GTU Department of Computer Engineering
CSE 344 - SPRING 2022
MIDTERM REPORT

Burak ÇİÇEK
1901042260

1.Design Decisions and Concept

-I have three files here: Client, Servant and Server.

-Client request creates threads as many as the number of valid lines in the file.

-I collect the messages it reads into a common struct array.

-Each thread transmits its related request to the server with its own socket.

-The server reads the requests from the socket.

-The commonMessage struct is a common message struct for both client and servant

.

-If there is a non-busy thread, the non-busy thread on the server immediately receives the message and starts to apply the necessary actions.

-There is a queue in the server process to put the requests in the correct order.

-Threads receive all common messages from this queue.

-It can use the monitor feature with the condition variable by keeping the size of the queue.

-First I measure the word count of the message.

-I forgot to mention that I have a flag for separating message types in my common message struct.

-If typeOfMessage is equal to 0, the message came from the Client. If it is equal to 1 then the message has come from the servant.

-In the server thread, both messages are processed differently, so such a distinction is needed.

-There is an array in the server to keep the information of the servant. I can know which cities the servant holds in this array.

-If wordCount is equal to 5, I can send my direct message to the relevant servant.

-If wordCount is equal to 4 then I send the message to all the servants.

-After sending my message, I am waiting for servant's reply to forward it to servant's client.

-On the other hand, if the servant sends the message, I get the city information he keeps in the relevant block and other important information about him (Unique port number, Process ID, etc.)

-In the Servant, there are a few jobs in the main thread first

-We create a port for ourselves from the port allocated to us by the server and send it to the server.

-.I'm waiting for the lower bound to the port that the server gave us to do the unique port. In this way, each servant has a unique port number.

-Then I listen to the socket I opened from this port so that the server can send me the client's messages

-I made a queue at the servant to make our work easier.

.-The Servant thread starts off by first shredding the message it receives.

-Then I do my operation again according to the length of the message.

-By the way, I forgot to mention. Main thread keeps relevant part of servant in dataset in linked list.

-Anyway, if the word count is equal to 5, I immediately start my operations by city. Otherwise, I am doing it in a normal way, not city specific.

-After preparing the required answer, I send it to the server with the help of socket (Unique port).

-I receive a message from the server waiting for a message from the Servant and send the answer over the socket opened by the client's thread.

-The client collects the whole responses and when the last thread terminates, the client also terminates.