

# PROGRAMACIÓN AVANZADA

Programación paralela con OpenMP



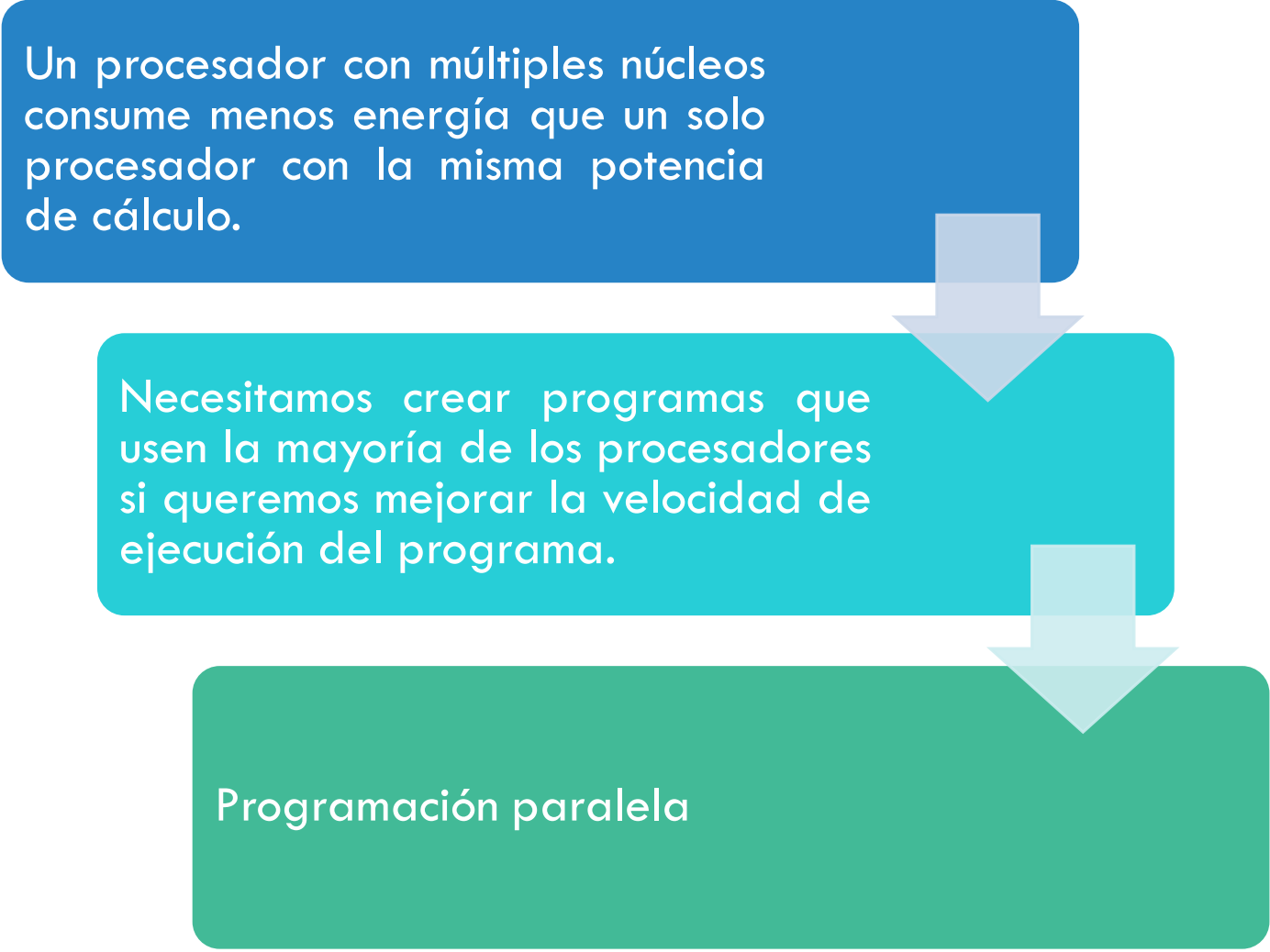
Programación paralela usando  
OpenMP.



"Introducción a OpenMP" de  
Tim Mattson.  
(<https://youtu.be/6jFkNjhJ-Z4>)

# CONTENIDO

Un procesador con múltiples núcleos consume menos energía que un solo procesador con la misma potencia de cálculo.



```
graph TD; A[Un procesador con múltiples núcleos consume menos energía que un solo procesador con la misma potencia de cálculo.] --> B[Necesitamos crear programas que usen la mayoría de los procesadores si queremos mejorar la velocidad de ejecución del programa.]; B --> C[Programación paralela];
```

Necesitamos crear programas que usen la mayoría de los procesadores si queremos mejorar la velocidad de ejecución del programa.

Programación paralela

# MOTIVACIÓN

# DEFINICIONES

## Concurrencia

- Múltiples tareas se ejecutan lógicamente al mismo tiempo. Ejemplos:
  - Multiplicar matrices.
  - Medición del valor de los criterios de división en un árbol de decisión.

## Paralelismo

- Se ejecutan múltiples tareas al mismo tiempo.

No todos los programas concurrentes pueden implementarse usando programación paralela. Debe identificar la concurrencia de su programa antes de decidir si se puede implementar utilizando un código paralelo de programación.

# OPENMP (*OPEN MULTI-PROCESSING*)

API para implementar aplicaciones multiproceso.

- Directivas del compilador.
- Rutinas de la biblioteca.

¿Qué es una API? → Interfaz que ofrece una librería para su uso.

**OpenMP**



Simplifica la escritura de programas multiproceso (o multi-hilo).



# COMPILAR Y EJECUTAR PROGRAMAS QUE USAN OPENMP

## En Code::Blocks

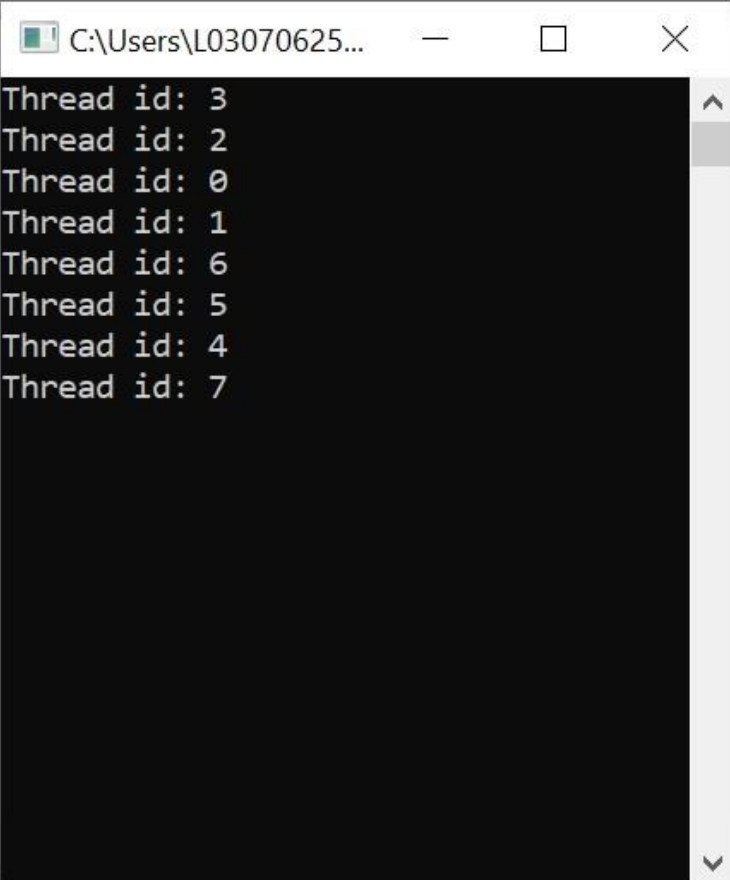
- Ir a:
  - Settings / Compiler... /
- En Compiler settings / Other compiler options:
  - Escribir: **-fopenmp**
- En Linker settings / Link libraries:
  - Add **libgomp-1.dll**
  - Se debe encontrar en la carpeta: [C:\Program Files \(x86\)\CodeBlocks\MinGW\bin\](#)

## Usando gcc:

- `gcc -fopenmp myProgram.c`
- `./myProgram.out`

# LISTADO DE LOS IDENTIFICADORES DE LOS HILOS

```
#include <omp.h>
#include <stdio.h>
int main(void) {
    #pragma omp parallel
    {
        int id = omp_get_thread_num();
        printf("Thread id: %d\n", id);
    }
    getchar(); }
```



```
C:\Users\L03070625...
Thread id: 3
Thread id: 2
Thread id: 0
Thread id: 1
Thread id: 6
Thread id: 5
Thread id: 4
Thread id: 7
```

# DESCRIPCIÓN GENERAL DE OPENMP

¿Cómo interactúan los hilos?

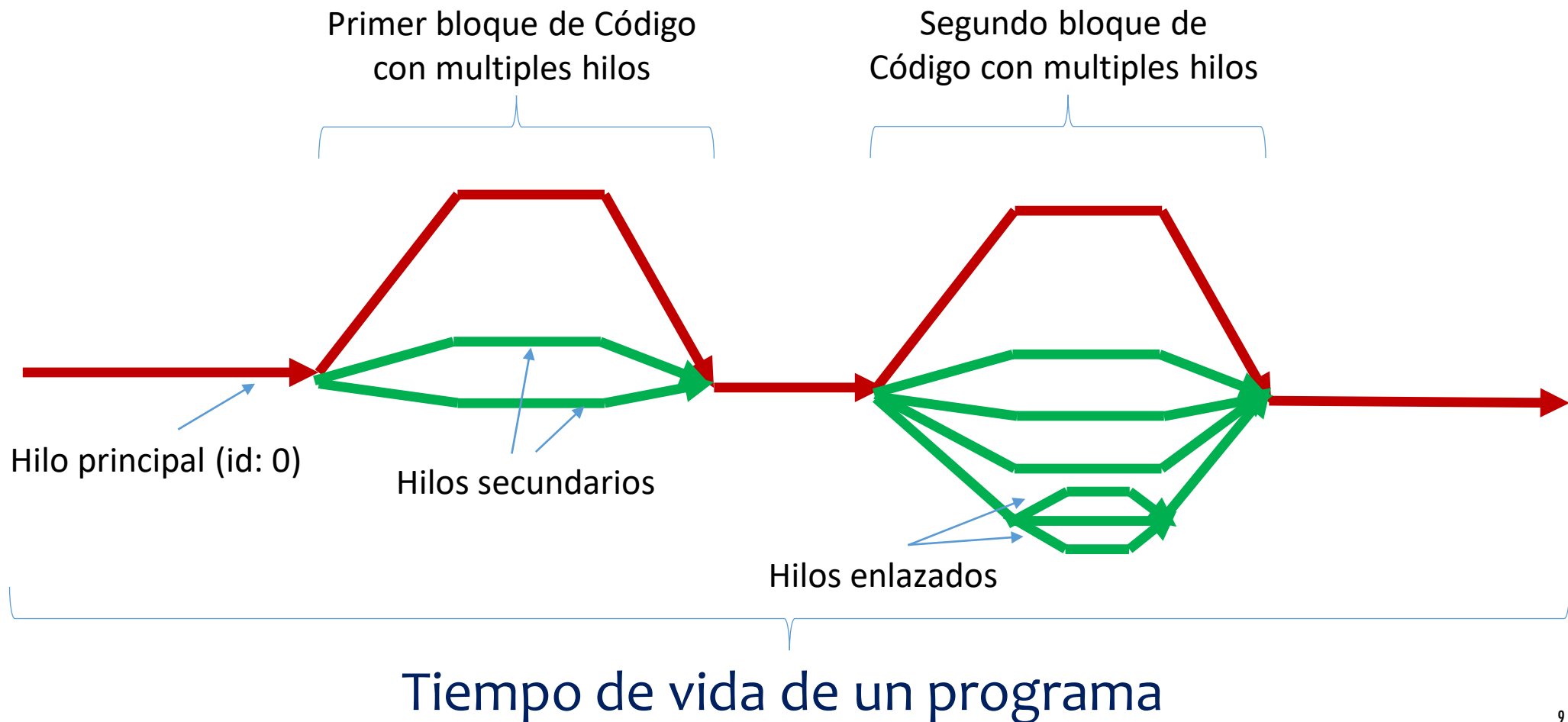
- Los hilos comparten el espacio de memoria.
- Por lo que se produce *condición de carrera*:
  - Varios hilos acceden al mismo espacio de memoria.
  - La salida del programa cambia en cada ejecución.



- Necesitamos una estrategia de sincronización.
- La sincronización consume mucho tiempo.



# EL PARALELISMO FORK-JOIN



# EJEMPLO:

CADA UNO DE 4 HILOS SUMA DOS ELEMENTOS DE UN ARREGLO DE 8 ELEMENTOS

```
#include <stdio.h>
#include <omp.h>

void sumas(int id, int *a) {
    int suma = a[2*id]+a[2*id+1];
    printf("La suma del hilo %d es:
%d \n", id, suma);
}
```

```
int main() {
    int *A = calloc(8, sizeof(int));
    for (int i = 0; i < 8; i++)
        A[i] = i + 1;
    omp_set_num_threads(4);
    #pragma omp parallel
    {
        int ID = omp_get_thread_num();
        sumas(ID, A);
    }
    free(A);
    printf("Programa terminado\n");
}
```

## EJERCICIO

Calcular la integral de una función en un intervalo es equivalente a calcular el área bajo la curva de esa función (**AUC**) en el intervalo dado. Este cálculo puede aproximarse como una suma de áreas de rectángulos más pequeños dentro del intervalo.

1. Busque información acerca del método del rectángulo para calcular integrales.
2. Implemente un programa que de manera secuencial calcule la integral (**AUC**) de la función:  
 $f(x) = \sqrt{x}$ .
3. Modifique el programa para que el área de cada rectángulo se calcule en paralelo.

**MUCHAS GRACIAS**