Descripción Práctica 4
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.- Observa el programa *sinCandadosA.c* , luego compila y corre el programa:
$gcc *sinCandadosA.c* -o sCA -fopenmp
$./sCA

Observa que al correr el programa varias veces el desplegado de cada hilo puede tener cualquier orden.
El valor final de la variable **suma** siempre es 213?

Para que se note con más claridad el cambio del valor de **suma** vamos a modificar el archivo haciendo lo siguiente

-copia el archivo *sinCandadosA.c* al archivo *sinCandadosB.c:*
$cp sinCandadosA.c sinCandadosB.c

- modifica el archivo sinCandadosB.c poniendo las líneas 12 a 19 dentro de un ciclo while infinito

- deja solamente una instrucción de desplegado, la que se da al terminar el pragma parallel, condicionándola de la siguiente manera:

```
if(suma != 213)
   printf("\nbye, suma total=%d, id:%d\n\n",suma,id);
```

- corre el programa y observa que efectivamente, el valor de suma puede ser diferente en varias corridas

3.- Para eliminar el problema de obtener diferentes valores de salida para el mismo valor de entrada, vamos a utilizar un mecanismo de sincronización que permite la exclusión mutua entre hilos: un CANDADO.

-copia el archivo *sinCandadosB.c* al archivo *sinCandadosC.c:*
$cp sinCandadosB.c sinCandadosC.c

- modifica el archivo sinCandadosC.c. Declara dentro del main el candado C1, de la siguiente manera:

```
int main (int argc, char *argv[])
{
  int id, suma;
  omp_lock_t C1;
```

- Después de hacer la declaración, invoca a la operación de inicialización del candado:

```
  omp_init_lock(&C1);  //el candado queda abierto por default
```

- invoca a las operaciones equivalentes al lock y unlock vistas en clase, para que la sección crítica se ejecute con exclusión mutua

omp_set_lock(&C1); //Operación equivalente a Lock, para cerrar un candado

omp_unset_lock(&C); //Operación equivalente a unlock, para abrir un candado

- corre nuevamente el programa y observa que ahora no se generan valores diferentes a 213.

- Elimina el ciclo while infinito y modifica el programa para que ahora el hilo main cierre el candado, después de su inicialización, y antes de que se invoque al pragma omp parallel. Explica el comportamiento que se obtiene al correr el programa.

4. Construye un nuevo programa, en donde un conjunto de hilos realiza la suma paralela de los elementos de un arreglo de enteros inicializados de manera aleatoria.