Comp 4981 - Assignment 3

Comprehensive Documentation

Morgan Ariss

A00989042

Anthony Vu

A00940922

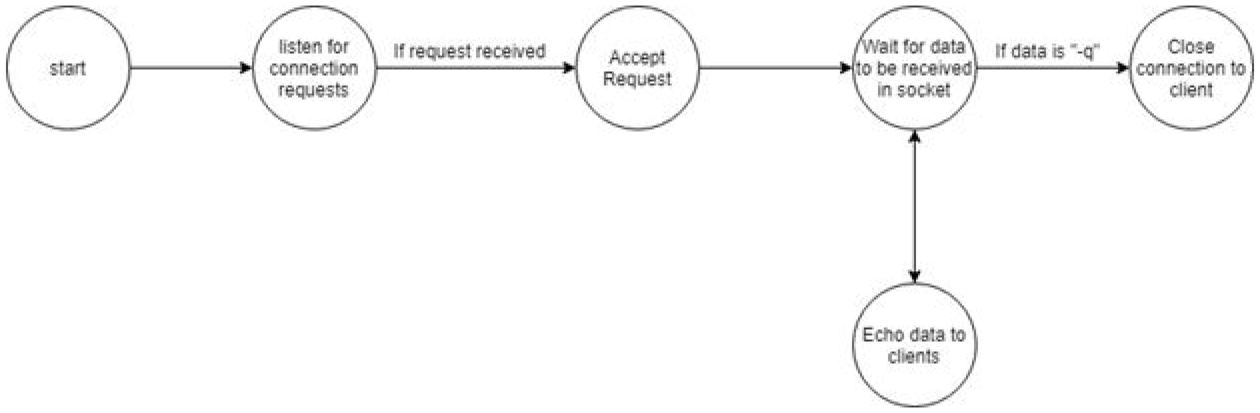
Introduction

The purpose of this assignment is to write and test a simple chat client/server application. Where in the server accepts connections on a specified port and allow clients to connect to it; once clients have established a connection, it will echo whatever it receives to all other connected clients.
Additionally to design and implement a chat client that really is no more complex in this case then an echo client. Each client will have the ability to send text strings to the server and will also be able to see the text sent by all other clients.
Both client and server will not include a GUI they will be console applications in LINUX.

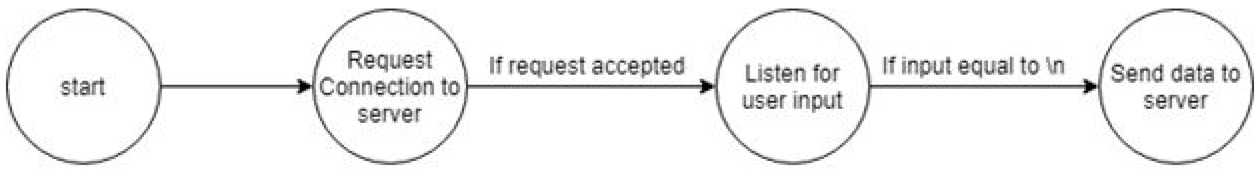
State Diagram

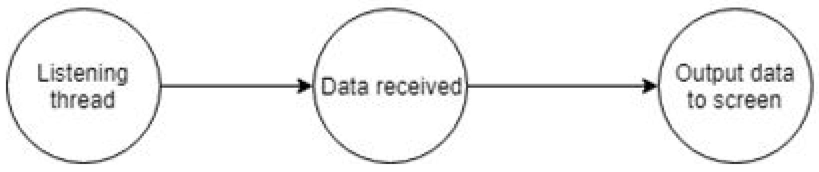
Server
If a user has requested to connect to the server, accept the request and start listening for data to be received from the socket. If there is data:

- If the data is equal to “-q”, close the connection to the client
- Otherwise, send the data to all connected clients except the initial sender


start
listen for connection requests
If request received
Accept Request
Wait for data to be received in socket
If data is "-q"
Close connection to client
Echo data to clients

Client
If the server has accepted the client's request, the user may begin sending data to the server. The data is stored in a saved buffer. Whenever the user presses the enter key, the data is sent. If the user enters "-q", exit the program. If the user enters "-s", save a log of the chat messages.





Pseudocode

Server Section
This is the server side code for the chat room that handles all connection requests and relays messages to all clients.

main()
Precondition: This is the entry point for the program.
This state is called immediately after the program is started, it is responsible for initializing the ports and sockets for client connections. Once the connection is made the client is entered into the relay list so that it will receive messages from the other clients.

```
main()
{
        Initialize sockaddr struct
        Initialize buffer

        Switch(user_input)
                Case 1:
                        Default port
                Case 2:
                        User specified port
                Default:
                        usage/help

        Create stream socket
        Call setsockopt with SO_REUSEADDR

        Bind an address to the socket

        Listen for connections
        Set all indexes in client array to -1
        Loop
        {
                Get number of set indexes in client array
                If new connection
                        Accept client connection and add client to hashmap
                If too many clients
                        exit
                Check all clients for data
                        If index in client is not set
                                Continue
                        If set
                                Read data from socket
                                If data is equal to -q
                                        Set client index to -1
                                        Delete client from hashmap
                                If data is equal to -s
                                        Continue
                                Send data to all clients except the client that sent the data
        }
}
```

Client Section
This is the client side code for the chat room where clients can send connection requests and messages to the server.

main()
Precondition: This is the entry point for the program.
This state is called immediately after the program is started, it is responsible for initializing the connection to the server. It sends the clients messages to the client and listens for responses from the server.

```
main()
{
        Instantiate hostent struct
        Instantiate sockaddr struct
        Instantiate buffer
```

```
    Get host from console argument[1]

    switch()
    Case 2:
        Set default port
    Case 3:
        Set port from console argument[2]
    Default:
        Usage

    Create the socket
    Get host info

    Connect to the server
    Create read thread, run outputMsg
    Loop while connected
    {
        Get user input
        Send buffer to the server
        If data is -q
            Close socket
            End program
        If data is -s
            Open file
            Write savedbuffer to a local file
            Close file pointer
        Reset buffer
    }
}
```

outputMsg()
Precondition: Called in read thread
This function is responsible for displaying all data sent through sockets from the other two processes. It is constantly listening for any data. It takes in a socket for the I/O.

```
outputMsg()
{
    While n bytes read is less than buffer length
        Keep reading data into read buffer
    While character in read buffer is not equal to EOF
        Store data into saved buffer
    Print read buffer
    Flush stdout
```

```
        Reset read buffer
}
```

Helper Functions Section

add_user()
Precondition: A new client connection has been formed
This function adds the new client to the list of connected users so their messages can be relayed.

find_user()
Precondition: Client is contained in hashmap
This function finds a client in the hashmap and returns the user struct
delete_user()
Precondition: A client needs to be removed from the hashmap
This function removes a client connection from the list to lighten the load on the server.

system_fatal()
Precondition: An error has occurred; and the program must be killed
This function prints the error message that it is passed to perror, and then sends the terminate signal for the function to be killed.