ECU : Programmation système

Enseignant :
Dr Moustapha BIKIENGA (bmoustaph@yahoo.fr)

Buts et principes
Les étudiants travailleront en binômes.

A rendre
Vous devez créer un fichier unique RAR ou ZIP, nommé par exemple "Travail_groupe_xx_xx_xx.rar" (xx_xx_xx =concaténation des noms des membres du groupe). Cette archive doit contenir tout fichier relatif à votre projet.
L'archive doit contenir entre autres un rapport (pdf, doc, odt, …) permettant notamment d'identifier les membres du binôme avec les adresses. Le rapport doit aussi permettre d'avoir un aperçu du travail réalisé. L'archive doit également contenir les codes sources commentés, les exécutables, le manuel d'utilisation…

Envoyez l'archive à l'adresse mail bmoustaph@yahoo.fr. Si vous ne recevez pas un mail accusant réception dans les 10 jours qui suivent, renvoyez vos documents.

Date limite
La date limite de remise de projet doit impérativement être respectée :
le dimanche 30 novembre 2020 au plus tard.

Bon travail !

DESCRIPTION DU SUJET

Le projet a pour objectif la réalisation d'une application de chat client/serveur en C permettant:

- d'échanger des messages entre 2 utilisateurs, entre plusieurs utilisateurs, ou à destination de la totalité des utilisateurs connectés sur le réseau;
- de s'envoyer des fichiers.

Pour le réaliser, vous travaillerez de manière incrémentale en créant un version de base qui va évoluer jusqu'à une version définitive. Chaque version doit être archivée.

Version 1 : Etablissement d'un modèle client/serveur
Dans cette version on doit retrouver les fonctionnalités suivantes :

- Création d'un client qui se connecte en TCP à un serveur renseigné par un port et une adresse IPv4
- Création d'un server avec une socket d'écoute qui accepte la connexion et gère les données entrantes.
- Le client doit pouvoir prendre une chaîne de caractère en entrée, l'envoyer au serveur et recevoir de ce dernier la même chaîne de caractère.
- La connexion doit se couper lorsque le client envoi '/quit'. Les sockets crées doivent être fermées et la mémoire allouée aux structures de données doit être libérée.

Version 2 : Serveur Multi-client

- Les clients se connectent au serveur renseigné par un port et une adresse IPv4.

%localhost > ./client port_number IP_address

- Le client doit avoir la possibilité d'envoyer et recevoir des données au/depuis le serveur
- La connexion doit se couper lorsque le client envoie '/quit'. Les sockets créées doivent être fermées et la mémoire allouée aux structures de données doit être libérée.

%> /quit
%> [Server] : You will be terminated
%> Connection terminated
%localhost >

- Le serveur, en recevant une chaîne de caractère depuis un client, doit répéter cette chaîne uniquement à ce même client.

% > Hello World!
% > [Server] : Hello World!

- Le serveur doit pouvoir maintenir 20 (et pas plus) connexions simultanées sans "crasher". Le 21ième client devra se voir refuser la connexion.

%localhost > ./client port_number IP_address
%> Server cannot accept incoming connections anymore. Try again later.

Version 3 : Gestion utilisateurs

- Une fois la connexion établie avec le serveur, le client DOIT s'identifier par son pseudo (commande /nick)
- Le serveur DOIT gérer plusieurs utilisateurs et plusieurs connexions. Les utilisateurs (pseudo) et leurs infos liées à leur connexion (numéro de socket, socket fd) sont stockés par le serveur.
- Le serveur DOIT tenir compte d'un utilisateur ou de son changement de pseudo.
- un client DOIT pouvoir obtenir du serveur la liste des utilisateurs connectés. (commande /who)
- un client DOIT pouvoir obtenir du serveur des informations sur un utilisateur en particulier (commande /whois)
- Le serveur DOIT garder son rôle répétitif

Version 4 : Création de l'application de Chat

- Un utilisateur DOIT pouvoir envoyer un message à tous les autres utilisateurs (broadcast).
- Un message envoyé NE DOIT pas être retransmis à l'expéditeur.
- Un utilisateur DOIT pouvoir envoyer un message privé à un autre utilisateur (unicast).
- Un utilisateur DOIT pouvoir créer un salon.
- Le serveur DOIT retourner un message d'erreur à l'utilisateur qui demande la création d'un salon déjà existant
- Un utilisateur DOIT pouvoir rejoindre et quitter un salon.
- Le serveur DOIT détruire le salon lorsque son dernier occupant le quitte.
- Un message envoyé dans un salon NE DOIT PAS être transmis à d'autres utilisateurs que ceux présents dans le salon (multicast).

Version 5 : Transfer de fichier

- Un utilisateur (l'émetteur) DOIT pouvoir envoyer un fichier à un autre utilisateur (le récepteur)
- Lors du transfert d'un fichier, le récepteur DOIT donner son approbation
- Lors du transfert d'un fichier, le récepteur et l'émetteur doivent avoir confirmation que l'envoi s'est déroulé