TP2 : FTP

L'objectif de ce TP est de réaliser une application client/serveur suivant le protocole FTP (File Transfer Protocol) dans une version restreinte, qui ne fournit pas toutes les fonctionnalités usuelles.

1 Fonctionnement

FTP est un protocole basé sur TCP permettant le *transfert de fichiers*, de manière indépendante de l'OS des machines mises en jeu. Il fournit une interface de type UNIX au client, qui peut ainsi (après identification) lister, lire et écrire des fichiers.

1.1 Communication

FTP utilise deux sockets de communications entre le client et le serveur : une socket de contrôle et une socket de données. Les commandes passent par la socket de contrôle, tandis que les fichiers transférés passent par la socket de données. La socket de contrôle est ouverte par le serveur, traditionnellement sur le port 21[1]. La socket de données peut être ouvert soit par le client (mode actif), soit par le serveur (mode passif). Ceci permet de pouvoir utiliser le protocole FTP sans que le client ne doive signifier à son firewall de laisser un port disponible en écoute.

1.2 Identification

Les commandes d'identification sont

- `USER <login>`
- `PASS <password>`

Attention, le protocole FTP n'est pas sécurisé. Toutes les informations sont envoyées en clair sur le réseau.

Cette authentification permet de gérer des permissions (côté serveur) pour chaque utilisateur. Ainsi, on peut facilement cantonner un utilisateur à un certain sous-dossier. Le login *anonymous* est spécial : il permet à un utilisateur de se connecter au serveur sans avoir de compte créé au préalable. Dans ce cas, il peut fournir n'importe quel mot de passe (la convention veut qu'on fournisse son adresse mail comme mot de passe lors d'une telle connexion).

1.3 Commandes classiques

Les commandes à implémenter dans ce TP sont:

- `LIST` : renvoie la liste des fichiers du dossier courant (similaire à `dir` sous Windows ou `ls` sous Unix).
- `CWD <dossier>` : déplace le répertoire courant.
- `RETR <fichier>` : demande le téléchargement du fichier.

De nombreuses autres existent, par exemple pour renommer, créer ou supprimer des fichiers.

[1] Comme vous n'avez pas les droits root sur les machines, il vous faudra utiliser un port non standard.

1.4 Mode actif/passif

Dans le mode actif, c'est le client qui ouvre un port sur sa machine. Il envoie ensuite la commande `PORT a,b,c,d,e,f` sur la socket de commande. Le serveur se connecte à l'adresse IP `a.b.c.d` sur le port $256e + f$ puis y envoie le fichier.

Dans le mode passif, le client doit demander au serveur sur quel port se connecter grâce à la commande `PASV`. Il reçoit alors une réponse de la forme `a,b,c,d,e,f`, et peut alors se connecter à l'adresse IP `a.b.c.d` sur le port $256e + f$.

1.5 Réponses du serveur

Chaque réponse du serveur contient un code (entier à 3 chiffres) et un message. Le premier chiffre du message indique son type :

- 1xx : l'action demandée a commencé mais n'a pas encore fini
- 2xx : l'action demandée a été effectuée avec succès
- 3xx : plus d'informations sont requises
- 4xx : l'action demandée n'a pas été effectuée mais pourra être retentée plus tard
- 5xx : l'action demandée est impossible

Une liste exhaustive est disponible ici. Voici quelques exemples courants:

- 150 Ouverture de la connexion en cours
- 200 OK
- 331 Utilisateur reconnu, en attente du mot de passe (réponse du serveur à une commande `USER`)
- 430 Identifiant ou mot de passe incorrect
- 501 Erreur de syntaxe

2 À vous de jouer

Dans ce TP, vous devez implémenter un serveur FTP permettant de répondre aux commandes USER/PASS, DIR/CWD/RETR et PASV/PORT.

Le serveur aura un utilisateur (login: toto, mot de passe: password) mais acceptera les connexions `anonymous`.

Les fichiers/dossiers seront localisés sur votre machine dans un dossier `Ressources`. Ce dossier contiendra des fichiers et un sous-dossier `toto`. L'utilisateur `anonymous` pourra accéder qu'aux fichiers dans `Ressources` mais pas dans `toto`, tandis qu'un client identifié comme toto pourra accéder au contenu de `toto`.