

# Assingment-5

---

## Group Details:

1. Adarsh Dhakar → 22CS01040
2. Avik Sarkar → 22CS01060
3. Debargha Nath → 22CS01070
4. Soham Chakraborty → 22CS02002

## Github Repository Link:

[https://github.com/adarshdhakar/cn\\_lab\\_sheet5/](https://github.com/adarshdhakar/cn_lab_sheet5/)

## Images Directory Link:

[https://github.com/adarshdhakar/cn\\_lab\\_sheet5/images](https://github.com/adarshdhakar/cn_lab_sheet5/images)

## Report.pdf Link:

[https://github.com/adarshdhakar/cn\\_lab\\_sheet5/Report.pdf](https://github.com/adarshdhakar/cn_lab_sheet5/Report.pdf)

## Demo Video Link:

[https://github.com/adarshdhakar/cn\\_lab\\_sheet5/Demo.mp4](https://github.com/adarshdhakar/cn_lab_sheet5/Demo.mp4)

```
+-----+
|
| How to send a message?
| 1. By default the message will be sent to 'All' users.
| 2. ${NAME} MESSAGE          -> To send the message to a particular user
| 3. ${NAME1,NAME2,...} MESSAGE -> To send message to a particular set of users
| 4. /rooms                   -> To list all the rooms with the users in them
| 5. /join ROOM_NAME          -> To join an existing room or to create a room
| 6. /leave                    -> To leave the joined room
|
+-----+
+-----+
| Enter your name: Soham
+-----+
You: hello
```

1.

### Why C++ over C ?

- Use of **string** makes it easier to handle text messages instead of relying on character arrays where we would have used `strcpy()` and `strcat()`.
- Use of **set<int>** to maintain list of active clients efficiently.
- Use of **map<string,int>** makes it easier to maintain mapping of active clients to their respective `socketfd`. This allows for quick lookups, insertions, and deletions.
- Use of **map<string, set<int>>** to store the **socketfd** of the members in a **room**.
- Use of **map<int, string>** to store the **rooms name** where a particular **socketfd** belongs to.

### What all libraries used ? Functionalities provided by these.

**#include <bits/stdc++.h>**

Functions used: `string`, `set`, `iostream`

**#include <netinet/in.h>**

Provides structures for internet addresses (**sockaddr\_in**)

**#include <netdb.h>** (only in Client)

Contains **gethostbyname()** to resolve hostnames to IP addresses.

**#include <pthread.h>**

Used for multi-threading:

- Creating threads for handling multiple clients on server.
- Creating separate threads for reading and writing for the client.

**#include <unistd.h>**

Provides access to the POSIX operating system API and contains system calls related to process control, file handling and I/O operations.

Functions used: **read**, **write**, **close**, **shutdown**, **sleep**

# Features and Functionality:

## Broadcast Messages

‘hi’ sent by Adarsh is sent broadcasted everyone.

### Adarsh

```
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx:~/Desktop/CN Lab/LabSheets/cn_lab_sheets$ ./client localhost 1800

How to send a message?
1. By default the message will be sent to 'All' users.
2. $[NAME] MESSAGE -> To send the message to a particular user
3. $[NAME1,NAME2,...] MESSAGE -> To send message to a particular set of users
4. /rooms -> To list all the rooms with the users in them
5. /join ROOM_NAME -> To join an existing room or to create a room
6. /leave -> To leave the joined room

Enter your name: Adarsh
You:
Debargha: Hello
You: Hi
You: 
```

### Debargha

```
debargha-nath@debargha-nath-HP-Laptop-14e-drxxx: /CN_Lab_3/cn_lab_sheets$ ./client 192.168.28.183 1800

How to send a message?
1. By default the message will be sent to 'All' users.
2. $[NAME] MESSAGE -> To send the message to a particular user
3. $[NAME1,NAME2,...] MESSAGE -> To send message to a particular set of users
4. /rooms -> To list all the rooms with the users in them
5. /join ROOM_NAME -> To join an existing room or to create a room
6. /leave -> To leave the joined room

Enter your name: Debargha
You:
Avik joined the chat.
Active Clients: 2
You:
Avik: Hi
You: Hello
You:
Adarsh joined the chat.
Active Clients: 3
You:
Adarsh: Hi
You: 
```

### Avik

```
avik@avik-swift-SFG14-72T:~/Desktop/khat_socket/cn_lab_sheets$ ./client localhost 1800

5. /join ROOM_NAME -> To join an existing room or to create a room
6. /leave -> To leave the joined room

Enter your name: Avik
You:
Debargha joined the chat.
Active Clients: 2
You: Hi
You:
Debargha: Hello
You:
Adarsh joined the chat.
Active Clients: 3
You:
Adarsh: Hi
You: 
```

## Private Messaging

Messages sent using ‘\$’ commands are sent to the specified people only.

Messages sent by Adarsh using \$[Avik,Debargha] are sent to only Avik and Debargha

## Direct Messaging

### Adarsh

```
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx:~/Desktop/CN Lab/LabSheets/cn_lab_sheets$ ./client localhost 1800

How to send a message?
1. By default the message will be sent to 'All' users.
2. $[NAME] MESSAGE -> To send the message to a particular user
3. $[NAME1,NAME2,...] MESSAGE -> To send message to a particular set of users
4. /rooms -> To list all the rooms with the users in them
5. /join ROOM_NAME -> To join an existing room or to create a room
6. /leave -> To leave the joined room

Enter your name: Adarsh
You:
Avik->You: Hi
You: $[Avik,Debargha] Hello
You:
Client Debargha has been banned from the server. [Reason: Time Out]
Active Clients: 2
You:
Debargha:
You: 
```

### Avik

```
Adarsh joined the chat.
Active Clients: 3
You: $[Adarsh] hlo
You:
Adarsh->You: Hello
You: 
```

### Debargha

```
Adarsh joined the chat.
Active Clients: 3
You:
Adarsh->You: Hello
You: 
```

## Join a Room

Messages sent after joining room are only sent to the members in the room.

Avik

```
You: /join room1
You:
Joined private room: room1
You: are you here
You:
Avik (in room1): are you here
You:
Debargha (in room1): Are you there
You: yes
You:
Avik (in room1): yes
You: □
```

Debargha

```
You: /join room1
You:
Joined private room: room1
You: /rooms
You:
Available Rooms:
room1 (2 users): Debargha Avik
You: Are you there
Avik (in room1): are you here
You:
Debargha (in room1): Are you there
You:
```

## Leave a Room

Check available rooms and members

```
Joined private room: room1
You: /rooms
You:
Available Rooms:
room1 (2 users): Debargha Avik
You: Are you there
Avik (in room1): are you here
You:
Debargha (in room1): Are you there
You:
```

## Exit Functionality

User **Soham** exited using the exit command.

```
+-----+
| How to send a message?
| 1. By default the message will be sent to 'All' users.
| 2. ${NAME} MESSAGE -> To send the message to a particular user
| 3. ${NAME1,NAME2,...} MESSAGE -> To send message to a particular set of users
| 4. /rooms -> To list all the rooms with the users in them
| 5. /join ROOM_NAME -> To join an existing room or to create a room
| 6. /leave -> To leave the joined room
+-----+
+-----+
| Enter your name: Soham
+-----+
You: hello
You: exit
Bye!!
```

## Timeout (120 sec)

After 2 minutes the user **Adarsh** is banned due to time out, and he can't send any more messages now.

```
adarsh-dhakar@adarsh-dhakar:~$ ./client localhost 1800
adarsh-dhakar@adarsh-dhakar:~$ ./client localhost 1800
How to send a message?
1. By default the message will be sent to 'All' users.
2. ${NAME} MESSAGE -> To send the message to a particular user
3. ${NAME1,NAME2,...} MESSAGE -> To send message to a particular set of users
4. /rooms -> To list all the rooms with the users in them
5. /join ROOM_NAME -> To join an existing room or to create a room
6. /leave -> To leave the joined room
+-----+
+-----+
| Enter your name: Adarsh
+-----+
You:
Debargha: Hello
You: Hi
You:
You have been banned from the server. [Reason: Time Out]
```

## Explanation of Flow and Working:

### Server (threaded):

#### 1. Main Function / Initialization:

The server initiates by creating a **server socket as any typical socket**. Upon establishing this socket, the server will make a **thread** to handle that client's communication. This ensures the server can handle multiple clients simultaneously without blocking or waiting on individual connections.

#### 2. Client Handler Function:

The Client function is invoked within a thread to manage the client's communication. Initial variables are set as required. Connection acceptance and client identification is done.

#### 3. Timeout Thread:

Client spawns a separate timeout monitoring thread for each client. This thread is responsible for periodically checking the client's inactivity time and ensuring that the client does not exceed the allowed timeout limit.

#### 4. Message Processing:

Messages are processed and parsed based on the following commands.

- a) **By default** the message will be sent to '**All**' users.
- b) **\$/[name]message** → To send the message to a particular user
- c) **\$/[name1,name2,..]message** → To send message to a particular set of users
- d) **/rooms** → To list all the rooms with the users in them
- e) **/join ROOM\_NAME** → To join an existing room or to create a room
- f) **/leave** → To leave the joined room
- g) **exit** → To exit the chat

**Note:** joining another room automatically takes you out of your current room.

---

### Client:

#### 1. Main Function / Initializations:

The Client is responsible for handling the **bidirectional communication** between the client and server. To efficiently manage the flow of messages, the client utilizes **two distinct threads**—

- one for reading messages from the server
- and another for writing messages to the server.

## **2. Read Thread:**

It constantly reads data from the server's socket and processes or displays it accordingly.

## **3. Write Thread:**

It waits for the user to input messages and sends them through the socket connection (using write() ).

The write thread is responsible for gathering and formatting user input or system-generated messages and dispatching them to the server.

Both threads operate independently, ensuring that sending or receiving messages doesn't block the other operation.

---

2.

### **i) Develop a Chat Server program using threads which can:**

- **Handle multiple clients at the same time.**

Multiple client's read and write are handled at the same time, using threads.

- **A client can join/disconnect from the chat.**

**Exit** functionality is implemented for the same.

- **Two clients can chat via server.**

Private messaging is implemented.

- **A client can choose to broadcast the message to all clients alive.**

Messages broadcasting is handled as a default case.

- **Add more functionalities**

**Functionalities like :**

a) **timeout**

b) **create and join a group**

c) **message and leave a group are also implemented**

## ii) Design the Chat Server program (in place of threads) use select() system call to connect multiple clients.

Chat Server Program is in **server\_select.cpp**

```
Feb 16 15:48
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx: ~/Desktop/CN Lab/LabSheet5/cn_lab_sheet5
ERROR on binding: Address already in use
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx: ~/Desktop/CN Lab/LabSheet5/cn_lab_sheet5$ ./server_select 1880
Server started on port 1880
Adarsh joined the chat.
Active Clients: 1
You:
Avik joined the chat.
Active Clients: 2
You:
Client Adarsh has been banned from the server. [Reason: Time Out]
Active Clients: 1
Avik left the chat.
Active Clients: 0
Adarsh joined the chat.
Active Clients: 1
You:
Avik joined the chat.
Active Clients: 2
You:
Adarsh left the chat.
Active Clients: 1
```

```
Feb 16 15:51
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx: ~/Desktop/CN Lab/LabSheet5/cn_lab_sheet5
Enter your name: Avik
Welcome Avik!
You:
Adarsh->Avik,: Hello
You: Hello
You:
Avik: Hello
You: /join room1
You:
Joined private room: room1
You:
Adarsh (in room1): How are you
You: I am fine
You:
Avik (in room1): I am fine
You: /leave
```

```
Feb 16 15:52
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx: ~/Desktop/CN Lab/LabSheet5/cn_lab_sheet5
Enter your name: Adarsh
Welcome Adarsh!
You:
Avik joined the chat.
Active Clients: 2
You: ${Avik} Hello
You:
Avik: Hello
You: /join room1
You:
Joined private room: room1
You:
Adarsh (in room1): How are you
You:
Avik (in room1): I am fine
```

```
Feb 16 15:53
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx: ~/Desktop/CN Lab/LabSheet5/cn_lab_sheet5
Server started on port 1880
Adarsh joined the chat.
Active Clients: 1
You:
Avik joined the chat.
Active Clients: 2
You:
Client Adarsh has been banned from the server. [Reason: Time Out]
Active Clients: 1
Client Avik has been banned from the server. [Reason: Time Out]
Active Clients: 0
```

## iii) Design the Client program for the chat-server.

Client Program is in **client.cpp**

```
Feb 16 15:54
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx: ~/Desktop/CN Lab/LabSheet5/cn_lab_sheet5
adarsh-dhakar@adarsh-dhakar-HP-Laptop-15s-fq5xxx: ~/Desktop/CN Lab/LabSheet5/cn_lab_sheet5$ ./client localhost 1880

How to send a message?
1. By default the message will be sent to 'All' users.
2. ${NAME} MESSAGE -> To send the message to a particular user
3. ${NAME1,NAME2,...} MESSAGE -> To send message to a particular set of users
4. /rooms -> To list all the rooms with the users in them
5. /join ROOM_NAME -> To join an existing room or to create a room
6. /leave -> To leave the joined room
7. exit -> To exit the chat

Enter your name:
```