

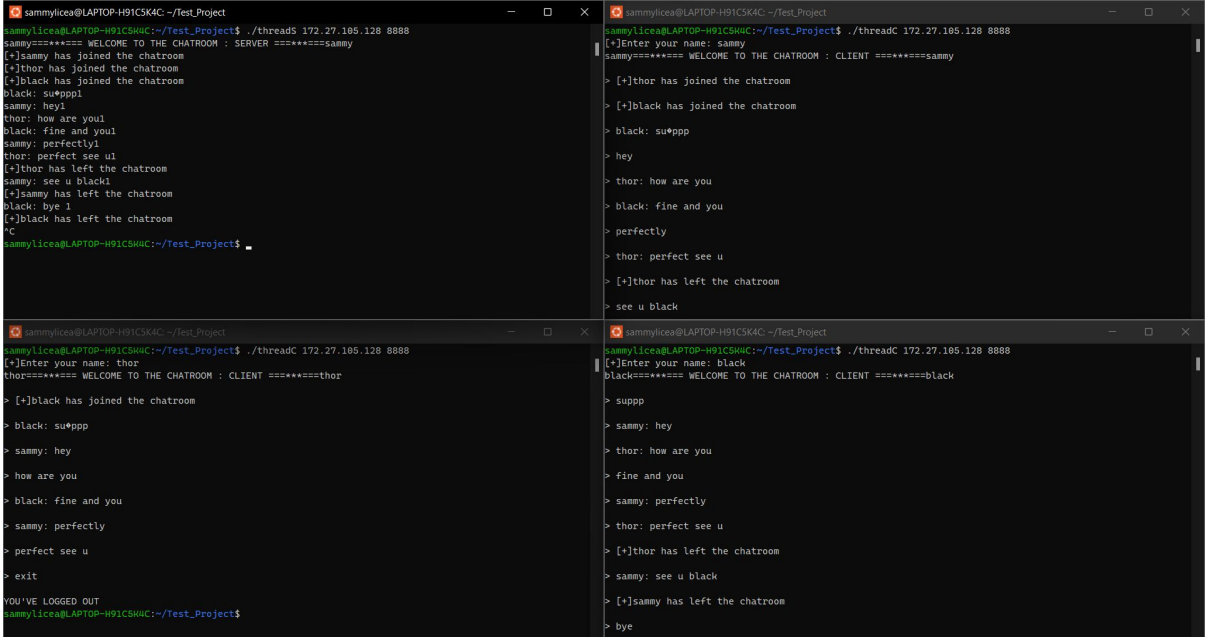


# Multi-client chatroom

Development of a Client-Server application

# Motivation

Client-Server sockets that enable message sending and receiving using threads



```
sammy@lisa:~/LAPTOP-H91CSK4C:~/Test_Project$ ./threadS 172.27.105.128 8888
sammy===== WELCOME TO THE CHATROOM : SERVER =====sammy
[+]sammy has joined the chatroom
[+]thor has joined the chatroom
[+]black has joined the chatroom
black: su*ppp
sammy: hey!
thor: how are you!
black: fine and you!
sammy: perfectly!
thor: perfect see u!
[+]thor has left the chatroom
sammy: see u black!
[+]sammy has left the chatroom
black: bye 1
[+]black has left the chatroom
^C
sammy@lisa:~/LAPTOP-H91CSK4C:~/Test_Project$

sammy@lisa:~/LAPTOP-H91CSK4C:~/Test_Project$ ./threadC 172.27.105.128 8888
[+]Enter your name: sammy
sammy===== WELCOME TO THE CHATROOM : CLIENT =====sammy
> [+]thor has joined the chatroom
> [+]black has joined the chatroom
> black: su*ppp
> hey
> thor: how are you
> black: fine and you
> perfectly
> thor: perfect see u
> [+]thor has left the chatroom
> see u black

sammy@lisa:~/LAPTOP-H91CSK4C:~/Test_Project$ ./threadC 172.27.105.128 8888
[+]Enter your name: thor
thor===== WELCOME TO THE CHATROOM : CLIENT =====thor
> [+]black has joined the chatroom
> black: su*ppp
> sammy: hey
> how are you
> black: fine and you
> sammy: perfectly
> perfect see u
> exit
YOU'VE LOGGED OUT
sammy@lisa:~/LAPTOP-H91CSK4C:~/Test_Project$

sammy@lisa:~/LAPTOP-H91CSK4C:~/Test_Project$ ./threadC 172.27.105.128 8888
[+]Enter your name: black
black===== WELCOME TO THE CHATROOM : CLIENT =====black
> suppp
> sammy: hey
> thor: how are you
> fine and you
> sammy: perfectly
> thor: perfect see u
> [+]thor has left the chatroom
> sammy: see u black
> [+]sammy has left the chatroom
> bye
```

---

# Source Code Description



# Server

1. Initialize libraries and global variables, as well as define client structure.

In main:

2. Initialize server and bound to a socket (socket(), bind()).
3. Server starts listening after connection (listen()).
4. If a client connects, a thread is created to manage message sending and receiving for that client (one thread per client).



# Client

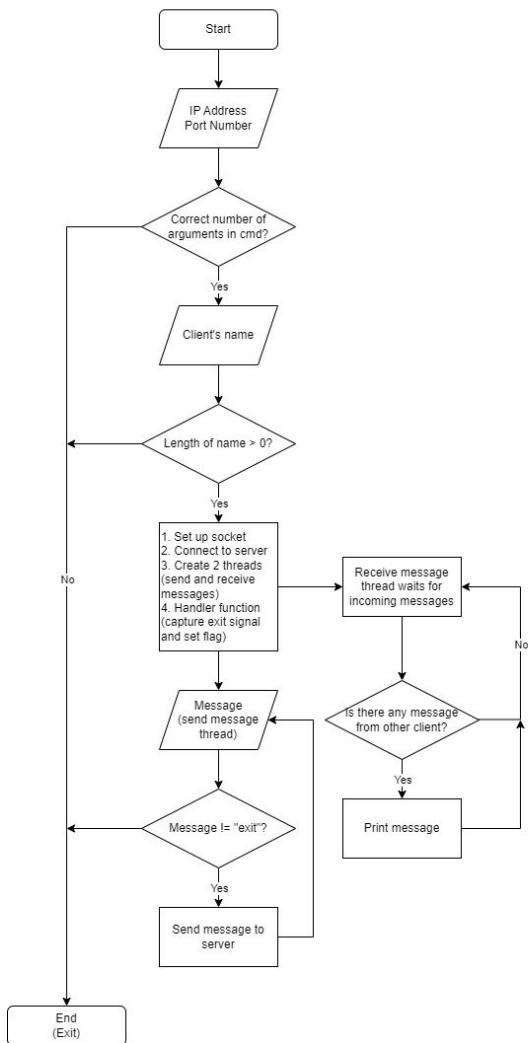
1. Initialize libraries and global variables.

In main:

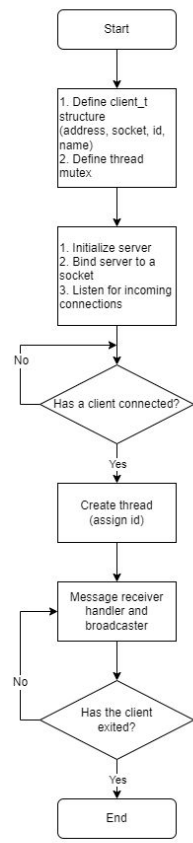
2. Wait for client's input name ( $0 < \text{name\_length} \leq 32$ ).
3. Set socket for the client, connect to server and create threads.
4. Sending message thread waits for user's message input (message != "exit").
5. Receiving message thread waits for incoming messages while client is connected to server, to print them.

z122301

Client's Code Flowchart



Server's Code Flowchart

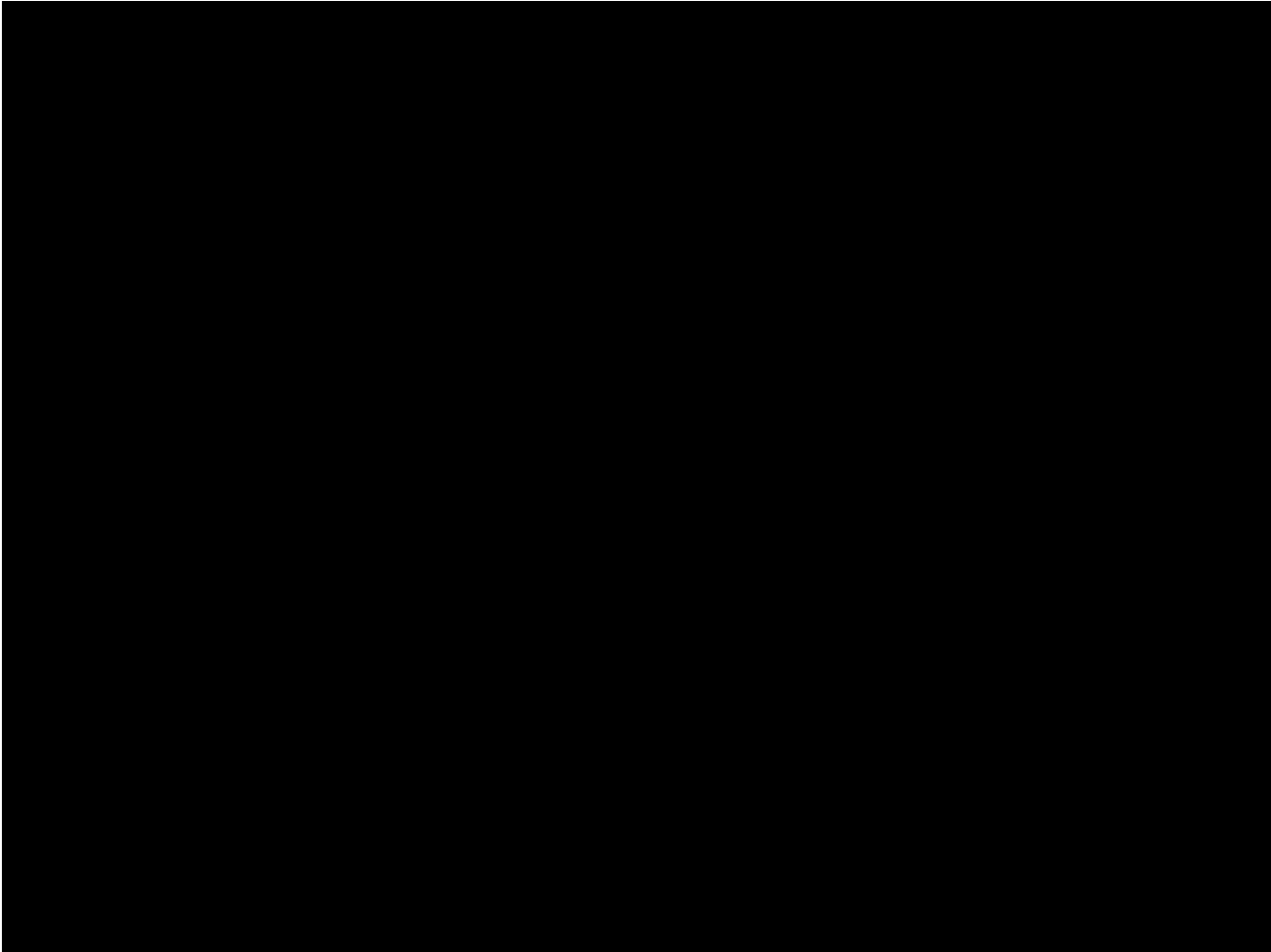


z122301

---

# Demonstration

z122301





z122301

---

# Thank you