Programmation Système
TP n° 9 : threads
21-22 mars 2016

Exercice 1 :

1. Écrire un programme à deux *threads*, partageant une variable initialisée à 0. Chaque *thread* doit incrémenter cette variable un million de fois. Faites en sorte que la variable vaille toujours deux millions à la fin.
2. Modifier le programme précédent pour que le *thread* auxiliaire décrémente la variable au lieu de l'incrémenter, *mais sans jamais la rendre négative* : si la variable est nulle, il rend la main (en dormant un peu ou en faisant un appel à `sched_yield()`), puis essaie de nouveau de décrémenter la variable.
3. Améliorer le point précédent en utilisant une variable de condition. Comparer les temps d'exécution.

Exercice 2 : crible d'Ératosthène parallèle

Ératosthène a écrit les entiers à partir de 2 sur le sol. Il constate que le premier entier est 2, il envoie un esclave barrer les multiples de 2. Il constate que le premier entier qui reste est un 3, il envoie un autre esclave barrer les multiples de 3. Il constate que l'entier suivant est 5, il envoie un troisième esclave barrer les multiples de 5, etc.

1. On suppose d'abord qu'Ératosthène dispose d'un nombre illimité d'esclaves. Implémentez le crible d'Ératosthène parallèle.
2. On suppose maintenant qu'Eratosthène a un nombre limité d'esclaves. Modifiez votre programme pour qu'il ne crée jamais plus de n *threads* à la fois, où n est un paramètre de ligne de commande. (Vous pourrez implémenter un sémaphore à l'aide d'une variable globale de type `int` et d'une variable de condition.)
3. Que pensez-vous de la localité en mémoire de cet algorithme ? Proposez un crible d'Eratosthène parallèle qui a de meilleures propriétés.