TD2 Programmation parallèle et distribuée
L3 Informatique 2018-2019
Threads, mutex, sémaphores

Exercice 1

Écrire un programme qui crée un thread qui initialise un tableau de réels avec des valeurs aléatoires entre 0 et 999. Le nombre d'éléments du tableau est passé en paramètre. Dans le même temps, le thread principal lit un réel x au clavier.
Lorsque le tableau est fini de générer, le programme crée un thread qui renvoie 1 si l'élément x est dans le tableau, et 0 sinon.

Exercice 2

Écrire un programme avec un compteur global compt, et qui crée deux threads :
- Le premier thread itère l'opération suivante : on incrémente le compteur et attend un temps aléatoire entre 1 et 8 secondes.
- Le deuxième thread affiche la valeur du compteur toutes les trois secondes.
Les accès au compteur seront bien sûr protégés par un mutex. Les deux threads se terminent lorsque le compteur atteint une valeur limite passée en argument (en ligne de commande) au programme.

Exercice 3

Créer un programme qui a en variable globale un tableau de N double, avec N=100. Dans le main, le tableau sera initialisé avec des valeurs réelles aléatoires entre 0 et 100, sauf les valeurs tableau[0] et tableau[99] qui valent 0.
Le programme crée alors deux threads :
- Le premier thread remplace chaque valeur tableau[i], avec i = 1,2,...,98 par la moyenne (tableau[i-1]+tableau[i]+tableau[i+1])/3. Il attend ensuite un temps alléatoire entre 1 et 3 secondes ;
- Le deuxième thread affiche le tableau toutes les 4 secondes.

Exercice 4

1) Les sémaphores permettent de réaliser simplement des rendez-vous. Deux threads T1 et T2 itèrent un traitement 10 fois. On souhaite qu'à chaque itération le thread T1 attende à la fin de son traitement qui dure 2 secondes le thread T2 réalisant un traitement d'une durée aléatoire entre 4 et 9 secondes. Écrire le programme principal qui crée les deux threads, ainsi que les fonctions de threads en organisant le rendez-vous avec des sémaphores.

2) Ecrire une version plus générale dans laquelle N threads doivent se donner rendez-vous, N étant passé en argument au programme. Les threads ont tous une durée aléatoire entre 1 et 9 secondes.

Exercice 5

Un thread émetteur dépose , à intervalle variable entre 1 et 4 secondes, un octet dans une variable globale à destination d'un processus récepteur. Le récepteur lit cet octet à intervalle variable aussi entre 1 et 4 secondes. Quelle solution proposez-vous pour que l'émetteur ne dépose pas un nouvel octet alors que le récepteur n'a pas encore lu le précédent et que le récepteur ne lise pas deux fois le même octet ?

Exercice 6

Des processus producteurs produisent des objets et les insèrent un par un dans un buffer de n places. Bien entendu des processus consommateurs retirent, de temps en temps les objets (un par un). Résolvez le problème pour qu'aucun objet ne soit ni perdu ni consommé plusieurs fois. Écrire une programme avec N threads producteurs et M threads consommateurs, les nombres N et M étant saisis au clavier. Les producteurs et les consommateurs attendent un temps aléatoire entre 1 et 3 secondes entre deux objets. Les objets sont des entiers que l'on stocke dans un tableau de n entiers avec gestion LIFO (i.e. last in first out : dernier entré premier sorti). S'il n'y a plus de place, les producteurs restent bloqués en attendant que des places se libèrent.