

TP1 : Gestionnaire de fichier

Par:

Johnny Khoury - 1980764

John Maliha – 1984959

Daniel Capelo - 1922984

Laboratoire de INF3405 – Réseaux informatiques

Hiver 2021

Soumis à

Monsieur Mehdi Kahdi

École Polytechnique de Montréal

Le 24 février 2021

**Introduction**

Lors de ce laboratoire, nous avons eu l’occasion d’explorer les concepts de base des réseaux informatiques. En effet, nous avons touché aux sockets, aux serveurs et aux clients. Le but de ce laboratoire était de réaliser un programme qui permet au client et au serveur d’échanger des informations en utilisant des sockets et d’utiliser des threads pour le développement d’applications.

**Présentation**

Pour arriver à atteindre ce but, il a fallu implémenter des fonctions. Tout d’abord, le langage utilisé est le Java. Nous avons commencé par créer une fonction qui vérifie si l’IP entrée par l’utilisateur est valide. Pour arriver à déterminer cela, la fonction IPVerifier() qui est une fonction de type booléenne va vérifier 3 conditions avant de retourner la valeur de sortie. Tout d’abord, la fonction va regarder si la valeur de l’entrée est nulle. Si c’est le cas, le programme va s’arrêter, car il n’est pas possible d’avoir une valeur nulle comme IP. De plus, afin d’avoir un IP valide, il faut qu’il y ait 4 parties toutes séparées par des points. Il n’est pas possible d’avoir plus que 4 parties ou moins. Si c’est le cas, la fonction retourne false et l’utilisateur doit entrer une nouvelle valeur. Finalement, la fonction va vérifier si les valeurs qu’elle contient sont contenues entre 0et 255. Si toutes ces conditions sont respectées, on peut affirmer que la valeur de l’IP entrée par l’utilisateur est conforme.

Passons maintenant à la valeur du port. Le programme doit également demander d’entrer une valeur de port à l’utilisateur. La fonction PortVerifier() prend en paramètre une valeur de port que l’utilisateur doit entrer. Cette fonction va vérifier si la valeur entrée est contenue entre 5000 et 5050 inclusivement, soient les conditions imposées par l’énoncé du laboratoire. Si c’est le cas, la valeur du port est valide, sinon, l’utilisateur doit entrer une nouvelle valeur de port pour l’adresse qu’il désire.

Ensuite, nous avons une autre fonction qui permet de lire les données qu’entre l’utilisateur. Celle-ci est nommée readFromConsole(). Elle sert à récupérer les valeurs entrées.

La fonction cd() permet de se déplacer dans les fichiers. Pour ce faire, nous devons changer le currentDirectory qui indique notre position actuelle. Ainsi, en faisant la commande ls, cela nous permet d’afficher les éléments à notre position où on se déplace. Nous vérifions ensuite si le nouveau chemin à emprunter existe. S’il n’existe pas, on affiche un message d’erreur. Dans le cas contraire, on se déplace à l’intérieur de celui-ci. Nous avons également codé la commande «..» qui permet de reculer dans le chemin emprunté.

Pour poursuivre, nous avons dû coder la commande ls. Cette commande permet de lister les éléments contenus dans un fichier. Pour ce faire, nous avons utilisé un tableau dans lequel nous avons passé les éléments contenus dans le fichier dans lequel nous nous trouvons. Puis, nous faisons afficher tous ces éléments jusqu’à ce que la fin du tableau soit atteinte.

De plus, nous avons créé une fonction createDir() qui permet de créer un fichier directement à partir de la console de sortie. Cette fonction retourne l’état du fichier, c’est-à-dire qu’elle nous indique si le dossier a bien été créée ou s’il y a eu une erreur de création. Si le fichier a bien été créé, on affiche un message de succès, sinon on affiche un message d’erreur.

Par ailleurs, nous avons créé une fonction qui permet d’afficher la date et l’heure de réception d’une commande. Nous avons utilisé une librairie déjà existante en java, soit «java.text.SimpleDateFormat» et «java.util.Date». Ces librairies vont nous permettre d’utiliser des objets qui nous retournerons la date et l’heure du système et de l’afficher dans le format que l’on veut. Cette fonction prend en paramètre l’adresse et la commande effectuée que nous affichons avec la date et l’heure.

Pour poursuivre, nous avons écrit une fonction qui permet d’envoyer des fichiers. La fonction sendFile(). Cette fonction se trouve dans client et elle permet créer un nouveau chemin pour envoyer un fichier tant que le fichier complet a été envoyé. La boucle while lit tous les bits du fichier afin de pouvoir les envoyer un à un.

Par la suite, après avoir été capable d’envoyer un fichier, il faut être en mesure d’en recevoir un aussi. La logique est très semblable de la fonction sendFile(), mais la différence est qu’ici, on lit tous les bits du fichier plutôt que de les envoyer dans la boucle while.

Pour les deux prochaines fonctions, nous les avons écrites dans le serveur. Il s’agit des mêmes fonction sendFile() et receiveFile(). Ainsi, on peut accéder aux fichiers désirés via le serveur donc nous pouvons faire l’échange de fichiers, tel qu’attendu.

Le but d’avoir créé des fonctions dans le client et dans le serveur dépendait des besoins que nous voulions répondre. Si les méthodes s’occupaient des besoins sur le système, nous les placions dans le fichier client. Par ailleurs, le fichier serveur sert à faire la connexion avec le client. Le programme va créer un serveur composé de sockets et, lorsqu’il reçoit une connexion du client, il va créer des threads qui permettra de communiquer avec le client. C’est ainsi que nous nous sommes assurés que l’échange de fichier était possible.

**Difficultés rencontrées**

Durant l’élaboration du code de ce laboratoire, nous avons rencontré quelques difficultés à certaines étapes du développement. Tout d’abord, en tant qu’équipe, il était difficile de se trouver un horaire commun pour pouvoir travailler sur le labo. Heureusement, nous avions des périodes de disponible sur l’horaire pendant le cours de laboratoire. De plus, nous avons fait un groupe sur discord pour pouvoir communiquer entre nous si jamais quelqu’un avait une question. Ainsi, même si nous nous étions séparés les parties, tout le monde pouvait mettre la main à la pâte. Effectuer un laboratoire à distance n’est pas chose facile, mais nous avons su nous débrouiller pour que celui-ci soit un succès.

Par ailleurs, nous avons également rencontré des difficultés lors de l’élaboration du code. En effet, nous avons eu des difficultés lorsqu’est venu le temps de coder la fonction IPVerifier(), car nous devions faire en sorte que toutes les possibilités d’erreurs s’y retrouve. De plus, puisque cette fonction comporte plusieurs condition if et certaines d’entre-elles sont imbriqués les unes dans les autres, il était parfois difficile de gérer tout cela pour que la fonction retourne la bonne valeur en fonction de l’entrée de l’utilisateur.

Enfin, nous avons aussi eu de la difficulté pour connecter le serveur et le client. La principale difficulté observée à cette étape est que nous n’avions pas d’idée comment faire la connexion, puisqu’il s’agissait d’un concept tout nouveau. Toutefois, avec de la recherche et en séparant le problème en sous-problèmes moins complexes, nous avons été capables de comprendre comment effectuer cette étape.

**Critiques et améliorations**

Ce laboratoire a été très intéressant, car il nous a permis de voir quelque chose de nouveau et de complètement différent de ce qu’on a l’habitude de faire. Toutefois, pour améliorer ce laboratoire, il serait intéressant de voir plus d’exemple sur la façon de coder ou d’avoir des exemples concrets qui nous permettrait de travailler de façon plus efficace. En effet, puisqu’il s’agit de quelque chose de tout nouveau, dans le sens où nous n’avons jamais coder un programme avec des clients et des serveurs, il aurait été utile d’avoir des exemples explicatifs pour mieux nous aider ou pour nous donner des pistes de réflexion.

La matière vue en cours est très intéressante, mais il y a une légère divergence avec ce laboratoire, car nous voyons simplement la matière de façon théorique. Nous n’avons pas eu d’exemple de code qui pouvait nous aider. À la fin de ce laboratoire, nous avons la base pratique et théorique, ce qui est très bien, mais selon-nous, présenter des exemples, soit dans le cours, soit dans l’énoncé nous aurait été utile.

**Conclusion**

Pour conclure, le but de ce laboratoire était de réaliser un programme qui était capable de demander à l’utilisateur d’entrer une adresse IP ainsi qu’un port et de faire la connexion entre le client et le serveur pour pouvoir échanger des fichiers. Il a fallu également faire en sorte que certaines commandes de base soient possibles à être effectuer, comme se déplacer d’un dossier à un autre, de créer des répertoires, de lister les éléments d’un dossier, etc. À la fin de ce laboratoire, nous pouvons affirmer avec certitude que nous avons atteint l’objectif du laboratoire, car tous les requis fonctionnent bien et notre programme fait ce que nous nous attendions à ce qu’il fasse. Ce laboratoire a été utile de plusieurs façons notamment, car nous avons eu la chance d’utiliser des threads, des sockets et de travailler avec des serveurs et des clients, chose que nous n’avions jamais fait auparavant. Ainsi, il nous a fait découvrir quelque chose de nouveau, mais d’essentiel. Il nous a permis d’ouvrir nos horizons et d’avoir une plus grande connaissance sur l’importance et la complexité du monde des réseaux informatiques. Enfin, ce laboratoire a comblé nos attentes, car il nous a permis d’avoir des exemples complémentaires à la matière théorique que nous voyons au cours. Il nous a permis d’avoir des exemples concrets de l’utilité des sockets et des threads en plus d’avoir touché aux serveurs et aux clients. Bref, ce laboratoire nous a grandement intéressé et nous a été très utile.