INFO-F-201 – Systèmes d'exploitations

Projet de programmation système en C/C++

Salon de discussion

1 Enoncé

Actuellement, le système de discussion instantanée de l'ULB est assuré par l'application Microsoft Teams. Cependant, pour des raisons philisophiques et budgétaires, les autorités de l'ULB souhaitent développer leur propre solution en interne et vous demandent de développer cet outil en C/C++. Afin de garantir l'indépendance de l'application, l'utilisation de librairies externes est proscrit. De plus, pour un résultat de la meilleure qualité possible, les autorités de l'ULB suggèrent des groupes de deux étudiants.

2 Exigences

L'application de chat est logiquement divisée en deux parties distinctes : un client et un serveur qui interagissent comme montré dans la Figure 1.

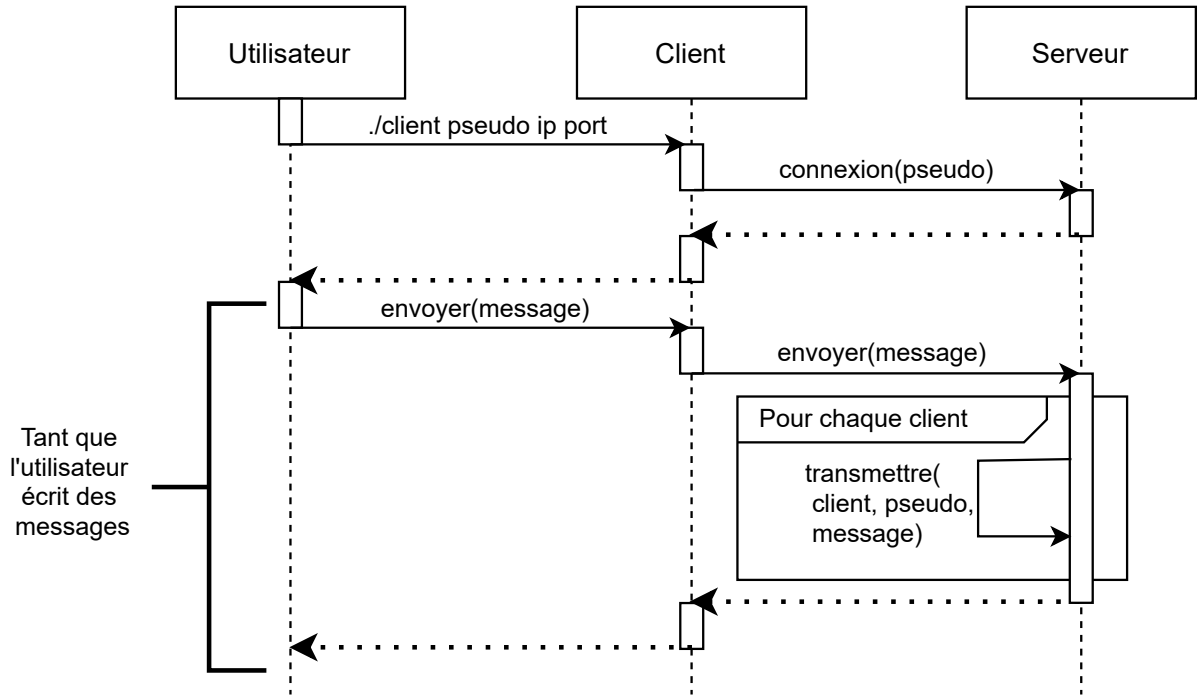

FIGURE 1 – Diagramme de séquence

2.1 Serveur

Le serveur porte deux rôles.
- Gérer les connexions et déconnexions des clients
- Transmettre les messages reçus à tous les clients, y compris celui qui l'a envoyé.

On lance le serveur avec la commande

```
$ ./serveur <port>
```

2.2 Client

Le client est l'application que chaque utilisateur va utiliser pour discuter avec d'autres personnes. Le client se présente sous forme d'application dans le terminal.

On lance le client avec la commande

```
$ ./client <pseudo> <ip_serveur> <port>
```

2.3 Fin du programme

Le client peut se terminer de deux manières différentes
- L'utilisateur souhaite se déconnecter, et appuie sur `CTRL+D`
- La connexion est perdue avec le serveur

Dans les deux cas, le client se termine proprement et affiche un message à l'utilisateur.

Le serveur s'arrête en lui envoyant un `SIGINT`, par exemple via `CTRL+C`. Veillez à gérer ce signal correctement pour que le serveur se termine proprement.

3 Implémentation

3.1 Client

Etant donné que le client doit pouvoir recevoir des messages et en envoyer simultanément, il vous est demandé d'implémenter ces fonctionalités à l'aide de **threads** noyaux (`pthread`). N'oubliez pas de gérer les accès concurrents aux ressources partagées.

3.2 Serveur

Le serveur doit lui aussi gérer la réception de messages simultanés depuis ses différents clients. Il vous est demandé d'implémenter cette simultanéité à l'aide de l'appel système **select**.

3.3 Format des messages

Lorsqu'un client envoie un message (après connexion au serveur), celui-ci comprend deux informations :

1. un timestamp
2. le message

Si le timestamp a une taille fixe, ce n'est pas le cas du message. Par conséquent, chaque message envoyé sera précédé de la longueur du message en nombre d'octets. Un message complet correspond donc à ce qui est illustré dans la Table 1, **dans cet ordre exact**.

message	
Champ	Type
Longueur du message	`size_t`
Timestamp	`time_t`
Message	`char*`

TABLE 1 – Structure d'un message

Remarque : Si vous avez correctement implémenté la séquence expliquée dans la Figure 1 et si les messages sont formatés conformément à la Table 1, vous devriez être capable de tester votre client/serveur avec le serveur/client d'un autre groupe.

4 Critères d'évaluation

Ce projet est à réaliser par **groupe de deux**.

Programmation système Ce projet de va principalement évaluer votre compétence à manier correctement les outils liés aux systèmes d'exploitation vus au cours des TPs (threads, sockets, mutex, ...) dans le langage C. La pertinence des outils utilisés ainsi que la manière dont ils sont utilisés (trop, pas assez, au mauvais endroit, trop longtemps, ...) sont evalués.

Rapport Ce projet doit contenir un rapport dont la longueur attendue est d'environ trois pages de contenu (maximum cinq). Ce rapport décrira succintement le projet, les choix d'implementation si nécessaire, les limitations de votre projet, les difficultés rencontrées et les solutions originales que vous avez fournies.

Code Votre code sera aussi évidemment examiné en termes de clarté, de documentation, de commentaires et de structure.

Votre projet doit compiler avec `g++` sans erreur ni avertissement (warning) avec le flag `-Wall` dans les salles machines. Si votre code ne compile pas, votre note sera automatiquement 0/20. Chaque warning vous fera perdre 1 point sur la note totale.

4.1 Pondération

- Programmation système **/12**
- Rapport **/4**
- Code **/4**

5 Remise du projet

Vous devez remettre un fichier zip contenant
- vos sources
- un Makefile
- votre rapport au format PDF

Vous devez soumettre votre projet sur l'**Université Virtuelle** pour le dimanche **19 décembre 2021 23h59** au plus tard.

Retards

Tout retard sera sanctionné d'un point par tranche de 4h de retard, avec un maximum de 24h de retard et devra être soumis sur l'université virtuelle.