TP8 - Sockets bis

Exercice 0 - Instructions

Il y a un exercice à rendre, sur 6 points.

Si l'exercice est fini, vous avez l'opportunité de refaire d'anciens exercices et de me demander de l'aide. Profitez-en, ~~la fin~~ le contrôle final est proche !

Exercice 1 - `irc`

Lire l'ensemble de l'exercice avant de commencer à coder

Comportement attendu

Nom du fichier : `votrelogin_irc.c`

Le but de cet exercice est de se renforcer sur les sockets.

On va implémenter un mini-serveur de messagerie dans le style d'IRC.

Les clients seront des `nc`, comme dans le TP précédent.

Un serveur gère une salle de discussion.

Quand un client se connecte, le serveur lui demande son pseudo, puis envoie `<pseudo> a rejoint le chat` à tous les autres clients.

Quand un client envoie une ligne de texte, le serveur envoie `<pseudo>: <ligne de texte` à tous les autres clients.

Implémentation :

On pourra repartir du code du TP précédent et réutiliser les fichiers `socket.c`/`socket.h`.

On devra gérer un tableau de file descriptors (les numéros des sockets). On peut le faire de taille fixe (ex : 100)

La fonction `server` se contente de faire ça en boucle :

- accepter une nouvelle connexion
- placer le `int` dans le tableau
- créer un nouveau thread pour gérer ce socket

Chaque thread :

- Transforme le file descriptor en `FILE*` avec `fdopen`
- Demande son pseudo au client
- Lit la ligne reçue et la stocke comme un pseudo.
- Envoie `<pseudo> a rejoint le chat` à tous les clients
- Puis, en boucle:
- Lit une ligne
- Renvoie `<pseudo>: <ligne de texte>` à tous les clients

Fonctions utiles

- `socket.c` :
  - `server_socket` (pour créer la socket serveur)
- `sys/socket.h` :
  - `accept` (pour accepter une nouvelle connexion)
- `pthread.h` :
  - `pthread_create` (pour lancer un nouveau thread)
- `stdio.h` :
  - `fdopen` (thread: pour avoir un `FILE *` pour communiquer avec son client)
  - `fprintf` (thread: pour demander son pseudo à son client)
  - `dprintf` (thread: pour envoyer des informations aux autres clients)
  - `fgets` (thread: pour lire une ligne de son client)
- `string.h` :
  - `strcpy` (pour copier l'input vers le pseudo)
  - `strlen` (pour enlever le retour à la ligne à la fin du pseudo)

Exécution

Testez votre programme avec au moins 3 clients.

Bonus :

- Le serveur envoie tous ses messages en jaune, pour distinguer plus facilement ce que tape le client de ce qu'il reçoit du serveur.
- Mettre un mutex pour bloquer toute opération sur le tableau de sockets, afin d'éviter les race conditions.

Démo :