Descripción Práctica 1
Programación Concurrente
15-I
Generación de Tareas en forma de hilos usando OpenMP

1.-Abre una terminal y conéctate al servidor del laboratorio (IP: 148.206.41.113), usando la cuenta común "concurrente", de la siguiente manera:

$ssh concurrente@148.206.41.113

password: programacion

2.- Ejecuta el comando **ls** y observa que hay un archivo llamado "holaomp.c" Haz una copia de ese archivo creando otro que tenga tu matricula al final del nombre, por ejemplo:

$**cp** holaomp.c holaomp_208312940.c

ejecuta nuevamente el comando **ls** para checar que se ha creado la copia

2.-Abre el archivo nuevo usando el editor de textos nano, de la siguiente manera:

$**nano** holaomp_208312940.c

ese editor permite hacer modificaciones inmediatamente que se abre el archivo. Observa los comandos de la parte inferior del editor, de los cuales podemos utilizar ctrl-o para guardar los cambios y ctr-x para salir del editor.
Modifica tu archivo para que el mensaje en el printf diga :

Hola Mundo soy **tunombre**

3.- Compila el archivo y corre el programa de la siguiente manera:

$gcc holaomp_208312940.c -o hola_208312940 -fopenmp
$**./**hola_208312940

Observa la salida del programa.

4.- Modifica el programa para que cada hilo despliegue lo siguiente:

Hola Mundo soy **tunombre** numero de Hilo= **id_del hilo**

Para obtener el id del hilo debes invocar a la función:

```
id = omp_get_thread_num();
```

la variable id la puedes declarar antes de la directiva pragma omp parallel, luego indicar que es una variable privada para cada hilo:

```
  int id;
#pragma omp parallel private (id)
```
otra opción equivalente es declarar la variable id dentro del pragma omp parallel:

```
#pragma omp parallel
{  int id;
.....
}
```

las variables declaradas dentro de las llaves son por default privadas para cada hilo. Prueba la dos versiones.

5.- Modifica el programa para especificar el número de hilos que se van a crear, invocando a la función omp_set_num_threads(N_HILOS), antes de la directiva #pragma omp parallel, de la siguiente manera:

```
#define N_HILOS 4  //esta definición va después de los include


 omp_set_num_threads(N_HILOS);
#pragma omp parallel private (id)
```

prueba el programa creando 4, 8, y 16 hilos

6.- Cambia el valor de la constante N_HILOS a 2 y declara una nueva constante:

```
 #define NUM_ELEMS 20  //esta definición va después de los include
```

y dentro del main declara un arreglo de enteros de tamaño NUM_ELEMS (int arreglo [NUM_ELEMS];), enseguida inicializa todo el arreglo con 1's.

En el #pragma omp parallel, declara el arreglo como una variable compartida:

```
#pragma omp parallel private (id) shared (arreglo)
```

Luego, dentro del pragma omp parallel, haz que cada hilo sume una mitad del arreglo, en base a su identificador. El hilo 0 sumaría la primer mitad y el hilo 1 suma la segunda mitad, cada hilo debe desplegar el resultado obtenido.

7.- Modifica el programa para que el hilo principal despliegue las dos sumas parciales, después de terminar el pragma omp parallel

8.- Modifica el programa para que funcione teniendo N_HILOS, es decir, cada hilo, en base a su identificador, deberá sumar una parte del arreglo, y el hilo principal despliega las sumas parciales al terminar el pragma omp parallel.
Vamos a suponer que el tamaño del arreglo siempre será un múltiplo de N_HILOS.