Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3086 Programación de microprocesadores

Ciclo 2 de 2,024
Temario A

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 cada subproceso debe de tener su propia instancia de una variable.
   b) Shared: Especifica que una o varias variables deben compartirse entre todos los subprocesos.
   c) Firstprivate: Determina que cada subproceso debe de tener su propia instancia de una variable y que la variable se debe inicializar con el valor de la variable.
   d) Barrier: Especifica una barrera explicita en el punto en donde el construct aparece.
   e) Critical: Restricte la ejecución del bloque estructurado asociado a un single thread en el momento.
   f) Atomic: Se asegura que se acceda a una locación especifica de almacenamiento automáticamente, en vez de exponerla a la posibilidad de múltiple, simultáneamente leyendo y escribiendo hilos que pueden resultar en valores indeterminables.

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.

   variable1 (compartida): Debido a que esta variable es compartida, los incrementos realizados por cada hilo se reflejan en el valor final de variable1. Como múltiples hilos pueden modificarla simultáneamente, el resultado puede no ser lo que se espera debido a posibles condiciones de carrera.

   variable2 (privada): Cada hilo tiene su propia copia de variable2, por lo que el valor de variable2 dentro del ciclo es independiente para cada hilo.

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.

Análisis:

Generación de un Arreglo Aleatorio:

- Define un arreglo a de tamaño N = 131072.
- Rellena este arreglo con números aleatorios usando la función rand() % N.

Inserción Manual del Valor key:

- El valor de key es fijado en 42.
- El programa inserta manualmente el valor key en tres posiciones específicas del arreglo a.

Conteo del Número de Ocurrencias de key:

- Utiliza la función count_key() para contar cuántas veces aparece key en el arreglo.
- Imprime el número de apariciones de key en el arreglo.