INF3405 – Réseaux informatiques

Automne 2019

TP 1 : gestionnaire de fichiers

Groupe 3

2038408 – Clément Prime

1879536 – Jacob Dorais

Soumis à : Bilal Itani

21 Octobre 2019

Introduction

Le but du gestionnaire de fichiers est de faire deux applications qui communiqueront avec un lien client-serveur. Le serveur sera actif en tout temps à partir d'un ordinateur fixe. Le client pourra se connecter de n'importe où. Plusieurs clients devront pouvoir se connecter au serveur à la fois. Lorsque connecté au serveur, le client devra être capable de naviguer à l'intérieur du serveur pour pouvoir télécharger des fichiers, téléverser des fichiers et créer des dossiers. Avec une telle application et en utilisant notre propre serveur, nous n'aurons plus besoin d'utiliser les serveurs infonuagiques qui coûtent cher.

Présentation

Création du serveur et du client

À la création d'un serveur, un message apparaît à l'utilisateur lui demandant d'indiquer une adresse et un port pour que les clients pour se connecter. Un système de vérification se lance pour s'assurer du bon format de l'adresse et du port. Le même processus est effectué à la création d'un client. L'utilisateur doit alors bien indiquer l'adresse du serveur auquel il veut se connecter.

Multiples clients

Comme l'application serveur devait être capable de répondre à plusieurs clients à la fois, nous avons dû avoir recours au threads. L'application serveur principale va servir d'accepter les nouveaux clients, créer une autre thread pour gérer le nouveau client et continuer à regarder pour de nouveaux clients. La classe clientHandler dérive de la classe Threads alors lorsqu'il est créé, il est exécuté sur un nouveau thread et lui se concentre sur un seul des clients. Le clientHandler va attendre les commandes de son client et va pouvoir les exécuter.

Gestion des commandes

Les commandes sont gérées par le clientHandler. Celui-ci attend un message du client. Lorsqu'il reçoit quelque chose, il va afficher des informations concernant ce massage (adresse du client, date de réception), puis va vérifier qu'il s'agit bien d'une des commandes possibles. Si oui, il l'exécutera et sinon, il enverra un message d'erreur au client qui ira l'afficher à l'utilisateur.

Commandes

Les commandes disponibles par le client après la connexion sont les suivantes:

-ls
-cd [dossier]
-mkdir [nom]

-download [nom]
-upload [nom]
-exit

La commande ls permet d'afficher les fichiers et dossiers dans le répertoire actif. Le répertoire par défaut est le C: . La commande distingue les fichiers des dossiers en affichant [FILE] ou [FOLDER] suivant les cas.

La commande cd permet de naviguer à travers les dossiers en suivant la commande du nom du dossier désiré. La commande cd .. nous permet de naviguer vers le dossier parent. La commande cd … nous permet de retourner au répertoire par défaut: C: . Si le dossier recherché n'existe pas, le serveur enverra un message d'erreur au client qui ira l'afficher à l'utilisateur.

La commande mkdir nous permet de créer un nouveau dossier dans le répertoire actif en suivant la commande du nom du dossier souhaité. Si le dossier existe déjà, un message d'erreur sera envoyé.

La commande download nous permet de télécharger un fichier qui se trouve dans le répertoire actif du serveur vers le bureau du client. Pour se faire, le serveur envoie au client la longueur en nombre d'octets au client ensuite envoie tous les octets du fichier par paquets de 1000. En même temps, à la réception de ces paquets, le client affichera la progression du téléchargement. Si le fichier demandé n'existe pas, un message d'erreur sera envoyé.

La commande upload permet de téléverser un fichier du client vers le serveur. Le fichier téléversé par le client doit d'abord se trouver sur le bureau et sera téléversé dans le répertoire actif du serveur. La méthode d'envoi du fichier est équivalente à celle du download.

La commande exit permet de couper la connexion avec le serveur et de fermer le client. Le serveur continue de fonctionner normalement.

<u>Réception des données par le client</u>

À la suite de l'envoi d'une commande, le client s'attend à une réponse du serveur pour savoir si la commande à fonctionné ou non. Pour sortir de cette phase d'attente, le serveur doit envoyer au client un indicateur de fin de message.

Difficultés rencontrées

Durant l'élaboration de la commande download, nous avons fait face à un problème lorsque le fichier était plus gros que quelques octets, car les données ne peuvent pas toutes être transférées dans un seul paquet. Nous avons dû séparer les fichiers en

paquets de 1000 octets. Nous avons pris 1000 octets, car comme nous avons vu en classe, avoir de petits paquets réduit l'efficacité du transfert, car si plus de paquets sont nécessaires pour une même quantité de données, plus d'en-têtes seront nécessaires et donc il y aura plus d'information à transférer pour la même quantité de données. Maintenant que nous devons passer plusieurs paquets, nous envoyons d'abord la grosseur totale du fichier pour que le client sache quand arrêter de lire les données et il lit les données par paquets jusqu'à avoir transféré la totalité des données. Une fois terminé, le fichier est complet dans l'ordinateur du client.

Pour ce qui est de la commande upload, nous avons dû définir où chercher les fichiers. Cette commande est très similaire à download mais de manière inversée, le serveur devient le receveur et le client devient le donneur. Avec le download, le client peut naviguer pour aller chercher où est le fichier que l'on veut télécharger, mais nous ne pouvons pas parcourir les fichiers du côté client. Nous aurions pu imiter le système de navigation pour serveur, mais la gestion des deux répertoires aurait rendu l'interface encombrante. De plus cette fonctionnalité n'était pas dans nos requis. Nous avons donc restreint les activités du client dans son bureau. Nous avons fait ce choix, car il s'agit d'un répertoire facilement accessible et cela correspondait à download qui mettait les fichiers directement sur le bureau.

Critiques et amélioration

Pour améliorer le laboratoire, il serait bien de clarifier comment procéder pour ce qui est du répertoire des fichiers du client, car le client peut naviguer dans le système de fichier du serveur, mais lorsque vient le temps de téléverser un fichier, nous devons nous limiter aux fichiers qui sont sur le bureau, comme précisé dans les difficultés rencontrées.

Une autre bonne amélioration qu'on pourrait apporter au laboratoire serait d'intégrer une petite partie sécurité, rien de trop extravagant, juste de demander un mot de passe histoire que si on veut tester avec nos appareils, de ne pas les rendre complètement vulnérables à l'internet. Cela nous permettra aussi de nous sensibiliser avec la sécurité informatique en ce qui a trait aux réseaux.

Conclusion

Ce laboratoire a permis de mieux comprendre la communication entre client-serveur et nous a familiarisé avec les fonctions de bases permettant de mettre en place une communication que ce soit au niveau de la création des sockets ou de l'envoi de fichiers par exemple.