Synthèse

1 Présentation

Le but de ce projet est de réaliser un lanceur de commande, sous la forme d'un démon qui répond aux requêtes émises par des clients. Le projet sera organisé de la manière suivante :

1. Un « démon » est à l'écoute des demandes de connexion des clients via un tube.
2. Un programme « client » qui utilisera le tube pour demander une connexion.
3. Des Thread qui prendront en charge les requêtes des clients via une zone de mémoire partagée (shm).
4. Des commandes développées lors des séances de TP : lsl, mypwd, ...

2 Figure

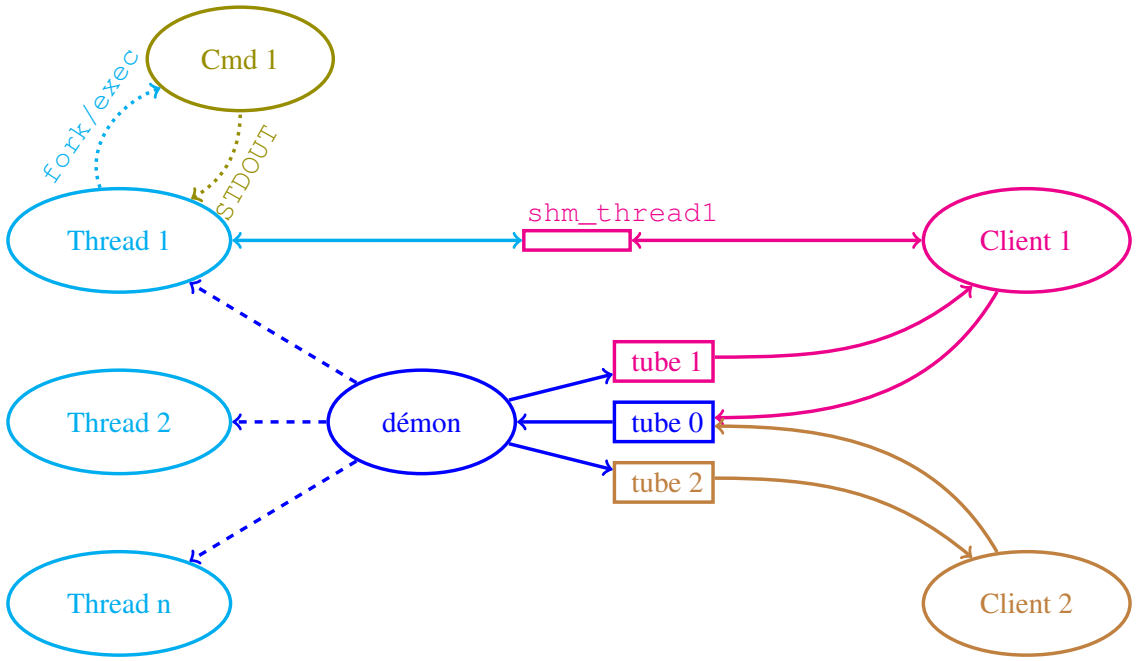

FIGURE 1 – Schéma de l'architecture client/démon

Attention : Pour ne pas surcharger la figure, l'ensemble des liens de client 2 ne sont pas tous représentés.

3 Séquence de traitement des requêtes

Lors de la phase d'initialisation, le démon récupère les paramètres enregistrés dans un fichier de configuration nommé `demon.conf`. Les paramètres enregistrés dans ce fichier sont les suivants :

MIN_THREAD = n	Nombre minimal de threads démarrés à un instant donné
MAX_THREAD = m	Nombre maximum de threads démarrés à un instant donné
MAX_CONNECT_PER_THREAD = p	Nombre de connexions gérées par chaque thread. (non pris en compte si p =0)
SHM_SIZE = q	Dimensions de la SHM associée à un thread (voir remarque)

Chaque paramètre est associé à une valeur de type entier positif ou nul.

Le démon initialise MIN_THREAD thread, qui seront en attente de connexion.
Un client qui souhaite se connecter au serveur envoie une demande « SYNC » dans le tube du démon.
Si le démon dispose d'au moins un thread disponible (thread initialisé et en attente), il retourne au client les informations nécessaire pour se connecter à la SHM associée à ce thread. Sinon, si le nombre MAX_THREAD n'est pas atteint, alors le démon initialise un nouveau thread pour gérer la connexion et transmet au client les informations nécessaire pour se connecter à la SHM associée à ce nouveau thread. Si aucun thread n'est disponible et si le nombre MAX_THREAD est atteint, alors le démon répond négativement par l'envoi du message « RST »

Remarque sur les valeurs des paramètres lus dans le fichier de configuration :

- Si MIN_THREAD > MAX_THREAD, alors le démon affiche un message d'erreur et se termine.
- Si MAX_CONNECT_PER_THREAD=0, alors le thread ne se termine jamais (i.e. ce paramètre ne sera pas pris en compte).
- SHM_SIZE représente la taille (en octets) allouée pour les échanges entre thread et client. Vous êtes libre dans l'organisation de son contenu.

4 Fonctionnement des threads

Chaque thread pourra prendre en charge un client à un instant donné. La communication entre client et thread s'effectue via une zone mémoire partagée (SHM) dont la taille est définie par le paramètre SHM_SIZE :

- Le client dépose sa requête dans la SHM,
- Le thread effectue un fork/exec pour exécuter la commande demandée par le client et récupère les résultats,
- Le thread dépose ensuite ces résultats dans la SHM,
- Le client récupère les résultats de la commande demandée dans la SHM.
- Lorsque le client a terminé sa requête, il peut soit terminer sa connexion par l'envoi de la commande « END », soit déposer une nouvelle requête.
- Lorsque le client met fin à la connexion, le thread décrémente alors son nombre de connexions à gérer.
- Lorsque le nombre de connexions à gérer atteint 0 et que MAX_CONNECT_PER_THREAD est différent de 0, alors le thread se termine proprement et est supprimé de la liste des threads initialisés ; sinon il reste en attente d'une nouvelle connexion.

5 Pool de thread

La gestion des threads est la suivante :

- Le démon initialise MIN_THREAD threads au démarrage du système, et leur transmet la valeur du paramètre MAX_CONNECT_PER_THREAD.
- A tout instant, le démon dispose des informations sur l'état des threads, notamment le nombre de threads initialisés qui doit toujours être compris entre MIN_THREAD et MAX_THREAD, ainsi

que la liste des threads en attente. Il doit, lorsque le nombre de threads initialisés descend en dessous de MIN_THREAD, créer autant de threads que nécessaire pour atteindre MIN_THREAD.
— Pour chaque demande de connexion, le démon associe au client le premier thread disponible qui devra gérer les requêtes selon le principe décrit précédemment.
— Le démon peut créer des threads supplémentaires jusqu'à atteindre MAX_THREAD pour prendre en charge les nouvelles connexions, si aucun des threads initialisés n'est disponible.
— Chaque fois qu'un client met fin à sa connexion, le thread qui lui était associé décrémente son paramètre p. Lorsque la valeur de p atteint 0, alors le thread se termine proprement et est supprimé de la liste des threads initialisés.
Attention : Dans le cas où le paramètre MAX_CONNECT_PER_THREAD est initialisé à 0, le thread ne se termine jamais.

6 Compléments

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 votre application.

7 Travail à réaliser

En plus des codes sources correctement documentés et d'un makefile pour générer l'ensemble des applications, vous rendrez :
— Un 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 seul ou par des groupes d'au plus deux étudiants.

La date de rendu est fixée au plus tard pour le 6 janvier 2020. Une soutenance d'environ 10mn sera organisée par chaque enseignant, pendant laquelle vous expliquerez le contenu de votre code, et procéderez éventuellement à une démonstration rapide de votre application.