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
      - Especifica que, en una lista de miembros de clase, esos miembros son accesibles solo desde funciones miembro y friend de la clase.
   b) Shared
      - Especifica que una o varias variables deben compartirse entre todos los subprocesos
   c) Firstprivate
      - Especifica que cada subproceso debe tener su propia instancia de una variable y que la variable se debe inicializar con el valor de la variable, ya que existe antes de la construcción paralela.
   d) Barrier
      - Proporciona un mecanismo de coordinación de subprocesos que bloquea un grupo de subprocesos de tamaño conocido hasta que todos los subprocesos de ese grupo hayan alcanzado la barrera.
   e) Critical
      - Sección que protege los datos en un programa de multi hilos. Cuando un hilo entra a una sección critica, no puede entrar ningún otro hasta que el primero haya salido.
   f) Atomic
      - Asegura que una ubicación de almacenamiento específica sea accedida de manera atómica, en lugar de exponerla a la posibilidad de que múltiples hilos la lean y escriban simultáneamente, lo que podría resultar en valores indeterminados.

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.

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

Referencias

TylerMSFT. (2023). Cláusulas de OpenMP. Microsoft.com. https://learn.microsoft.com/es-es/cpp/parallel/openmp/reference/openmp-clauses?view=msvc-170

Milek. (2024). How to use critical section. Stack Overflow. https://stackoverflow.com/questions/49722736/how-to-use-critical-section