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**
   Una variable que tiene una copia independiente para cada hilo en ejecución.

   b) **shared**
   Es una variable que es compartida por todos los hilos, y que los cambios que esta tenga serán acumulativos en cada hilo.

   c) **firstprivate**
   Es un tipo de variable privada en la que cada copia privada se inicializa con el valor de una variable compartida antes de que comience el proceso paralelo.

   d) **barrier**
   Es una directiva usada para declarar un punto de sincronización en el que todos los hilos deben llegar antes de que alguno pueda continuar.

   e) **critical**
   Es una directiva usada en el código para darle acceso a solo un hilo, el cual puede acceder a una región de código específica en un momento dado.

   f) **atomic**
   Es una operación que no puede ser interrumpida por otros hilos.

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.

Resultado

```
El resultado de la suma es: 1784293664
El tiempo del proceso es: 0.000000 segundos
```

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.

Resultado

```
Factorial de 10 es 3628800
Serie de Fibonacci hasta 100: 0 1 1 2 3 5 8 13 21 34 55 89
El valor maximo en el arreglo es 5438
```

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.

Resultado

```
Hilo 5: variable 1 = 7, variable 2 = 10
Hilo 1: variable 1 = 45, variable 2 = 2
Hilo 4: variable 1 = 45, variable 2 = 8
Hilo 9: variable 1 = 45, variable 2 = 18
Hilo 8: variable 1 = 45, variable 2 = 16
Hilo 2: variable 1 = 45, variable 2 = 4
Hilo 0: variable 1 = 45, variable 2 = 0
Hilo 6: variable 1 = 45, variable 2 = 12
Hilo 7: variable 1 = 45, variable 2 = 14
Hilo 3: variable 1 = 45, variable 2 = 6
```

Diferencias observadas

En la variable 1, al ser esta una variable pública, todos los hilos modifican a la misma variable, por lo que la modificación de la variable será acumulativa, lo que en el valor final reflejara la suma de todos los procesos.

En la variable 2, al ser esta una variable privada, todos los hilos tienen una copia de esta, lo que permite que todos los cambios realizados en un hilo no afecten a los demás.

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.

Resultado

```
Numero de veces que [key] aparece: 7
```

Referencias

- ***private Clause. (n.d.).*** ***https://www.openmp.org/spec-html/5.2/openmpsu37.html***
- ***shared Clause. (n.d.).*** ***https://www.openmp.org/spec-html/5.2/openmpsu36.html***
- ***firstprivate Clause. (n.d.).*** ***https://www.openmp.org/spec-html/5.2/openmpsu38.html***
- ***barrier Construct. (n.d.).*** ***https://www.openmp.org/spec-html/5.0/openmpsu90.html***
- ***critical Construct. (n.d.).*** ***https://www.openmp.org/spec-html/5.0/openmpsu89.html***
- ***atomic Construct. (n.d.).*** ***https://www.openmp.org/spec-html/5.0/openmpsu95.html***