REZK Hania
SURESH Sabana
L3 IM2D

Projet Système : messagerie instantanée

Exécution du programme:

Pour exécuter le programme:

1- Ouvrir votre terminal (qui jouera le rôle du serveur) et accéder au dossier qui contient voter code

2- Compiler les 2 codes:
gcc -Wall -o server server.c
gcc -Wall -o client client.c

3-Ouvrir au moins un autre terminal (qui jouera le rôle du client) et accéder au dossier qui contient voter code

4- Exécuter : ./server dans votre terminal qui joue le rôle du serveur et ./client dans vos terminaux qui jouent les rôles de clients.

5- Envoyer des messages entre les clients.

Rapport du projet:

Les deux outils principaux que nous avons utilisés lors de ce projet sont: les threads et les sockets.

Les sockets pour la mise en place du serveur (dans server.c) ont été configurés pour accepter les connexions entrantes des clients. Une fois qu'un client s'est connecté avec succès, un thread distinct est créé pour gérer la communication avec ce client spécifique. Cela a permis à plusieurs clients de se connecter simultanément au serveur et d'interagir de manière indépendante.

D'un autre côté, les sockets du côté client (dans client.c) ont été utilisés pour établir une connexion avec le serveur. Une fois la connexion établie, deux threads distincts ont été créés : l'un pour envoyer des messages au serveur et l'autre pour recevoir les messages du serveur en parallèle. Cette approche a permis une interaction dynamique entre les clients, où les messages peuvent être envoyés et reçus en temps réel.

Les threads ont permis à plusieurs clients de se connecter du côté du “server.c”, du côté “client.c” ces threads ont assurés que les opérations de réception et d'envoi de messages ne bloquent pas le flux de communication pour les autres clients connectés.

Nous devons également noter l’utilisation des mutex qui nous a permit d’attribuer à chaque utilisateur un numéro de client unique. On a utilisé une variable globale qui s’incrémente à chaque fois qu’un utilisateur se connecte et qui attribue alors à cet utilisateur sa valeur comme numéro de client. Cela a été fait correctement avec les mutex et les “lock” ,“unlock” pour ne pas avoir plusieurs threads qui modifient la valeur de la variable globale en même temps.

Enfin, nous avons optimisé la synchronisation de la réception des messages côté client en débloquant les sockets avec le flag O_NONBLOCK. Cela a permis au client de surveiller les entrées sans se bloquer lorsqu'il n'y avait pas de données disponibles immédiatement.