TP1 Systèmes Concurrents

Mahmoud Dkhissi

05 November 2023

1 Première approche : les fourchettes sont des ressources critiques.

La version de base (PhiloBase.java) utilise un tableau de sémaphores pour gérer les fourchettes, où le nombre de sémaphores correspond au nombre de philosophes (n). Chaque sémaphore qui représente une fourchette est initialement configuré avec une ressource disponible (valeur 1). Lorsqu'un philosophe souhaite commencer à manger, il suit un protocole strict : d'abord, il prend la fourchette à sa droite, puis celle à sa gauche. Une fois qu'il a terminé de manger, il rétablit l'ordre initial en reposant d'abord la fourchette à sa droite, puis celle à sa gauche.

Pour illustrer le phénomène d'interblocage dans la version de base, il suffit d'ajouter un délai de 100000 ms (Thread.sleep(100000)) entre la prise de la fourchette droite et celle de la gauche. L'interblocage se produit lorsque tous les philosophes tentent de prendre leur fourchette droite simultanément.

Une solution initiale (PhiloAlter.java) consiste à permettre au premier philosophe de prendre en premier la fourchette droite, puis la fourchette gauche, tandis que pour les autres philosophes, on inverse l'ordre de prise de fourchettes. Cette approche est conçue pour prévenir la situation d'interblocage que nous avons examinée précédemment.

La deuxième solution (Philo2.java) implique de donner d'abord la fourchette droite à un philosophe, puis de vérifier si la fourchette gauche est disponible. Si la fourchette gauche est disponible, le philosophe peut la prendre. En cas de non-disponibilité de la fourchette gauche, le philosophe relâche la fourchette droite. Cette approche garantit que le processus ne se bloque pas, car le philosophe est prêt à s'adapter en fonction de la disponibilité de la fourchette gauche.

2 Deuxième approche : contrôler la progression d'un philosophe en fonction de l'état de ses voisins.