3BOUKROU Malik
DURAND Jean-Frédéric
23/02/2016

TP 1 : SERVEUR FTP CAR

I) Introduction

L'objectif de ce TP est d'implémenter un serveur FTP qui gère les commandes de base de tout serveur FTP existant, de plus ce serveur doit respecter la norme RFC 959.
L'utilisateur doit donc être capable de se connecter à ce serveur via n'importe quel client FTP valide et effectuer des transferts de fichiers depuis ou vers ce serveur.

Parmi la longue liste de commandes qu'un client peut exécuter sur un serveur FTP, nous devions implanter les suivantes :

- USER
- PASS
- LIST
- RETR
- STOR
- QUIT
- PWD
- CWD
- CDUP

Nous expliquerons en détail l'utilité et le fonctionnement de ces différentes commandes dans la doc fournie.

II) Architecture

Nous avons décider de découper l'application comme suit :

- une classe **Serveur** qui s'occupe d'initialiser et de lancer le serveur
- une classe **FtpRequest** qui est appelée dans un nouveau thread pour chaque nouveau client qui souhaite se connecter au serveur et qui gère les différentes commandes appelées par le client ainsi que les réponses qui lui sont fournies
- une classe **Authentification** qui regroupe les principales variables globales du serveur comme les comptes utilisateurs, le numéro de port du serveur, le nombre autorisé de connexion simultanées...etc
- une classe **Main** qui crée une instance de Serveur et qui met en ligne ce serveur

III) Code Samples

```
package main;

public class Authentification {

    //Liste factice d'utilisateurs avec leur mdp associÃ©
    public static String[] username = {"admin", "user", "malik", "jf"};
    public static String[] password = {"admin", "user", "malik", "jf"};

    //NumÃ©ro de port du serveur
    public static int port = 3000;

    //Nombre max de connexion simultannÃ©es
    public static int nbLimit = 20;

}
```

On voit ici dans la classe Authentification l'initialisation des différentes variables globales à l'application.
Ce qui rend l'application maniable en cas de modification de ports ou de comptes utilisateur.

```
public void processRequest(String req){
    if(req != null){
        if (req.split(" ").length > 1){
            this.cmd = req.split(" ")[0];
            this.data = req.split(" ")[1];
        }
        else{
            this.cmd = req;
            this.data ="";
        }
        System.out.println("cmd :"+cmd);
        System.out.println("data :"+data);
    }

    if (cmd.contains("USER"))
        processUSER();
    if (cmd.contains("PASS"))
        processPASS();
    if (cmd.contains("RETR"))
        processRETR();
```

Dans le processRequest, on récupère la requête envoyée par le client, et ensuite on découpe selon le caractère espace et on appelle la fonction relative au traitement que souhaite réaliser le clients.

```
 private void processUSER(){
     if (!this.auth){
         for (int i=0; i < Authentification.username.length; i++){
             if ((this.data.equals(Authentification.username[i]))){
                 this.user = Authentification.username[i];
                 this.pass = Authentification.password[i];
                 this.auth = true;
                 sendToClient(331," user ok");
                 this.nbConnect += 1;
                 this.path = "/home/"+this.user;
             }
         }
         if(!this.auth){
             sendToClient(530," bad user");
             processQUIT();
         }
     }
     else{
         sendToClient(530," bad user");
         processQUIT();
     }
 }
```

On voit par exemple ici dans le processUser, ce qui est effectué lorsque le client appelle la commande USER.

S'il n'est pas authentifié, on va aller chercher le username/mot de passe associé au username qu'il a entré.

```
 private void processPASS(){
     if (this.auth && this.pass.equals(this.data) && this.nbConnect <= Authentification.nbLimit){
         sendToClient(230," login ok");
     }
     else{
         sendToClient(530," bad password");
         processQUIT();
     }
 }
```

Et ensuite, lorsqu'il appelle la commande PASS on va regarder si le pass qu'il nous donne en paramètre matche avec le pass associé à l'user qu'il a précédemment entré.