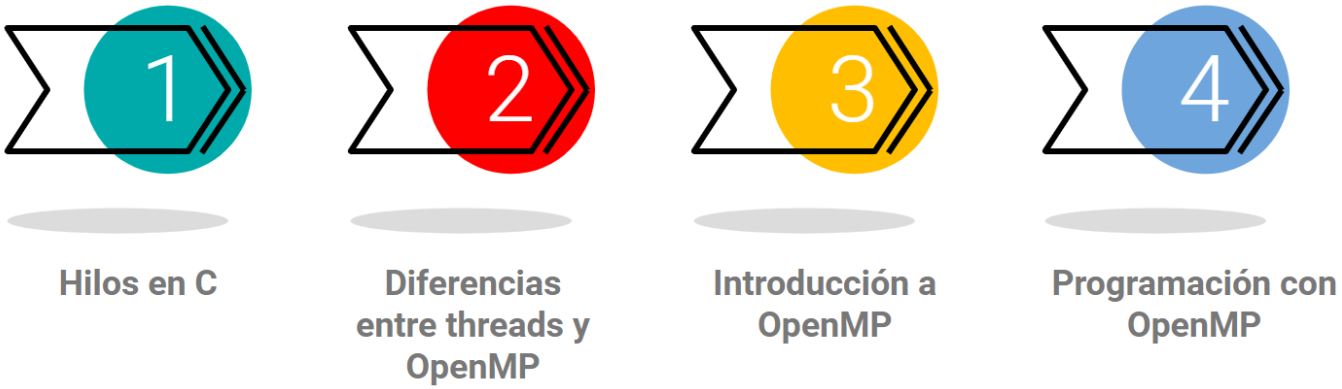
Unidad 8: OpenMP

Contenido

Contenido de la unidad



Hilos en C

> **Nota**
>
> **Veamos cómo se programan**
>
> Ver el Notebook «threadsC.ipynb»

Diferencias entre threads y OpenMP

API y nivel de abstracción

- Threads en C son de bajo nivel.
- OpenMP es de más alto nivel.

Ejecución paralela

- Threads en C permiten una ejecución paralela fina-grano.
- OpenMP se centra en la ejecución paralela a nivel de bucles y funciones.

Sincronización

- Threads en C proporcionan primitivas de sincronización de bajo nivel.
- OpenMP proporciona constructos de sincronización de más alto nivel.

Paralelismo de bucles

- OpenMP permite paralelizar bucles de forma automática.
- Threads en C, esto debe hacerse manualmente.

- **API y nivel de abstracción**: Los threads en C son una API de bajo nivel que requieren un mayor conocimiento del programador sobre cómo funcionan los hilos y cómo sincronizarlos. En cambio, OpenMP es una API de más alto nivel que proporciona constructos de programación paralela más sencillos para el usuario.
- **Ejecución paralela**: Los threads en C permiten una ejecución paralela fina-grano, lo que significa que el programador puede controlar exactamente qué se ejecuta en cada hilo. En cambio, OpenMP se centra en la ejecución paralela a nivel de bucles y funciones, lo que significa que el programador tiene menos control sobre cómo se ejecuta el código en paralelo.
- **Sincronización**: Los threads en C proporcionan primitivas de sincronización de bajo nivel como mutexes y semáforos para controlar el acceso concurrente a secciones críticas. OpenMP, por otro lado, proporciona constructos de sincronización de más alto nivel como barreras y locks implícitos.
- **Paralelismo de bucles**: OpenMP permite una forma sencilla de paralelizar bucles utilizando la directiva #pragma omp parallel for. Esta directiva permite paralelizar un bucle de forma automática en varios hilos sin necesidad de crear manualmente cada hilo. Los threads en C también pueden paralelizar bucles, pero el programador debe realizar esta tarea manualmente.

OpenMP versus threads

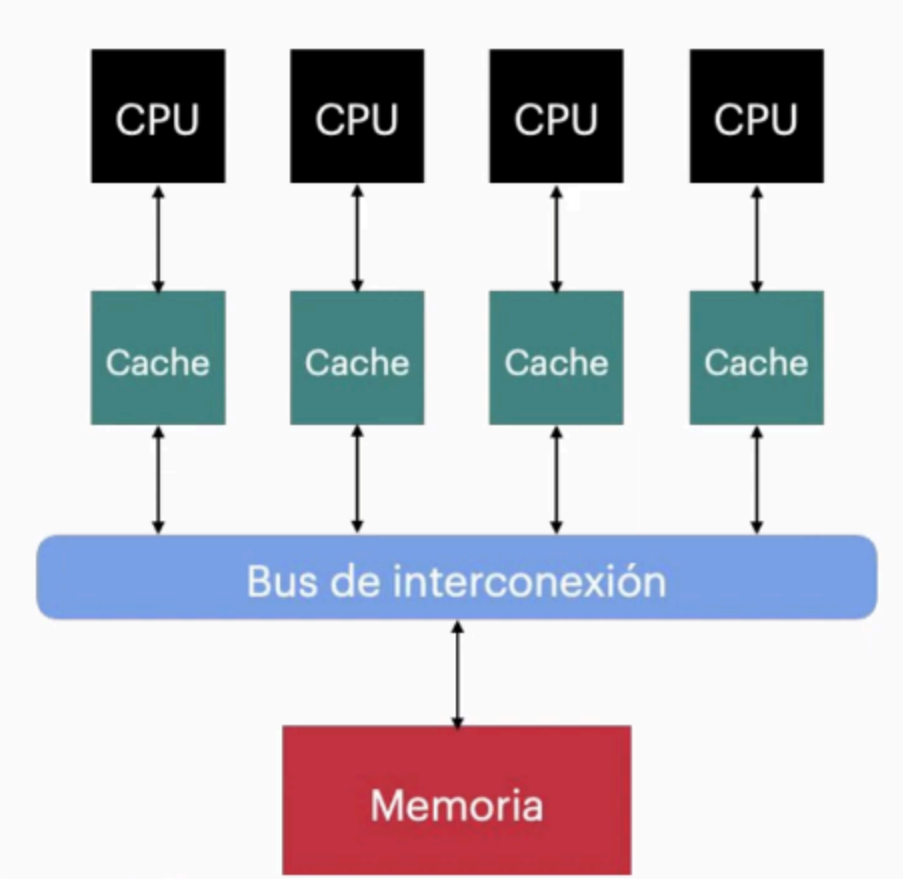
- Los threads en C proporcionan un mayor nivel de control y flexibilidad, pero requieren un mayor conocimiento del programador.
- OpenMP es más fácil de usar y proporciona una mayor portabilidad y una mayor abstracción del hardware subyacente.
- Tanto los threads en C como el uso de OpenMP en C permiten crear programas paralelos que pueden ejecutar tareas en múltiples procesadores o núcleos de forma simultánea.

Introducción

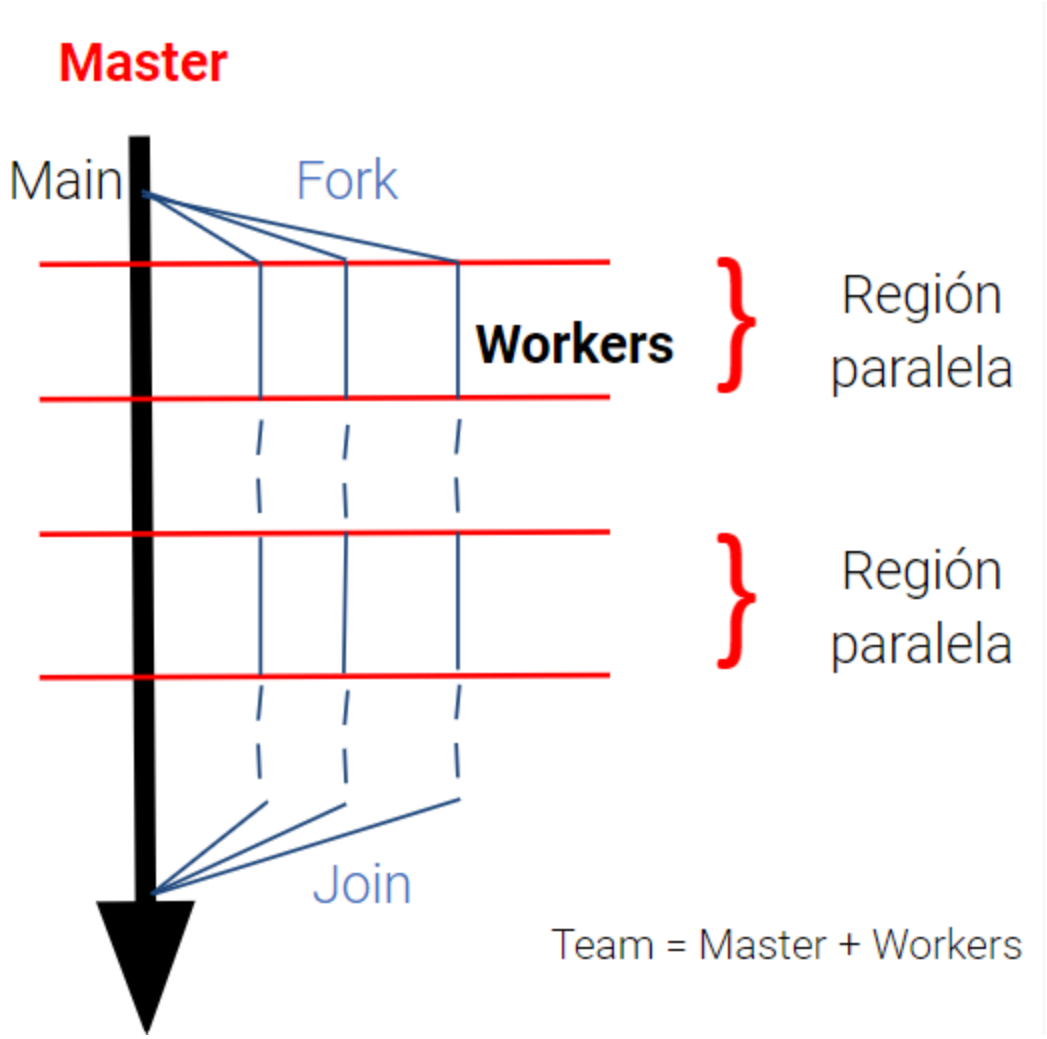
Características

- Es estándar para la programación de aplicaciones en el modelo de memoria compartida.
- Está soportado en lenguajes de programación como: C, C++ y Fortran
- Está basado o compuesto de:
  - Directivas: desarrollar elementos de programación paralela
  - Funciones o librerías: útiles para según se necesiten
  - Variables de entorno: pasar información a los programas
  - Runtime: ambiente de ejecución

Modelo para OpenMP



Modelo de ejecución en OpenMP



- El programa comienza en el hilo de ejecución principal (main).
- Los hilos de trabajo (Workers) son generados en las regiones paralelas.
- En medio de las regiones paralelas los hilos son pausados.
- OpenMP se basa en Fork/Join

Directivas y región paralela

Directivas

- **En programación en general**: las directivas son una construcción del lenguaje que indican cómo un compilador debería procesar un bloque de código
- **En OpenMP**: las directivas indicarán cómo el compilador deberá interpretar ciertas reglas de código para la ejecución en paralelo

Directivas OpenMP

- Lenguaje orientado a directivas.

  #pragma

- No son parte de la especificación o comportamiento de C
- Si un compilador no tiene soporte, simplemente ignora las directivas

Región paralela - especificando

Paralelismo expresado de manera explícita

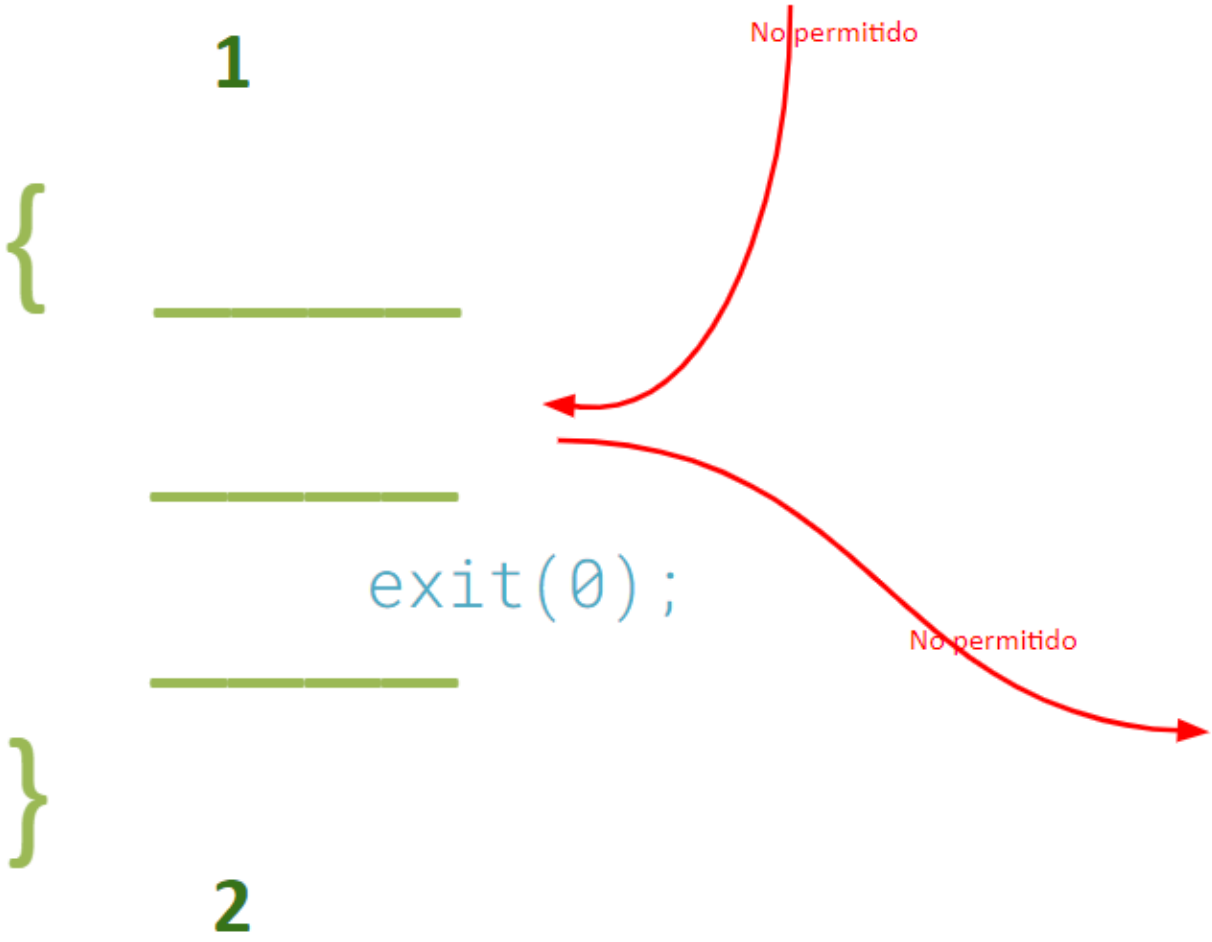
```
#pragma omp parallel
{
    // Bloque estructurado
}
```

Región paralela - bloque estructurado

Características:

- Tiene exactamente un punto de entrada en la parte superior
- Tiene exactamente un punto de salida en la parte inferior
- No se permite ramificación hacia afuera o hacia adentro
- Si se permite la terminación del programa

Región paralela - bloque estructurado



Manejo de recursos

¿Cuántos hilos para mi programa?

Lo puedo definir usando una variable de entorno

OMP_NUM_THREADS = 6

Lo puedo definir usando una cláusula num_threads():

#pragma omp parallel num_threads(6)

- Modifican el funcionamiento de las directivas
- Permite que el programador especifique el número de hilos que va a ejecutar el siguiente bloque de código

Programación con OpenMP

Nota

Veamos cómo se programan

Ver el Notebook «OpenMP_en_C.ipynb»