Projet Programmation avancée en C

Loya Dylan, Lacaze Yon

05 Décembre 2022

Table des matières

1 Organisation du code

1.1 Fichiers .c

— **master.c** : Ce fichier met en place les tubes permettant la communication avec le client et créé le premier worker en lui donnant le numéro 2.
— **master_client.c** : Ce fichier contient les fonctions permettant au client d'envoyer des requêtes au master et de lire les réponses du master vers le client.
— **client.c** : Ce fichier contient l'appel aux fonction définie dans le fichier **master_client.c** pour dialoguer avec le master.
— **master_worker.c** : Ce fichier contient toutes les fonctions utiles à la création et à la destruction des moyens de communications entre le master et le worker. il contient aussi les fonctions permettant le dialogue entre le master et le client.
— **worker.c** : Ce fichier reçois les ordre du master, lui réponds ou crée un autre worker si il le faut. Il se met ensuite en attente d'un autre ordre du master ou d'un précédent worker.

1.2 Fichiers .h

Les fichiers .h contiennent les déclarations des fonctions créés dans les fichiers .c portant le même nom.
Ils contiennent aussi les déclarations des constantes que l'on utilisera dans les fichiers .c qui inclus ces fichiers.

1.3 Fichiers .sh

Un script shell **run.sh** à été créé pour facilité la compilation du projet. Il fait appel à la commande **make clean** pour supprimer tout les fichiers **.d** et **.o**, puis à la commande **make all** pour recompiler et enfin au script **rmsempipe.sh** pour supprimer les tubes nommés créés par le programme si celui ci ne s'arrête pas de la bonne façon.

2 Protocoles de communication

2.1 Communication entre le master et le client :

Plusieurs clients peuvent être connectés en même temps au master, il faut donc s'assurer qu'un seul client puisse parler au master et que chacun des autres attende son tour. Pour mettre cette synchronisation en place nous avons créé un sémaphore grâce à la fonction **createSemClient**. Dans cette fonction on crée le sémaphore et on l'initialise avec la valeur 1, de cette façon il n'y a qu'un seul client qui peut effectuer l'opération -1 et entrer en section critique. Le client connecté fait ensuite un +1 quand il a fini et les autres clients peuvent se connecter.
Une fois en section critique les clients communiquent avec le master grâce à des tubes nommés créés par le master à son lancement. Il utilise pour cela la fonction **createFifos** qui crée un tube nommé pour l'écriture du master vers le client et un autre tube nommé pour la lecture du master vers le client.

2.2 Communication entre le master et les worker et des worker entre eux :

Le master ne peux discuter qu'avec le premier worker et il peut recevoir des messages de n'importe quel worker. Pour réaliser cela nous utilisons des tubes anonymes créés avec la fonction **pipe**. Le master crée deux tubes anonymes pour écrire et recevoir des worker puis se fork et lance le premier worker en lui passant les file descriptors permettant de recevoir du master et de lui écrire.
Quand un worker se fork pour créer un nouveau worker, il lui passe en argument un file descriptor pour recevoir des messages du précédent worker et un file descriptor pour envoyer des messages au master.