CP372 Assignment 1

Benjamin Scmid - 169042790
Ben Clark - 169017292

Part A): Code Description

The code required for assignment one implements a client-server communication system using Python and associated socket and threading libraries. The server is multi-threaded, allowing it to handle multiple users at the same time, up to 3 users at once. The server uses a specific port and accepts incoming client connections, managing a cache of the user's details, specifically the start and end times.

Key Functionalities:

- Connection Management
    - The server listens for incoming connections and either accepts or rejects the user based on the current number of users relative to the max_clients limit which is set to 3. Once connected the server welcomes the user with a message, if the server is at capacity it will send a busy message.
- Client Communication (Commands)
    - Status: The server returns a list of all users and their connection times
    - List: Returns a list of available files in a specific directory on the server
    - get <filename>: Request a specific file, which the server will send in chunks, ending the transfer with a “EOF” marker.
    - exit: A user can end their connection to the server by using the “exit” command. This removes them from the active user list and logs the end time.
- Thread Safety
    - By using “client_lock” we ensure that only one thread can modify “current_clients” at a time. This helps keep the values accurate and consistent.
    - The same is done by “cache_lock” to keep the data accurate for “client_cache”
- File Transfer:
    - The server supports file transfer by sending requested files in chunks of 1024 bytes to the user.

Part B) Difficulties Faced

- Managing Concurrent Users:
    - A challenge was ensuring that multiple users could connect to the server and communicate without issues such as data corruption. This was addressed by using the thread locks mentioned above to synchronize access to “client_cache” and

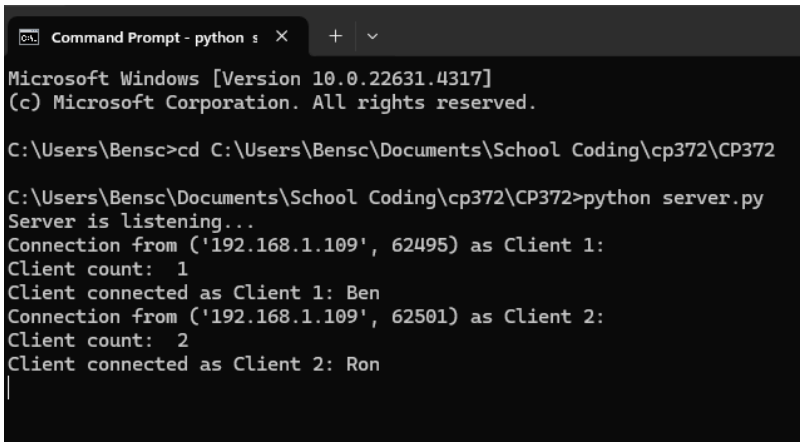
"current_clients". Without the locks multiple threads could have accessed or modified these variables at the same time causing inconsistent or wrong data.

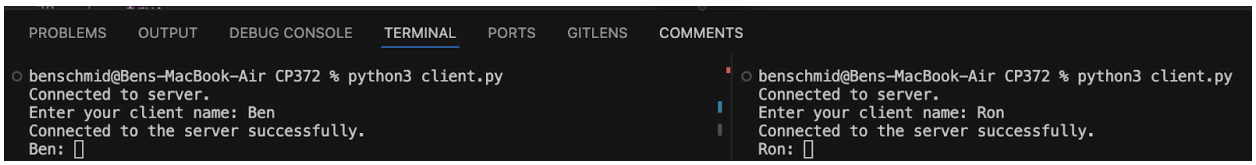
- File Transfer
    - During file transfer, we had to make changes to allow for various chunks of data to be sent before prompting the user to accommodate larger files.
    - Another challenge was informing the user that the transfer had been completed. To solve this, we had the server send an "EOF" marker after the file content, allowing the client to determine the end of the transfer and stop reading.

Part C) Test Results:

For the test results, a windows machine will be running the server in command prompt, and the client will be running on a mac machine using VS Codes built in terminal.

Rubric Requirements 1-3:

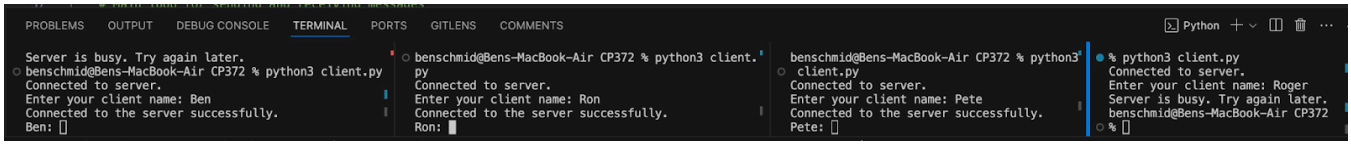




The top screenshot is shown to be running the server, and the bottom screenshots shows 2 clients successfully connecting to the server.

Rubric Requirements 4:

```
C:\Users\Bensc\Documents\School Coding\cp372\CP372>python server.py
Server is listening...
Connection from ('192.168.1.109', 62495) as Client 1:
Client count:  1
Client connected as Client 1: Ben
Connection from ('192.168.1.109', 62501) as Client 2:
Client count:  2
Client connected as Client 2: Ron
Connection from ('192.168.1.109', 62530) as Client 3:
Client count:  3
Client connected as Client 3: Pete
Connection from ('192.168.1.109', 62532) as Client 4:
Max clients reached. Rejecting Client 4: Roger
Current Clients:  3
```



As shown, when a 4th client attempts to join the server, they are rejected since the max clients allowed at a time is 3.

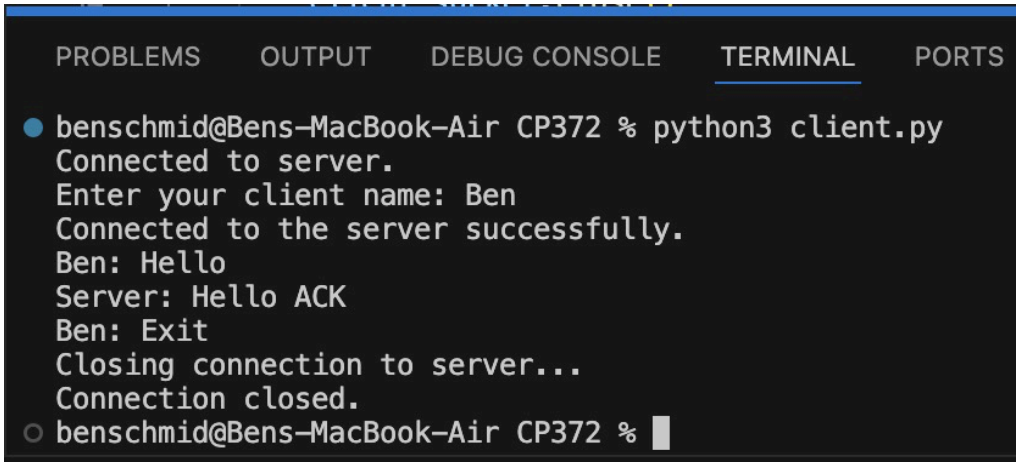
Rubric Requirements 5:

```
benschmid@Bens-MacBook-Air CP372 % python3 client.py
Connected to server.
Enter your client name: Ben
Connected to the server successfully.
Ben: Hello
Server: Hello ACK
Ben:
```

When client sends message Hello, server responds with “Hello ACK”

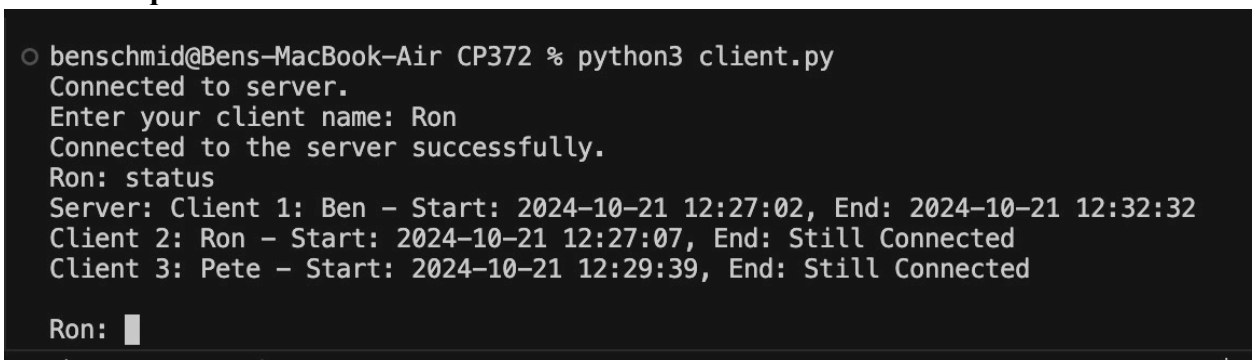
Rubric Requirements 6:

```
C:\Users\Bensc\Documents\School Coding\cp372\CP372>python server.py
Server is listening...
Connection from ('192.168.1.109', 62495) as Client 1:
Client count:  1
Client connected as Client 1: Ben
Connection from ('192.168.1.109', 62501) as Client 2:
Client count:  2
Client connected as Client 2: Ron
Connection from ('192.168.1.109', 62530) as Client 3:
Client count:  3
Client connected as Client 3: Pete
Connection from ('192.168.1.109', 62532) as Client 4:
Max clients reached. Rejecting Client 4: Roger
Current Clients:  3
Client 1: Ben has disconnected.
Current Clients:  2
```



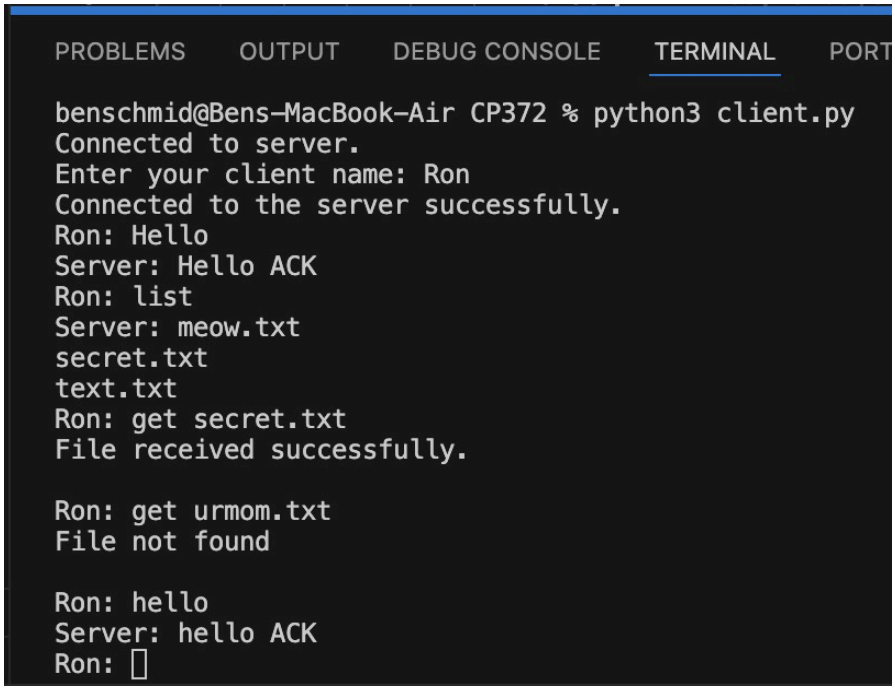
When client types “exit”, the server cleanly disconnects clients and makes room for other clients

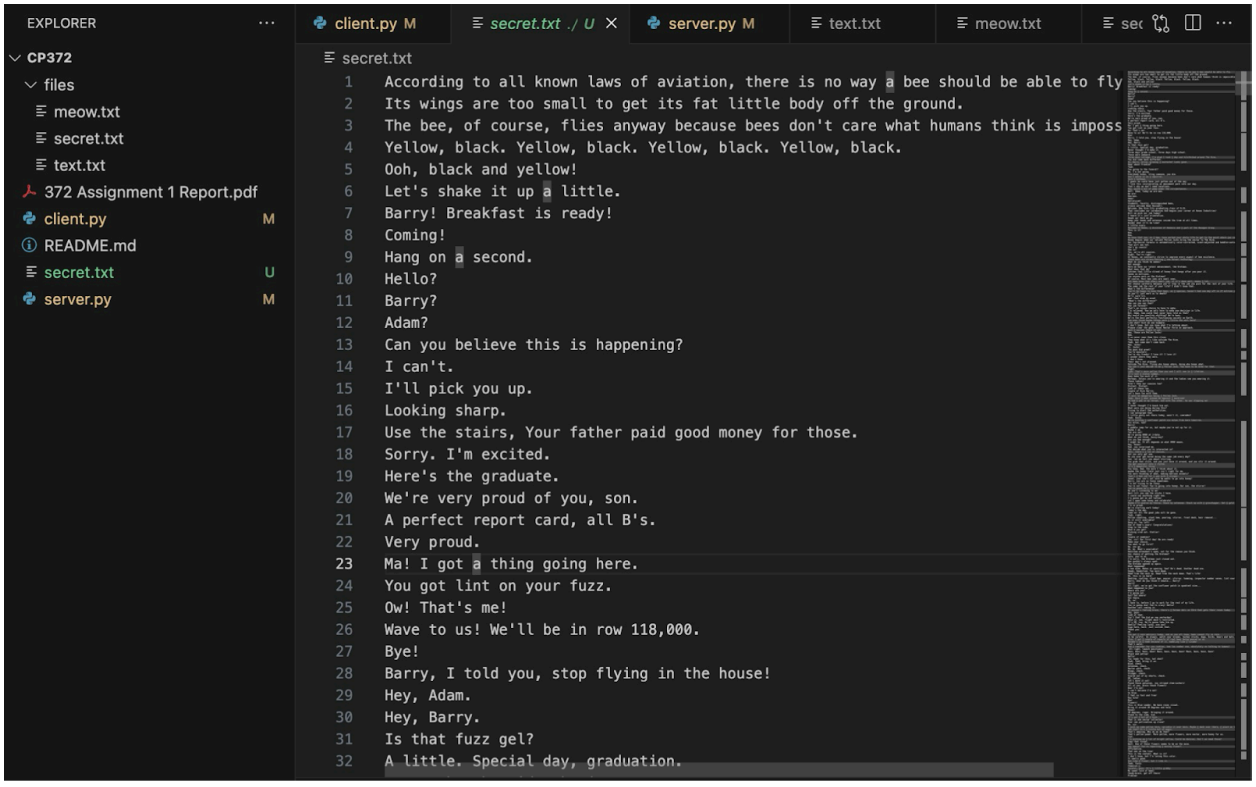
Rubric Requirements 7:



Server maintains connection details, including connection and disconnection time.

Rubric Requirements 8, 9:





When the user types “list,” it lists all the files in the files directory. Typing in get “file_name” will download the file and its contents to the current directory that client is running on. In this example, user types “get secret.txt” and the file is written to the directory. Furthermore, when typing in a file that does no exist, the server tells the client that the file does not exist

Part D): Possible Improvements

Given more time, the biggest improvement would be enabling direct client-to-client communication. Currently, all communication is routed through the server. By implementing a peer-to-peer connection, clients could exchange messages directly after establishing an initial connection to the server.

Additionally, other improvements include

- File transfer between client and client
- User Authentication
- Implementing a proper graphical user interface instead of command line