Multithreaded chat server and client

Bhargava Sreekantappa Gayathri(010109580)
San Jose State University

***Abstract*—This project demonstrates a multithreaded chat server and client. The server starts a new thread for every client that connects to it. Every client has two threads, one to listen to the server and one more to wait for the user to input commands.** *(Abstract)*

***Index Terms*—Server; Client; Threads; Multithread; Socket.** *(key words)*

I. Introduction

In the client/server programming model, a server is a program that awaits and fulfills requests from client programs in the same or other computers. A given application in a computer may function as a client with requests for services from other programs and also as a server of requests from other programs.

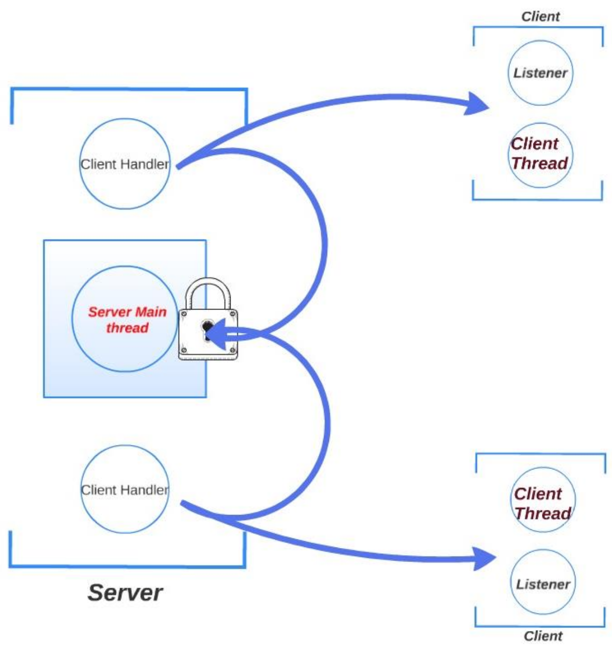
In this project I have written a simple application of multithreading where in the users will be able to do a group chat with each other. The different threads of the server will synchronize with each other using a mutex.

II. Objectives

The objectives of the client server implementation were:

1. Create a server which will allow multiple people to have a discussion online.
2. The server should spin a thread to service every new client.
3. Client address book is shared among all client threads on the server and it has to be protected using a mutex lock.
4. Create a client program which will talk to the server and run on the client machine.
5. The client program should provide an interface between the client user and the server.

III. Block diagram



IV. Implementation

As we can see from the block diagram, the server client model has 4 parts:

1- Threads running on server:
   a. Server thread (main thread)
   b. Server's client handle thread
2- Threads running on client:
   a. Client process(main thread)
   b. Client's listening thread

Let us look into the details of each thread and its function.

1.a Server thread (main thread)

- Initialize the server and listen on port 8080.
- Initialize the link-list for storing the clients.
- Initialize Mutex to protect shared data.
- Keep accepting new requests and create a thread to handle the request.

1.b Server's client handle thread

- Wait to receive the data from the client thread.
- Once data received, if the option is:
  - send:
    - Get the mutex to the client list.
    - send the data to all the client threads running on the client machine.
    - release the mutex.
  - exit:
    - Get the mutex to the client list.
    - delete the client node in the link list.
    - release the mutex.
    - close the socket

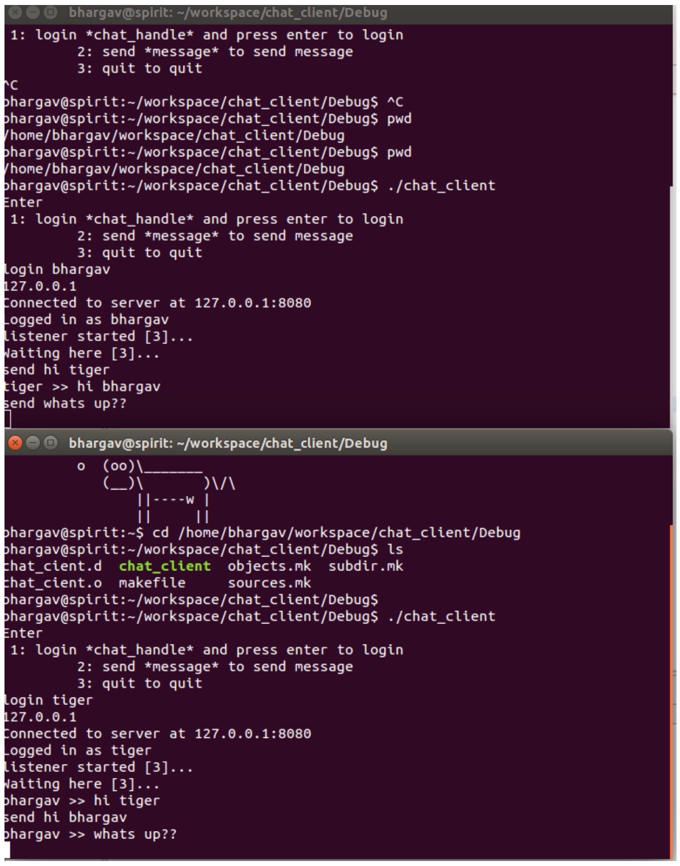
2.a Client process(main thread)

- Get the input the client user. If the input is:
  - exit:
    - send packet with exit option to the server and end the process.
  - login:
    - Connect with server.
    - register username.
    - create a new thread for listening on the connection.
  - send:
    - Create a packet with option send and sent it to the server.

2.b Client process(main thread)

- Wait for recv() to return data.
- Print the received information.

V. SCREENSHOT



bhargav@spirit: ~/workspace/chat_client/Debug

```
 1: login *chat_handle* and press enter to login
        2: send *message* to send message
        3: quit to quit
^C
bhargav@spirit:~/workspace/chat_client/Debug$ ^C
bhargav@spirit:~/workspace/chat_client/Debug$ pwd
/home/bhargav/workspace/chat_client/Debug
bhargav@spirit:~/workspace/chat_client/Debug$ pwd
/home/bhargav/workspace/chat_client/Debug
bhargav@spirit:~/workspace/chat_client/Debug$ ./chat_client
Enter
 1: login *chat_handle* and press enter to login
        2: send *message* to send message
        3: quit to quit
login bhargav
127.0.0.1
Connected to server at 127.0.0.1:8080
Logged in as bhargav
listener started [3]...
Waiting here [3]...
send hi tiger
tiger >> hi bhargav
send whats up??
```

bhargav@spirit: ~/workspace/chat_client/Debug

```
        o  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
bhargav@spirit:~$ cd /home/bhargav/workspace/chat_client/Debug
bhargav@spirit:~/workspace/chat_client/Debug$ ls
chat_cient.d  chat_client  objects.mk  subdir.mk
chat_cient.o  makefile     sources.mk
bhargav@spirit:~/workspace/chat_client/Debug$
bhargav@spirit:~/workspace/chat_client/Debug$ ./chat_client
Enter
 1: login *chat_handle* and press enter to login
        2: send *message* to send message
        3: quit to quit
login tiger
127.0.0.1
Connected to server at 127.0.0.1:8080
Logged in as tiger
listener started [3]...
Waiting here [3]...
bhargav >> hi tiger
send hi bhargav
bhargav >> whats up??
```

VI. ACKNOWLEDGMENT

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this Operating Systems project. I am thankful for their aspiring guidance, invaluable constructive criticism and friendly advice during my project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

I express my warm thanks to Prof. Frank for his guidance and support during the course.

REFERENCES

[1] https://globaljournals.org/GJCST_Volume12/2-Enhanced-Chat-Application

[2] http://chatinc.blogs.ua.sapo.pt/389.html

[3] https://github.com/dtolj/simple-chat-client-server/blob/master/chat.c

[4] http://stackoverflow.com/questions/14884450/chat-server-socket

[5] http://www.igniterealtime.org/projects/openfire/.

[6] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAAahUKEwivhZ6-lYfHAhXTMogKHcpcDDA&url=https%3A%2F%

[7] https://computing.llnl.gov/tutorials/pthreads/