ECEC 353 Assignment 3

William Fligor, Avik Bag

August 2016

1 Design

For this assignment, the aim is to implement a chat server and chat client system with the use of inter process communication techniques. The chat server will be able to accept up to 10 clients and allow for messages to the board and also individually to any targeted client. The chat client should be able send messages to the board and any individual message.

1.1 Server

The client is established to allow for the clients to communicate with. When it's started up, it sets up 10 separate shared memory spaces. Each of these are corresponding to a client. So anytime a server is running, the client can start up and it will be assigned a shared memory allocation which can link it to the server group. This shared memory space is what will be accessed between the client and the server process. Each of these shared memory spaces are blocked off with the help of semaphores, 5 to be exact. Each of them are responsible for tracking a specific stage in the process. The server shuts down properly and clears out and deallocates any shared memory by implementing a signal handler that triggers on a SIGINT (which normally is ctrl+c).

- sem_tx waits until data can be transmitted
- sem_rx waits until data can be received
- sem_writer waits until the writer is done writing
- sem_reader waits until the reader is done reading
- sem_ack waits until the reader has cleared its data available tag and the server can move on

1.2 Client

The client essentially looks for the given server name that is passed via command line arguments. If the server the client is looking for does not exists then it'll spit out an error. If found, the server is going to check whether there is a spot open in that given server. Once that is established, the client can continue to send messages to the board or to any particular user.

1.3 Challenges

Challenges faced were mainly focused around communication. Establishing a communication protocol that meet the design goals took the longest. The design goals,

- Allow for TX and RX communication between clients and servers
- Not allow clients to read other clients private messages

The second design goal resulted in the implementation that was developed, using 10 shared memory spaces to separate client communication in order to give complete control to the server and prevent clients having access to other clients private messages.

The second biggest challenge was reading from STDIN while not blocking the loop that communicates with the server. This was accomplished using ioctl to read on the next loop after the return button was pressed.