Système distribué et système d'exploitation

Projet Barman

ROBIN Florian - DI Giorgio Maxime

Les fichiers contenues dans le projet :
pour le langage C

client.c, serveur.c (processus_de_communication), tcp.c, tcp.h, main.c, barman.c, controle.c (processus_de_contrôle), fifo.c, fifo.h, tireuses.c, tireuses.h

pour le java :
Fournisseur.java, Commande.java, Biere.java, IBiere.java
pour compiler tous les fichiers il suffit d'utiliser la commande "make all"
ensuite il faut exécuter les fichiers dans l'ordre suivant en respectant les arguments passés en paramètres.

Tout d'abord il faut changer l'adresse ip qui est contenue dans le fichier Fournisseur.java afin de la remplacer par l'adresse de la machine actuelle. Il faut ensuite mettre cette adresse dans le fichier Commande.java

1- ouvrir un terminal et exécuter "rmiregistry &" (si le code a besoin d'être exécuté une nouvelle fois il faut fermer le terminal et le réouvrir

2- ouvrir un terminal et exécuter cette commande "java Fournisseur"

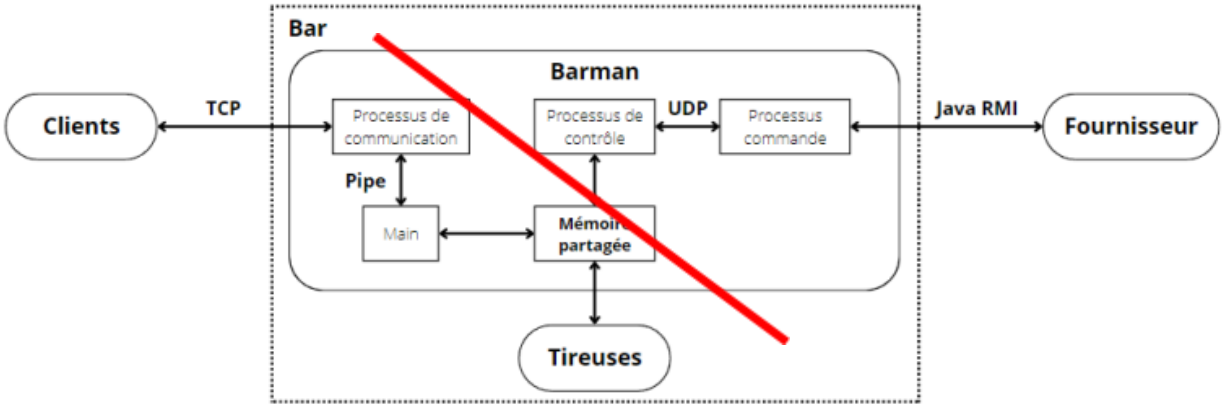
3- ouvrir un terminal et exécuter java Commande port_UDP (exemple : 3652)

4- ouvrir un terminal et exécuter la commande :

"./barman.out port_TCP(7895) ip_machine_barman port_UDP(3652)

5- ouvrir un terminal et exécuter la commande "./client.out ip_machine_barman port_TCP(7895)

Répartition du travail au sein du groupe :



Florian robin : toute la partie client, processus de communication, tcp, main, pipe, tireuses. La mémoire partagée a été faite en commun avec mon binôme.

Maxime Di Giorgio : processus de contrôle, processus de commande, connexion udp, java rmi, fournisseur.

ensuite les commentaires, gestion d'erreurs et de mémoire ainsi que certains détails ont été effectué par le binôme.

PARTIE SYSTÈME DISTRIBUÉ :

Pour cette partie on peut notamment relever la connexion tcp entre le client.c et le serveur.c, le choix a été fait de créer une fonction dans le fichier tcp.c en incluant tcp.h afin d'alléger le code du serveur.c, de plus la connexion est multi-client grâce aux threads présents dans le serveur.c. controle.c permet la connexion udp avec le fichier Commande.java. De plus, Commande.java est relié au Fournisseur.java grâce au java rmi. Il y a également l'interopérabilité entre le fichier controle.c et commande.java, les deux fichiers communiquent via la socket udp avec des langages différents. le fichier commande.java agit comme un serveur et le fichier controle.c agit comme un client. Il a été décidé de créer et d'initialiser la mémoire partagée dans le fichier tireuses.c lié au fichier tireuses.h. les fichiers main.c et controle.c ont accès à la mémoire partagée en utilisant les fonctions présentes dans le fichier tireuses.h.

Cependant la partie sur l'ordonnanceur préemptif qui permet de donner tour à tour la main aux 3 processus grâce aux threads et aux signaux en théorie pour simuler un pseudo-parallélisme n'a pas pu être réalisée par faute de manque de compréhension mais le fichier barman.c contient permet de lancer main.c, serveur.c et controle.c grâce à des processus, ils sont lancés à tour de rôle mais reste en exécution en arrière plan. La mémoire partagée a été implantée pour mettre à jour une variable entre les 3 processus mais cela n'a pas donné le résultat attendu car l'implémentation avec l'ordonnanceur n'a donné lieu qu'à des erreurs.

PARTIE SYSTÈME D'EXPLOITATION :

Pour cette partie, les sémaphores sont présentes et initialisés dans le fichier tireuses.c, les fonctions présentes dans ce fichier sont utilisées grâce aux tireuses.h dans le fichier main.c et le fichier controle.c. ensuite le fichier main.c possède un ordonnanceur fifo codé dans fifo.c et lié au fifo.h afin de déterminer quel client sera servi en premier. l'ordonnanceur préemptif n'a cependant pas été réalisé comme expliqué dans la partie précédente. Dans le fichier controle.c un processus est mis en place afin de vérifier toutes les 2 secondes l'état des fûts via la mémoire partagée. Les pipes sont créées dans le fichier main.c et ouvertes dans dans le même fichier, elles sont également ouvertes dans le fichier serveur.c, ce sont des tubes de communication nommés, un sert à l'écriture et l'autre à la lecture. le répertoire pour créer les pipes doit se situer dans le répertoire /tmp/ afin de contourner les permissions de la fac qui empêchent la création de pipes. Les thread sont également présents dans le fichier serveur.c. Les processus (fork) sont utilisés dans barman.c. et controle.c. pour finir les signaux sont utilisés dans plusieurs fichiers également : client.c, serveur.out, main.c et barman.c.

Brèves explication des fichiers avec les fonctions :

client.c : affichage du menu, création de socket, communication avec le serveur, gestion de signal d'interruption pour les éventuelles erreurs, le switch permet de traiter les différentes demandes du client, handle_beer_order(); permet de gérer correctement la communication

avec le serveur. la fonction read_integer(); agit comme une gestion d'erreur, elle force le client à entrer un entier entre 1 et 4 pour choisir une option du menu.

tcp.c et tcp.h : SocketServeurTCP(); permet de créer la socket et de la lier pour le serveur.c

serveur.c : sig_atomic_t est un type de données primitif qui permet le bon fonctionnement des opérations critiques, cela empêche l'interruption par un signal. Ouverture des pipes, lecture ecriture sur pipes et socket, mise en place de thread, gestion de signal, gestion demande de client, handle_beer_order(); permet de traiter la commande de biere dans le cas 1, pour le cas 2 le code n'a pas été mis dans une fonction, cela aurait été préférable mais faute de temps le code est laissé tel quel. Le cas 2 permet de demander quelle bière est présente dans le fût.

main.c : la fonction prepare_drink(); permet de préparer une bière en agissant sur la mémoire partagée, la mémoire partagée, les sémamphore et les tireuses sont initialisés également, création des pipes, ouverture des pipes, appel de l'ordonnanceur fifo. la fonction server_process(); permet de traiter les messages du serveur et de lui renvoyer la boisson en appelant la fonction prepare_drink(); qui associe les messages du serveur à la quantité et au type de bière blonde ou ambrée.

fifo.c et fifo.h : ordonnanceur fifo pour les requêtes.

tireuses.c et tireuses.h : création mémoire partagée, création fûts, création sémpahore, fonction pour retourner l'état des fûts, fonction serve_beer(); pour décrémenter la quantité de bière servis.

barman.c : lance en arrière plan via des fork() main.out, controle.out et serveur.out, variable partagée entre les 3 processus pour savoir quel processus est en cours.

controle.c : udapte_keg(); permet de mettre à jour le nouveau fût qui a été commandé en remplaçant le nom et la quantité. handle_received_message();permet de récupérer les informations du fût envoyer par la socket udp. Création de socket udp, check_tap_status fonction qui vérifie toute les 2 s l'état du fût.

Commande.java : création socket udp, connexion java rmi, empêcher barman d'écrire autre chose que ce qui est demandé. affichage menu de bières blondes si le fût de bière blonde est vide, même procédé pour la bière ambrée.

Fournisseur.java, IBiere.java et Biere.java sont vos fichiers.