Lionel Kha (z5317093)
Language used: Python 3.7.3

Structure

Client.py: client entry file
- Takes in an input which is the server port number

Server.py: server entry file
- Same input as Client.py

Credentials.txt: a list of registered usernames and their associated passwords

The messages sent between the client and server are stored in a JSON format. Each message sent to the server, except for the authentication process, will contain the user's command and their username and any other arguments required. The server will send back a message depending on the outcome of the command e.g. success, fail, thread does not exist etc.

How authentication works:

The client side will prompt the user to enter a username which will be sent to the server. The server will check Credentials.txt and reply with one of the following message:

“New user”
- The client will ask the user for a password to register
- User enters a password and the server will store the username and password in credentials.txt

“Already logged in”
- The server has checked the list of active users and found the inputted username has been logged in
- The client will ask the user for another username

“Existing user”
- The server has checked that the user is registered and is not logged in
- The client will ask the user for the password
- If the password is incorrect, the client will continue to ask for the password
- There is not limit to this as the spec does not specify

How each command works:

Once the user has logged in, the client will prompt the user a list of possible commands they can enter. The user can enter a command and the required arguments. If the wrong amount of arguments are provided, the client will notify the user to enter a new command.

Create thread (CRT):
- The client will send a JSON message containing the command name, username and the thread name to the server
- The server checks if the thread exists, which is done by checking for the thread’s name in the list of threads
- If successful, the thread name will be added to the list and the user is notified on the client side
- If unsuccessful, the client will prompt the user for another command

Remove Thread (RMV)

- The server checks if the thread exists and using os.remove() to remove the thread from the server directory
- If the thread doesn't exist, an error message will appear on the client side and the user is prompted for another command

List threads (LST)
- The server will check if there are any threads in the thread list
- If there are none, the client side will notify the user
- If threads exist, the server uses pickle.dumps() to send the list to the client to print to the user

Read thread (RDT)
- The server will create an list and store each line of the selected file except for the first line (which states the owner of the file)
- The server uses pickle.dumps() to send the list to the client to print to the user

Post message (MSG)
- The server opens the thread if it exists and count the number of existing messages
- The server then appends the message with the message number, owner and the message to the thread file

Edit message (EDT)
- The server finds the corresponding message with the correct number and edits the message after checking ownership

Delete message (DLT)
- The server rewrites the entire file except for the selected message
- After finding the desired message, every line that has a digit as the first character is edited to move the message number by 1

Upload file (UPD)
- After checking all the requirements (if thread exists and if file exists in thread), a TCP connection is opened
- The file is transferred as a copy into the server directory
- TCP connection is closed after the file has been transferred

Download file (DWN)
- After checking all the requirements, a TCP connection is opened
- The file is transferred as a copy into the client directory from the server directory
- TCP connection is closed after the file has been transferred

Trade off

- When removing a thread, all files that are part of it are not deleted from the directory but users cannot download it because the server checks the thread file if a user has uploaded files.
- JSON is used to send messages between the client and the server
- I have decided to use python lists to keep track of the active threads and users which allows me to easily append and remove elements when required.
- Pickle is used for RDT and LST to send a list to the client side
    - Client side can't access the thread files because it is another directory
- Multi threading has been incorporated to allow multiple users from different directories to simultaneously communicate with the server

- There was a problem with the tcp connection on the client side, So I had to fix it with time.sleep() as suggested on this ED post but it does not 100% fix the problem.
- I implemented a try and except for when the server side is abruptly killed
- Will not work if tcp is used multiple times and the server closed and opened (But assumed it will not be restarted when marking)
- **NOTE:** when starting the server and an error appears, please try a different port number

Edge cases

- The upload command will not work if a user is called “uploaded” and has posted a message which contains the filename. For example: 1. uploaded: test.txt
- If a user post multiple messages in a thread and removes all of them, there is a trailing new line
    - This trailing new line will break commands like EDT and DLT

Borrowed code:

- Some parts of my UDP and TCPcode have been adapted from:
    - https://webcms3.cse.unsw.edu.au/COMP3331/22T1/resources/70201
    - https://webcms3.cse.unsw.edu.au/COMP3331/22T1/resources/70203
    - https://webcms3.cse.unsw.edu.au/COMP3331/22T1/resources/70202
    - https://webcms3.cse.unsw.edu.au/COMP3331/22T1/resources/70204
    - The sample code provided
- The delete command includes some of this code:
    - https://stackoverflow.com/questions/4710067/how-to-delete-a-specific-line-in-a-file
- The upload command includes some of the following code:
    - https://stackoverflow.com/questions/27241804/sending-a-file-over-tcp-sockets-in-python
    - https://nikhilroxtomar.medium.com/file-transfer-using-tcp-socket-in-python3-idiot-developer-c5cf3899819c
- The download command includes some of the following code:
    - https://stackoverflow.com/questions/29110620/how-to-download-file-from-local-server-in-python