TP8

December 2018

1 Objectifs

Durant ce TP, vous allez écrire deux petits programmes: cuisinier.c et serveur.c. Ce TP est une implémentation de mémoire partagée et de sémaphore.

2 Description

Le cuisinier a besoin d'un temps aléatoire pour cuire une pizza. Quand il n'y a pas de pizza, le serveur attendra. Lorsque les pizzas sont disponibles, le serveur commence à servir. Lorsqu'il y a trois pizzas sur l'étagère, le cuisinier se repose. Le cuisinier et le serveur doivent savoir combien de pizza sont disponibles sur les étagères. Le cuisinier s'arrêtera après dix pizzas. Le serveur s'arrêtera quand il aura servi toutes les pizzas.

2.1 Mémoire partagée

Dans notre cas, les étagères sont la mémoire partagée entre les processus. Le segment de mémoire partagée sera créé par le cuisinier. Il définira la taille du segment et le mappera sur cette adresse de processus. Du côté du serveur, il doit ouvrir la mémoire partagée et mapper sur son adresse de processus. Lorsque le travail est terminé, le cuisinier et le serveur dissocient la mémoire partagée. Avant que le serveur se ferme, il supprime le segment de mémoire partagée.

La mémoire partagée ne fournit pas de mécanisme de synchronisation, c'est-à-dire qu'il n'existe pas de mécanisme automatique pour empêcher le second processus de commencer à lire avant que le premier processus ne termine l'écriture sur la mémoire partagée. Nous allons utiliser le sémaphore pour synchroniser l'accès à la mémoire partagée.

2.2 Sémaphore

Le sémaphore est une variable utilisée pour résoudre des problèmes critiques et pour réaliser la synchronisation de processus dans un environnement partagé. Les sémaphores de comptage sont équipés de deux opérations, notées historiquement P et V. L'opération V incrémente le sémaphore S et l'opération P le décrémente.

La valeur du sémaphore S correspond au nombre d'unités de la ressource actuellement disponibles. L'opération P attend ou dort jusqu'à ce qu'une ressource protégée par le sémaphore devienne disponible, heure à laquelle la ressource est immédiatement réclamée. L'opération V rend une

ressource disponible à nouveau une fois que le processus a fini de l'utiliser. Une propriété importante du sémaphore S est que sa valeur ne peut être changée qu'en utilisant les opérations V et P.

Un moyen simple de comprendre les opérations d'attente (P) et de signal (V) est le suivant:

wait: Si la valeur de la variable de sémaphore n'est pas négative, décrémentez-la de 1. Si la variable de sémaphore est maintenant négative, le processus en cours d'attente est bloqué (c'est-à-dire ajouté à la file d'attente du sémaphore) jusqu'à ce que la valeur soit supérieure ou égale à 1. Sinon, le processus continue son exécution après avoir utilisé une unité de la ressource.
signal: incrémente la valeur de la variable sémaphore de 1. Après l'incrément, si la valeur de pré-incrémentation était négative (ce qui signifie qu'il y a des processus en attente d'une ressource), un processus bloqué est transféré de la file d'attente du sémaphore vers la file d'attente prête.

2.3 Consignes et conseils pratiques

- Vous pouvez créer trois sémaphores pour gérer les deux processus. Un pour informer le serveur de commencer à servir. Un pour informer le cuisinier de se reposer. Un pour l'exclusion mutuelle d'étagère. Tous ces trois sémaphores seront créés dans cook.c.
- Avant de quitter, vous devez fermer le sémaphore et le dissocier. Lorsque tous les processus ont fini d'utiliser le sémaphore, vous pouvez le supprimer du système à l'aide de `sem_unlink`.