Synthèse

1 Sujet

Le but de ce projet est de réaliser un lanceur de commande. Le projet sera constitué de trois parties :

1. Une bibliothèque proposant l'implémentation d'une file synchronisée.
2. Le programme « lanceur de commandes » qui utilisera une file synchronisée afin de récupérer les commandes qu'il doit lancer.
3. Un programme « client » qui utilisera la file synchronisée du lanceur pour y déposer des demandes de lancement de commandes.

2 File synchronisée

Une file synchronisée est un type de données abstrait proposant (au moins) deux opérations : `enfiler` et `defiler`. Ces deux opérations ont, en plus des sémantiques classiques sur les files, la propriété d'être bloquantes quand la file est pleine (pour `enfiler`) ou vide (pour `defiler`). De plus les opérations sur les files synchronisées doivent être atomiques : si plusieurs tâches utilisent en même temps une file synchronisée, le résultat obtenu devra être équivalent à une utilisation séquentielle de la file.

Pour ce projet vous êtes libre de choisir d'implémenter une file permettant d'enfiler des messages prédéfinis de tailles fixes ou d'implémenter une file générique permettant d'enfiler tout type d'objets. La notation tiendra compte de la difficulté liée à l'implémentation de la file ainsi que de la pertinence des algorithmes de synchronisation.

3 Contraintes

— La file synchronisée utilisera un segment de mémoire partagée afin de permettre aux différents processus de placer et de retirer des demandes d'exécutions.
— En plus du nom de la commande à exécuter et de ses éventuels paramètres, une demande d'exécution contiendra une information sur les noms des tubes nommés qui seront utilisés en tant que sortie standard et sortie d'erreur par la commande à exécuter. Un second tube nommé pourra également être utilisé en tant qu'entrée standard de la commande.
— Chaque demande d'exécution reçue par le lanceur devra être traitée par un thread dédié.
— Les commandes envoyées par les clients sont de la forme `cmd1 | cmd2 | ... | cmdN` avec $N \geq 1$.
— Le résultat de l'exécution de la commande ainsi que d'éventuels messages d'erreurs seront retournés au client sur deux flux distincts.

Le lanceur devra gérer correctement les zombis et les demandes de terminaisons via des signaux.
Les différents acteurs devront libérer proprement les ressources utilisées lorsqu'ils se termineront.
Vous pourrez apporter toutes les améliorations que vous estimerez pertinentes à l'application.

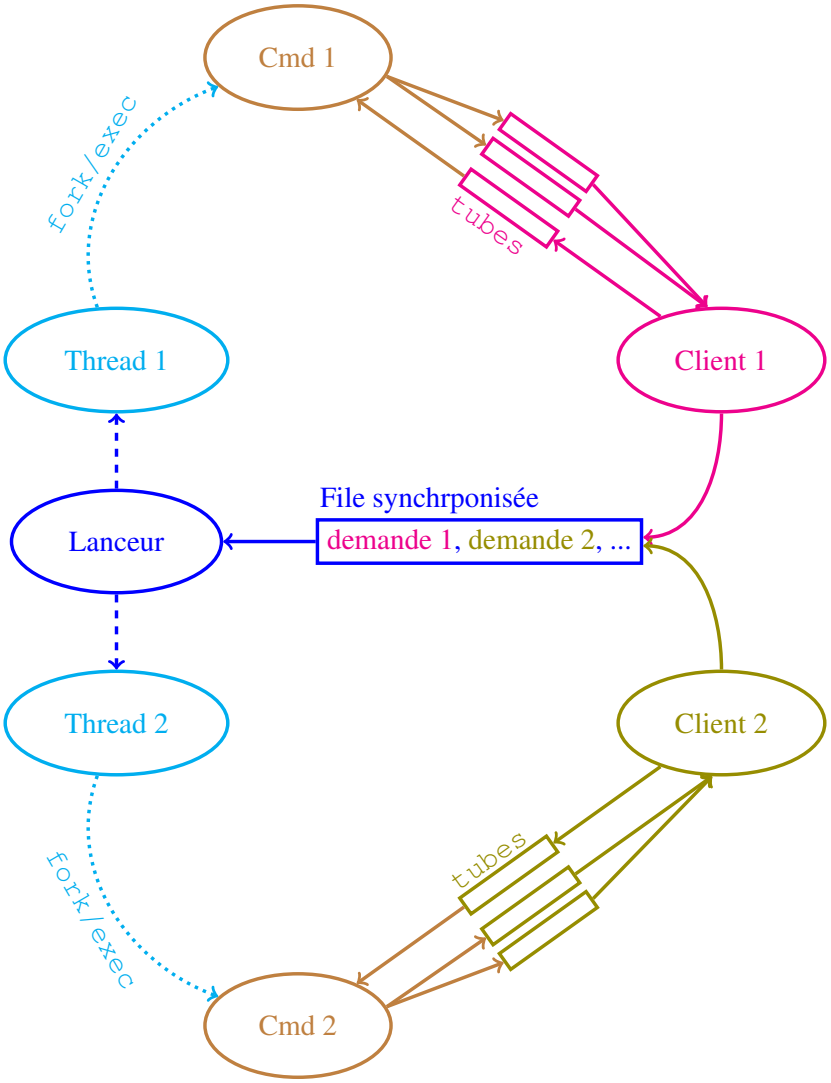

FIGURE 1 – Schéma de l'architecture client/démon

4 Travail à réaliser

En plus des codes sources correctement documentés et d'un makefile, vous rendrez :
— un petit manuel utilisateur explicitant comment utiliser votre application ;
— un manuel technique décrivant les solutions que vous avez été amenés à développer pour réaliser les différents modes de communication entre le lanceur et ses clients.
Le projet peut être réalisé par des groupes d'au plus deux étudiants.

La date de rendu est fixée au 31 décembre 2023.