# 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:**
2. **Private**

Las variables declaradas como private son privadas para cada hilo. Ya que cada hilo tiene su propia copia de la variable, así mismo los cambios realizados por un hilo no afectan a las copias de otros hilos.

1. **Shared**

Las variables que son declaradas shares son compartidas entre todos los hilos, así mismo todos los hilos pueden leer y escribir en la misma variable.

1. **Firstprivate**

Las variables de firstprivate se inicializan con el valor de la variable correspondiente en el hilo maestro antes de entrar en la región paralela.

1. **Barrier**

La directiva barrier sincroniza todos los hilos en un punto específico del código. Así mismo todos los hilos deben llegar a la barrera antes de que cualquiera de ellos pueda continuar.

1. **Critical**

La directiva de criticar define una sección del código que solo puede ser ejecutada por un hilo a la vez. Esto comúnmente se utiliza para proteger secciones críticas del código que no deben ser ejecutadas simultáneamente por múltiples hilos.

1. **Atomic**

La directiva de atomic asegura que una operación sea específica en una variable se realice de manera atómica, es decir que esta se realizara sin ninguna interferencia de otros hilos, esta es útil para operaciones simples como incrementos o actualizaciones.

1. **(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.
2. Define N como una constante grande, por ejemplo, N = 1000000.
3. Usa omp\_get\_wtime() para medir los tiempos de ejecución.

Nombre del archivo Lab5.c

1. **(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.
2. **(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.
   1. Usa la cláusula shared para gestionar el acceso a la variable1 dentro del ciclo.
   2. Usa la cláusula private para gestionar el acceso a la variable2 dentro del ciclo.
   3. Prueba con ambas cláusulas y explica las diferencias observadas en los resultados.
3. **(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.

**Bibliografias**

OpenMP Architecture Review Board. (n.d.). OpenMP API Specification. Retrieved from <https://www.openmp.org/specifications/>

OpenMP Architecture Review Board. (n.d.). *OpenMP API Specification*. Retrieved from <https://www.openmp.org/specifications/>