Travail pratique #2 - IFT-2245

Maude Sabourin et Jérémy Coulombe

2 avril 2018

Ce travail pratique a été rempli d'embûches d'incompréhension et de surprise. Ce rapport aura pour but de les énumérez et des les expliquer.

1 Problèmes rencontrées

Nous avons rencontrées plusieurs problèmes durant ce travail. Nous avons eu de la difficulté à faire en sorte que le client et le serveur communique entre eux. Par la suite nous avons dû changer l'ensemble de notre implémentation du serveur puisque nous avions un segfault et nous n'arrivions pas à trouver la raison pour laquelle nous l'obtenions. Durant cette ré-implémentation nous avons réduit les capacités de traimtements d'erreurs du serveur puisque nous considérions que le client ne devrait pas réalisé ce genre d'erreur s'il était bien implanté. Nous avons par la suite dû composé avec des problèmes qui n'apparaissait pas systématiquement ce qui rendaient l'identification de leur cause exacte difficile. Nous avons dû parfois procéder par élimination en appliquant soit des solutions trouvé en ligne relativement au problème rencontré, soit penser aux endroits d'où pourrait provenir le problème et essayer des solutions afin d'essayer de voir si le problème provenait vraiment de cet endroit. Bien que Valgrind et GDB était d'une aide précieuse, ils étaient d'une certaine utilité afin de découvrir l'endroit par exemple où nous avions un segfault et les valeurs que les fonctions ont reçus en paramètre mais dans ce travail pratique une bonne partie des problèmes devait être trouvé par raisonnment surtout ceux qui avait un lien avec la communication entre le client et le serveur. Nous avons eu droit à des omissions du compilateur. Par exemple, nous avions déclarer deux fois une mutex et nous n'avons eu aucun warning et aucune erreur. Nous avions aussi une connexion avec le serveur sur laquelle il n'y avait aucune information qui était envoyé ce qui causait un segfault. Il y a certaines erreurs dont nous n'avons jamais trouvé l'origine et ce n'est pas faute d'avoir essayer.

2 décisions que nous avons pris

Nous avons décidé décider que le client devrait établir une connexion avec le serveur à chaque fois qu'il voudra enoyer une commande et il fermerait cette connexion après avoir reçu une réponse du serveur. Nous avons dû prendre cet décsion puisqu'au départ nous avions développé le client et le serveur séparément et nous avions 2 vues complétement différentes de la dynamique des communcations entre le client et le serveur. Nous avions au départ penser utiliser la fonction getdelim, mais un de nos démonstrateurs nous a montré les problèmes que cette fonction pourrait causer. Nous avons dû pour ce travail pratique nous informer sur l'implémentation des sockets en c et sur les différentes fonctions relatifs au multithreading comme les mutex qui sont inclus dans le language C. Suite à ça, nous avons dû réimplanté la fonction que nous utilisions pour parser la commande reçu par le serveur puisque nous utilisions strtok, or cette fonction n'est pas réentrante donc elle n'est pas sécuritaire pour le multithreading. nous avons donc dû utiliser une de ses variantes qui était sécuritaire pour le multithreading. Nous avons découvert l'existence de cette fonction sur stackOverflow. Cet exemple reflète bien l'ensemble du tp puisqu'il nécessitait une grand effort de recherche afin de comprendre les méthodes nécessaires, afin de comprendre les messages d'erreurs et finalment comment éliminer cet erreur. Nous avions aussi commencé en utilisant des array dynamiques

mais lorsque nous avons décidé de tout réimplanté nous avons décidé que nous allions prendre l'option qui était d'ajouter le nombre de clients à la commnade beg au détriment de ne pas recevoir un point de bonus. Nous avons conservé un seul array dynamique dans lequel nous stockons chacun des mots de la commande reçu par le serveur ou de la réponse que le client reçoit du serveur.

3 Comment nous avons vécu ce travail

Ce travail a été une longue suite d'incompréhansion des erreurs puis de recherche des erreurs puis de corrections des erreurs et ça pratiquement sans arrêt. Il nous a permis d'en apprendre plus sur le multithreading et sur la dynamique client-serveur. Nous avons aussi porté une attention particulière à la gestion mémoire après avoir perdu des points lors du premier travail pratique sur cette aspect. Ce travail nous a fait autant maturer mentalement que physiquement.