实验八自测报告

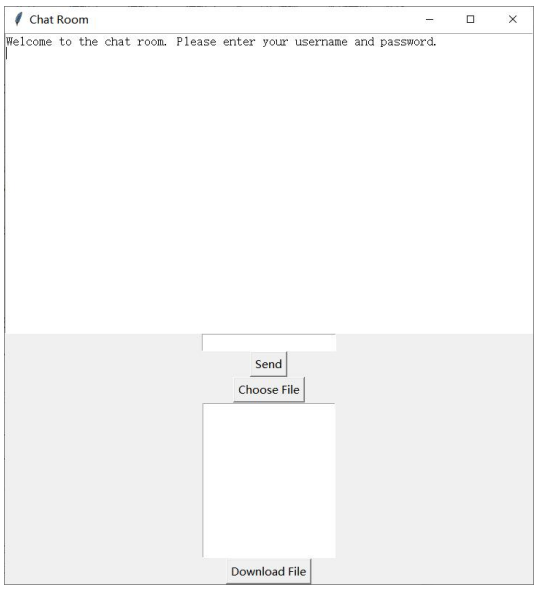
实现一个简单的聊天程序

一、说明完成的基本功能的情况，并充分测试，贴图说明。

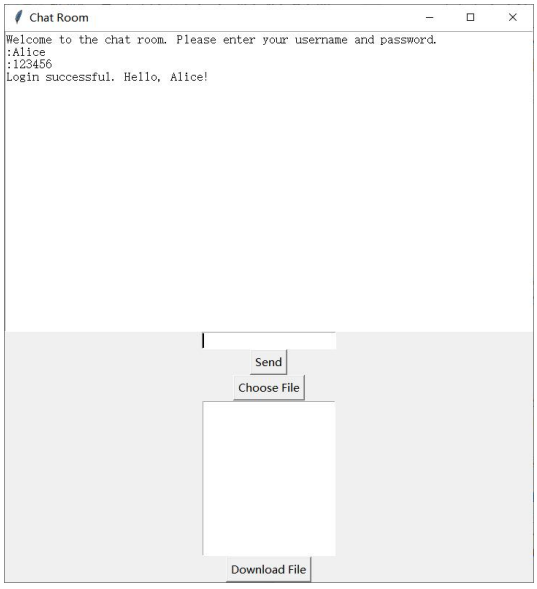
基本功能测试：

A. 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）。

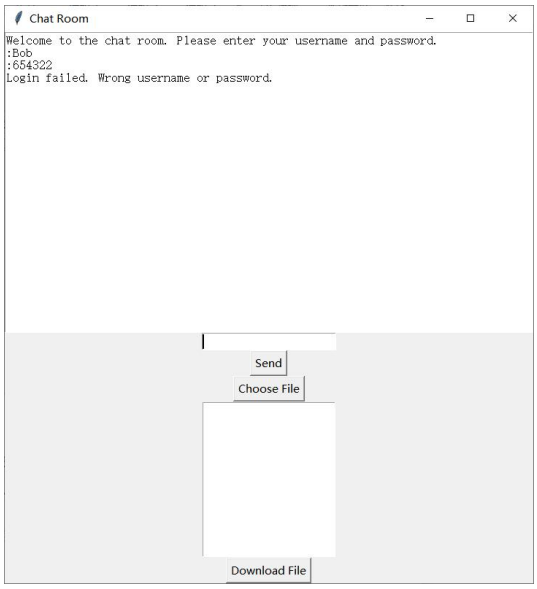
客户端启动后提示输入用户名和密码进行登录：



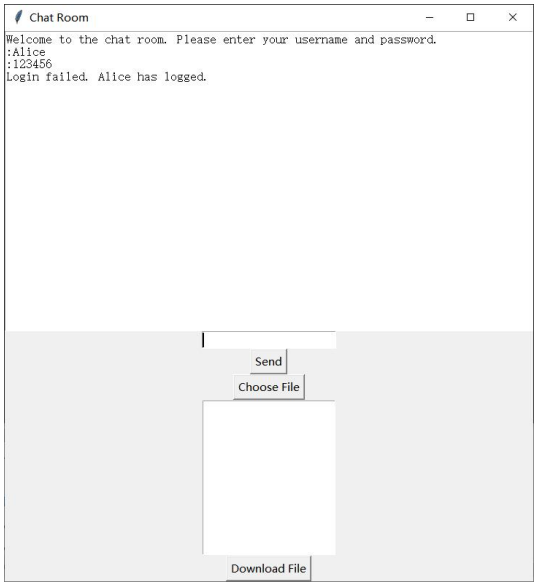
输入用户名和密码，经服务器验证正确后发送欢迎消息：



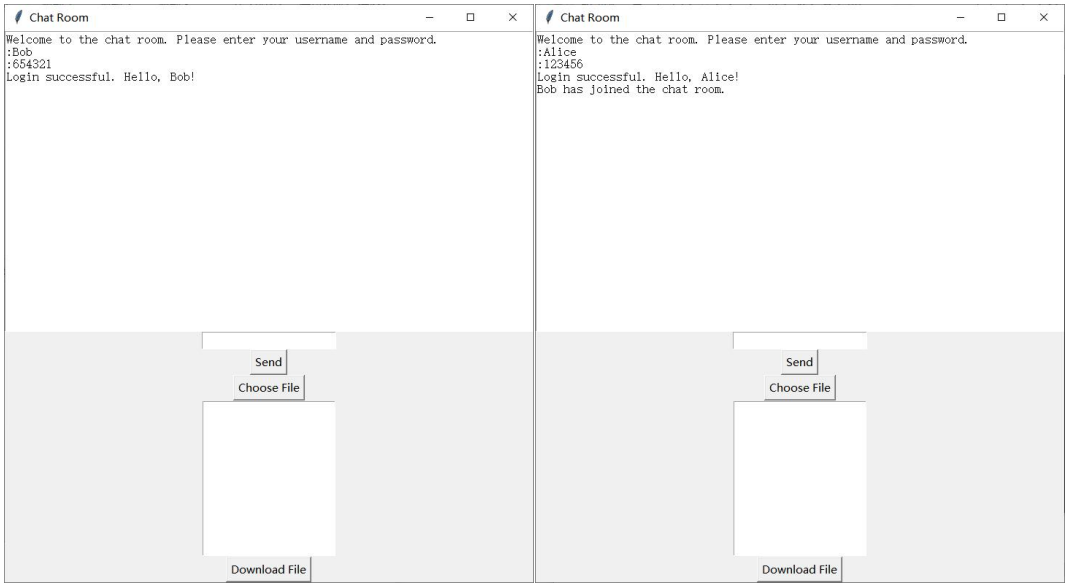
若错误则提示并断开连接：



若试图登录至一个已登录的账号则同样会提示并断开连接：

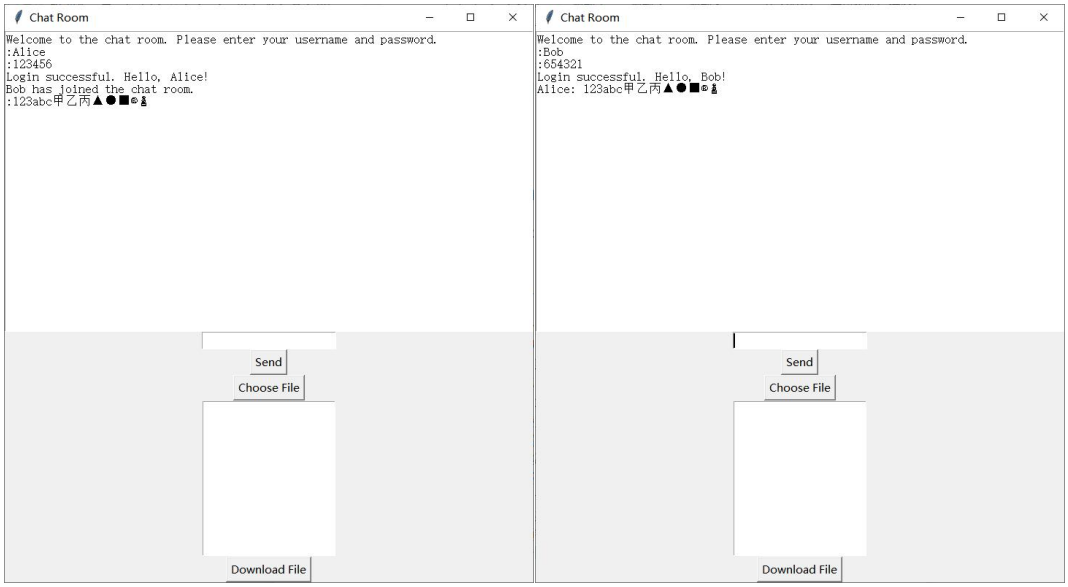


正确登录后会在其他已登录的客户端提示用户上线：



B. 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；

中英文和一些特殊字符均支持发送：



（2）支持收发文字消息的最大长度；

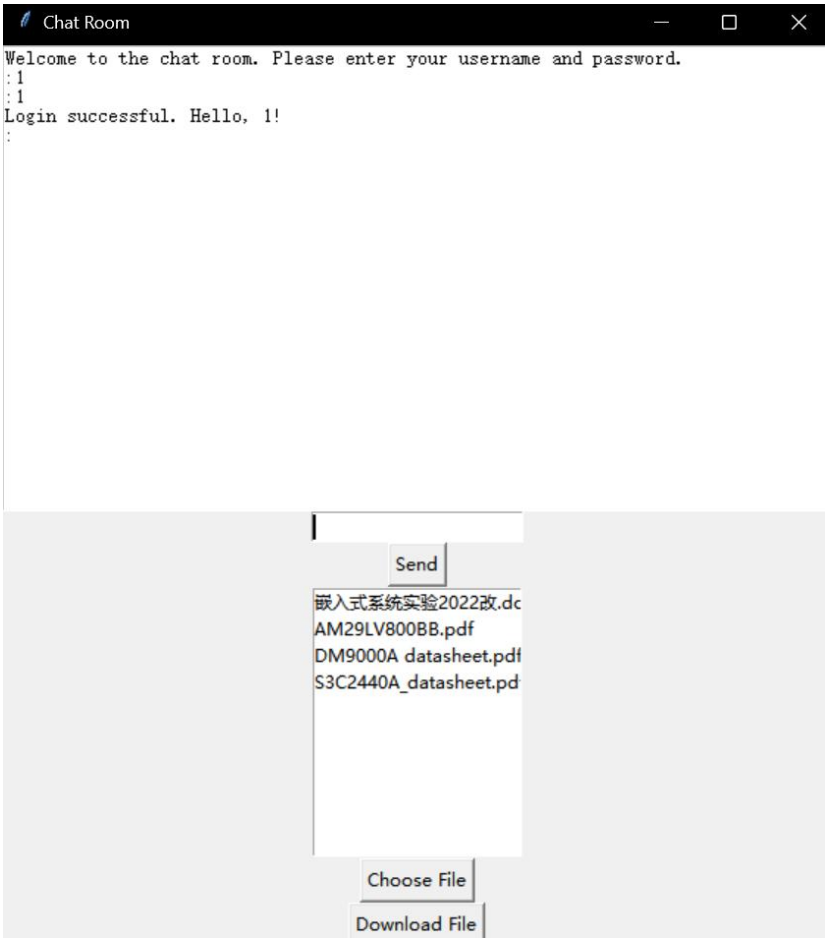
支持收发文字的最大长度是（1022-用户名长度），总消息最大长度为 1024，包含消息发送者':'消息，故文字消息最大长度为（1022-用户名长度）。

（3）支持多行文字吗？

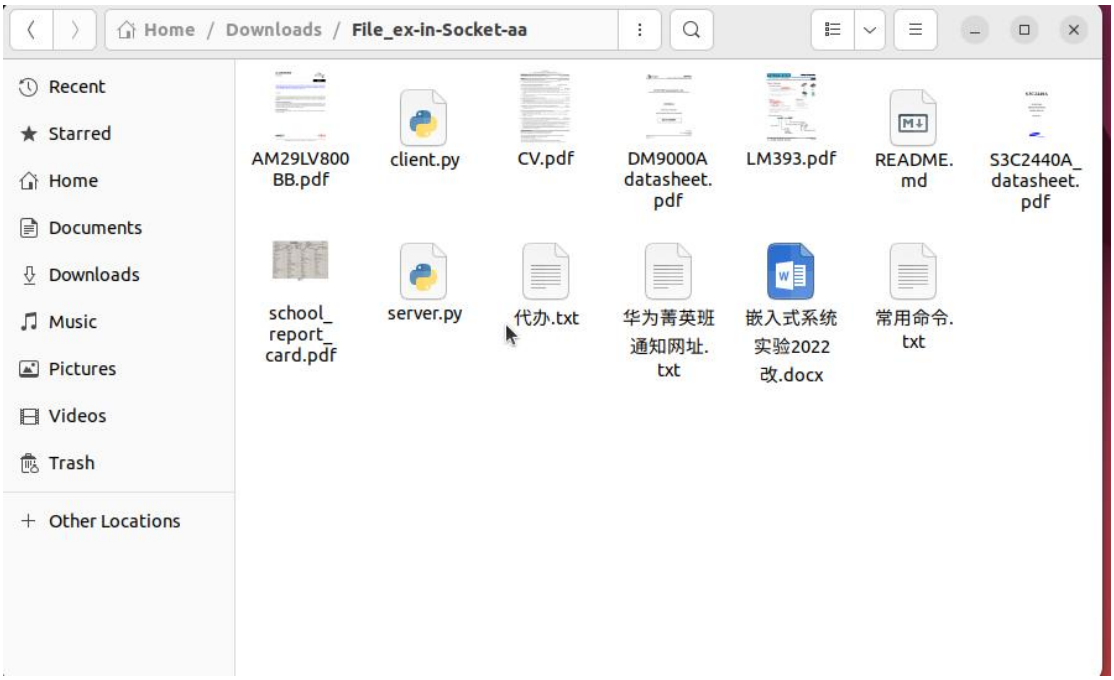
不支持多行文字发送，可以分多次发送多行文字。

C. 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。

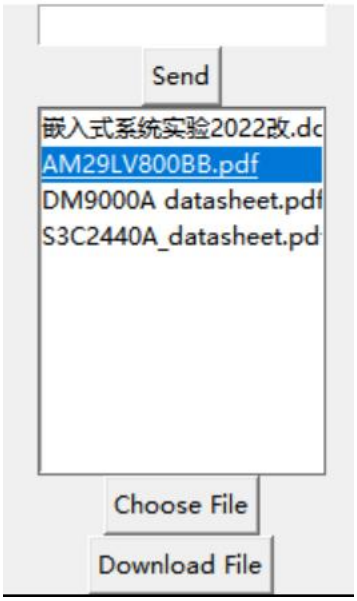
如下图传输了四个大于 10M 的文件

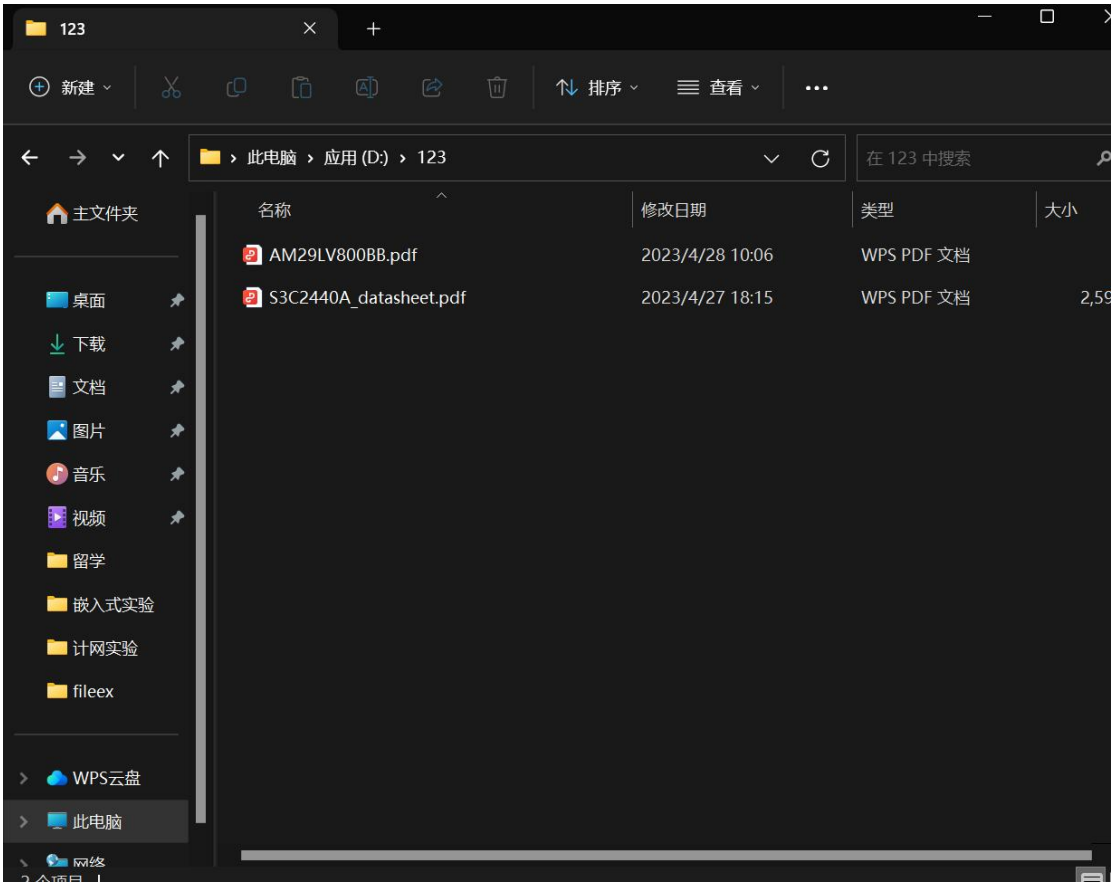


服务器端接受到文件

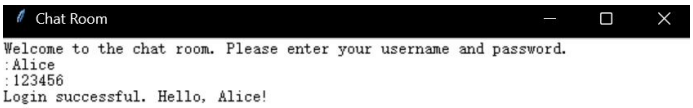


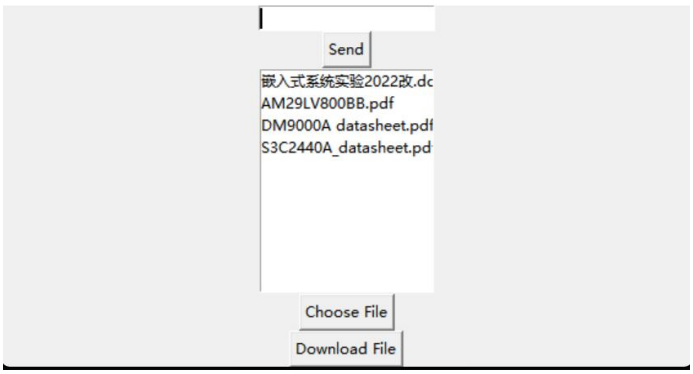
用户端可以下载文件





新的用户登录获取所有文件列表，并且可以下载：





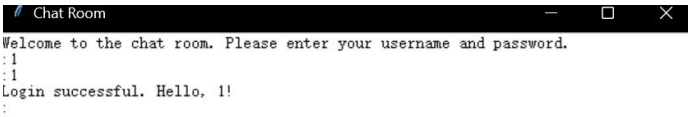
二、说明完成的高级功能的情况，并测试，贴图说明。

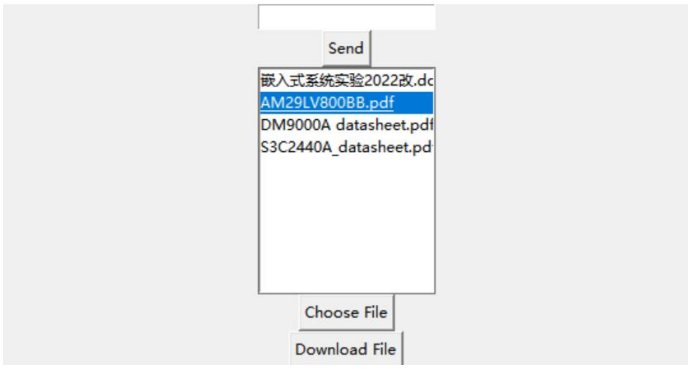
高级功能测试：

离线文件的支持；—— 给出具体的测试说明。

由于文件传输的方法类似于自定义的 FTP，每次会将文件保存在服务器，因此支持离线文件功能

将服务器关闭并重新打开，会发现仍然可以下载文件：





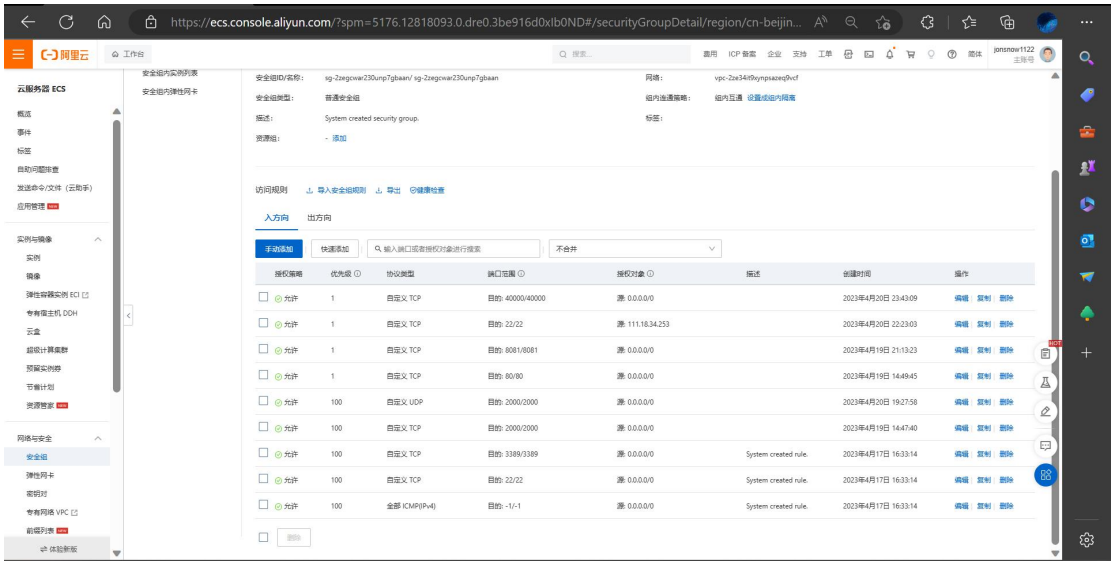
三、所有的测试必须在广域网上进行（建议服务器程序运行在云服务器上，客户端程序运行在本地的 PC 电脑上）。

使用阿里云 ubuntu 云服务器和本地电脑进行端口匹配

服务器公网 ip:112.126.75.159

服务端口为 2000

防火墙和安全组：



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;

本程序使用 C/S 模式通信。

（2）说明使用的具体协议。

本程序使用 TCP 协议通信。

（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】

服务器支持并发用户数在程序中不做限制，由服务器系统资源和内存限制。

（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。

由于文件收发和聊天使用同一条信道，故不支持同时进行收发文件和聊天。

（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）。

通过多进程方式实现多个并发 TCP 连接，每次有客户端登录时都会创建一个进程用于服务器与该客户端通信。

（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。

登录消息：服务器端存储了所有用户的用户名和密码，每次用户需要输入正确的用户名和密码，才能连接服务器开启服务

聊天文字：通过一个字符串通过 pickle.dumps()函数进行序列化传输，在服务器端使用 loads()函数反序列化解读 message。

文件传输：通过循环读取 message 完成大文件传输

（7）如果登录有安全性方面的设计，请特别说明。

五、自测报告中简要介绍 代码实现情况。

运行代码时先运行 server.py,再运行多个 client.py 作为客户端，代码细节解释见注释

client.py

```
import os
import socket
import threading
import pickle
import tkinter as tk
import time
from tkinter import filedialog



# 创建一个 socket 对象
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
flag = True
username = 1
BUFFER_SIZE = 1024    # 缓冲区大小
# 连接到服务器的 IP 地址和端口号
client.connect(("112.126.75.159", 2000))
#client.connect(("127.0.0.1", 8888))
# 创建一个图形用户界面
window = tk.Tk()
window.title("Chat Room")

# 创建一个文本框，显示聊天消息
text = tk.Text(window)
text.pack()

# 创建一个输入框，输入聊天消息
entry = tk.Entry(window)
entry.pack()

# 创建一个函数，发送聊天消息
def send_message():
    global flag, username
    if flag:
        # 获取输入框的内容，并清空输入框
        message = entry.get()

        # 判断消息是否以/开头，表示特殊指令
        if message.startswith("/"):
            # 判断是否是下载指令，格式为/download 文件名
```

```
                if message.startswith("/download "):
                    # 获取要下载的文件名
                    filename = message.split()[1]

                    # 向服务器发送下载请求，包含自己的 IP 地址和文件名，使用 pickle 模块序列化
                    client.send(pickle.dumps([client.getsockname(), filename]))
                else:
                    # 如果是其他指令，就在文本框中显示无效指令的消息
                    text.insert(tk.END, "Invalid command.\n")
            else:
                # 如果是普通的聊天消息，就直接发送给服务器，使用 pickle 模块序列化
                if isinstance(username, int):
                    username = message
                client.send(pickle.dumps(message))
                text.insert(tk.END, ":" + message + "\n")
        entry.delete(0, tk.END)

# 创建一个按钮，点击时调用 send_message 函数
button = tk.Button(window, text="Send", command=send_message)
button.pack()

# 创建一个函数，选择要发送的文件
def choose_file():
    global flag
    if flag:
        # 弹出一个文件选择器，获取选择的文件路径
        filepath = filedialog.askopenfilename()

        # 判断是否选择了文件
        if filepath:
            # 获取文件名和文件内容
            filename = filepath.split("/")[-1]
            filesize = os.path.getsize(filepath)
            message = (filename, filesize)
            client.send(pickle.dumps(message))

            with open(filepath, 'rb') as f:
                while True:
                    data = f.read(BUFFER_SIZE)
                    if not data:
                        break
                    client.send(data)
```

```
                listbox.insert(tk.END, filename)

# 创建一个列表框，显示已发送的文件列表
listbox = tk.Listbox(window)
listbox.pack()

# 创建一个按钮，点击时调用 choose_file 函数
button = tk.Button(window, text="Choose File", command=choose_file)
button.pack()



# 创建一个函数，下载选中的文件
def download_file():
    global flag
    if flag:
        # 获取列表框中选中的文件名
        filename = listbox.get(tk.ACTIVE)
        print('df')
        # 向服务器发送下载请求，包含自己的 IP 地址和文件名，使用 pickle 模块序列化
        client.send(pickle.dumps([username, filename]))
        #client.getsockname()

# 创建一个按钮，点击时调用 download_file 函数
button = tk.Button(window, text="Download File", command=download_file)
button.pack()

# 定义一个函数，接收服务器的消息
def receive_message():
    # 循环接收服务器的消息，使用 pickle 模块反序列化
    while True:
        message = pickle.loads(client.recv(1024))
        print(type(message))
        # 判断消息的类型
        #if not message:
        #    continue
        if isinstance(message, str):
            # 如果是字符串，就是普通的聊天消息，直接在文本框中显示
            text.insert(tk.END, message + "\n")
            if message.split()[0] == "Login" and message.split()[1] == "failed.":
                flag = False
                break
        elif isinstance(message, dict):
            # 如果是字典，就是已发送的文件列表，循环遍历并添加到列表框中
```

```
                for filename in message:
                    listbox.insert(tk.END, filename)
            elif isinstance(message, list):
                #message = pickle.loads(client.recv(1024))
                filename, filesize = message
                print('ls')
                with open(f'D:/123/{filename}', 'wb') as f:
                    received_size = 0
                    filesize = int(filesize)
                    print(1)
                    while received_size < filesize:
                        print('#')
                        data = client.recv(BUFFER_SIZE)
                        received_size += len(data)
                        f.write(data)
            '''
            elif isinstance(message, bytes):
                # 如果是字节串，就是文件对象，包含文件内容
                filecontent = message

                # 弹出一个文件保存器，获取要保存的文件路径
                filepath = filedialog.asksaveasfilename()

                # 判断是否选择了文件路径
                if filepath:
                    # 将文件内容保存到本地，并在文本框中显示文件名
                    with open(filepath, "wb") as f:
                        f.write(filecontent)
                    text.insert(tk.END, f"File {filepath.split('/')[-1]} saved.\n")'''


# 创建一个线程，执行 receive_message 函数
thread = threading.Thread(target=receive_message)

# 启动线程
thread.start()

# 进入图形用户界面的主循环
window.mainloop()
```

server.py

```
import os
import socket
import threading
```

```
import pickle
import time

# 创建一个 socket 对象
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
BUFFER_SIZE = 1024  # 缓冲区大小
# 绑定 IP 地址和端口号
server.bind(("127.0.0.1", 8888))

# 监听连接请求
server.listen()

# 定义一个字典，存储用户名和密码
users = {"Alice": "123456", "Bob": "654321", "Charlie": "111111", "1": "1"}

# 定义一个列表，存储已连接的客户端
clients = []

# 定义一个字典，存储已连接用户
cliname = {}

# 定义一个字典，存储已发送的文件对象
files = {}


def receive_file(client):
    # 接收文件大小字符串，并转换成整数
    filesize_str = client.recv(10).decode()
    filesize = int(filesize_str)

    # 用一个变量记录已接收的字节数
    received = 0

    # 创建一个空的字节对象，用于存储文件内容
    filecontent = b""

    # 在一个循环中不断地接收文件内容，直到接收完毕
    while received < filesize:
        # 每次接收 4096 个字节
        chunk = client.recv(4096)

        # 将数据块追加到文件内容中
        filecontent += chunk
```

```
            # 更新已接收的字节数
            received += len(chunk)

        # 返回文件内容
        return filecontent


# 定义一个函数，处理每个客户端的消息
def handle_client(client):
        # 获取客户端的地址
        address = client.getpeername()
        print(f"New connection from {address}")

        # 向客户端发送欢迎消息
        client.send(pickle.dumps("Welcome to the chat room. Please enter your username and password."))

        # 接收客户端的用户名和密码
        try:
            username = pickle.loads(client.recv(1024))
            password = pickle.loads(client.recv(1024))
        except:
            print(f"{address} has disconnected")
            return

        # 验证用户名和密码是否正确
        if username in users and users[username] == password:

            #验证用户是否已经登陆
            if username in cliname.keys():
                client.send(pickle.dumps(f"Login failed. {username} has logged."))
            else:
                # 如果正确，向客户端发送登录成功消息，并广播给其他客户端
                client.send(pickle.dumps(f"Login successful. Hello, {username}!"))
                broadcast(f"{username} has joined the chat room.", client)

                # 将客户端添加到列表中
                clients.append(client)
                cliname[username] = client.getpeername()

                # 向客户端发送已发送的文件列表
                client.send(pickle.dumps(files))

                # 循环接收客户端的消息
```

```
                while True:
                #try:
                    # 接收客户端的消息，使用 pickle 模块反序列化
                    message = pickle.loads(client.recv(1024))

                    # 判断消息的类型
                    if isinstance(message, str):
                        if message.startswith("@"):
                            ater = message.split()[0][1:]
                            if ater in cliname.keys():
                                send_message(f"{username}: {message}", cliname[ater])
                            else:
                                client.send(pickle.dumps(f"user {ater} not found!"))
                        else:
                            # 如果是一般字符串，就是普通的聊天消息，直接广播给其他客
户端
                            broadcast(f"{username}: {message}", client)
                    elif isinstance(message, tuple):
                        # 如果是元组，就是文件对象，包含文件名和文件内容
                        #message = client.recv(BUFFER_SIZE)
                        filename, filesize = message
                        files[filename] = "1"
                        with open(filename, 'wb') as f:
                            received_size = 0
                            filesize = int(filesize)
                            while received_size < filesize:
                                data = client.recv(BUFFER_SIZE)
                                received_size += len(data)
                                f.write(data)

                        # 广播给其他客户端有新文件可下载
                        broadcast(f"{username} has sent a file: {filename}. You can download it
by typing /download {filename}", client)
                    elif isinstance(message, list):
                        # 如果是列表，就是下载请求，包含请求者的用户名和要下载的文件
名
                        requester, filename = message

                        # 判断文件名是否在字典中
                        if filename in files.keys():
                            # 如果在，就将对应的文件对象发送给请求者
                            send_file(filename, requester)
                        else:
                            # 如果不在，就向请求者发送文件不存在的消息
```

```
                            send_message(f"File {filename} does not exist.", requester)
                    '''
                    except:
                        # 如果发生异常，就表示客户端断开连接，从列表中移除，并广播
                        clients.remove(client)
                        broadcast(f"{username} has left the chat room.", client)
                        print(f"{address} has disconnected")
                        break
                    '''
        else:
            # 如果不正确，向客户端发送登录失败消息，并断开连接
            client.send(pickle.dumps("Login failed. Wrong username or password."))
            client.close()

# 定义一个函数，向所有其他客户端广播消息
def broadcast(message, sender):
    # 循环遍历客户端列表
    for client in clients:
        if client != sender:
            client.send(pickle.dumps(message))

# 定义一个函数，向指定的客户端发送消息，使用 pickle 模块序列化
def send_message(message, receiver):
    # 循环遍历客户端列表
    for client in clients:
        # 获取客户端的地址
        address = client.getpeername()
        # 如果地址的第一个元素（IP 地址）等于接收者的用户名，就向其发送消息
        if address == receiver:
            client.send(pickle.dumps(message))
            break

# 定义一个函数，向指定的客户端发送文件对象，使用 pickle 模块序列化
def send_file(filename, receiver):
    # 循环遍历客户端列表
    for client in clients:
        # 获取客户端的地址
        address = client.getpeername()
        # 如果地址的第一个元素（IP 地址）等于接收者的用户名，就向其发送文件对象
        if address == receiver:
            #client.send(pickle.dumps(filecontent))
            filesize = os.path.getsize(filename)
            message = [filename, filesize]
            client.send(pickle.dumps(message))
```

```
                with open(filename, 'rb') as f:
                    while True:
                        data = f.read(BUFFER_SIZE)
                        if not data:
                            break
                        client.send(data)
                break

# 打印服务器启动的消息
print("Server is running...")

# 循环接受连接请求
while True:
    # 接受一个连接请求，并返回一个客户端对象
    client, address = server.accept()
    time.sleep(1)

    # 创建一个线程，执行 handle_client 函数，传入客户端对象作为参数
    thread = threading.Thread(target=handle_client, args=(client,))

    # 启动线程
    thread.start()
```