Rapport

I. Organisation du code

Le code du projet est structuré en plusieurs répertoires, chacun ayant des fichiers spécifiques qui remplissent des fonctions distinctes. Voici un aperçu de l'organisation des fichiers, ainsi que leur but :

1. **Répertoire CLIENT**
   - **client.c** : Point d'entrée pour le client, gère l'interaction avec l'utilisateur et l'envoi des requêtes au serveur.
   - **client_arret.c** : Gère l'arrêt de l'orchestre.
   - **client_somme.c** : Gère la demande de somme de deux nombres.
   - **client_compression.c** : Gère la compression de chaînes de caractères.
   - **client_sigma.c** : Gère le calcul de la moyenne d'une série de nombres.
2. **Répertoire SERVICE**
   - **service.c** : Point d'entrée pour les services, gère les requêtes des clients et les redirige vers le service approprié.
   - **service_somme.c** : Implémente la logique pour le service de somme.
   - **service_compression.c** : Implémente la logique pour le service de compression.
   - **service_sigma.c** : Implémente la logique pour le service de calcul de la moyenne.
3. **Répertoire ORCHESTRE**
   - **orchestre.c** : Gère la coordination entre les clients et les services, en s'assurant que les requêtes sont traitées correctement.
   - **compil.sh** : Script de compilation pour l'orchestre.
4. **Répertoire CONFIG**
   - **config.c** : Gère le fichier de configuration, qui définit les services disponibles et leurs états.
   - **config.txt** : Fichier de configuration contenant les informations sur les services.
   - **test_config.c** : Fichier de test pour vérifier le bon fonctionnement de la gestion de configuration.
5. **Répertoire UTILS**
   - **myassert.c** : Fournit une fonction d'assertion personnalisée pour une gestion des erreurs simplifiée.
   - **io.c** : Contient des fonctions d'entrée/sortie pour la conversion de types.
6. **Répertoire CLIENT_ORCHESTRE**
   - **client_orchestre.c** : Gère la communication entre le client et l'orchestre.
7. **Répertoire ORCHESTRE_SERVICE**
   - **orchestre_service.c** : Gère la communication entre l'orchestre et les services, y compris la gestion des sémaphores.

II. Protocoles de communication

Les protocoles de communication dans ce projet reposent principalement sur des tubes nommés (FIFO) et des sémaphores. Voici un aperçu des protocoles utilisés :

1. **Communication Client-Orchestre** :
   - Les clients envoient des requêtes à l'orchestre via un tube nommé. Les requêtes incluent des informations sur le service demandé.

- L'orchestre répond aux clients avec un message d'acceptation ou d'erreur, indiquant si le service est disponible ou si une erreur s'est produite.

2. **Communication Orchestre-Service** :
   - L'orchestre envoie des messages au service via un tube anonyme, y compris des mots de passe et des codes de service.
   - Les services traitent les requêtes et renvoient les résultats à l'orchestre, qui les transmet ensuite au client.
3. **Sémaphores** :
   - Les sémaphores sont utilisés pour gérer l'accès concurrent aux ressources partagées, garantissant que deux clients ne peuvent pas interagir avec le même service en même temps.

III. Problèmes rencontrés

Bien que le projet soit en grande partie fonctionnel, plusieurs problèmes ont été identifiés :

1. **Gestion des erreurs** :

Certaines parties du code ne gèrent pas correctement les erreurs, notamment lors de l'ouverture des tubes. Des messages d'erreur sont affichés, mais le programme continue de s'exécuter, ce qui peut entraîner un comportement inattendu.

2. **Synchronisation** :

La synchronisation entre les clients et les services pourrait être améliorée. Actuellement, il existe des temporisations (comme **sleep(1)**) qui ne sont pas idéales pour garantir que les clients ont fini de traiter les requêtes avant que l'orchestre ne continue.

3. **Situation de Réalisation**

Au cours de la réalisation du projet, j'ai dû accomplir tout le travail seul, car un membre de l'équipe n'a pas pu participer pour des raisons de santé. Cela a posé plusieurs défis, mais j'ai fait de mon mieux pour terminer le projet dans les délais impartis.

4. **État Actuel du Projet**

Bien que j'aie essayé de terminer l'intégralité du code pour le projet, le système ne fonctionne pas encore correctement. Certaines fonctionnalités ne fonctionnent pas comme prévu et pourraient nécessiter des ajustements ou des corrections. J'ai réalisé plusieurs tests pour identifier les problèmes, mais il me faudra encore du temps et des efforts pour le finaliser.