Système Unix

my_irc

Contact
b-psu-330@epitech.eu

EPITECH.
L'ECOLE DE L'INNOVATION ET DE
L'EXPERTISE INFORMATIQUE

Table des matières

Détails administratifs

- Les sources doivent être rendues dans le répertoire `PSU_année_myirc`
  ex : PSU_2013_myirc pour l'année 2013-2014
- L'intégralité de votre projet devra compiler avec un unique Makefile.
- Le binaire du serveur sera `server`.
- Le binaire du client sera `client`.
- Votre Makefile devra contenir une règle `server` et une `client` permettant de compiler les binaires éponymes.
- Ce projet est à réaliser en binôme
- Votre rendu devra contenir un fichier `auteur` avec les logins de chaque membre du groupe, séparés par un ';'

Sujet

- Il s'agit dans ce projet de réaliser un client et un serveur IRC.

Indices Pour ceux qui ne savent pas ce qu'est un IRC, c'est une sorte de CHAT, ou encore un système de discussion en temps réel, qui gère des groupes de discussion appelés "channel", et permet des échanges de fichiers.

- La communication réseau se fera au travers de sockets TCP.
- Votre serveur devra accepter plusieurs connexions simultanées.

Attention, l'utilisation de **fork** est interdite. Vous devrez donc impérativement utiliser **select**

- Votre serveur ne devra pas être bloquant

Un seul **select** est autorisé au sein de votre projet

Indices Cela n'a rien à voir avec des sockets non bloquantes qui, elles, sont interdites (donc pas de fcntl(s, O_NONBLOCK))

- Votre serveur devra implémenter la notion de channel.
- Votre serveur devra respecter la RFC 1459 (Internet Relay Chat Protocol)
- Votre client devra au minimum gérer les commandes suivantes :
  - `/server _host_[:_port_]` : se connecte à un serveur
  - `/nick _nickname_` : définit le surnom de l'utilisateur au sein du channel
  - `/list [string]` : liste les channels disponibles sur le serveur. N'affiche que les channels contenant la chaîne "string" si celle-ci est spécifiée.
  - `/join _channel_` : rejoint un channel sur le serveur
  - `/part _channel_` : quitte le channel
  - `/users` : liste les utilisateurs connectés au channel (liste des nicknames bien entendu)
  - `_message_` : envoie un message à tous les utilisateurs connectés au channel.
  - `/msg _nickname_ _message_` : envoie un message à un utilisateur spécifique
  - `/send_file _nickname_ _file_` : envoie un fichier à un utilisateur.
  - `/accept_file _nickname_` : accepte la réception d'un fichier en provenance

d'un utilisateur du channel.

Contraintes

Votre code sera non seulement non bloquant, mais **devra** également utiliser des tampons "tournants circulaires" afin de sécuriser et optimiser l'envoi comme la réception des différentes commandes et réponses.

Il vous appartient de produire un code "propre", vérifiant absolument toutes les erreurs et tous les cas pouvant amener des problèmes. Dans le cas contraire, nous n'aurons aucun mal à rendre votre serveur inopérationnel (et donc non fonctionnel).

Indices `man nc + Ctrl-D`

Votre client peut être graphique. Vous avez donc la possibilité d'utiliser une librairie graphique (Gtk, SDL, ...) tant que c'est une librairie C, voire C++.

Cependant, la partie traitement réseau devra impérativement être réalisée au travers des fonctions de la lib C ... (pas de QtNetwork par exemple).

Fonctions interdites

- fork

Fonctions autorisées

- la librairie C