INFO-F-201 – Systèmes d'exploitations
Projet de programmation système en C/C++
Salon de discussion
Rapport

Morari Augustin - Bourgeois Noé

19 December 2021

Contents

1 Introduction

Toute organisation doit avoir un moyen pour faire communiquer ses employés instantanément à distance. Un outil pareil est crucial pour le travail en groupe et il peut augmenter le taux de réussite de ces groupes.
L'ULB utilise un outil pareil qui s'appelle Microsoft Teams.
Cependant, pour des raisons philisophiques et budgétaires, nous avons développé notre propre solution interne en C.
En plus, afin de garantir l'indépendance de l'application, aucune librairie externe n'a été utilisée.

1.1 Présentation du programme

Notre programme est composé d'un serveur et d'un ou plusieurs client(s) qui communiquent entre eux.
Le serveur est lancé en premier et le client s'y connecte avec une adresse ip valide et un port ouvert.

1.2 Compatibilité

Ce programme doit être exécuté dans un terminal Linux.

1.3 Server

Le serveur doit être lancé avec la commande qui suit :
./serveur ⟨port⟩
Il prend en paramètre le port qu'on va utiliser pour communiquer à travers le socket. Le serveur détient une base de données qui sauvegarde les pseudos de tous les clients qui se connectent au serveur et supprime leur pseudo si ceux derniers se déconnectent.
Grâce à notre implémentation de l'appel système *select*, le serveur peut gérer la réception de messages simultanés depuis différents clients.
Une fois le message et le timestamp lu, le serveur renvoie le message, le timestamp et le pseudo à tous les clients, y compris celui qui l'a envoyé.
Si un client décide de se déconnecter un message de déconnection avec son pseudo est affiché.
L'exécution du serveur peut être arrétée par un signal SIGINT, par exemple par la combinaison Ctrl+C.

1.4 Client

Le client doit être lancé avec la commande qui suit :
./client ⟨pseudo⟩⟨ip_serveur⟩⟨port⟩
L'utilisateur est constamment demandé d'envoyer un message et il peut en envoyer un en appuyant sur la touche Enter.
Grâce au multithreading notre client peut également recevoir ses propres messages ou les messages envoyés par d'autres utilisateurs en même temps.
Une fonctionnalité utile de notre programme est le blocage des accès concurrents aux ressources partagées. Ceci est géré par les appel à *mutex.*
Le client peut se terminer si l'utilisateur appuie sur Ctrl+D. Il peut également terminer son exécution s'il perd la connexion avec le serveur, un message sera affiché dans ce cas-ci.

2 Connexion

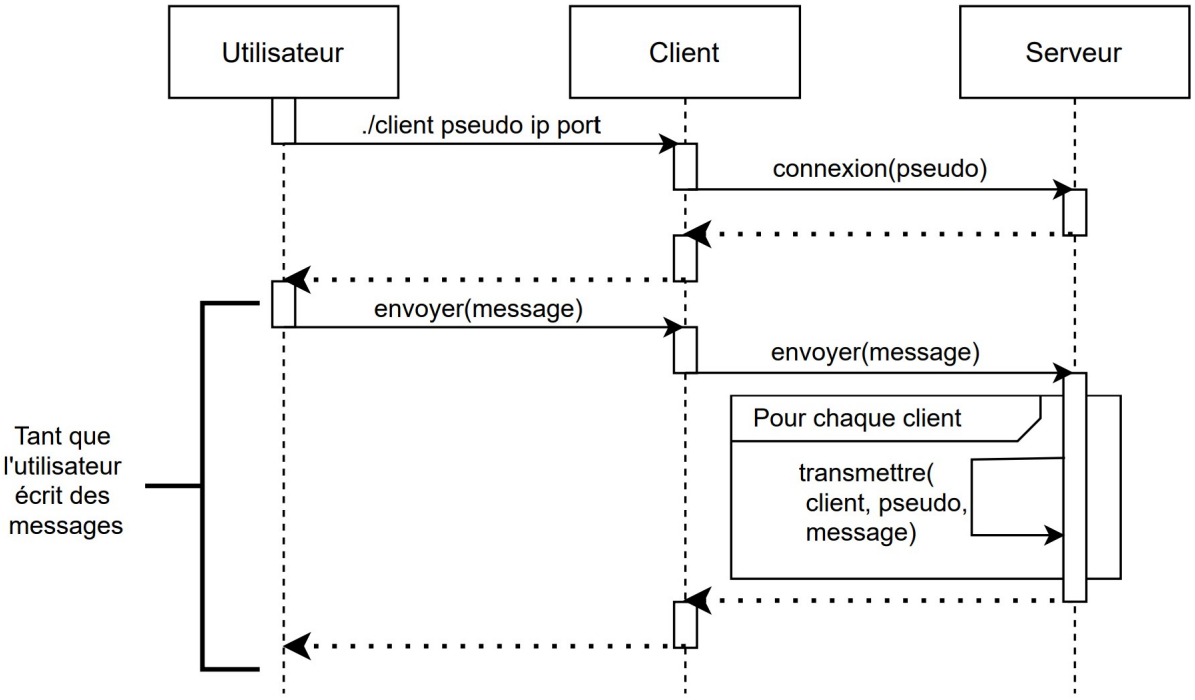

FIGURE 1 – Diagramme de séquence

source: énoncé

3 Pseudo

Après connexion, le pseudo est envoyé par le client au serveur et celui-ci l'enregistre dans une base de données afin de l'associer aux futurs messages provenant de ce client. À la déconnexion d'un client, le pseudo est supprimé de la base de données.

4 Timestamp

Chaque message envoyé par le client est accompagné d'un timestamp indiquant le jour et l'heure de l'envoi.

5 Conclusion

5.1 Limitations

Le salon peut actuellement accueillir un maximum de 1024 clients connectés en même temps. Mais cette constante est modifiable dans le programme à défaut d'être paramétrée à l'exécution. Nous n'avons pas testé les limites d'accueil mais elles sont probablement bien supérieures au besoin du commanditaire.

5.2 Améliorations possibles

- suppression des anciens prompts dans l'affichage terminal
- interface utilisateur
- annonce de connexion d'un utilisateur aux autres utilisateurs déjà connectés