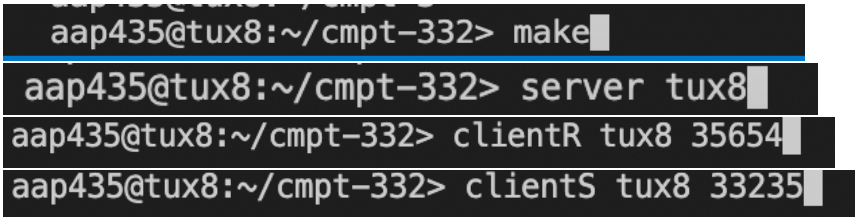
Henry Fang & Aasrija Puchakatla
hef052 & aap435
11233914 & 11211759
CMPT 332 A4

In this assignment, we have clientS where the client sends the message to the server, clientR which receives the messages and the server which receives the message from the sender client and sends the message to the receiver client. We were successfully able to implement our server where we have two threads that accept connections for sender clients and receiver clients, a thread-per connection approach where each thread is created as a detached thread and we have also implemented lock and condition variables for synchronization among the threads.

To start our program, in the terminal, type in make in terminal and enter in server tux# in 1 terminal, clientR tux# port# in the second terminal and clientS tux# port# in the third terminal.

Ex:



Once this is done you will get prompter to enter a message in the terminal for clientS and once you enter a message, it will be sent to the server which will receive the message, send it to the receiver client and the receiver client will receive the message.

clientS.c:
In clientS.c, we connect the client to the server, we create out sockets and prompt our user to enter hostname, port and message. Once the user has entered a message, we send this to the server.

clientR.c:
In clientR.c, we connect with the server to receive message from the server and print the message to the client in the terminal.

server.c:

In server.c, we connect with clientS.c and clientR.c. Client sender (clientS.c) and server receiver are connected while server sender is connected to client receiver (clientR.c).

In main(), we initialize our locks and create our two types of threads for receiving and sending. For our functions named respective to receiving receive data from the sender client while functions with sending in their names in this file, send data to receiving data to reciever client.

For receving_connection_thread:

-we implement code for connections with receiving the message from sender client

-bind and got connection, and create our detached threads

-we have created a thread for every connection made with the sender client and new clients

For send_connection_thread:

- we implement code for connections with sending the message to receiving client

- bind and got connection, and create our detached threads

-we have created a thread for every connection made with the receiver client and new clients

For myRec(void *args):

-receives message from client sender

-params: void *args

- in the while loop we receive the num of bytes from the message that was sent to us from sender client

-when the num of bytes=0, we close the receiving thread

-when a message is received, we allocate the message to put together IP addr, port number and message all together

-we again use locks and set recMessage=1 for our conditionals

For mySend(void*args):

- sends message to client receiver

-param: void *args

- in the while loop we use mutex locks to wait until a message

-has been recieved to send to receiver client

-once the message has been sent, we erase our message so we can assign it a new message

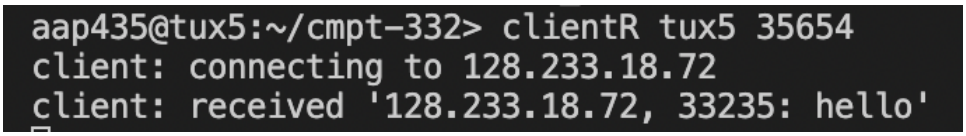
-once we are out of this loop, we close our sockets and unlock our locks

-when we stop waiting for a message, we also unlock our locks

Here is an example of our run:

Server terminal:

Receiver client terminal:



Sender client terminal: