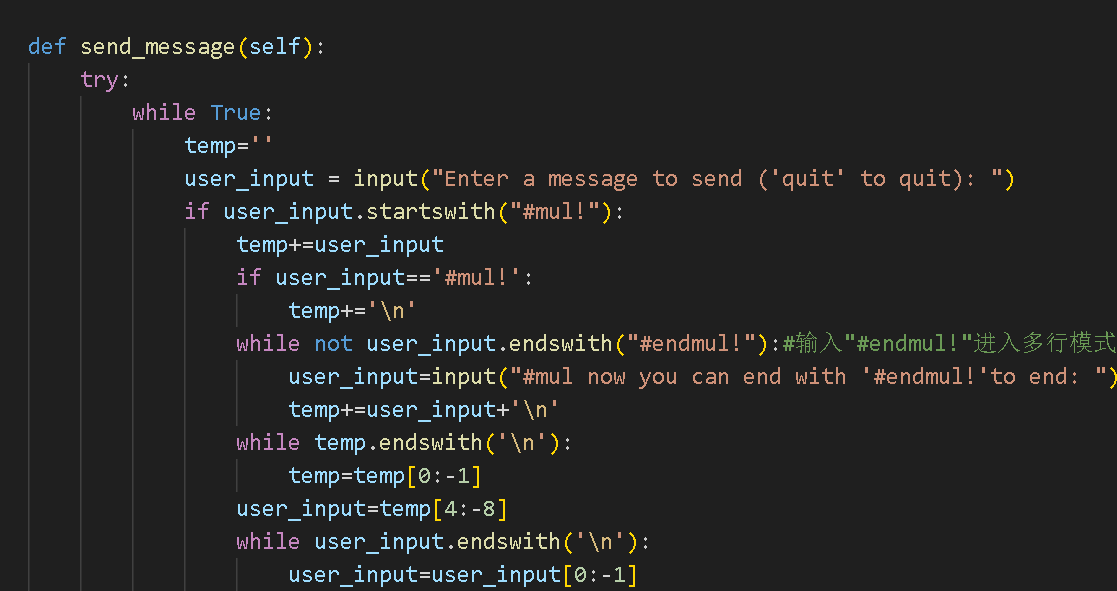
**自主编写：向胤兴**

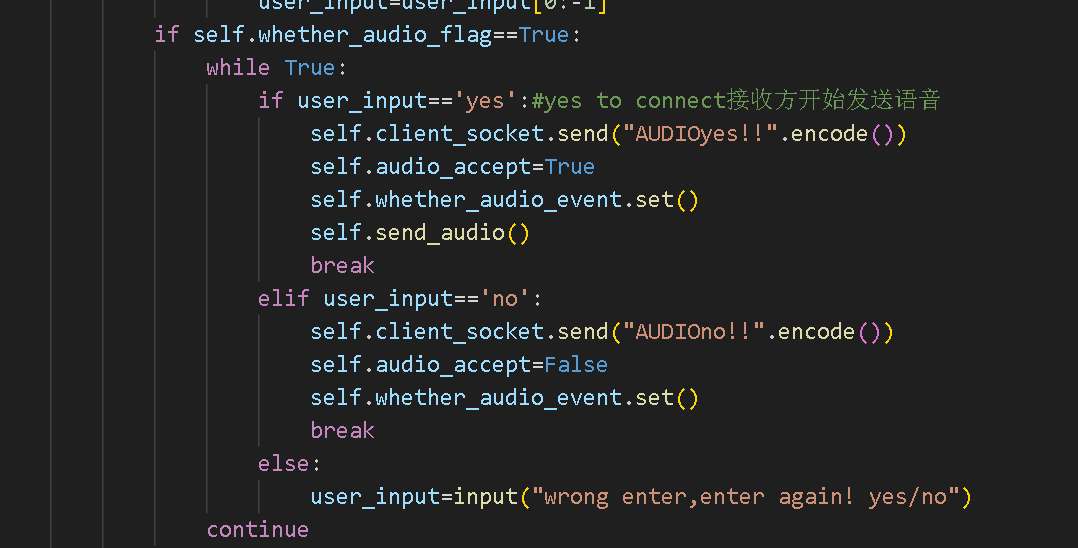


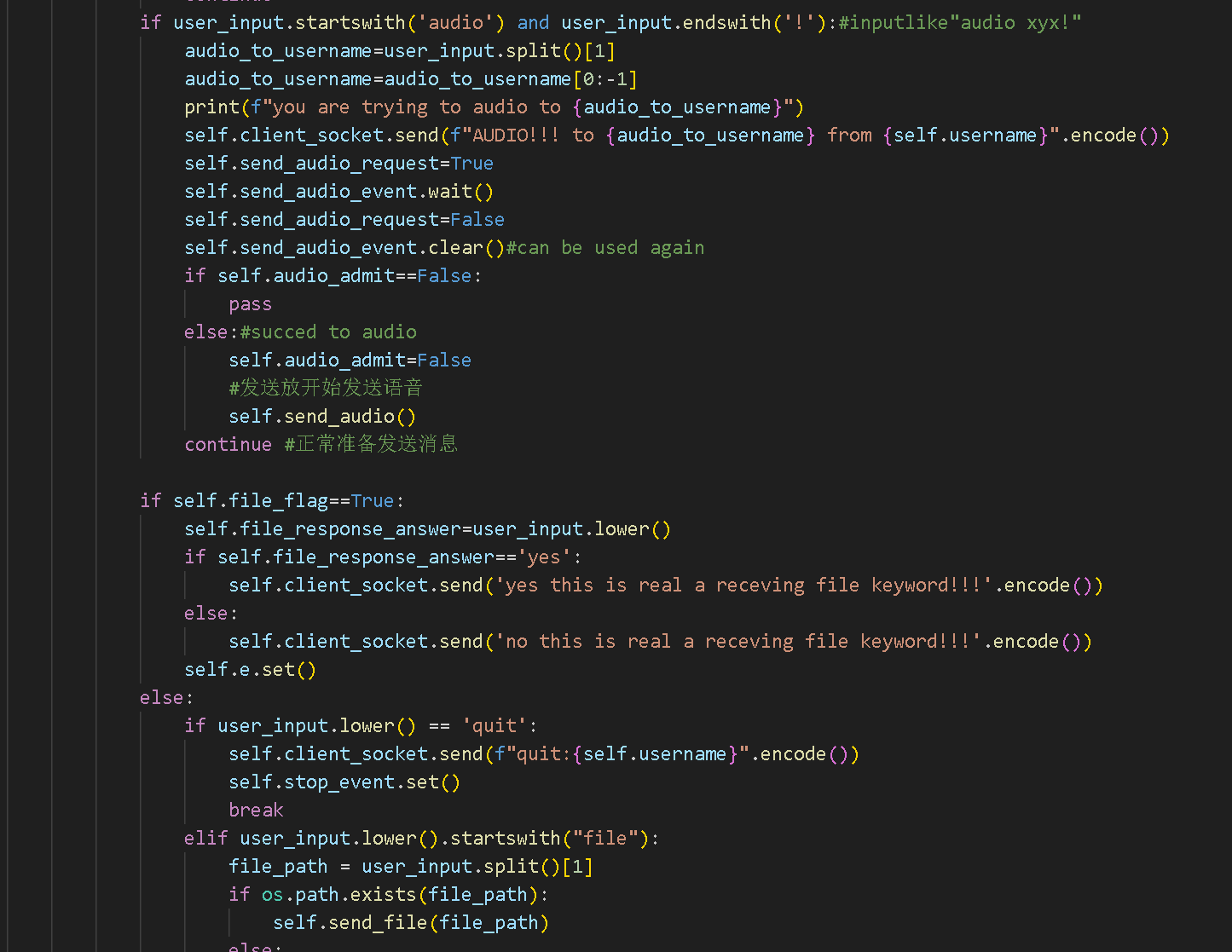
当客户端启动并连接上一个服务器时，会打印用户登录提示信息，然后监听用户键盘输入，需要用户输入用户名和密码进行登录，用户登录成功后会创建并启动一个发送进程send\_thread和一个接收进程rec\_thread，分别用于发送消息和接收消息，最后使用join方法让主线程等待两个这两个子线程结束，确保主线程结束之前，所有工作都已完成。

（六）关键代码6：客户端发送消息方法

**自主编写：向胤兴**



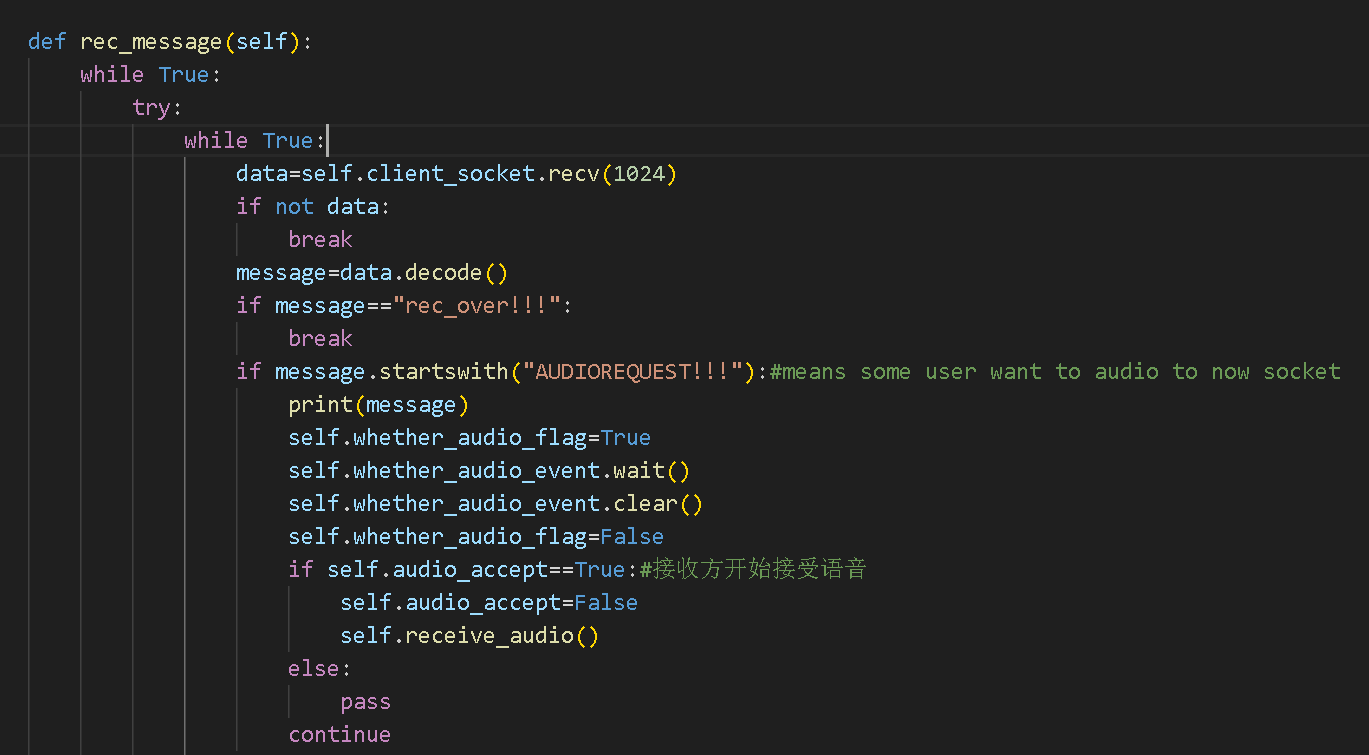


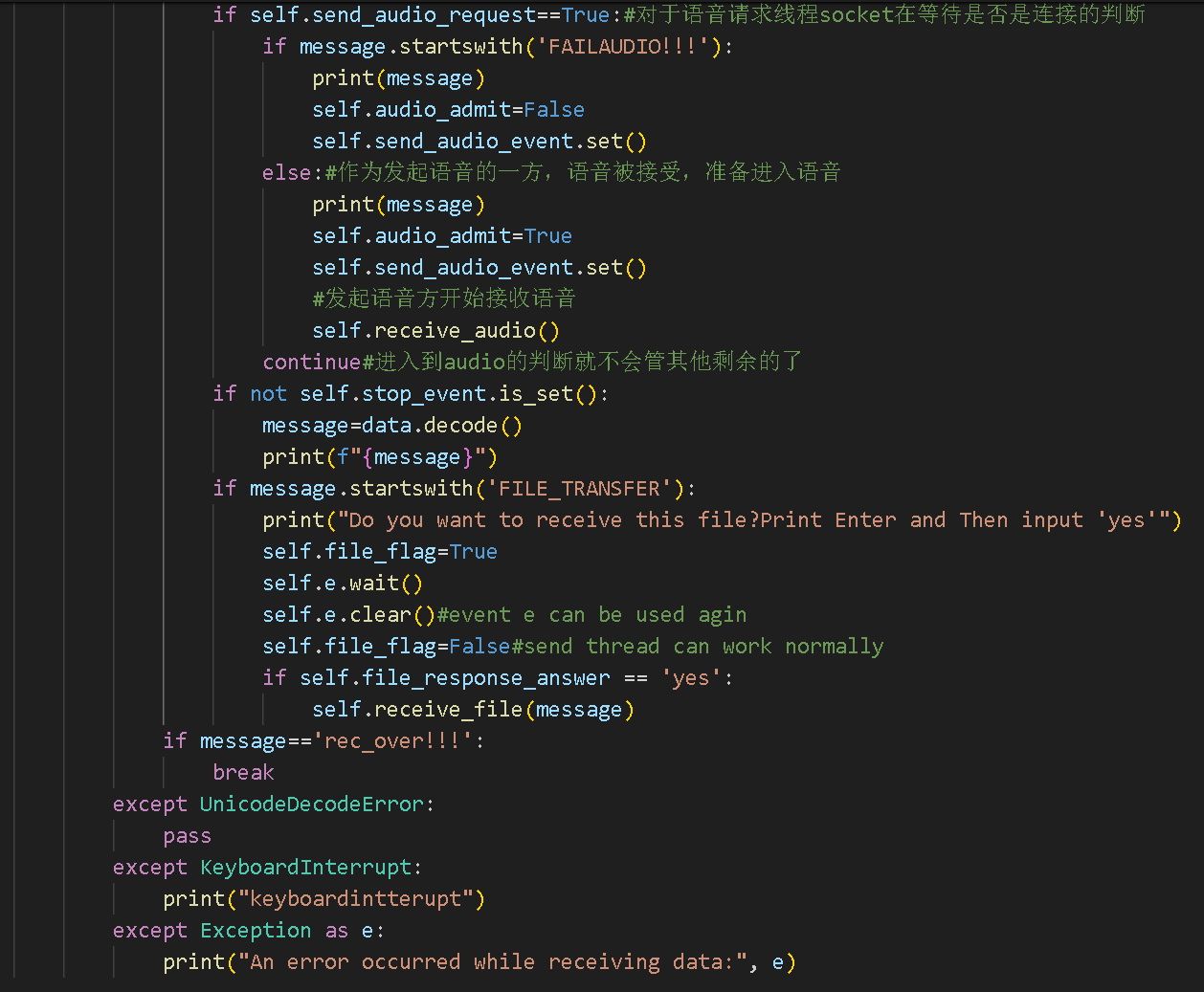


该方法的主要功能是，对用户输入的内容进行时刻监听，接收然后处理，处理后给服务器发送相应的内容；用户输入的内容觉得了接下来要使用的功能。如果用户简单输入文字聊天，输入内容开头不包含特殊功能指定命令，那客户端就直接将内容发给服务器，再经服务器转发给其他客户端；如果用户要进行文件传输，需要输入命令file <file\_path>发送指定路径下的文件，然后客户端会给服务器发送FILE\_TRANSFER....信息表示需要进行文件传输；用户要进行语音聊天也是类似文件传输，不多赘述。

1. 关键代码7：客户端接收消息方法

**自主编写：向胤兴**

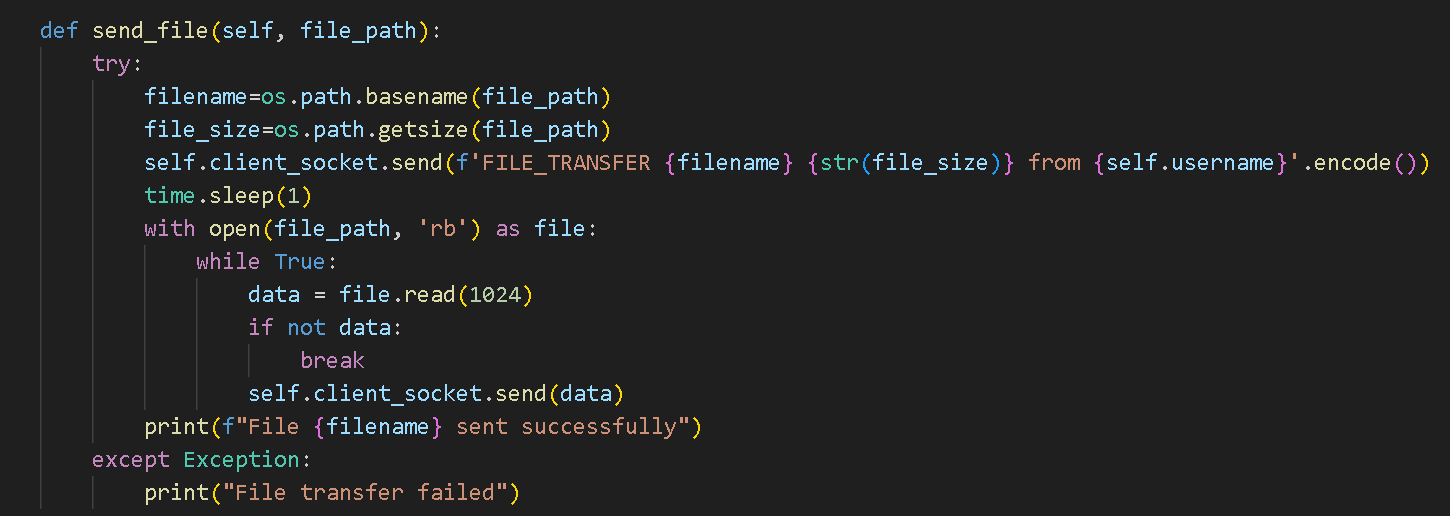




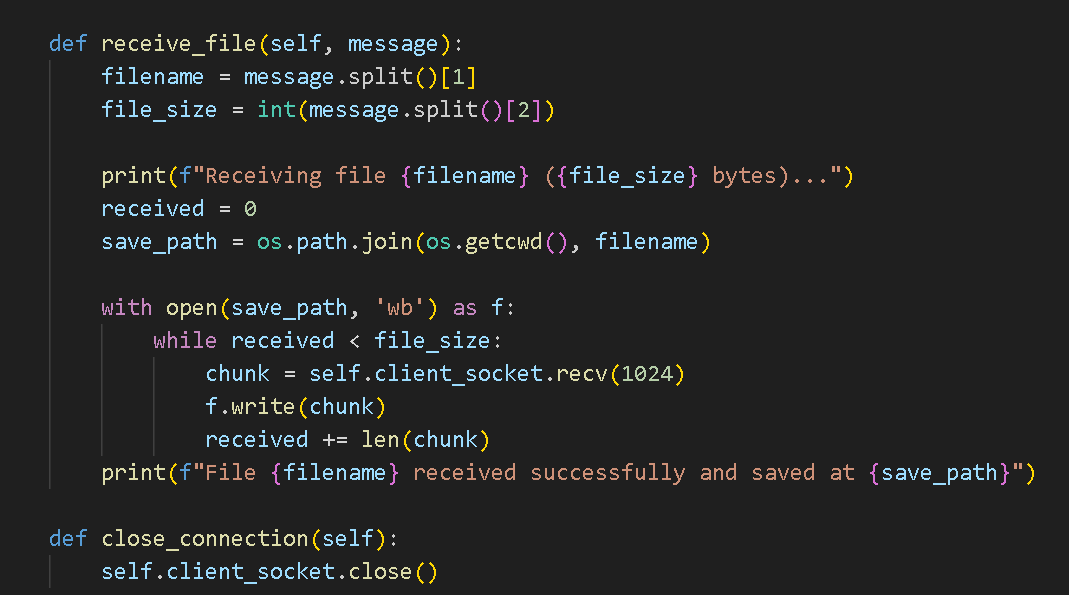
该方法的主要功能是通过recv方法监听并接收服务器发送的消息，对消息进行解码并处理，配合发送客户端完成指定功能。对服务器发送的信息进行解码后，首先判断信息开头是否有特殊功能命令，若没有就是普通文字聊天，就只需要普通接收并打印消息；如果信息开头有FILE\_TRANSFER，则说明发起用户想要给你传输一个文件，然后可以选择是否接收文件；如果信息开头有AUDIOREQUEST!!!，则说明发起用户想和你进行语音聊天，然后可以选择是否进行语音连接。

1. 关键代码8：客户端文件发送和接收方法

**自主编写：向胤兴、曾水靖**



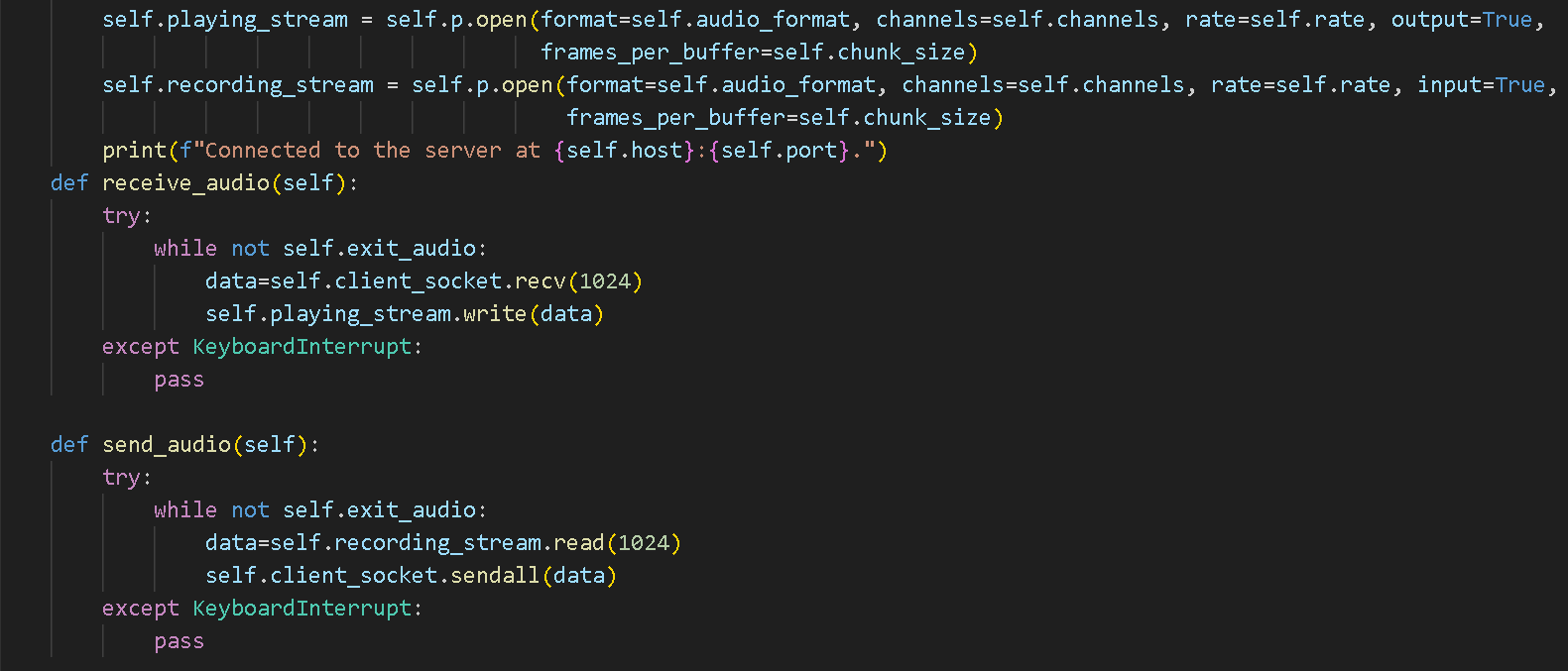
文件发送方法以文件路径为输入参数，通过os模块根据输入的文件路径解析出文件名和文件大小，然后将文件名和文件大小发送给服务器，然后再打开文件，将文件内容读出存到临时变量，然后再发送出去。



文件接收方法以服务器发送过来的的FILE\_TRANSFER <file\_name> <file\_size> from <sender\_name>消息为输入，通过split方法从输入消息中分离出文件名和文件大小，通过os模块join方法获得当前目录路径，根据接收文件名新建一个空文件，然后打开文件，根据接收文件大小将接收到的文件内容写入文件中。

1. 关键代码q：客户端语音发送和接收方法

**自主编写：曾水靖**



在初始化方法中声明定义了音频输出流playing\_stream和输入流recording\_stream；在音频发送方法中，无限循环通过pyaudio模块的read方法将语音录制到输入流中，然后转换成音频数据发送给服务器；在音频接收方法中无限循环监听和接收服务器发送的消息，然后将接收到的音频数据写入音频输出流中，写入音频输出流也就是将语音实时播放出来。

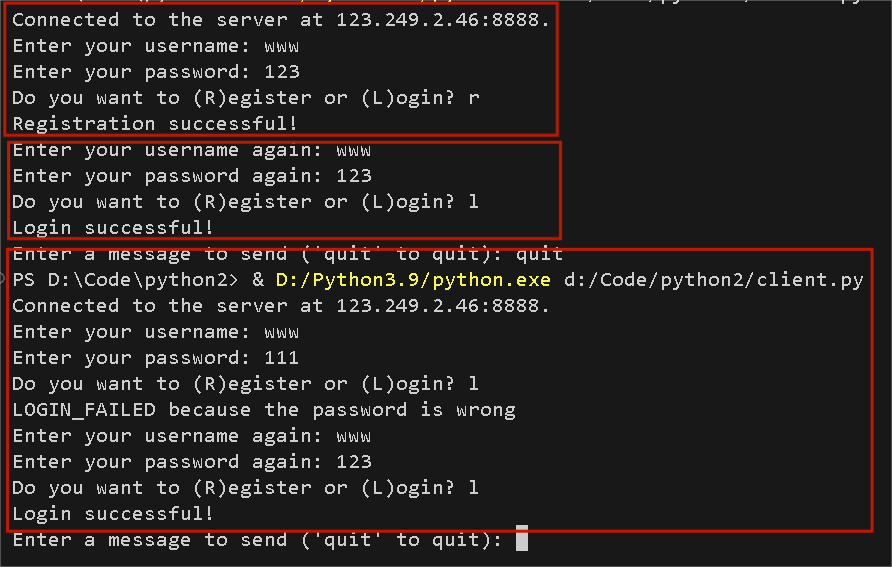
1. 测试及结果分析

【测试过程应当比较详尽，把所有的功能都测试覆盖了，还要注意错误情况的处理。】

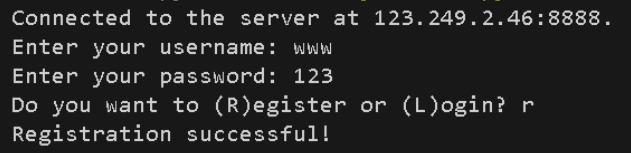
**基本功能测试：**

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）。

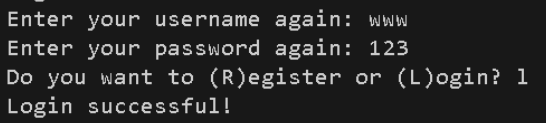
这是一个总体的新客户端连接服务器，退出然后重新登录的过程



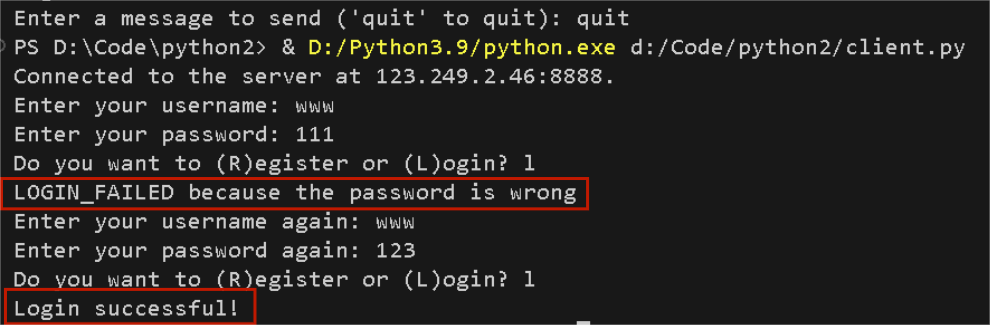
过程分为三个小步骤，首先新用户连接服务器需要注册账号



这里注册新用户www，密码123，注册完成后需要重新登录

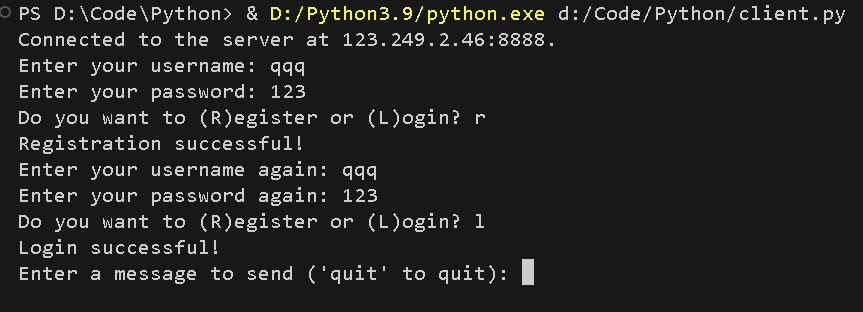


重新登录这里密码正确，登录成功。然后退出账号重新登录



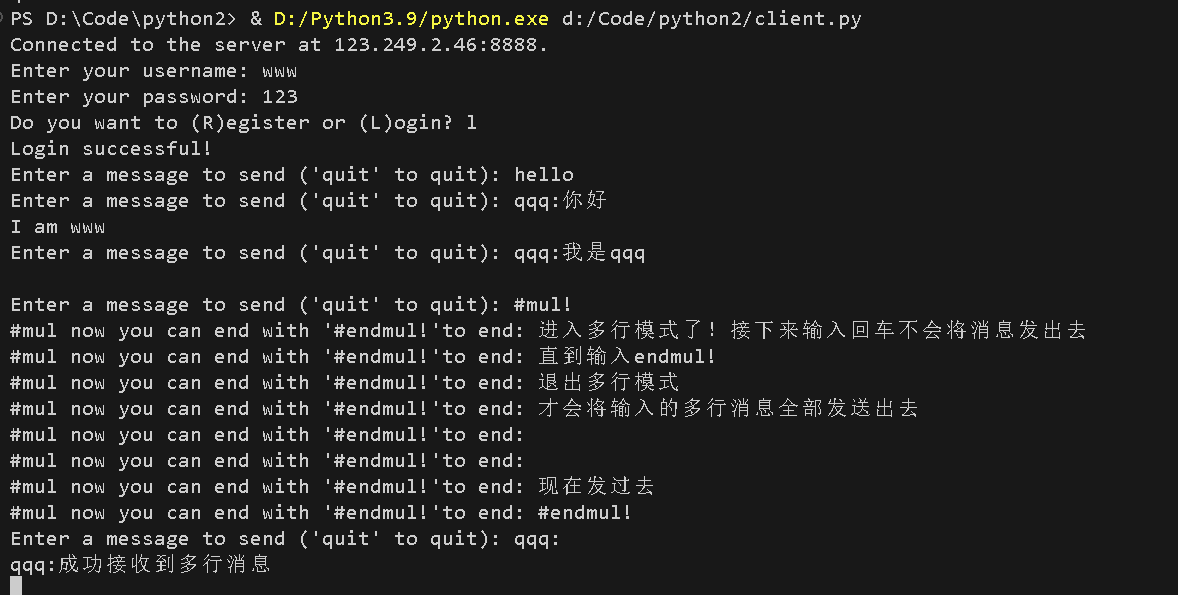
这里输入了错误的密码111，所以登录失败，然后再用正确的密码123登录，登录成功。

下面是另一个客户端新用户qqq登录过程

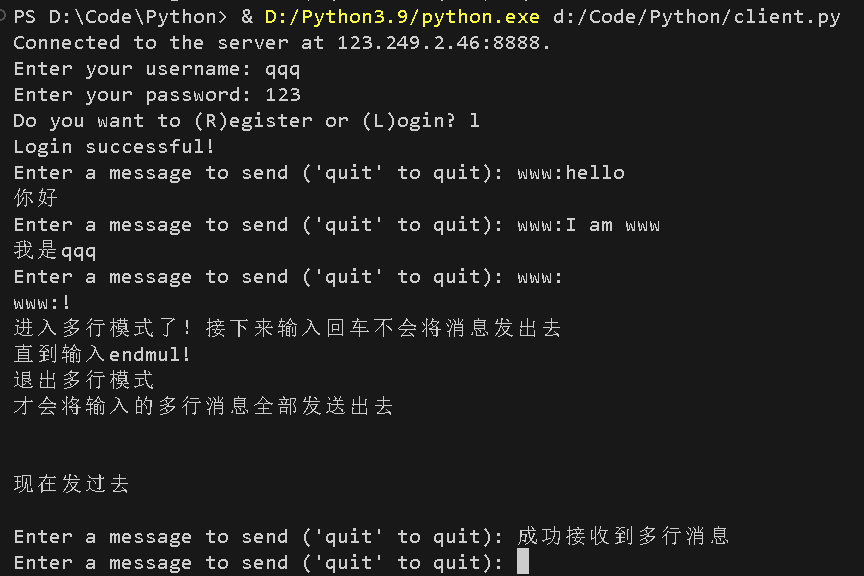


1. 两个用户的文字聊天；—— 文字聊天可以参照微信、QQ，比如 （1）中英文的支持；（2）支持收发文字消息的最大长度； （3）支持多行文字吗？

用户www界面

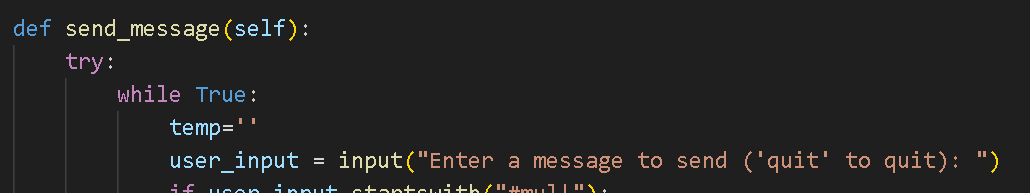


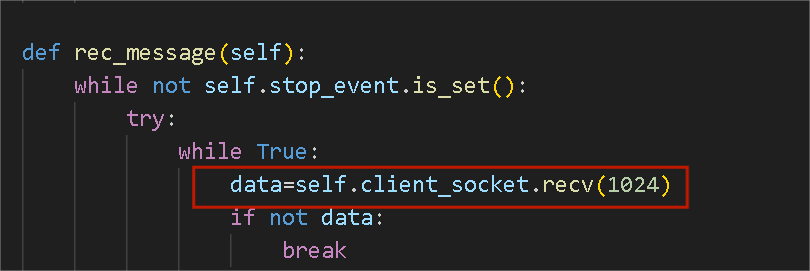
另一个用户qqq界面



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；当在命令行中输入#mul!然后回车就会进入多行输入模式，接下来输入换行符不会将消息发出去，直到输入#endmul!结束多行输入模式回车之后之前输入的多行输入就一并被发送出去。

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。

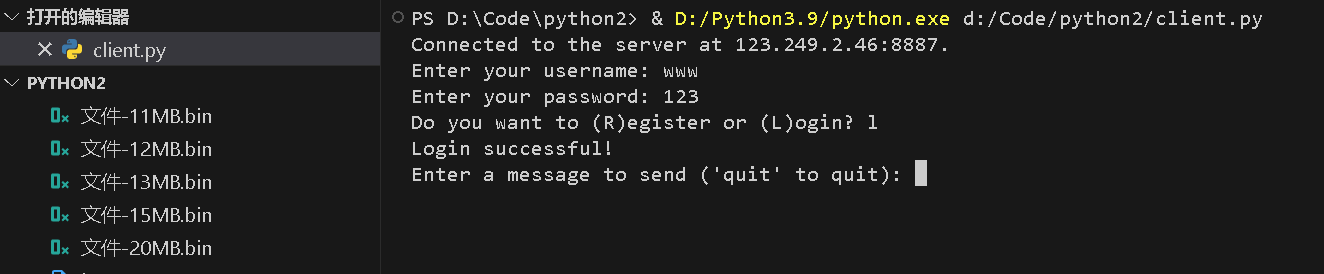




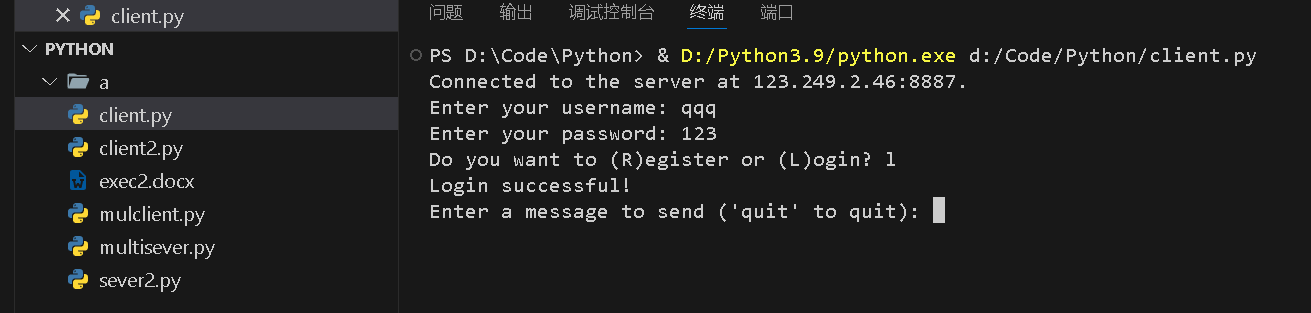
1. 用户之间传输文件，包括二进制的大文件（比如 10MB）。 —— 建议传输3-4个大约10MB大小文件，如PDF、word文档、图片等。

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

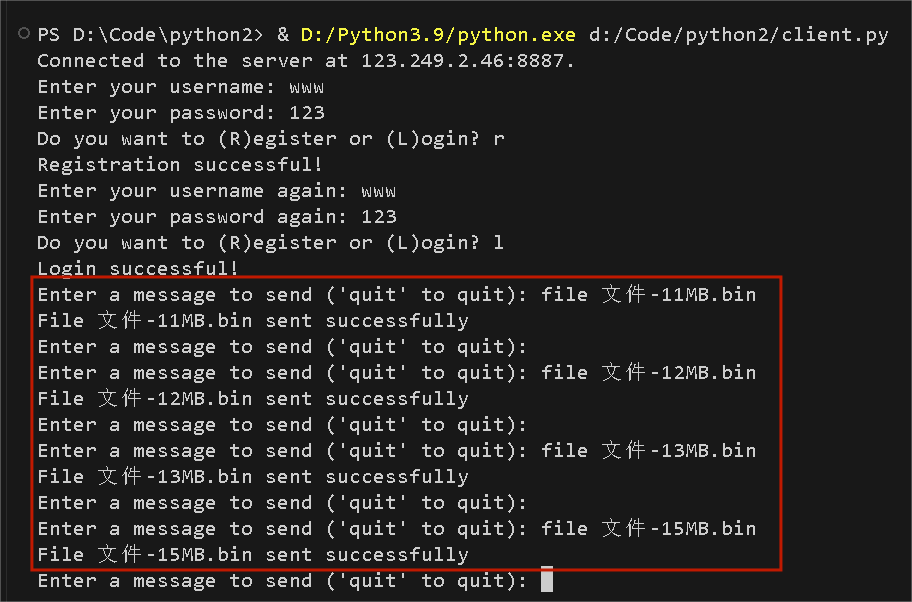


（2）这是用户qqq的界面，可以看到在当前目录下没有与www相同的文件



1. 接下来是传输过程

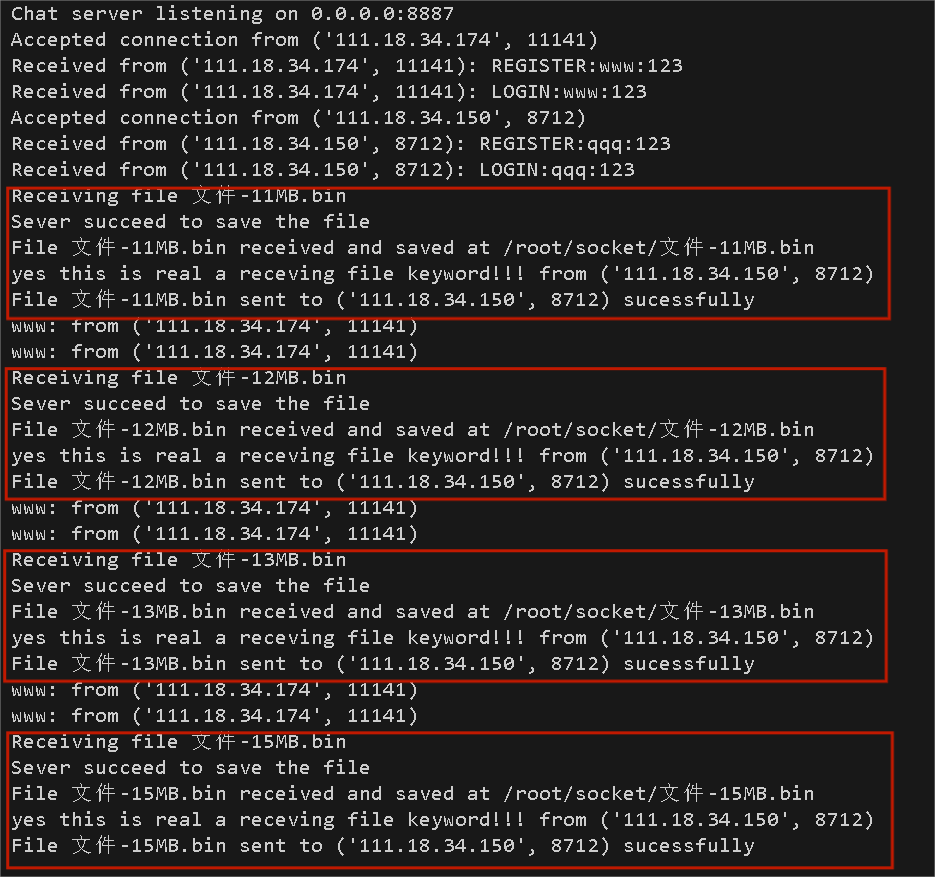
用户www发送文件界面



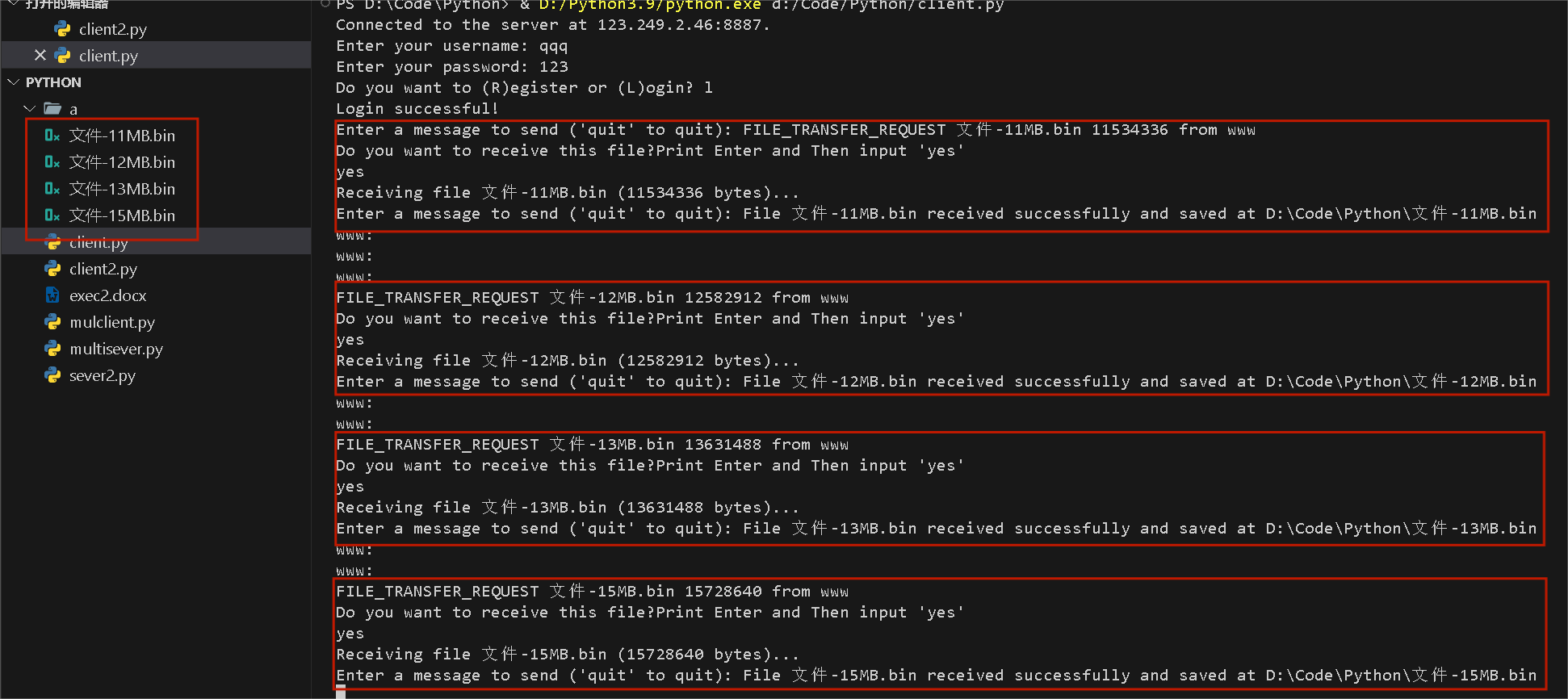
用户www通过输入指令file <file\_path>来发送指定路径下的文件，这里文件在当前目录下直接输入文件名，然后将文件发送到服务器，经服务器下载再广播给其他用户。如图www进行了4次不同大小文件发送，并且都显示发送成功。

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。

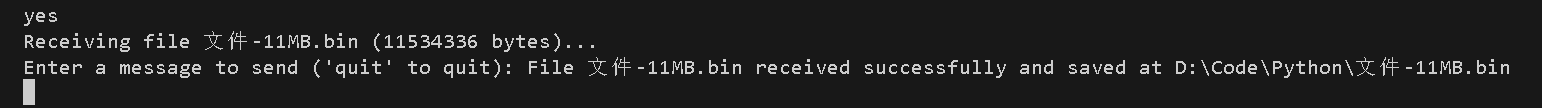
具体文件接收过程如下：当服务器将文件广播出去时，除了发送文件的客户端之外的其他客户端收到一条消息显示“FILE\_TRANSFER\_REQUEST <文件名> <文件大小> from <用户名>”，



然后选择是否接收该文件

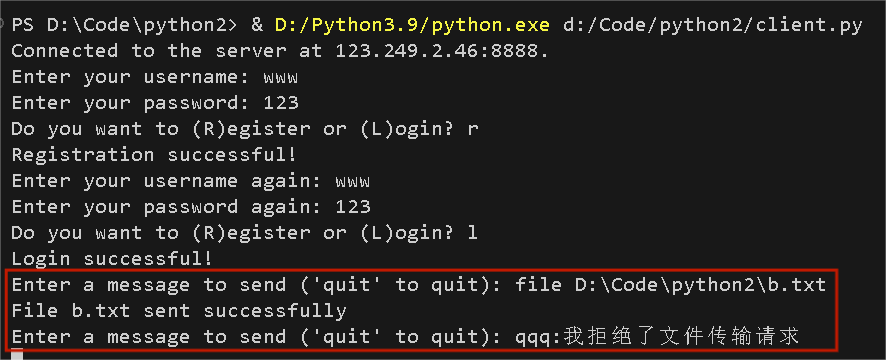


如果输入yes，就开始下载文件到默认当前目录下

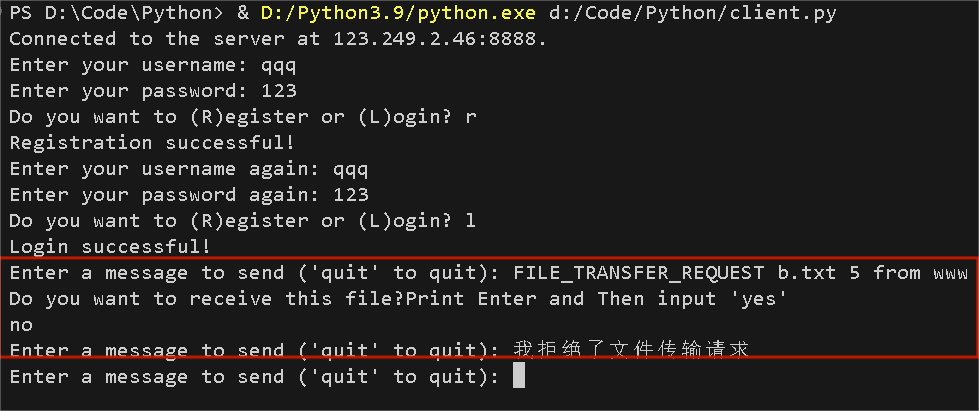


用户也可以拒绝接收文件

www给qqq发送b.txt文件



qqq收到文件传输请求并拒绝



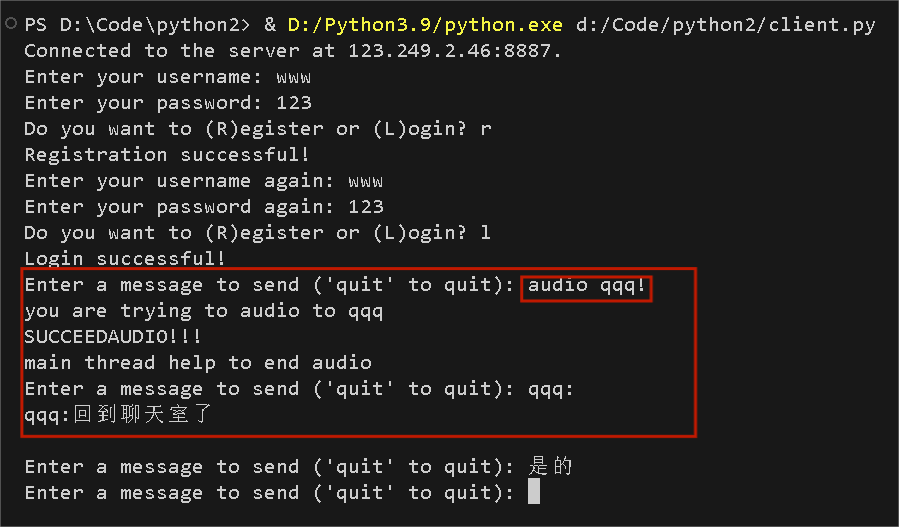
拒绝文件传输后可以回到正常的聊天。

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。

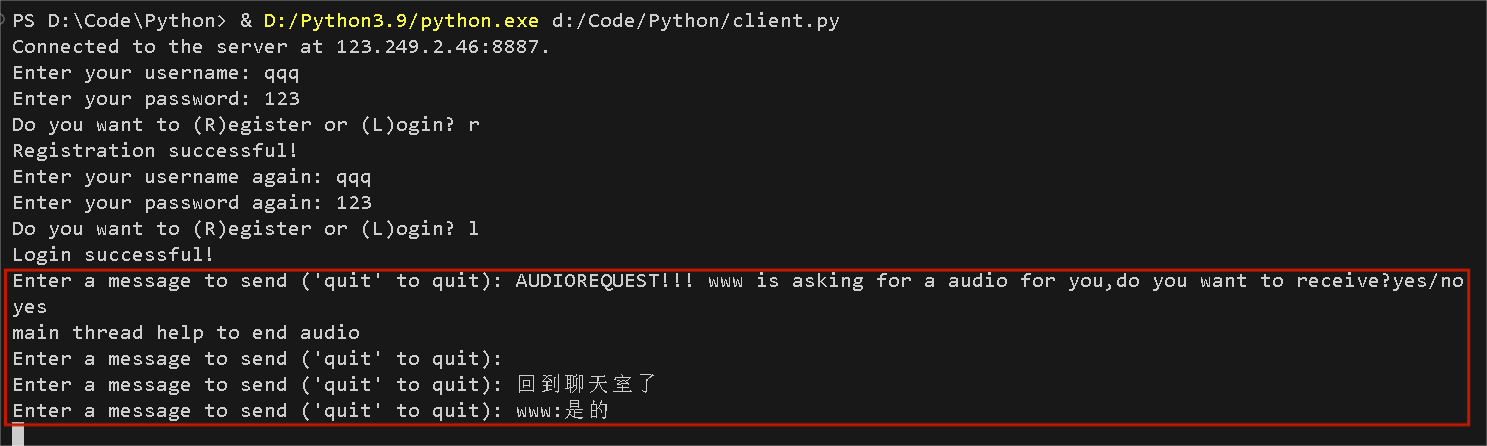
**高级功能测试**

C.语音聊天 —— 测试并说明语音聊天双方的网络连接状况和语音通话质量。

www向qqq发起语音请求



qqq接受语音聊天



语音聊天开始后在键盘输入Ctrl+C手动中断来结束语音聊天。语音聊天质量良好，有些许回声，无电流声，无丢包现象，声音连贯清晰。

1. 实验结论

TCP/IP协议栈的实用性：

实验成功验证了TCP/IP协议栈在构建可靠的网络通信中的有效性。

TCP协议确保了数据的可靠传输，包括错误检测、流量控制和拥塞控制。TCP协议的可靠性和稳定性在语音通信中尤其明显，与使用UDP协议的同学的实验效果相比，我们使用TCP协议实现的语音通话质量高很多，但同时使用UDP的同学在传输速率上比我们使用的TCP协议进行的文件传输要快许多。

Socket编程的可行性：

Python的socket库提供了简单易用的API，使得实现C/S模式的网络通信变得可行。

通过对socket的编程，可以深入理解网络通信的底层机制。

客户端与服务器的交互：

客户端能够成功连接到服务器，并发送和接收数据。

服务器能够监听来自客户端的连接请求，处理请求并返回响应。

多线程/多进程处理的优势：

实验中使用了多线程或多进程来处理并发连接，可以观察到这种设计能够显著提高服务器的性能和吞吐量。

多线程/多进程的设计使得服务器能够同时处理多个客户端的请求。

我们在此实验中使用多线程，与多进程相比，它更轻量级更节约资源，也更便于对变量进行共享，但同时也引起了对临界资源需的竞争问题，需要引入线程锁来解决。

异常处理和错误检测来提高程序鲁棒性：

实验中的异常处理机制能够有效地处理网络通信中的错误和异常情况，如连接中断、数据传输错误等。

通过错误检测和处理，可以确保程序的健壮性和稳定性。

1. 总结及心得体会

经过本次实验，我产生了以下心得体会

基础知识的重要性：

在实验过程中，我深刻体会到了计算机网络基础知识的重要性，如TCP/IP协议栈的工作原理、socket编程的基本概念等。这些基础知识是理解和实现网络通信的关键。

动手实践的价值：

通过动手实践，我不仅加深了对理论知识的理解，还学到了很多实际编程中的技巧和经验。比如如何处理网络异常、如何优化数据传输效率等。也让我更加直观的认识到了UDP和TCP两大传输协议之间的区别，对其有了更直观的认识。

调试与排错的挑战：

在实验过程中，由于初次进行较复杂的网络通信编程+多线程模式，调试过程中遇到了很多问题。经过本次实验，获得了多线程调试程序的经验。

对网络通信的深入理解：

通过这次实验，我对网络通信有了更加深入的理解。我了解了数据如何在网络中传输、如何确保数据的完整性和安全性等问题。让我对计算机网络有了更深入的理解。

附件

1.源码文件

2.相关文档

# 自测报告内容：

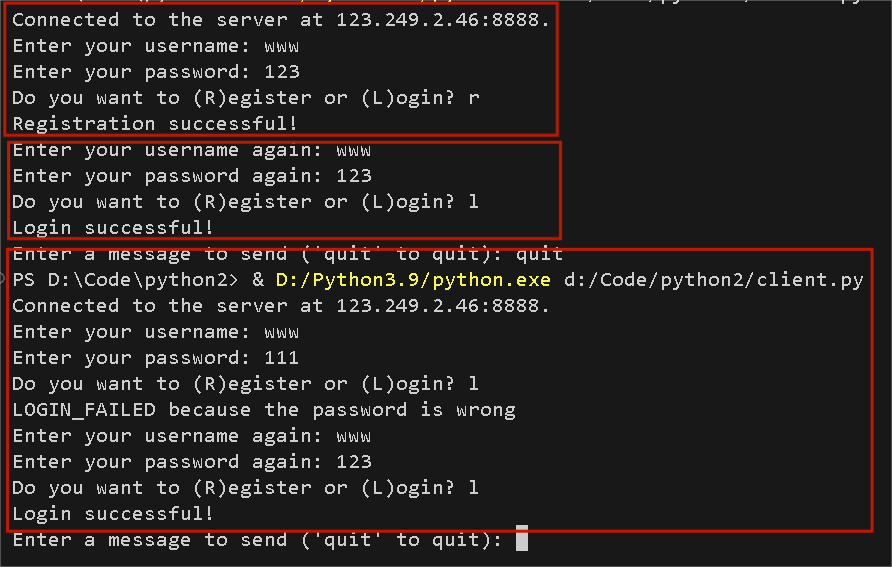
8.2.2 实现一个简单的聊天程序

一、说明完成的基本功能的情况，并充分测试，贴图说明。

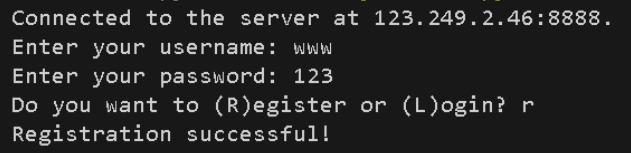
**基本功能测试：**

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）。

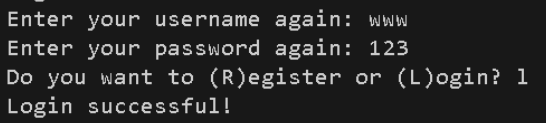
这是一个总体的新客户端连接服务器，退出然后重新登录的过程



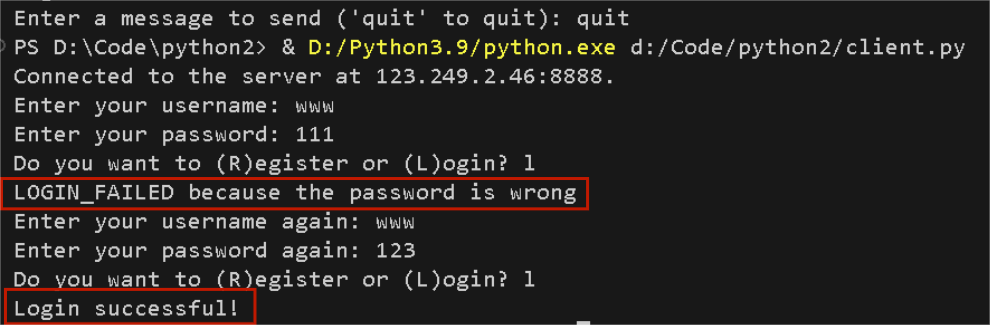
过程分为三个小步骤，首先新用户连接服务器需要注册账号



这里注册新用户www，密码123，注册完成后需要重新登录

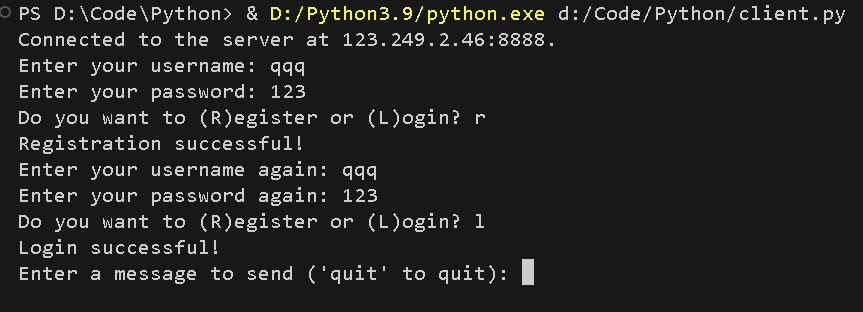


重新登录这里密码正确，登录成功。然后退出账号重新登录



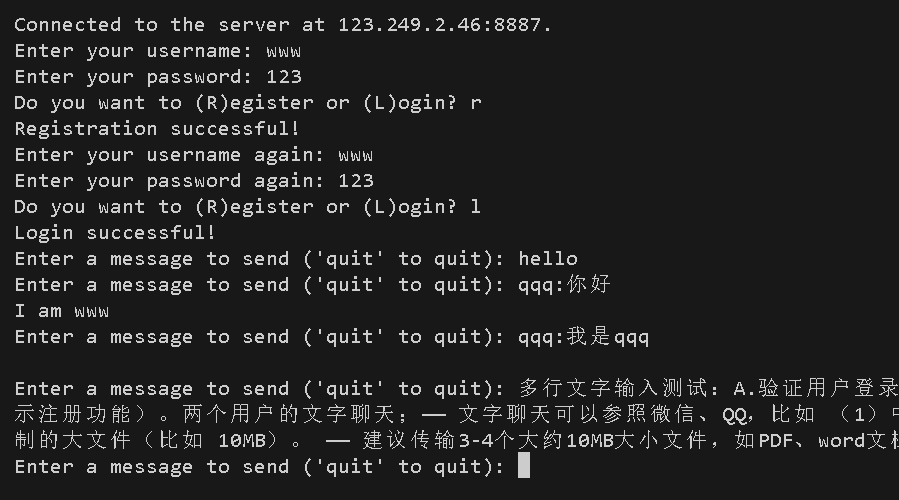
这里输入了错误的密码111，所以登录失败，然后再用正确的密码123登录，登录成功。

下面是另一个客户端新用户qqq登录过程

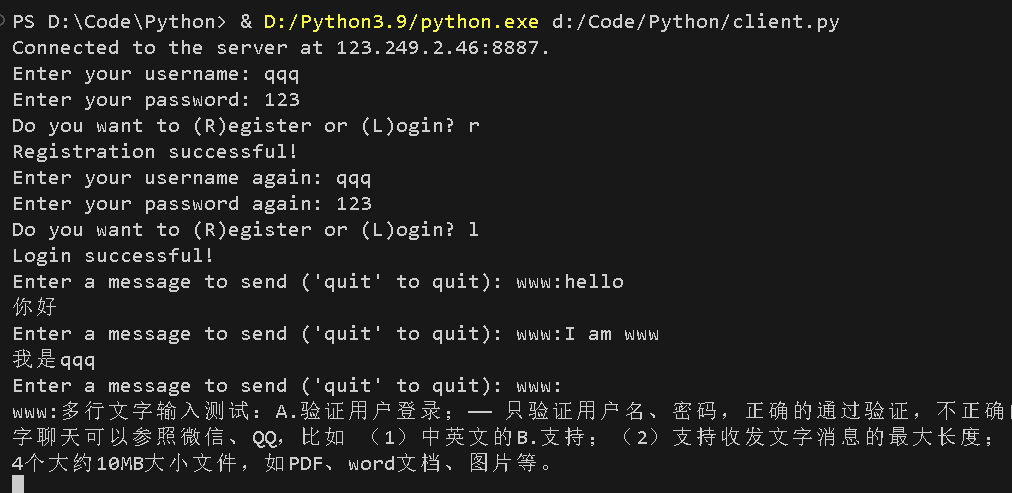


1. 两个用户的文字聊天；—— 文字聊天可以参照微信、QQ，比如 （1）中英文的支持；（2）支持收发文字消息的最大长度； （3）支持多行文字吗？

用户www界面

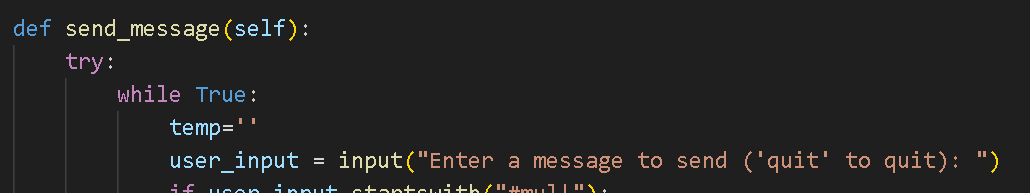


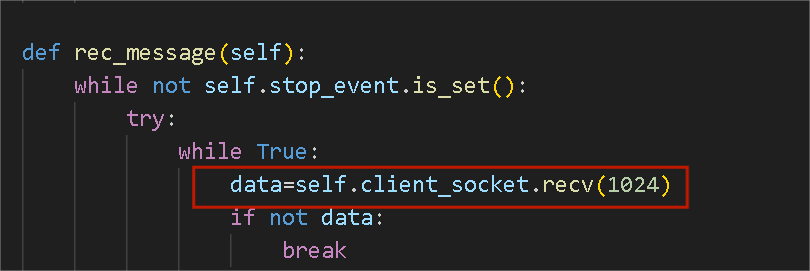
另一个用户qqq界面



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。

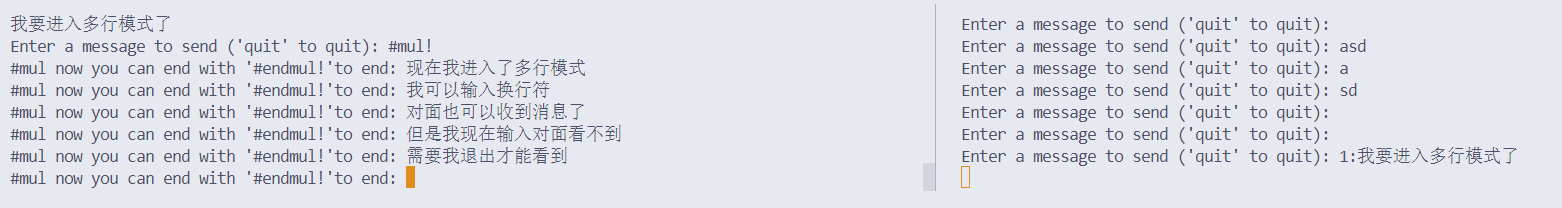
至于收发消息最大长度，发送消息没有长度限制，接收消息限制最大长度为1024字节，这点从代码中分析体现。所以虽然发送方可以无限输入消息，但由于接收方接收窗口为1024字节，其实际发送出的消息也就只有1024字节。



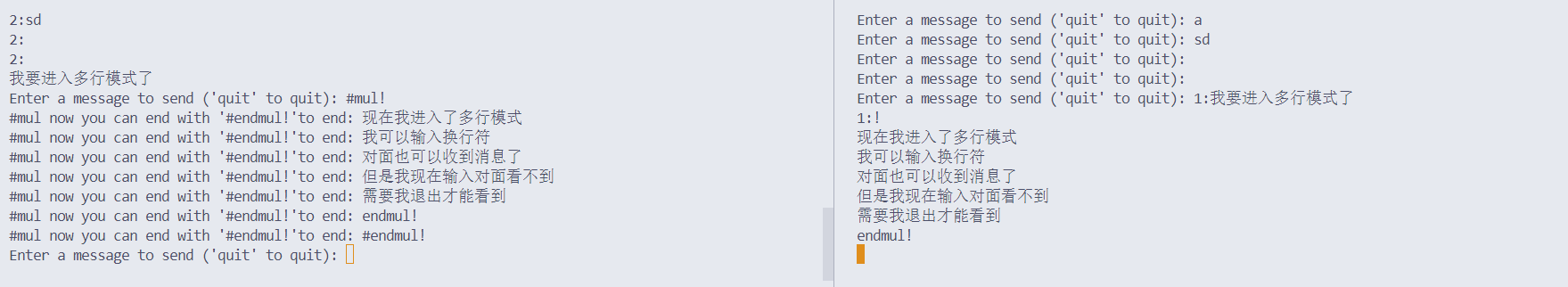


1. 多行文字输入展示：

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：

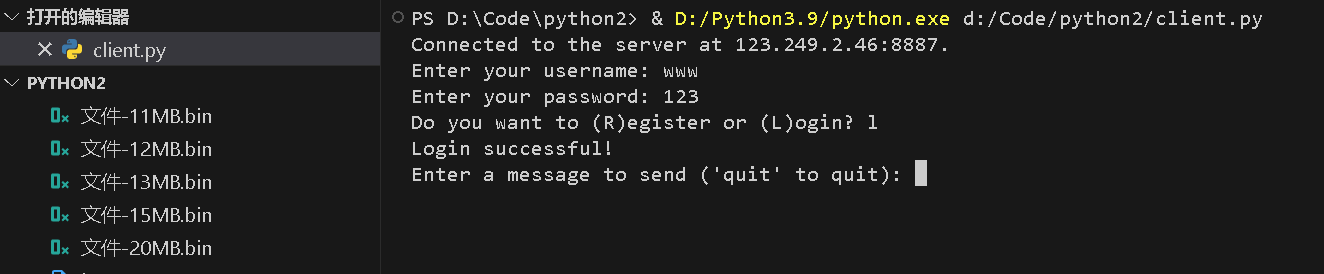


接收方直接接受大段文字。

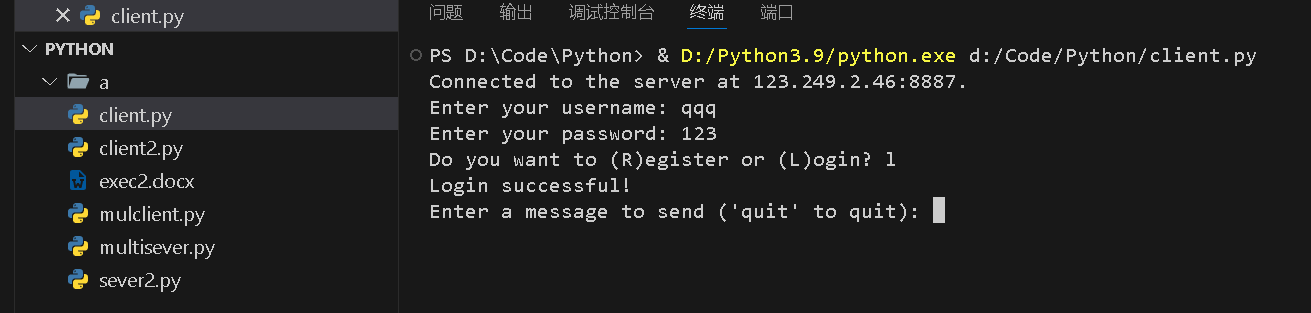
1. 用户之间传输文件，包括二进制的大文件（比如 10MB）。 —— 建议传输3-4个大约10MB大小文件，如PDF、word文档、图片等。

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

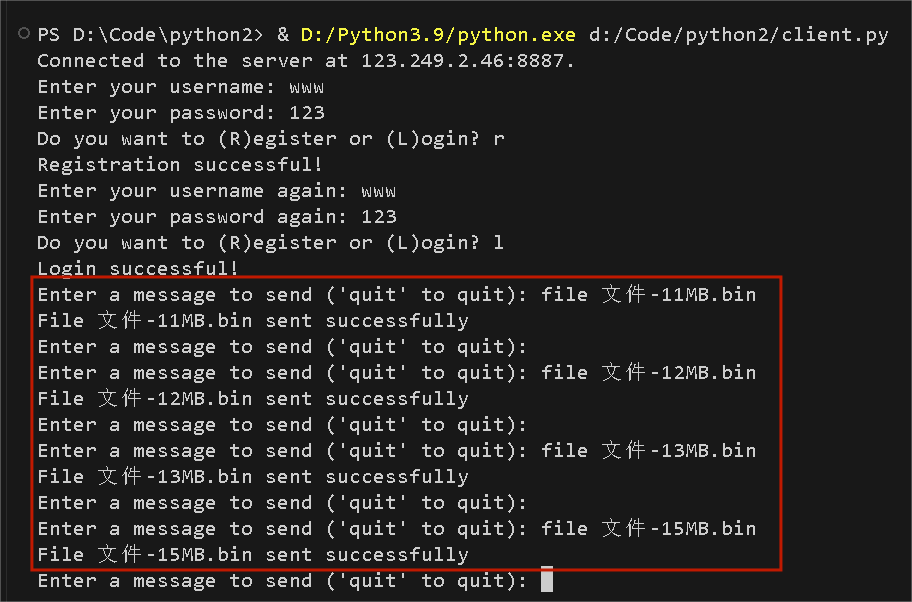


（2）这是用户qqq的界面，可以看到在当前目录下没有与www相同的文件



1. 接下来是传输过程

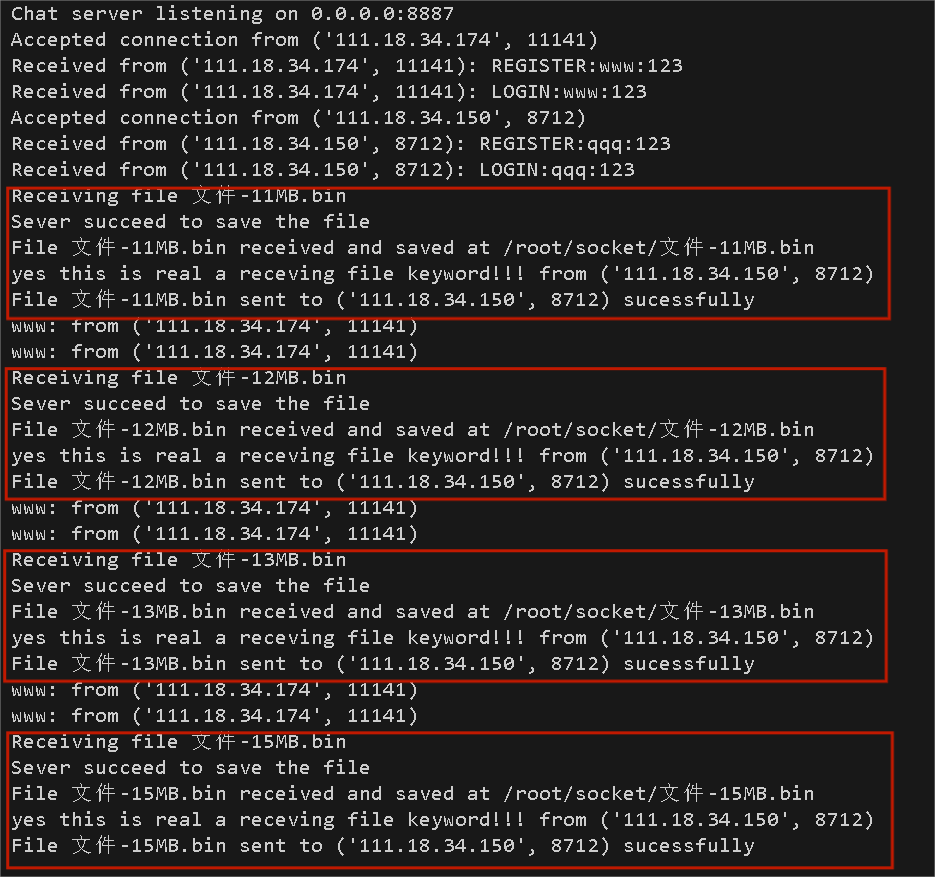
用户www发送文件界面



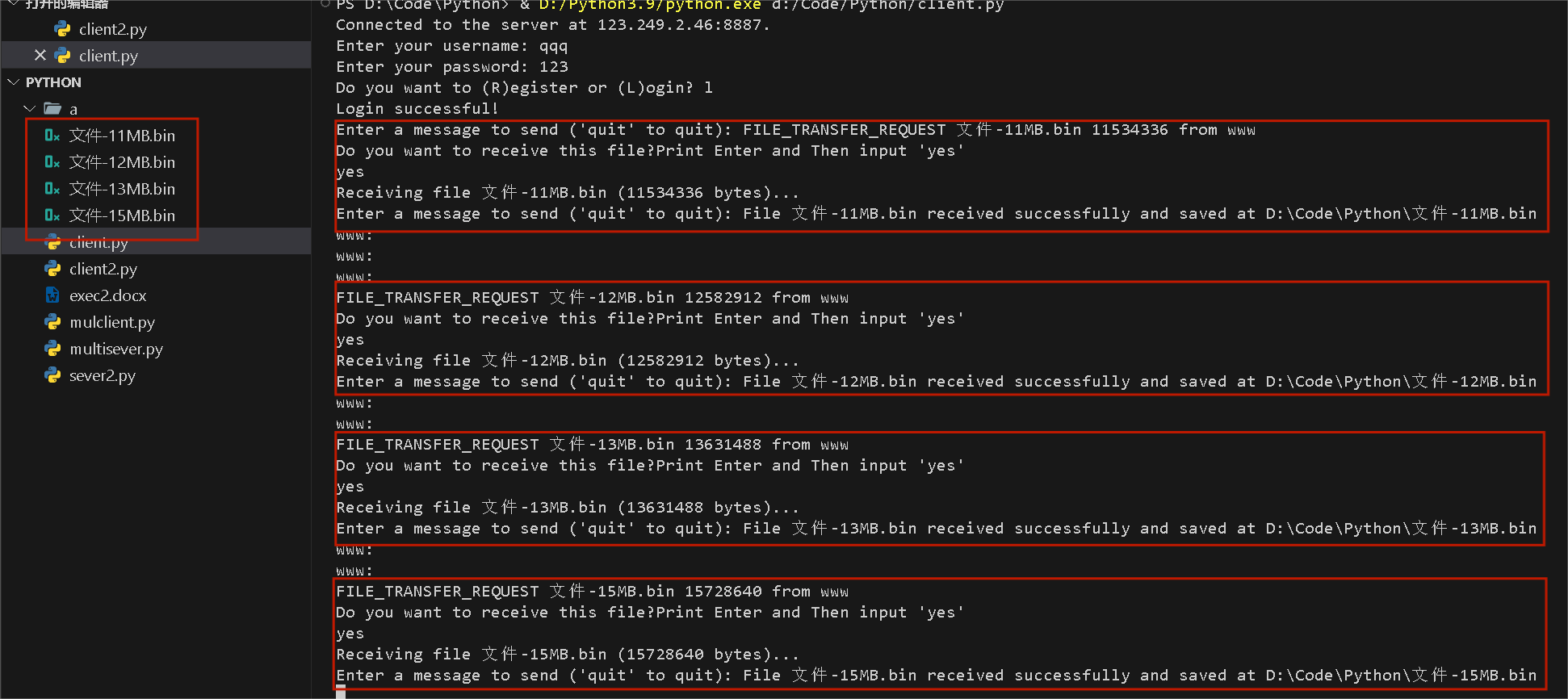
用户www通过输入指令file <file\_path>来发送指定路径下的文件，这里文件在当前目录下直接输入文件名，然后将文件发送到服务器，经服务器下载再广播给其他用户。如图www进行了4次不同大小文件发送，并且都显示发送成功。

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。

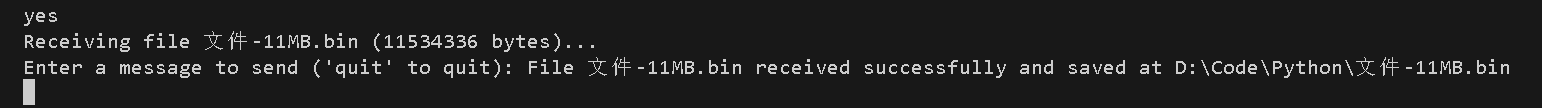
具体文件接收过程如下：当服务器将文件广播出去时，除了发送文件的客户端之外的其他客户端收到一条消息显示“FILE\_TRANSFER\_REQUEST <文件名> <文件大小> from <用户名>”，



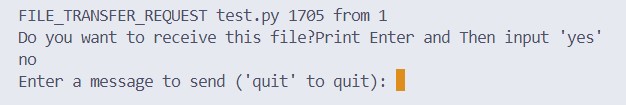
然后选择是否接收该文件



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。

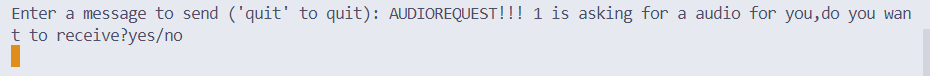
**密码加密传输：**用户在登录时，密码是以明文形式发送给服务器的。为了增加安全性，可以在客户端对密码进行加密处理，然后再发送给服务器，服务器端再对密码进行相应的解密验证。这样可以避免密码在网络传输过程中被窃取。

**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。

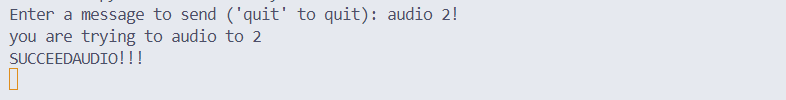
**注册结果处理**：服务器在用户注册时，会返回注册成功或失败的消息。如果用户名已经被注册，服务器会返回注册失败的消息，这样可以防止同一用户名被多次注册。

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：

