NETWORK PROTOTYPE

Introduction:

This is a prototype of a server-client network. It currently supposes multi-client registration, broadcasting messages, and private messaging. The server monitors all the user-inputs, so it would be easier to debug. It current supports 16 threads, and the number of threads can be increased as farther requirements come.

Design:

Once the server socket is created, the server loops for new clients to join. Whenever a new client joins, the server creates a new thread for the client; the server also notifies every other client that is current on this network. Each thread awaits user's input, upon receiving a message, the server checks if the message starts with a '-' character, which indicates that it is a special command. We currently support two different special commands, "-list" and "-pm": -list will provide the user all the user ids that are currently connected to the server. -pm (properly "-pm USER_ID MESSAGE") will search for user with the given id, and send a private message to that users, once the message is delivered, the server will notify the sender, that the message is delivered successfully. A client can disconnect by "C + c", the server can terminate by "C + c" as well, and will also terminate when all the clients disconnect.

Challenges:

One of the biggest challenges was how the clients send its message to the server. We have tried "fgets(msg_in, stdin)", "scanf("%s", msg_in)", "getchar(msg_in)", "scanf("%[^\n]\n", msg_in)", and "scanf("%[^\n]s", msg_in)", which all resulted jamming the transmission, with fgets() being the least destructive: user has to press ENTER twice, and scanf() being the worst: printing the same input indefinitely. We spent a dozen of hours on this supposedly trivial problem, and finally settled on scanning every char individually.

Another problem we had early on, was that the server would terminate prematurely, if a client disconnects. The problem was resolved by adding the join function after our main loop.