2024-25

AR

**Exercice1 :**

Soit la boucle suivante :

for (int i = -10; i < 10; i++) {

x++;

}

Sur une machine monoprocesseur, quelles sont les valeurs extrêmes (min, max) du domaine de valeurs possibles qu’est susceptible de prendre la variable partagée x, initialisée à 0, après exécution complète par entrelacement du code ci-dessus par 2 threads distincts ?

Justifiez vos réponses.

Rep :

**Valeur max :**

aucun entrelacement des threads, ils s'executent sequentiellement : T0 en entier, puis T1

cela implique : 0 + 20 + 20 = 40

**Valeur min :**

x = x + 1 se decompose en :

mov r, x

add r, 1

mov x, r

1. T0 : mov r, x (=0)

changement contexte

2. T1 : exécute sa boucle jusqu'à n-1 (19) => x = 19

changement contexte

3. T0 : continue depuis le point 1.

add r, 1

mov x, r

=> x = 1

changement contexte

4. T1 : commence sa dernière boucle; x = à 1 du point précédent:

mov r, x

changement contexte

5. T0 : fini son exécution => x = 20

6. T1 : fini sa dernière boucle (depuis point 4) :

add r, 1

mov x,r

7. => x = 2

**Resultat : (2,40)**

**Exercice2 :**

1. Cherchez par vous-même la définition d’une fonction réentrante, et donnez un exemple.

Une fonction *ré-entrante* peut aussi être appelée simultanément depuis de multiples threads, mais uniquement si chaque invocation utilise ses propres données

// fun1() et fun2() sont reentrant

int fun1(int i)

{

    return i \* 5;

}

int fun2(int i)

{

   return fun1(i) \* 5;

}

2. Quelle est la différence entre une fonction réentrante et thread-safe ?

Une fonction *thread-safe* peut être appelée simultanément depuis de multiples threads, même si les invocations utilisent des données partagées, car la totalité des références sur les données partagées sont sérialisées.

Par conséquent, une fonction *thread-safe* est toujours *ré-entrante*, mais une fonction *ré-entrante* n'est pas toujours *thread-safe*.

**Exercice3 :**

Refaire l’exercice de la baignoire avec les threads Posix, est ce qu’il y a une différence dans

les résultats par rapport au programme séquentiel ?

rep :

Voir le code