Operating Systems - Assignment 1
(Client-Server Model and Threads)

Motivation:

In class, we have taught the client-server model and threads. This assignment asks you to use both techniques to develop an on-line chatting application (e.g., LINE).

Assignment Content:

In this assignment, you need to create a server and some clients. When a client wants to send messages to another client, it has to log in to the server first. The server should maintain the mapping between the username and its socket address. When a client logs in, all other active clients will see that the client is on the line. They will also be informed whenever the client logs out. In addition, two clients can send messages with each other. The server takes charge of relaying messages between them. Here is an example. Suppose that there are two clients, say, Alice and Bill. Alice has logged in to the server. Then, Bill tries to connect to the server:

$ connect 192.168.1.1 1234 Bill

Here, the server's IP address is 192.168.1.1 and its port is 1234. The last parameter indicates the username of the connected client. Suppose that the IP address of this client is 192.168.2.3 and its port is 5678. The server has to map 192.168.2.3 to Bill. Moreover, Alice will see the following information:

<User Bill is on-line, socket address: 192.168.2.3/5678.>

If Alice wants to talk to Bill, she can use the following command:

$ chat Bill "Hello, Bill."

When Alice talks to a guy that does not exist in the system, the server has to notify her:

$ chat HandsomeGuy "Hello~"

<User HandsomeGuy does not exist.>

When Bill wants to leave the system, he can use the following command:

$ bye

In this case, all other on-line clients have to be notified:

<User Bill is off-line.>

In this case, if Alice wants to talk to Bill, the following case will happen:

$ chat Bill "Lend me some money."

<User Bill is off-line.>

Whenever a client connects to the server, the server creates a thread to serve that client. The server maintains a "whiteboard" (i.e., a memory space) for communication. If a client A wants to say something to another client B, A's thread writes words on the whiteboard, the server notifies B's thread, and B's thread reads these words and sends them to B. According to the previous example, the whiteboard's content may be as follows:

Communication	Whiteboard
Alice: $ chat Bill “Hello, Bill.”	[2025 March 5, 14:30:45] Alice is using the whiteboard. <To Bill> Hello, Bill.

On the server side, your program needs to show the whiteboard’s content in real time. Just like what OS should do, Since the whiteboard is shared by all threads, the server must handle synchronization. If two clients are sending messages concurrently, the server has to ensure that the whiteboard can be written by at most one thread at any time. You need to demonstrate this behavior to TA and also show somethings on the whiteboard.

Requirements:

- You need to use UNIX socket programming and Pthread in this assignment.
- You must provide a makefile. TA will deduct your points if there is no makefile or the makefile is erroneous.
- You must submit a README file along with your program. The README file should briefly describe how you write your codes (for example, the idea of your program).
- You must demonstrate your program. TA will announce the demonstration time.

Grading Policy:

You need to submit your codes and demonstrate your program to TA. The due day of this homework is **4/2**. You will get **no point** if you do NOT demonstrate your program (even if you submit codes). Discussion among your classmates is encouraged. However, plagiarists will get **ZERO point**. Below are the points you can get in this homework:

- Main program: 90%
- Code comments & README: 10%