COMP3331 Assignment Report

HANQING GUAN(z5213081)

This is COMP3331 assignment report. I will describe my code about the design of the client and server, authentication and 11 commands. I implemented functionality for handling multiple concurrent clients in this code. I use python 3.7.

Description of the client.py and server.py

Firtst at all, both of them, check if the input is valid. If not, exit. Besides, the client and server both use TCP to implement.

Client.py

I use the multiple thread in client.py.
When the client connect to the server, then the client will get input and send message to server, at the same time, the thread check_alive in client will check if the server is alive every 0.1s. If the check_alive find the server cannot be connected, that means server is closed, clientSocket will close and the program will exit. If the server can be connected, it will send an empty message to server and close this socket, wait for next check.
When the client connect to the server, then the client will get input and send message to server, then after the client receive the complete messages from server the client will automatically go in next loop and give the prompt and information. Besides, if there is no input to client, the client will automatically go in next loop and give the prompt and information.

Server.py

for my code from server.py, some code use the server of Muti-threaded Code(Python) in COMP3331 Assignment webpage. I use this to build the multiple thread in server.
In server.py, if the shutdown is false, the server will continue accept socket. Then, for every connectionsocket, they will go to function server_run() and finish the order of message from client.
In serverrun(), at first of the loop, server receive the message from client. If message is empty, that means it just from check_alive to test if server is alive, so just close this connection socket and break. If message is not empty and first = TRUE, that means a new useful client connected to server, give a message in server. Then, check the commands.
In server.py, there is a clients list will record every client's connection socket and their client address, username that the client login.
I create a code file, serverHelper.py to contain the helper function of server.

Authentication

Authentication

In this process, my program assume credential.txt exists in server directory as mentioned in assignment specification.
When client.py starts, the authentication will occur. After the username and password is entered, client will put "LOGIN", username and password in one message, and the first word in this message will be "LOGIN" to make the server can understand what this message want to do. Then, the client send message to server.

After that, the server will check if the input is valid, if the username already logged by other client, if username exists, if the password for this username is correct. Anyone of the 4 check above is false, the server will give an appropriate message to client. If the username exists and password is correct, the server will give "welcome to forum" message to client, and add this client's connection socket, clientaddress and the username they login in clients list.

Client receive message. If the message is “welcome to forum”, client break this loop go to next step: enter command.
If the message is “Invalid username”, that means the username does not exists, the client will ask to register a new username, then enter new username with its password. Server will receive the message start at “REGISTER”, it will check if the input valid. If the input is valid, add new username and password to credential.txt. And server return a message to client and tell client “register successfully, please log in”, then the loop continue to wait for the login. In this process, there may be something can improve. I tried to user regular expression to check if the username and password contain the characters mentioned in assignment specification. But then I realize those characters contain all characters I know except space( ). Thus, I think we do not need to check this, because we already check about the space( ). However, maybe use regular expression to check will be better, for now, it is also okay for this program.

For the other message, client just output the message and goes to next loop to wait for log in successfully.

11 commands

If an invalid command is selected, an error message from server to client will be shown to the user and they will be prompted to select one of the available actions.

CRT: Create Thread

At first, server check if the input valid as format “CRT threadtitle”. If false, return an error message to client.
Then, server check if the thread exists, if exists, return an error message to client. If does not exist, create a file for this thread (first line is creator username), add this thread to list threads and return a successful message to client.

MSG: Post Message

At first, server check if the input valid as format “MSG threadtitle message”. If false, return an error message to client.
Then, server check if the thread exists, if does not exist, return an error message to client. If exists, open thread file add message to end of the file as the format “messagenumber username: message” and return a successful message to client.

DLT: Delete Message

At first, server check if the input valid as format “DLT threadtitle messagenumber”. If false, return an error message to client.
Then, server check if the thread exists, check if the message number exists, check if the username is the user want to delete. If any checks fail, return a corresponding error message to client. If all checks pass, delete this message and update the other messages number in file, return successful message to client. During update process, I get the old content in file, then in content, I delete the message, then change the message number of subsequent messages, then rewrite thread file with new content.

EDT: Edit Message

At first, server check if the input valid as format “EDT threadtitle messagenumber message”. If false, return an error message to client.
Then, server check if the thread exists, check if the message number exists, check if the username is the user want to edit. If any checks fail, return a corresponding error message to client. . If all checks pass, edit the message and return successful message to client. During edit process, I get the old content in file, then in content, I replace old message to new message, then rewrite thread file with new content.

LST: List Threads

At first, server check if the input valid as format “LST”. If false, return an error message to client.
Get all threads exist (check file exists and thread exists in list), return threads list to client. If no threads, return “no threads to list”.

RDT: Read Thread

At first, server check if the input valid as format “RDT threadtitle”. If false, return error message to client.
Then, server check if thread exists, if not, return a corresponding error message to client. If exists, get all line except the first line of the thread file send to client. If there is only one line in thread file, send message “thread xxx is empty” to client.

UPD: Upload file

At first, client check if the input valid as format “UPD threadtitle filename”. If false, client output message and go to next loop. And I assume the file should be binary files.
Then, the client check if the file exists in client, if not, client output message and go to next loop. If exists, client send message to server. Server receive message and check if thread exists and if this file in this thread exist. IF thread exists and this file in this thread does not exist, send “You can upload file” to client. Otherwise, return a corresponding error message to client.
If the client receives “You can upload file”, the client will get file’s data and send data to server. The server will receive data and write to file named threadtitle-filename. Then, server will add upload message to this thread file as format “Username uploaded filename” and send successful message to client.
There are some points that can be improved. Because I just test two files given in COMP3331 assignment webpage, if use too large file maybe we should make only the server working on loading file. And there may be limit of file size.

DWN: Download file

At first, server check if the input valid as format “DWN threadtitle filename”. If false, return error message to client. And I assume the file should be binary files.
Then, the server check if the thread and if the file in this thread exists, if not, return a corresponding error message to client. If exists, send “You can download file” to client and get file’s data send to client, then send successful message.
If the client receives “You can download file”, receive the file data from server and save it in current directory as filename, then receive successful message.
The improve points are same with UPD.

RMV: Remove Thread

At first, server check if the input valid as format “RMV threadtitle”. If false, return an error message to client.
Then, server check if the thread exists and if the tread is created by this user that want to remove thread. If false, return a corresponding error message to client. Otherwise, delete the thread file, thread upload files and remove thread form threads list, return successful message to client.

XIT: Exit

Server send goodbye message to client, and remove this client from clients list. Client receive goodbye and close.

SHT: Shutdown

At first, server check if the input valid as format “SHT admin_password”. If false, return an error message to client.
Then, server check if the password is correct. If password is incorrect, send a corresponding error message to client. Otherwise, delete all files in server directory except the “server.py” and “serverHelper.py”, close all clients connection, close server socket and send goodbye shutdown message to client. Then exit.
Once the client check if the server closed, the client will receive last message from server and close automatically.