Anne-Elisabeth Lelièvre 16 084 130

Olivier Beauséjour 16 099 883

Devoir en Processus concurrents et parallélisme

Devoir #3

Rapport de travail

Dans le cadre du cours IFT630

Présenté à :

Gilles Brunet

Université de Sherbrooke

3 août 2018

**Table des matières**

[1. Démarche et recherches 2](#_Toc520450369)

[2. Description du mode de déploiement 3](#_Toc520450370)

[3. Références 4](#_Toc520450371)

# Démarche et recherches

Pour notre démarche, nous avons bien perfectionné nos connaissances sur l’implémentation d’un serveur en c++. En effet, nous avons déjà fait un transfert d’un fichier client/serveur dans notre cours de télématique. Par contre dans ce cours, nous avons utilisé du C# pour le faire et donc pour le faire en C++ il a fallu nous inspirer d’une implémentation sur internet (voir les références). Or, ce code n’était pas complétement fonctionnel et donc nous avons donc dû rajouter et modifier certaines parties du code (voir le dossier *Shared* avec les fichiers *Messages.cpp*/*Messages.h* et *Sockets.hpp*). Au départ, le code supportait seulement l’envoi d’une chaîne de caractère en un seul fil d’exécution. Nous avons donc fait l’implémentation du multiple fils pour le serveur. La fonction *update* du serveur sert à mettre à jour le serveur et l’état de chaque client connecté et *poll* retournera le message suivant à traiter. Grâce à la collection type *vector*, l’itération serait toujours dans le même ordre, donc notre serveur retournera les messages des premiers clients en priorité. Nous avons aussi rendu bloquant la fonction *accept* pour chaque client en désactivant la fonctionnalité de *SetNonBlocking* de la base de code originale. Ainsi, chaque client exécutera sont *recv* puis son *send* dans son propre thread lors de l’appel de la fonction du Client *poll*. Après avoir fait l’implémentation des multiples fils d’exécution, nous avons fait l’envoie d’un fichier au lieu d’une chaîne de caractère. Finalement, nous avons fait l’appel des commandes *fork* et *exec* pour démarrer tous les clients ainsi que le serveur. Ces appels se font dans le fichier *main.cpp* du Client. Pour la capture des signaux, nous avons créé des *signal handler.* Cela nous permettait de gérer tous les signaux globalement. Lors d’un signal, nous affichons le signal reçu puis nous effectuons une déconnection propre autant du côté client que du côté server.

# Description du mode de déploiement

Tout d’abord, nous avons séparé les parties du client et du serveur dans 2 dossiers différents. Ainsi, chaque client et serveur ont leur propre exécution dans leur *main.cpp*. La logique dans Client.cpp contient la réception, l’envoie et la connexion du client dans des classes différentes bien indiqués (*ConnectionHandler, ReceptionHandler, SendingHandler, ClientImpl* et *Client*). Pour la partie serveur, la fonction *start* déclenche l’initialisation du serveur en créant le socket principal acceptant les connexions clientes. On fait donc la lecture du fichier reçu et on traite les données. On enregistre donc le fichier reçu de chaque client dans le répertoire /ift630Devoir3/cmake-build-debug/ identifié par chaque id du client. Nous avons créé les variables *isFile* pour distinguer l’envoi d’un fichier de celui d’une chaîne de caractère et *morePackets* pour indiquer la fin du fichier au serveur et *fileId* identifier le fichier*.* Après que les clients aillent envoyé leur fichier, nous affichons un message indiquant la fin du transfert. Nous avons aussi modifié l’implémentation de la fonction *send* du client afin de supporter l’envoi d’un fichier et créer une nouvelle fonction *sendFile* qui envoie par paquet de 1024+6 le fichier à envoyer.

Pour exécuter notre programme, vous devez indiquer en argument le numéro de port du client et du serveur. De plus, pour l’envoi du fichier nous avons mis un fichier du type *tp3.zip* dans le répertoire courant /ift630devoir3 pour tester notre programme. Ainsi, si vous voulez tester notre programme vous n’avez qu’à déposer un fichier dans ce répertoire et changer le *path* de ce nouveau fichier dans le fichier *main.cpp* du Client à la ligne 49 tel que : **char**\* pathFile = **"../tp3.zip"**;.

# Références

<https://bousk.developpez.com/cours/reseau-c++/TCP/08-premier-serveur-mini-serveur/>

<https://bousk.developpez.com/cours/reseau-c++/TCP/05-envoi-reception-serveur/>

<https://fr.wikipedia.org/wiki/Signal_(informatique)>

<https://forgetcode.com/c/1201-file-transfer-using-tcp>

<https://stackoverflow.com/questions/1543466/how-do-i-change-a-tcp-socket-to-be-non-blocking>

<https://en.cppreference.com/w/cpp/utility/program/signal>

<http://www.csl.mtu.edu/cs4411.ck/www/NOTES/process/fork/create.html>

<ftp://www.cs.uregina.ca/pub/class/330/Fork/fork.html>

<https://stackoverflow.com/questions/1002513/non-blocking-version-of-system>