Université de Strasbourg
Licence d'informatique – Semestre 5

Architecture des Systèmes d'Exploitation
2021/2022

TP 4 (noté)

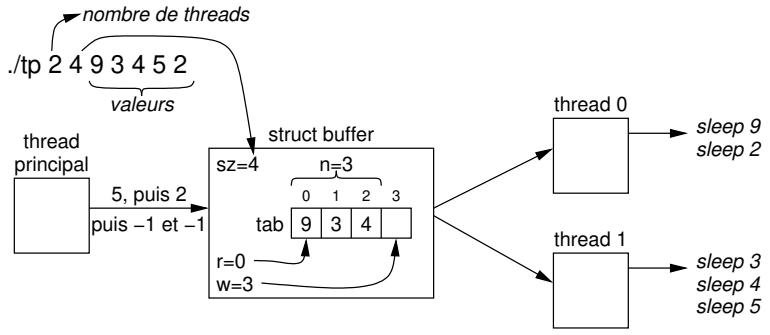
Épreuve du 17 novembre 2021
Durée : 1h30
Documents autorisés : supports disponibles sur Moodle, vos anciens TD/TP, pages de manuel

Ce TP a pour but de compléter un programme fourni sur Moodle avec des mécanismes de synchronisation basés sur les barrières et les conditions POSIX.

Le programme prend les arguments suivants :
- le nombre de threads à créer ;
- la taille d'un buffer circulaire, en nombre de cases ;
- une suite (éventuellement vide) de valeurs positives ou nulles.

Le thread principal crée le buffer et les threads, puis transmet les valeurs aux threads fils par l'intermédiaire du buffer. Les différents threads extraient à chaque itération une valeur du buffer et réalisent une attente (avec `sleep`) du nombre de secondes trouvé dans le buffer. Lorsque le thread principal a transmis toutes les valeurs, il écrit autant de valeurs « -1 » dans le buffer qu'il y a de threads pour les prévenir de s'arrêter. Une fois que tous les threads ont fini de traiter les valeurs reçues, chacun doit afficher le caractère « `*` » et se terminer.

Le schéma ci-après illustre le fonctionnement (les notations dans la structure `buffer` font référence au source fourni sur Moodle) : avec les arguments indiqués, le programme crée 2 threads numérotés 0 et 1 ainsi qu'un buffer de 4 cases. Dans l'état illustré sur la figure, le thread principal a déjà écrit les 3 premières valeurs dans le buffer, il lui reste à écrire les deux dernières, ainsi que les valeurs -1 servant aux threads à détecter l'ordre de se terminer. En supposant que le thread 0 lise en premier dans le buffer, il devra attendre 9 secondes, ce qui laissera au thread 1 le temps de lire les 3 valeurs suivantes et attendre 3, puis 4 et enfin 5 secondes. Après ses 9 secondes d'attente, le thread 0 n'aura plus que la dernière valeur à lire et attendra alors les 2 dernières secondes.



Pour être complété, le programme doit répondre aux exigences suivantes :

- il ne doit pas utiliser de variable globale ;
- il ne doit pas comporter d'attente active ;
- il ne doit pas comporter de problème de concurrence ;
- les appels à `sleep` doivent être réalisables en parallèle par les différents threads (i.e. sans avoir à attendre qu'un autre thread finisse son appel à `sleep`) ;
- l'affichage des « `*` » ne doit être réalisé que lorsque tous les threads ont épuisé les valeurs transmises et terminé les attentes correspondantes ;
- les mécanismes de synchronisation utilisés doivent être les **barrières** et les **conditions POSIX** (et donc implicitement les verrous d'exclusion mutuelle) à l'exclusion de tout autre mécanisme [1]. Pour éviter les réveils inutiles, vous prendrez soin de ne réveiller un thread que pour une bonne raison

Le programme fourni ne doit être modifié que pour ajouter les synchronisations aux endroits indiqués (cf. « `TODO` ») ainsi que les initialisations et passages d'arguments nécessaires. Vous ne devez pas modifier l'algorithme ou placer les synchronisations à d'autres endroits.

Vous trouverez sur Moodle des scripts simples de test.

1. Ne cherchez pas à utiliser votre implémentation des sémaphores avec les conditions, ce serait inadéquat et sanctionné.