Travail pratique #2 - IFT-2245

Jérémi Grenier-Berthiaumet et Olivier Lepage-Applin

15 mars 2019

Côté client : Sémaphore pour tenter d'éviter le busy waiting sur la variable 'count'. Fonction 'st_signal()' enlevée car le démonstrateur a dit qu'elle ne servait à rien. Fonctions 'xt_free()' ajoutées pour libérer les malloc des mains. Plusieurs mutex au lieu d'un seul pour que le traitement soit plus efficace (meilleur parallélisme). Tous les compteurs sont protégés par leur propre mutex, mais on n'avait pas de besoin d'un mutex pour ce qui est touché par un unique thread (i.e. le count++ dans ct_init) Certaines macros pour la lisibilité et rapidité de changement durant développement. Nouvelle méthode 'ct_init_server()' qui permet d'initialiser le serveur de manière plus efficace. Notre solution précédente était d'utiliser un mutex ainsi qu'un booléen dans 'ct_code()', mais on a alors autant de threads que de clients qui attendent que ce qu'un seul thread peut faire soit terminé. Détection de la dernière commande à l'intérieur du 'for' parce qu'on dirait bien qu'on n'était pas supposé écrire à l'extérieur des TODO. Évitement des malloc le plus possible (en passant en paramètre des pointeurs de variables déclarées localement).

Côté serveur : On a conservé la fonction qui ne reçoit que le header en premier, et appelons ensuite une fonction qui traite les arguments ensuite : cela nous permet de connaître la grosseur du buffer à remplir par la fonction qui lit le contenu du socket. Utilisation d'un array de fonction pour gérer de façon élégante les différents comportements du serveur.

Algo du banquier : (oli)