Série d'exercices 5 : Synchronisation

5.1 Vous voulez réaliser un certain traitement d'image en 3 étapes sous forme de pipeline. La première étape est le chargement de l'image, ensuite l'application d'un effet, et la dernière étape est la sauvegarde. Chaque item est passé au suivant par un tampon circulaire. Comment réaliser cette architecture avec des sémaphores? Décrivez quels sont les rôles de chaque fils d'exécution (producteur et consommateur).

On doit démarrer 3 fils d'exécution, un pour chaque étape. Il faut 2 tampons et 2 sémaphore par tampon, un pour attendre si le tampon est plein et l'autre pour attendre si le tampon est vide. Il faut également un signal pour indiquer qu'il faut quitter. Pour cela, on peut utiliser une valeur spéciale sentinelle, comme null tout simplement. Le premier fil d'exécution n'est qu'un producteur. Le second fil d'exécution est un consommateur et un producteur, tandis que le dernier fil n'est qu'un consommateur.

5.2 Dans quel cas on voudrait utiliser un verrou asymétrique read/write plutôt qu'un verrou à exclusion mutuelle?

Quand la structure de donnée à protégér est accédée souvent en lecture seule, mais rarement modifiée, alors un verrou asymétrique est préférable à une exclusion mutuelle, car plusieurs lecteurs peuvent accéder à la section critique sans problème.

5.3 Plusieurs processus veulent ajouter de l'information à la fin d'un fichier avec mmap() simultanément. Décrivez comment vous pourriez faire pour éviter qu'un processus écrase l'information d'un autre processus en train d'écrire, tout en étant le plus efficace possible?

5.4 Linux supporte uniquement le verrouillage de fichier coopératif (contrairement à un verrouillage obligatoire), c'est-à-dire que si on verrouille un fichier avec flock(), il faut que l'autre processus utilise flock() lui aussi. En d'autres termes, on ne peut pas s'assurer d'avoir l'accès exclusif à un fichier en forçant un autre fichier à attendre. Quels sont les avantages et les inconvénients de cette situation?

Il s'agit d'une vue optimiste à la concurrence, dans laquelle la plupart du temps, ce sera correct. Un avantage est que l'accès est toujours possible, par exemple pour des programmes de sauvegarde. Avec le verrouillage obligatoire, on pourrait oublier de fermer un fichier en cours d'édition, ce qui bloquerait la sauvegarde. L'inconvénient est que si un fichier est en cours de modification, il se peut qu'il ne soit pas cohérent. C'est la raison pour laquelle on ne peut pas tout simplement sauvegarder directement les fichiers d'une base de données pendant qu'elle fonctionne.

5.5 Réalisez deux scripts pour expérimenter avec les verrous sur un fichier. Le premier script modifie un fichier périodiquement, tandis que l'autre doit le sauvegarder périodiquement. Utiliser la commande flock pour assurer l'exclusion mutuelle entre les deux script.

On utilise flock sur un fichier spécifique, dont le chemin est connu des deux scripts. La commande de modification du fichier utilise flock avec ce fichier pour lancer le programme qui modifier le fichier principal. Le script de sauvegarde doit utiliser également flock pour exécuter la copie du fichier. Le système d'exploitation empêchera les deux opérations de se faire en même temps.

5.6 Faites les exercices du site https://deadlockempire.github.io/