Université Joseph Fourier, Grenoble
Licence d'Informatique L3

Introduction aux Systèmes et Réseaux

TP n°8 : Implémentation d'un pseudo serveur FTP

L'objectif du TP est de construire un serveur de fichiers inspiré par le protocole FTP [1]. Les différentes sections correspondent aux différentes étapes du projet.

Traiter les étapes dans l'ordre de leur présentation. En cas de doute, se référer à votre enseignant de TD/TP.

Rendu de code Chaque binôme devra rendre le code source commenté des programmes réalisés et un bref compte-rendu présentant : (a) les principales réalisations et (b) une description des tests effectués. Une démonstration des réalisations sera organisée en fin de semestre (détails indiqués en temps utile).

Date limite 29 mars 2019 au soir.

1 Serveur FTP concurrent

Le but de ce TP est de programmer un serveur FTP permettant uniquement de récupérer le contenu d'un fichier à distance à partir de son nom. La partie serveur doit être implémentée sous la forme d'un serveur multi-processus ayant un nombre fixe `NPROC` de processus exécutants. Lors de l'arrêt du serveur FTP, celui-ci veillera à terminer les `NPROC` processus exécutants qu'il aura lancé. Pour se faire, le serveur retransmettra le signal `SIGINT` à chacun de ses fils via la programmation d'un traitant de signal approprié. Pour cette étape, vous pouvez vous inspirer de votre réponse à la question 2.2 du TP7.

Le serveur ouvre une socket d'écoute sur le port 2121 [2]. Après réception du nom de fichier demandé par le client, le serveur le charge en mémoire en une seule fois. Il le transmet ensuite dans le flot TCP à destination de celui-ci. A la fin de la transmission, le serveur ferme la connexion avec le client (une seule demande de fichier par connexion).

On considère que le client se connectera au serveur FTP uniquement sur le port par défaut (2121 dans ce TP) ; ainsi ce port ne doit pas être spécifié par l'utilisateur mais codé en dur dans le code du client. Le fichier envoyé par le serveur FTP sera sauvegardé par le client dans son dossier courant. Un message sur la sortie standard confirmera la bonne réception du fichier. Ce message contiendra également des statistiques sur l'envoi (voir exemple ci-dessous). En cas d'erreur (e.g., fichier manquant, crash du serveur), le client affichera un message d'erreur et terminera proprement son exécution.

Exemple d'une exécution correcte :

1. `https://www.ietf.org/rfc/rfc959.txt`
2. Le port par défaut attribué aux serveurs FTP est le port 21. Dans ce TP, nous utiliserons le port 2121 pour éviter de lancer le serveur en mode superutilisateur. En effet, pour des raisons de sécurité, l'utilisation d'un port inférieur à 1024 nécessite des droits d'accès root.

```
$ ./clientFTP mon_serveur_FTP
Connected to mon_serveur_FTP.
ftp> get Nom_du_fichier
Transfer successfully complete.
X bytes received in Y seconds (Z Kbytes/s).
```

Remarque : on considère que tous les fichiers sont stockés à la racine du serveur FTP (pas de commande de type `cd`, `pwd`, ou `ls` à implémenter dans cette première étape).

2 Découpage du fichier

Modifier le chargement du fichier en mémoire pour le charger par blocs de taille fixe (au lieu de le charger en une seule fois). L'objectif étant de ne pas monopoliser la mémoire lors du transfert de fichiers volumineux.

3 Gestion simple des pannes coté client

Modifier la version précédente en y ajoutant une gestion des pannes coté client. Si un client crashe pendant un transfert, le serveur doit continuer à fonctionner correctement en nettoyant proprement les structures système. Si le client est relancé, il doit reprendre le transfert qui a été interrompu (ne télécharger que la partie manquante du fichier).

4 Serveurs FTP avec équilibrage de charge

Modifier la version précédente en ajoutant un répartiteur de charge (*load balancer*). Le répartiteur de charge (serveur maître) distribue les requêtes des clients entre plusieurs serveurs esclaves. Toutes les demandes de fichiers passent par le serveur maître qui les transmet ensuite aux serveurs esclaves en suivant une politique *Round-Robin* (un tourniquet simple). Vous devez trouver un moyen afin de faire communiquer le client et le serveur esclave. On considère que tous les serveurs esclaves disposent de tous les fichiers (les fichiers sont dupliqués).

5 Plusieurs demandes de fichiers par connexion

Modifier votre implémentation pour permettre aux clients d'effectuer plusieurs demandes de fichiers les unes après les autres, sans renouveler la connexion avec le serveur FTP. Plus précisément, le serveur FTP ne devra plus fermer la connexion après avoir envoyé le fichier au client. A la place, la connexion sera terminée par le client en utilisant la commande `bye`.

6 Commandes `ls`, `pwd` et `cd`

Modifier la version précédente afin d'enrichir votre serveur FTP avec les commandes suivantes :

— `ls` : retourne le contenu du répertoire courant du serveur FTP,
— `pwd` : affiche le chemin courant du serveur FTP,
— `cd` : change le dossier courant sur le serveur FTP.

Le serveur maître et ses esclaves ont donc une arborescence correspondant à une partie du système de fichier.

7 Commandes `mkdir`, `rm`, `rm -r` et `put`

Enrichir le serveur FTP avec les commandes suivantes :

— `mkdir` : permet de créer un dossier sur le serveur FTP,
— `rm` : permet de supprimer un fichier sur le serveur FTP,
— `rm -r` : permet de supprimer un dossier sur le serveur FTP,
— `put` : permet de téléverser un fichier sur le serveur FTP.

Si un client effectue ce genre de requêtes, le serveur auquel le client est connecté doit répercuter ces modifications sur tous les autres serveurs esclaves (chaque système de fichier doit être mis à jour au bout d'un temps fini). On ne demande pas de cohérence forte ici (si un client se reconnecte, il n'a pas la garantie de voir ces modifications immédiatement prises en compte).

8 Authentification

Les commandes implémentées dans l'étape précédente modifient les fichiers auxquels un serveur FTP donne accès et sont donc dangereuses. Pour protéger le serveur contre leur utilisation malveillante, mettre en place un système d'authentification où seulement certains utilisateurs (login :mot de passe) pourront les utiliser. Une tentative d'utilisation de ces commandes sans authentification préalable devrait échouer.