VIAL Fabien
COMBE Gaétan

TP: Serveur FTP

Lien:

HTTPS:
https://github.com/Bylack26/ProjetSRServer.git

SSH:
git@github.com:Bylack26/ProjetSRServer.git

Introduction:

Pour ce TP de création d'un serveur FTP, avec le protocole TCP, nous avons pris comme base le serveur Echo du TP précédent, auquel nous avons ajouté plusieurs fonctionnalités pour permettre le transfert de fichier. Bien que dans un premier temps ce transfert ne pouvait être réalisé que pour un fichier et un seul client, nous avons finalement pus proposer un modèle de serveur permettant de lancer des téléchargements depuis plusieurs clients sur un même serveur, et plusieurs téléchargement par session.

Les Étapes réussies:

Pour commencer nous avons réussi à mettre au point des solutions pour répondre à toutes les questions des étapes I et II. Ainsi les fonctionnalités actuellement implémentées sont :

- Un pool de processus fils de taille fixe (contenue dans la constante NB_PROC), pour un total de NB_PROC + 1 connexions clients possible (en comptant le serveur).
- Le support de tous types de fichier considéré, binaire comme texte.
- Un affichage lors de la fin d'un téléchargement
- Un découpage des fichiers en paquets de taille PAQUET_SIZE mis à 1000 octets.
- Implémentation des commandes "bye" et "get", pour une déconnexion, ou pour récupérer un fichier.
- Relance du téléchargement lors d'une panne client.

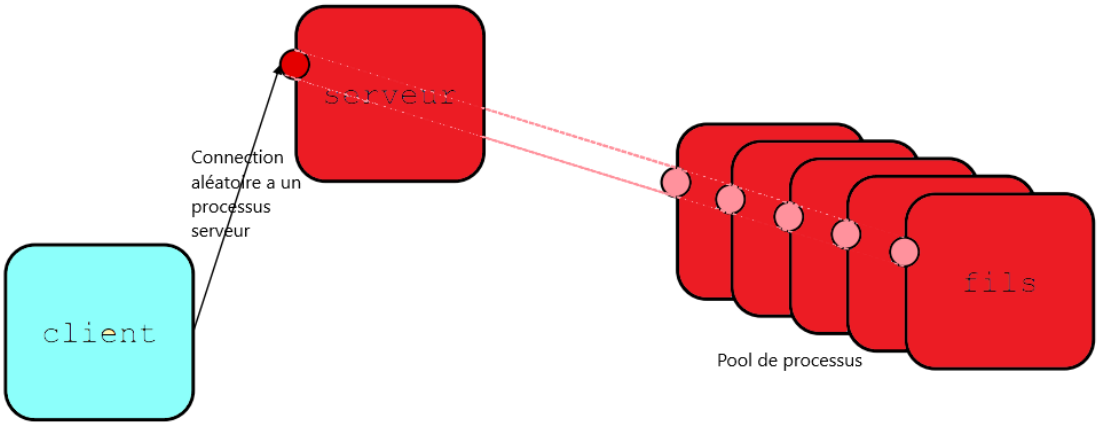
Plusieurs Tests peuvent être réalisé en suivant les instructions suivantes

Les Étapes non réussies:

L'étape III du TP n'a pas été réussi, par manque de temps. Sur le principe, nous pensions connecter le client au serveur maître puis renvoyer depuis le maître au client le socket d'un des serveurs esclaves, puis de dialoguer avec celui-ci. La difficulté aurait été de maintenir à jour tous les esclaves et notamment lorsque deux d'entre eux cherchent à accéder ou modifier le même fichier, il faudrait alors opérer une fusion des modifications. Mais encore une fois la mise en place du serveur FTP normal nous aura plus de temps que prévu.

Fonctionnement du serveur

Schéma des processus



En premier lieu, make permet de générer les exécutables "server" et "client" ainsi que le dossier de téléchargement appelé "DirClient/" dans lequel les fichiers récupérés depuis le serveur seront copiés. Puis il faut les lancer respectivement côté serveur et côté client dans cet ordre expressément. Ensuite le client, depuis la fonction "main", demandera quelle commande il devra envoyer au serveur. Les deux commandes implémentées sont "get" qui permet de récupérer un fichier depuis le serveur et "bye" qui permet de terminer une connexion avec le serveur. La commande est ensuite envoyée au serveur sous forme d'une valeur entière dont la valeur peut être retrouvée dans les constantes présente dans le fichier "serverFunction.h" sous les noms GET_FUNC (1) et BYE_FUNC(0).

Dans le cas où un téléchargement précédent aurait échoué, le client tentera de reprendre le téléchargement là où il s'est arrêté. La captation d'un échec précédent se fait lorsqu'il existe dans le dossier de téléchargement un fichier ".log". Par conséquent, il est nécessaire de disposer de ce dossier au lancement du client. Le fichier ".log" contient le nom du fichier à télécharger, la taille (**T**) de ce nom ainsi que le nombre de paquets déjà reçu.
Il ne reste plus qu'à envoyer au serveur une demande de téléchargement de fichier à partir des octets T*PAQUET_SIZE.

Commandes:

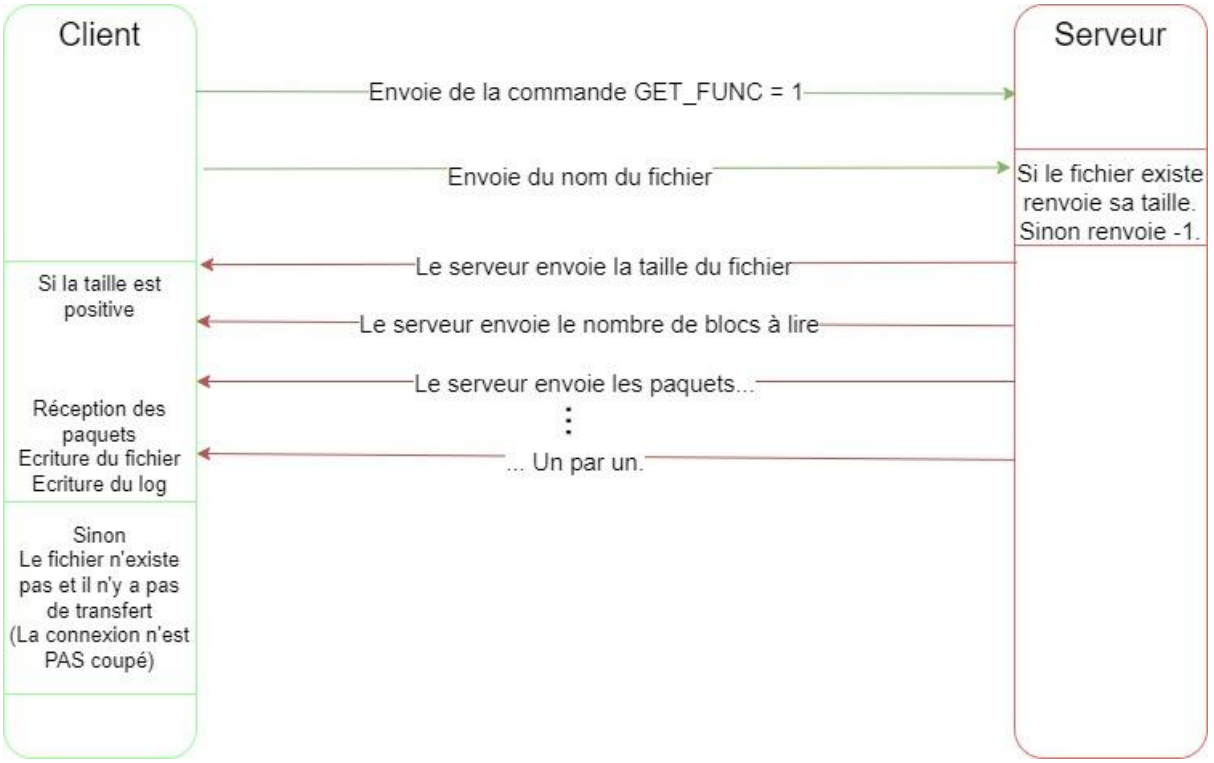
- ***“get”* :**

  Code: GET_FUNC 1:
  La commande “get”, lancée depuis le client, va appeler la fonction “recupereFichier”. Cette fonction va envoyer sur l’entrée standard un nom de fichier et l'envoyer au serveur qui pourra le lire via un Rio_readlineb, de la même manière que sur le serveur echo du TP précédent.
  Le serveur renverra la taille du fichier ce qui permettra au client de savoir si le fichier a été trouvé ou non.
  Enfin après avoir créer un fichier de même nom que le fichier commandé, dans le répertoire fixé par la constante “DIR” , le serveur enverra le nombre de blocs à lire, puis les paquets du fichier que le client écrira à chaque fois dans son propre fichier, les uns à la suite des autres. En cas de panne client un fichier log est écrit et permet de reprendre le téléchargement à la prochaine connexion (voir **Les logs et crashes**)
  Le client et le serveur attendent ensuite une nouvelle commande.

  Exemple: tapez **get** puis entrée. Sur la ligne d’après, taper le nom du fichier voulu. Cela donne:

  get
  ./Donnees/fichier.txt

Schéma du “get”:



- ***“bye”*:**

  Code: BYE_FUNC 0:
  La commande “bye”, lancée depuis le client, va en premier lieu envoyer un 0 en tant que commande au serveur, qui de son côté va fermer la connexion avec le client et remonter dans sa boucle principale d'exécution. Ainsi il pourra attendre une nouvelle connexion. Le client quant à lui, va couper sa connexion au serveur et s'arrêter.

Les logs et crashes:

Nous avons un fichier nommé .log qui possède les informations suivant: taille du nom de fichier suivis du nom du fichier (avec chemin depuis le répertoire courant du serveur. Puis il y a le nombre de paquets déjà reçu par le client avant que celui-ci ne crash.

Lorsqu’on se connecte au serveur, le client vérifie si un fichier .log existe dans son répertoire où il met ses fichiers téléchargés (appelé DirClient chez nous). Si ce .log existe, le client envoie au serveur une commande REPRISE_FUNC (2) signifiant qu’il y a eu un crash puis le client envoie les 3 données du fichier log. Sinon, il n’y a pas eu de crash et le serveur attendra donc une commande.

Dans le cas où un fichier ".log" existerait mais pas le fichier associé, le log sera supprimé et le client ainsi que le serveur reviendront en état d'attente de commande.

Lorsque le client télécharge un fichier, à chaque paquet de donné reçu, il supprime puis ré-écrit dans le fichier .log les 3 données. Quand le téléchargement est fini, le fichier .log est supprimé.

Test :

Téléchargement d'un fichier:

Lancez un "make" et executez le serveur (./server) puis le(s) client(s) (./client).
Tapez la commande "get", puis Entrée.
Choisissez un fichier (Le dossier ./Donnees/ contient different type de fichier dont ce pdf, ainsi qu'un Lorem Ipsum et une capture d'écran). Il faut taper le chemin relatif complet du fichier
Le serveur envoie le fichier et à la fin de l'envoie vous pouvez voir dans le dossier DirClient le fichier copié.
Vous pouvez maintenant relancer une commande.

Eteindre le client en plein téléchargement:

Dans le fichier serverFunction.c, dans la fonction taille(int connfd), dans la dernière boucle while se trouve un sleep(1). Dé-commentez-le et relancez une compilation (vous pouvez lancez un make clean optionnel avant).
Effectuez un téléchargement comme montré précédemment et tuez le client avec CTRL+C avant la fin du téléchargement (le fichier Fichier.txt est sûrement trop court pour avoir le temps de faire la manipulation rapidement).
Relancez ensuite le client. Le téléchargement reprend.

Variantes:

Avant de relancer le client, vous pouvez supprimer le fichier téléchargé. Le client affichera une erreur, le téléchargement ne recommencera pas et vous pourrez entrer une nouvelle commande.

Si vous supprimez le fichier .log caché, il ne sera pas possible de lancer une restauration.

Vous pouvez fermer le serveur entre temps, tant que vous ne supprimez pas le fichier téléchargé ni le .log.

Si vous lancez un autre client depuis un endroit où il a accès au dossier DirClient alors il reprendra le téléchargement.

Tuer le serveur avant le client:

Si vous tuez le serveur avant le client que ce soit pendant un téléchargement ou avant d'envoyer une commande, le client indiquera que le serveur a rencontré des problèmes et fermera sa connexion.