

Client to Client Communication using Shared Memory and Semaphores

Task:

You are required to produce two programs, chatserver and chatclient. There can be more than one instance of chatclient running at a time and the purpose of chatserver is to provide a link between all the available clients so that they can talk to each other.

Explanations and Methods:

1. Messageboard:

The messageboard is a structure which has variables for a destination, a source and a message.

```
struct shared_mem {  
    char dest[8];  
    char src[8];  
    char message[239];  
};
```

2. Server:

The server maintains a list of clients in a two dimensional array of characters and the number of clients is kept in a separate integer.

- **void insert(char *clientid):** This method takes an array of characters and inserts it into a 2d array which is a list of available clients.
- **int search2darr(char *phrase):** This method is used to search for a specific client in the list. This method called when retiring a client or when sending a message to a client.
- **void delete(char *phrase):** This method deletes the client from the list. It uses the above method to find the client and then it empties the index of that client.
- **void put_clients_in_msg(char *message):** This method loops through the client array and copies all available clients in provided argument and put a line break escape sequence between them.
- **void *inputs(void *args):** This method runs in a separate thread to check if someone entered @Quit command on the server.
- **int main():** The main method of the server program first creates a shared memory and attaches it to a pointer and then converts type of that pointer to shared_mem struct. Then it creates a thread for taking input and initializes a semaphore for resource synchronization. Then there are three conditions for @List, @Quit and connect, they all do their required work and also lock and release the shared memory using semaphore.

3. Client:

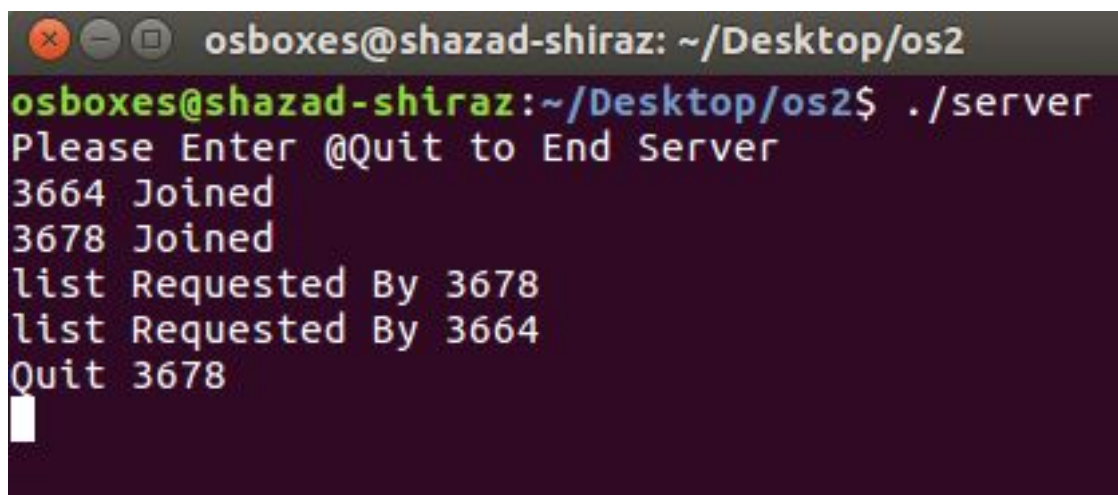
- **void *input(struct shared_mem *share):** This method is used for chatting between clients, requesting list of available clients and retiring clients.
- **void *alive(struct shared_mem *share):** This method sends an alive message to the server after an interval which is told by the server after at the time of connection.
- **int main():** This method assesses the shared memory and when the user send a message it locks the shared memory using semaphore.

How to use:

- Both server and client programs use threading.
- To compile server program: `gcc -o server server.c -lpthread`
- To compile client program: `gcc -o client client.c -lpthread`
- Run server only once with `./server`
- Run client with `./client`, to run more instances of client program each time we must recompile client.c and then run the program.

Screenshots:

1. Server



```
osboxes@shazad-shiraz: ~/Desktop/os2
osboxes@shazad-shiraz:~/Desktop/os2$ ./server
Please Enter @Quit to End Server
3664 Joined
3678 Joined
list Requested By 3678
list Requested By 3664
Quit 3678
```

2. Client

```
osboxes@shazad-shiraz: ~/Desktop/os2
osboxes@shazad-shiraz:~/Desktop/os2$ ./client
Please Enter @List for list,@Quit for endyoursession,clientid message for sending a message
@List
3664
3678

3664 hi bro
i am fine bro
@Quit
osboxes@shazad-shiraz:~/Desktop/os2$
```

3. Client

```
osboxes@shazad-shiraz: ~/Desktop/os2
osboxes@shazad-shiraz:~/Desktop/os2$ ./client
Please Enter @List for list,@Quit for endyoursession,clientid message for sending a message
@List
3664
3678

hi bro
3678 i am fine bro
Updated List
3664
```