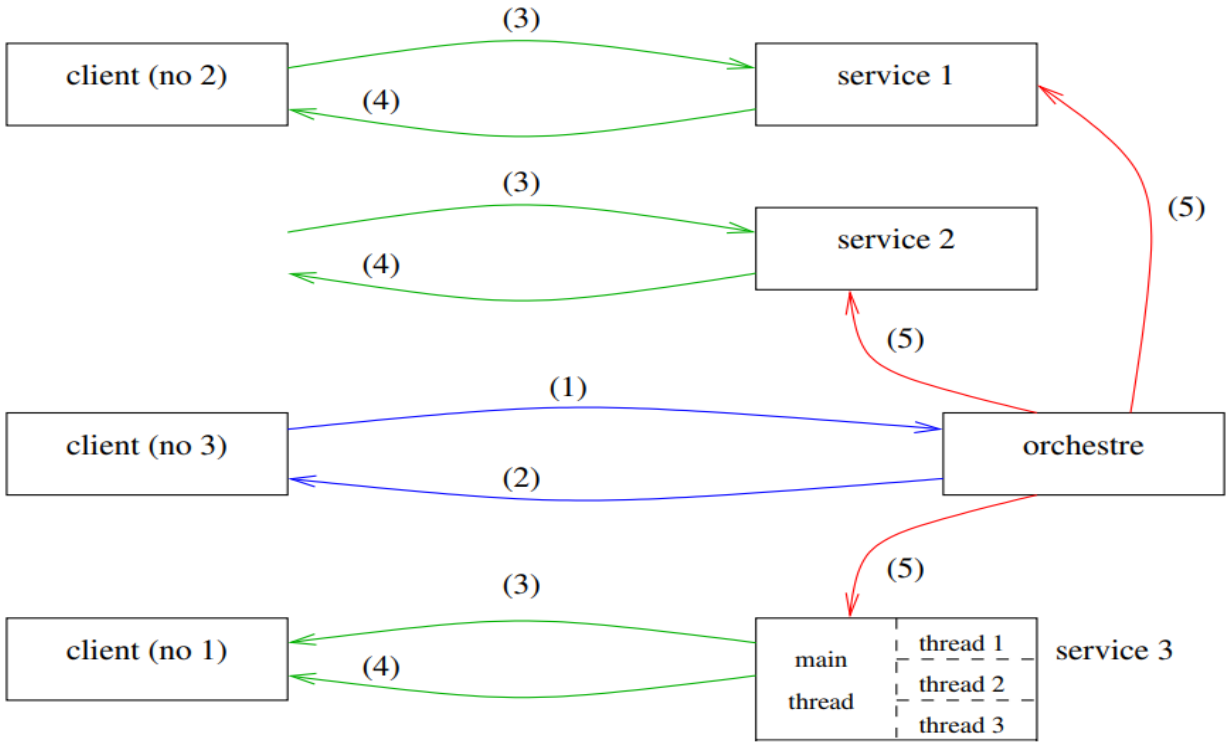
RAPPORT

CC2 : Projet



Esteban Mauricio & Adnane Laanani

Table des matières

I.Organisation du code

ORCHESTRE

orchestre.c : Processus principal dont le but est de démarrer les services et de répartir les différentes demandes des clients entre les services.

CLIENT

client.c : Processus principal dont le but est d'envoyer une demande qui sera traité par le service adéquat.

client_maximum.c/.h : Processus secondaire qui est un sous-programme de client, sont but est de gérer l'envoi des données pour le calcul du maximum, la réception du résultat retourné par le service et son affichage.

client_somme.c/.h : Processus secondaire qui est un sous-programme de client, sont but est de gérer l'envoi des données pour le calcul de la somme, la réception du résultat retourné par le service et son affichage.

client_compression.c/.h : Processus secondaire qui est un sous-programme de client, sont but est de gérer l'envoi des données pour la compression d'une chaîne, la réception du résultat retourné par le service et son affichage.

client_arret.c/.h : Processus secondaire qui est un sous-programme de client, contient du code pour une demande d'arrêt de l'orchestre.

SERVICE

service.c/.h : Processus (principal) lancé par l'orchestre avec un fork->execv, il contient le code principal d'un service et les différents appels aux fonctions des services spécifiques.

service_compression.c/.h : Contient le code spécifique à l'exécution de la compression par un service.

service_maximum.c/.h : Contient le code spécifique à l'exécution du calcul d'un maximum par un service.

service_somme.c/.h : Contient le code spécifique à l'exécution du calcul de la somme par un service.

service_somme.c/.h : Contient le code spécifique à l'exécution du calcul de la somme par un service.

CLIENT_ORCHESTRE

client_orchestre.c/.h : Module contenant les routines de communications entre l'orchestre et un client (tubes nommés et sémaphore).

ORCHESTRE_SERVICE

orchestre_service.c/.h : Module contenant les routines de communications entre l'orchestre et les services(tubes anonymes et sémaphores).

CLIENT_SERVICE

client_service.c/.h : Module contenant les routines de communication entre les clients et les services (tubes nommés).

UTILS

service.c/.h : Processus (principal) lancé par l'orchestre avec un fork->execv, il contient le code principal d'un service et les différents appels aux fonctions des services spécifiques.

service_compression.c/.h : Contient le code spécifique à l'exécution de la compression par un service.

service_maximum.c/.h : Contient le code spécifique à l'exécution du calcul d'un maximum par un service.

II.Les protocoles de communication

CLIENT_ORCHESTRE :

Entre le client et l'orchestre , on a utilisé une structure ComP, dont laquelle on va stocker le mot de passe créé par l'orchestre, envoyé au client et au service, et aussi les tubes nommée utilisée par le client pour communiquer avec le service.

Pour on a aussi défini des fonctions : pour la création des tubes client-orchestre, pour création de mutex entre l'orchestre et le client.

ORCHESTRE_SERVICE :

Entre l'orchestre et le service, on a utilisé une structure Order P, qui contient un booléen indiquant au service de soit traiter le client ou non, et aussi le mot de passe envoyé au service, pour que le service le vérifie avec celui de client.

CLIENT_SERVICE :

contient les fonctions de l'ouverture des tubes nommées, les fonctions d'envoi de réception de mot de passe entre le client et le service, les fonctions d'envoi et de réception d'un accusé de réception entre le client et le service.

Ce qui ne marche pas:

L'orchestre considère en permanence que les services sont occupés ce qui le pousse à refuser toutes les demandes des clients (à part client_arret qui est à part).

Dans le cas de client_arret, l'orchestre bloque après la boucle while au niveau de l'attente de la fin des traitements en cours (via les sémaphores) ce qui rejoint le 1er problème cité précédemment qui dit que les services sont déjà occupés.

Il y a eu beaucoup d'autres problèmes, mais qui ont été résolus juste avant la deadline du projet.

CONCLUSION

Nous aurons manqué légèrement de temps pour ce dernier problème, malgré un travail continue tout au long de ses 3 semaines de projet, ce qui pourrait être compensé par une coordination et une stratégie d'approche du sujet améliorée.

Méthode pour tester le projet:

Pour orchestre:

commencer par vérifier que les tubes nommés non pas été créer a la racine du projet (les supprimer si c'est le cas), puis exécuter les cmd ci-dessous dans l'ordre avant le (re)lancer le projet:

./rmsem.sh

./clean.sh

./compil.sh