Algorithmique et Programmation Parallèle

TD 1 MT – Threads POSIX, thread-safety, mutex

Exercice 1 : « Hello world ! »

1. Ecrire un programme créant un nombre de threads spécifiés en argument, chaque thread devant afficher une chaîne de caractères (par exemple « `Hello world !` ») sur la sortie standard.

2. Modifier le programme précédent pour que chaque thread affiche en outre un entier qui lui est passé en paramètre égal à l'ordre dans lequel il a été créé : le premier thread créé affiche 1, le second 2, etc. La chaîne devra être passée en argument, ainsi que l'identifiant du thread. Essayez de deux manières différentes :
   a) En créant la chaîne (« `Hello...` » + identifiant) **avant de la passer** au thread à créer (utilisez la fonction `sprintf`) ;
   b) **En passant la chaîne et l'identifiant séparément** au thread à créer; chaque thread se chargera d'appeler `printf` lui-même.

Exercice 2 : multithread-unsafe

Tout programme parallèle doit manier avec précaution des ressources partagées. Dans le cas de la programmation « multithreadée », les variables globales sont des variables communes et visibles de tous les threads. Accéder en écriture de façon concurrente à ce type de variables sans protection peut rendre le programme faux.

Ouvrir le fichier `mt_unsafe/exemple.c` et étudier le code.

1. Compiler le code une fois, et exécuter le plusieurs fois.
   Que remarquez-vous ?
   Comment expliquer le problème ?

2. Comment résoudre le problème ? (indication : utiliser les fonctions `pthread_mutex_lock / pthread_mutex_unlock`)

3. Mesurez le temps du programme incorrect d'origine (avec la commande `time`) et comparez le avec votre programme correct.
   Si votre programme est plus lent, proposez une nouvelle implémentation plus rapide.

Exercice 3 : Recherche parallèle du max d'un tableau

Ouvrir et étudier les sources du fichier `mt_max_tab/max_tab.c`. Ce programme prend en argument 2 entiers : le nombre d'éléments d'un tableau d'entiers et un nombre de threads à créer :

```
% max_tab.exe  nelts  nthreads
```

Pour l'instant, le programme ne créée aucun thread et recherche en séquentiel le maximum du tableau d'entiers (entiers compris entre 1 et 1000 et tirés aléatoirement).

Travail à effectuer : rechercher le maximum du tableau en créant *nthreads* threads et en affectant ce maximum dans la variable `maxv_mt` (déjà déclarée dans les sources). Implémenter 2 méthodes :

1. chaque thread recherche la valeur maximale dans une sous-partie du tableau, et renvoie cette valeur. Le thread principal recherche le maximum parmi les valeurs retournées ;

2. chaque thread met à jour une variable partagée entre les threads contenant le maximum.