Rapport – Travaux Pratique 1

Sommaire

Exercice 1

Dans ce programme, nous avons 3 threads différents :

- Le thread principal
- Le thread qui s'occupe de la fonction *foo*
- Le thread qui s'occupe de la fonction *bar*

Exercice 2

Nombre de threads	**Moyenne du temps de calcul sur 10 exécutions (ms)**
2	1687
4	1047
8	737
16	640

Plus le nombre de thread augmente, plus le temps de calcul diminue ce qui semble logique puisque le calcul est partagé entre les threads.

En moyenne le temps de calcul est de 1028 ms et l'écart type est 409 ms, ainsi on se rend compte d'une variabilité importante au niveau du temps de calcul. Cette variabilité est égale à environ la moitié de la moyenne. Ainsi, on voit qu'avec 4 threads on est dans la moyenne, alors qu'avec 2

threads on est nettement au-dessus de la moyenne. Pour 8 et 16 threads on est significativement en deçà de la moyenne.

Exercice 3

Dans ce programme, le résultat du calcul de chaque thread est écrit dans une même case mémoire ce qui peut produire un problème d'interblocage. Pour supprimer ce problème dans cet algorithme, j'ai décidé d'utiliser une section critique lors de l'écriture du résultat dans la variable globale partagée par tous les threads. Cette section critique sera protégée par un mutex.

Exercice 4

La variable globale de l'exercice précédent est maintenant encapsulée dans un moniteur qui gère son accès en section critique. Ce moniteur a seulement 2 méthodes d'accès, une méthode d'accès en lecture et une autre méthode d'accès en écriture pour incrémenter le résultat. On peut faire le lien avec la notion de lecteur/rédacteur, or ici on a un seul lecteur qui est le thread principal et plusieurs rédacteurs qui sont les autres threads qui calcule la valeur de pi. Ici l'accesseur de la variable protégée n'est pas en section critique et pour cause le lecteur qui est le thread principal, appelle cette fonction uniquement quand tous les autres threads se sont rejoints et qu'ils ne modifient donc plus la variable.

Exercice 5

Pour cet exercice l'utilisation de 2 moniteurs est nécessaire. Un premier moniteur va servir à stocker les nombres premiers « jumeaux », tandis que le second moniteur va distribuer le nombre à calculer. Le premier moniteur reprend le même principe que le moniteur de l'exercice précédent tandis que le second moniteur n'a qu'un seul accesseur qui donne à chaque thread la valeur à traiter.