Compte Rendu TP-Réseau

Serigne Saliou Gueye et Beaudoin Félix

Fonctionnement du programme:

Le dossier actuel "projet" contient deux répertoires (**fich_test et FTP_client_serveur**) et un fichier (**README.txt**).

- Dans le "**fich_test**", nous avons encore un répertoire test et un fichier test (**dossier_test et fichier_test.txt**).
- Dans le fichier "**FTP_client_serveur**", nous avons trois fichiers (**client.c, Makefile et serveur.c**).
  - Les fichiers "**client.c**" et "**serveur.c**" sont les codes sources respectifs pour le FTP client et le FTP serveur.
  - Le fichier **Makefile** permet de compiler de façon simplifiée les fichiers "**client.c**" et "**serveur.c**".

Lancement du programme:

Vous utilisez la commande *"**make**"* dans le répertoire "**projet/FTP_client_serveur**" pour faire la compilation . Après vous aurez les fichiers exécutable du serveur et du client ainsi que les fichier .o qui leur correspondent. Vous pouvez utiliser la commande *"**make clean**"* pour supprimer les fichiers .o (ils sont optionnels).

Pour faire fonctionner le programme, on exécute les nouveaux fichiers exécutables (serveur.o et client.o renommés en serveur et client) dans deux terminaux en mettant d'abord "**./serveur num_port**" (dans l'un des deux terminaux) puis "**./client num_port**" (dans l'autre). Nous n'avons pas spécifié le numéro de port pour permettre au serveur d'être hébergé sur une autre machine que localhost. Un message peut apparaître "**Erreur du bind**" cela signifie que l'adresse du port est déjà utilisé, vous devez alors prendre un autre port qui lui sera disponible.

Connexion du client:

_____Une fois la commande *"**./client num_port**"*, un message de bienvenue apparaîtra au client en lui disant qu'il doit entrer son identifiant. Le client pour se connecter au serveur doit choisir l'un des identifiants autorisés qui sont "1ABC, 2345, 5468, KE34, 9OLM, 9834, IK34, PO23, 5973, 3ERZ", identifiants qu'on peut bien sûr changer dans le code.
Si vous entrez un identifiant incorrect vous serez déconnecté, vous devrez alors relancer la commande *"**./client num_port**"*. Une fois le client identifié, un message de bienvenue apparaîtra, vous expliquant les commandes que vous pourrez réaliser sur le serveur.
Les commandes:

_____Pour chaque commande, nous avons un outil de mesure indique la durée d'exécution de la commande entre le client et le serveur.

Le programme vérifie que la commande tapée est autorisée pour le client. Si le client entre une commande qui n'est pas répertoriée, un message *"Commande introuvable"* apparaîtra.

Pour les commandes ls**, cd, pwd, get** et **put**, le nom de la commande sera envoyé sur le serveur envoyé par un socket en le stockant dans un buffer. Le serveur traitera alors les commandes en fonction de ce qu'il a reçu.

Fonction ls: Lorsque que le client entre *"**ls**"*, on le stock dans un buffer et il est envoyé par un socket au serveur. Le serveur va recevoir le socket, le regarder et lire "**ls**". Il va alors stocker tous les fichiers du répertoire (documents et répertoires) dans un fichier ***.ls.txt*** (Le point devant *"**ls**"* permet de créer le fichier tout en étant invisible afin de ne pas encombrer le répertoire et éviter d'avoir un fichier *"**ls**"* avec dedans *"**ls.txt**"*). Le serveur va alors copier dans un buffer tout ce qui se trouve dans le répertoire ouvert et le renvoyer au client. Le client est alors obligé de créer un fichier *"**.ls.txt**"* et d'écrire ce qu'il y a dans le buffer afin que le serveur ne soit pas spammé par la création du fichier par les clients. Par la suite on affiche ce qu'il dans son répertoire.

Fonction: Même principe pour les autres fonctions en général:

- Pour *"**cd**"*, le client envoie sa commande, le serveur change de répertoire et le client reçoit un message de confirmation ou d'erreur.
- Pour "***pwd***", on copie tout le chemin jusqu'à notre répertoire dans un fichier ***.pwd.txt*** et on envoie au client un message en lui disant son répertoire.
- Pour *"**get**"*, on demande au client de dire le fichier qu'il veut obtenir. Si le fichier n'existe pas, il reçoit un message d'erreur *"Pas de fichier de ce nom"*. On envoie au serveur le socket qui lui copie tout sur un fichier, le renvoie au client qui lui créer un fichier où il écrit ce qu'il a reçu, et s'affiche sur son terminal.
- Pour *"**put**"*, le client copie son fichier dans un fichier qui sera envoyé au serveur. Si le fichier n'existe pas, il recevra un message d'erreur *"Ce fichier n'existe pas dans le dossier local"*. Le serveur reçoit le socket avec le fichier a put dedans. Il copie tout un nouveau fichier et envoie le nouveau fichier au client. Si le contenu du message est identique, il recevra *"**Fichier bien stocké**"*, sinon *"Le stockage du fichier a échoué".*

Pour quitter le serveur, vous devez tapez *"**quit**"* et vous serez alors déconnecté.