UNIVERSITÉ DE SHERBROOKE

DÉPARTEMENT D'INFORMATIQUE

Devoir # 1 (8%)

IFT 630

Processus concurrents et parallélisme

Devoir à remettre au plus tard le **20 juin 2020**.

La correction de ce travail sera basée sur le bon fonctionnement et la clarté de vos programmes. Vous devez me soumettre votre code source à l'aide de Turnin (dans le projet «tp1») avant la date de remise. Un retard de moins de 48 heures sera supporté mais sera sujet à une pénalité de 10% par jour de retard.

Introduction

Le travail pratique contient 2 parties. La première est sur la résolution de problèmes classiques grâce aux sémaphores et aux moniteurs. La seconde porte sur l'implémentation de canaux de communication et sur la résolution d'un des problèmes classiques grâce à cette implémentation. Vos solutions peuvent être faites en SR, en JR ou en C++. Le travail doit être fait en équipe de 3 à 5 personnes.

(85%) Partie 1 - Problèmes classiques

Travail à faire

Vous devrez fournir plusieurs solutions aux problèmes décrits dans les pages suivantes en utilisant des primitives de synchronisation différentes. Voici les solutions demandées pour chaque primitives :

a) (65%) Les problèmes 1, 2 et 3 doivent être résolus en utilisant les sémaphores.
   i) Note : Si vous choisissez d'utiliser C++, utilisez «**`std::counting_semaphore`**» et les opérations «**`release`**» et «**`acquire`**» qui correspondent à «**`V()`**» et «**`P()`**».

b) (20%) Les problèmes 1, 2 doivent être résolus en utilisant les moniteurs (option -sw).
   i) Note : Si vous choisissez d'utiliser C++, vous devez fournir votre propre implémentation des moniteurs.

Description des problèmes

#1 Le problème des philosophes.

Soit une table circulaire sur laquelle se trouve 5 baguettes réparties équitablement. 5 philosophes se trouvent assis autour de la table, chacun ayant accès à deux baguettes. Un philosophe songe ou mange à l'aide de ses deux baguettes (les plus rapprochées). Pour manger, un philosophe à besoin des deux baguettes en questions. Si une des baguettes n'est pas disponible, il attend. Par conséquent, le philosophe possède trois états possibles : «songe», «veux manger» et «mange».

Points importants à considérer :

- Si un philosophe veux manger mais n'arrive pas à prendre ses deux baguettes, celui-ci se met en attente des baguettes mais continue d'avoir faim. Il ne peut pas retourner songer par peur de mourir de faim.
- L'algorithme de distribution des baguettes doit être équitable. Si un philosophe attend l'accès aux baguettes pendant un temps indéterminé, alors il y a littéralement un risque de famine (voir la définition de famine dans les algorithmes d'exclusion mutuelle).

- Manger prend un temps déterminé et fini.

#2 Le problème du parc d'attraction.

Soit un parc d'attraction possédant des montagnes russes sur lesquelles circule une voiture pouvant contenir au plus C passagers. Plusieurs personnes (N tel que N > C) attendent de façon répétitive pour faire un tour de voiture. La voiture ne part que si elle est pleine et elle fait 5 tours de piste. Il est important que, dans vos solutions, les personnes et la voiture soient implantées comme des processus.

Point important à considérer :

- Il faut que tous les passagers soient débarqués avant de débuter un nouvel embarquement. Vous devez donc prévoir un mécanisme de synchronisation qui signale aux clients en attente que la voiture est vide et qu'ils peuvent débuter l'embarquement.

#3 Le problème de la boutique du barbier.

Dans une petite ville, on retrouve une boutique de barbier ayant deux portes et quelques chaises. Les clients entrent par une porte et quittent par l'autre porte. À cause de la dimension restreinte de la boutique, un seul client à la fois ou uniquement le barbier peut se déplacer dans la boutique.

Le barbier passe sa vie à servir des clients, un à la fois. Quand il n'y a aucun client dans la boutique, le barbier s'endort. Quand un client arrive et trouve le barbier endormi, il réveille le barbier et s'assoit sur la chaise du barbier pendant que celui-ci lui coupe les cheveux. Si le barbier est occupé, le client s'assoit sur une chaise pour attendre son tour. S'il n'y a pas de chaise, le client s'en va.

Après avoir terminé la coupe de cheveux, le barbier ouvre la porte de sortie pour le client et la ferme lorsqu'il est sorti. S'il y a des clients en attente, le barbier appelle le suivant et attend qu'il soit assis pour le servir. Sinon, le barbier retourne dormir.

Dans votre solution, vous devez utiliser un processus distinct pour chaque client et un pour le barbier. Un client particulier doit se synchroniser avec le barbier pour toutes les étapes de la coupe de cheveux jusqu'à ce qu'il quitte après que la coupe soit terminée.

Vous devez avoir plus de processus clients que de chaises disponibles.

Points importants à considérer :

- Le barbier (processus) doit dormir (salle vide) et doit être réveillé lorsque cela est requis (arrivée d'un nouveau client). Dormir ici signifie mettre le processus en attente (sur un sémaphore, variable condition ou réception de message).
- Le client attend sur une chaise si le barbier est occupé. Il ne se fait pas couper les cheveux sur cette chaise. Lorsque le barbier devient disponible, il doit se déplacer vers la chaise du barbier.
- Le barbier attend que le client soit assis. Cette situation se produit lorsque client se déplace d'une chaise dans la salle d'attente vers la chaise du barbier. De même, lorsque le barbier se fait réveiller, le client n'est pas nécessairement assis. Il faut donc prévoir un mécanisme avec lequel le barbier attend que le client soit assis.
- Le barbier doit attendre que le client soit sorti pour appeler le prochain client. Il faut donc prévoir un mécanisme qui synchronise la sortie du client avec le barbier.

(15%) Partie 2 - Communication par messages

Travail à faire

a) (10%) Implémenter un canal de communication permettant l'envoi et la réception de messages par désignation directe. Votre implémentation doit respecter les contraintes suivantes :
    - La création du canal doit retourner deux objets :
        1) Un pour l'envoi de messages
        2) Un pour la réception de messages
    - Le canal possède un tampon dont la taille est spécifiée en paramètre au constructeur du canal.
    - Il doit être possible de créer plusieurs canaux disjoints.
    - Supporter les échanges entre plusieurs fils d'exécution.
        1) Ces échanges seront synchronisés à l'aide de primitives de synchronisation telles que les sémaphores, les moniteurs ou les variables conditions.
    - Lors d'un appel de réception sur un canal vide, l'appelant se met en attente jusqu'à ce qu'un message soit reçu.
    - Lors de l'envoi sur un canal plein, l'appelant se met en attente qu'un élément soit retiré du tampon.
b) (5%) Produire une solution au problème du parc d'attraction (problème #2 de la question #1) en utilisant votre implémentation des canaux de communication.