Fichiers Très Partagés

Introduction du sujet

FTP (File Transfert Protocol) est un projet basé sur des compétences réseau (encore, et oui), avec des notions de gestion d'accès, de droits, de multi-client et un peu de parsing. Le but étant de coder un client et un serveur FTP, simple.

Travail à réaliser

Partie 1 : Tout commence par le commencement

Coder deux exécutables, un client et un serveur, capables de communiquer entre eux (./lpf pour le client et ./lpf_server pour le serveur). Vous pouvez utiliser des **class réseaux** précédemment développées (par vous-même) ou en faire de nouvelles.
Les familles de fonctions à utiliser sont **socket**, **bind**, **listen**, **accept**, **connect**, **write**, **read**, **send** et **recv**.
Étant donné la nature du projet, vous pouvez, au choix, construire votre projet sur un modèle single thread (un seul thread server et utilisation de **select**/**poll**) ou sur un modèle multi-thread (server avec un thread par client et utilisation de **std::thread**)

La partie 1 est considérée comme validée si le programme client peut envoyer le message "ping" au server et que celui-ci peut répondre "pong", réponse que le client affiche.

Partie 2 : Traitement des commandes

Pour la partie 2, le client doit permettre l'envoie (via le réseau ...) d'un fichier au programme server et de télécharger un fichier depuis le programme serveur, vers le client, via les commandes :

Commandes client :
./lpf ip:port -upload filename (upload un fichier)
./lpf ip:port -download filename (download un fichier)

ip:port : IP et Port du serveur. Exemple : 127.13.37.1:20215
Filename : Nom du fichier a uploader ou à télécharger

Partie 3 : Plus on est de fous ...

Le serveur doit ensuite pouvoir gérer **plusieurs utilisateurs**. Lorsqu'un utilisateur souhaite uploader un fichier, si un dossier existe à son nom, son fichier est upload dans ce dossier. Si aucun dossier ne porte le nom de l'utilisateur, un dossier est créé à son nom.
Si un utilisateur demande à download un fichier, le serveur cherche ce fichier à l'intérieur du dossier appartenant à cet utilisateur.
Un utilisateur peut aussi supprimer un de ses fichiers se trouvant sur le serveur.

Commandes client :
./lpf user@ip:port -upload filename (upload un fichier)
./lpf user@ip:port -download filename (download un fichier)
./lpf user@ip:port -delete filename (supprime un fichier)

user@ip:port : Nom d'utilisateur, IP et Port du serveur.
Exemple : Reyna@127.13.37.1:20215

Partie 4 : Let's OSI Layer 7 ... and stuff

Pour sécuriser un peu les utilisateurs, la partie 4 ajoute un système d'authentification.
Lors de sa première connexion (via l'exécution d'une des commandes ci-dessus), un utilisateur se voit demander un mot de passe. Les mots de passes des utilisateurs sont sauvegardés dans un fichier "very_safe_trust_me_bro.txt", sur le serveur. Ce fichier contient une ligne par utilisateur et chaque ligne correspond à ce format : **username:password**
Exemple :

Solar:PraiseTheSun
BjarneStrouStrup:cpp17

Si l'utilisateur donne le bon mot de passe, sa commande est exécutée.

Si l'utilisateur n'existe pas, le serveur envoie une demande de création de mot de passe au client. Après la création du mot de passe et création du dossier utilisateur, la commande est exécutée.

Pour cette partie, il sera nécessaire de réfléchir à un **protocole de communication** afin que le client et le server puisse exécuter les différentes étapes du processus d'authentification.

Partie 5 : Un peu de folder management

Pour donner aux utilisateurs la possibilité d'organiser leur dossier distant, la partie 5 ajoute de nouvelles commandes client:

Commandes client :
./lpf user@ip:port -list [PATH]
Renvoie la liste des fichiers et dossiers présents le dossier PATH. Si le paramètre PATH n'est pas spécifié, le serveur renvoie la liste des fichiers et dossiers se trouvant a la racine du dossier de l'utilisateur.

./lpf user@ip:port -create foldername [PATH]

Crée un sous dossier dans le dossier PATH. Si le paramètre PATH n'est pas spécifié, le serveur créé un dossier à la racine.

./lpf user@ip:port -rm foldername [PATH]

Supprime le dossier PATH et les fichiers qu'il contient. Si le paramètre PATH n'est pas spécifié, le serveur supprime tous les fichiers et dossier de l'utilisateur.

./lpf user@ip:port -rename foldername PATH

Renomme le dossier PATH. Il est impossible, pour un utilisateur, de renommer son dossier racine.

Partie 6 : RGPD ?

Enfin, mettre en place, sur le serveur, un système de logs pour y sauvegarder l'ensemble des commandes reçues de chaque utilisateur, leur Ip, date de réception de la commande, les erreurs, si il y en a eu, etc ...

Pour aller plus loin...

Bonus :

- ➔ Ajout de commandes supplémentaires
- ➔ Implémenter une interface graphique (Qt (Cute !), SFML, SDL, autre ...)
- ➔ Respect de la RFC 959

Rendu

- ➔ Votre travail est évalué en soutenance avec un support et une revue de code.