Laboratorio 05

Competencias para desarrollar
Distribuir la carga de trabajo entre hilos utilizando programación en C y OpenMP.

Instrucciones
Esta actividad se realizará individualmente. Al finalizar los períodos de laboratorio o clase, deberá entregar este archivo en formato PDF y los archivos .c en la actividad correspondiente en Canvas.

1. **(18 pts.)** Explica con tus propias palabras los siguientes términos:

   a) **Private:** Genera una réplica exclusiva de una variable para cada hilo dentro de una sección paralela. Esto implica que cada hilo opera con su propia versión de la variable, y las modificaciones hechas por un hilo no tienen impacto en las versiones de los otros. Al abandonar la región paralela, las copias privadas se eliminan y la variable original permanece sin cambios.

   b) **Shared:** Una variable se comparte por todos los hilos en una sección paralela. Cada uno de los hilos tiene acceso y puede cambiar la misma instancia de la variable en la memoria. Esta característica es beneficiosa cuando se desea que los hilos colaboren en una tarea común, pero es importante prevenir situaciones de competencia.

   c) **Firstprivate:** Genera un duplicado exclusivo de una variable para cada hilo, y también configura cada copia con el valor que la variable poseía previo a ingresar en la zona paralela. Esto resulta beneficioso cuando se desea que todos los threads inicien con una condición inicial compartida, manteniendo el valor original de la variable.

   d) **Barrier:** Coordina los hilos en una sección paralela, obligándolos a detenerse y esperar hasta que todos lleguen a la misma barrera. Este método garantiza que una sección concreta de código no se ejecute hasta que todos los hilos hayan terminado la sección previa.

   e) **Critical:** Se emplea para salvaguardar bloques de código que no deben ser ejecutados al mismo tiempo por varios hilos. Solamente un hilo puede operar el código en una sección crítica a la vez, evitando así conflictos cuando se alteran recursos compartidos, aunque puede causar un bloqueo si múltiples hilos intentan ingresar a la sección simultáneamente.

   f) **Atomic:** Permite que operaciones básicas en variables compartidas, como incrementos, se realicen de forma atómica por un solo hilo. Esto garantiza que las operaciones no sean interrumpidas por otros hilos, reduciendo las condiciones de carrera sin la sobrecarga de una sección crítica completa.

2. **(12 pts.)** Escribe un programa en C que calcule la suma de los primeros N números naturales utilizando un ciclo ***for*** **paralelo**. Utiliza la cláusula **reduction con +** para acumular la suma en una variable compartida.
   a) Define N como una constante grande, por ejemplo, N = 1000000.
   b) Usa omp_get_wtime() para medir los tiempos de ejecución.

3. **(15 pts.)** Escribe un programa en C que ejecute tres funciones diferentes en paralelo usando la **directiva #pragma omp sections**. Cada sección debe ejecutar una función distinta, por ejemplo, una

que calcule el factorial de un número, otra que genere la serie de Fibonacci, y otra que encuentre el máximo en un arreglo, operaciones matemáticas no simples. Asegúrate de que cada función sea independiente y no tenga dependencias con las otras.

4. **(15 pts.)** Escribe un programa en C que tenga un ciclo for donde se modifiquen dos variables de manera paralela usando #pragma omp parallel for.
    a. Usa la cláusula shared para gestionar el acceso a la variable1 dentro del ciclo.
    b. Usa la cláusula private para gestionar el acceso a la variable2 dentro del ciclo.
    c. Prueba con ambas cláusulas y explica las diferencias observadas en los resultados.

5. **(30 pts.)** Analiza el código en el programa Ejercicio_5A.c, que contiene un programa secuencial. Indica cuántas veces aparece un valor key en el vector a. Escribe una versión paralela en OpenMP utilizando una descomposición de tareas **recursiva**, en la cual se generen tantas tareas como hilos.

Enlace al Github que incluye el código:
https://github.com/357U4RD0/Labs_Micro

6. **REFLEXIÓN DE LABORATORIO: se habilitará en una actividad independiente.**