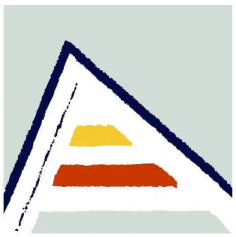
INGENIERÍA DE LOS COMPUTADORES
UNIVERSIDAD DE ALICANTE
CURSO 2023/2023



Universitat d'Alacant
Universidad de Alicante

PRÁCTICA 3/LABORATORIO 3

Tarea 1: Estudio del API OpenMP (Tarea Individual)

PROFESOR
Pau Climent Pérez, pau.climent@gcloud.ua.es

AUTORES
Javier Sala Pérez, jsp67@gcloud.ua.es

Tarea 1: Estudio del API OpenMP

1.- Introducción

En esta tarea vamos a realizar un estudio sobre la api de OpenMP y su uso con GNU GCC, para ello estudiaremos diversos ejemplos que hemos obtenido de internet.

2.-OpenMP

OpenMP es un API (Interfaz de programación de aplicaciones) , se utiliza para la programación multiproceso de memoria compartida, pudiendo trabajar y dividir las tareas en distintos hilos.

Sintaxis:

```
# pragma omp <directiva> [cláusula [ , ...] ...]
```

Su implementación está integrada, en nuestro caso, en cualquier compilador de C y C++ de GNU versión 4.2 o superior.

Para su compilación en deberemos añadir en las CFLAGS -fopenmp

3.- Ejemplos de aplicación

#pragma omp

Uso compartido de trabajo paralelo:

parallel

Define una región paralela, que es el código que ejecutarán varios subprocesos en paralelo.

for

Hace que el trabajo realizado en un bucle for dentro de una región paralela se divida entre subprocesos.

sections

Identifica las secciones de código que se van a dividir entre todos los subprocesos.

single

Permite especificar qué se debe ejecutar una sección de código en un único subproceso, no necesariamente en el subproceso principal.

Subproceso principal y la sincronización:

maestra

Especifica que solo el subproceso principal debe ejecutar una sección del programa.

crítica

Especifica que el código solo se ejecuta en un subproceso cada vez.

barrier

Sincroniza todos los subprocesos de un equipo; todos los subprocesos se pausan en la barrera hasta que todos los subprocesos ejecutan la barrera.

atomic
: Especifica una ubicación de memoria que se actualizará atómicamente.

flush
: Especifica que todos los subprocesos tienen la misma vista de memoria para todos los objetos compartidos.

Ordered
: Especifica que el código de un bucle paralelizado for debe ejecutarse como un bucle secuencial.

Ejemplos:

atomic
Especifica una ubicación de memoria que se actualizará atómicamente.

```
// omp_atomic.cpp
// compile with: /openmp
#include <stdio.h>
#include <omp.h>

#define MAX 10

int main() {
   int count = 0;
   #pragma omp parallel num_threads(MAX)
   {
      #pragma omp atomic
      count++;
   }
   printf_s("Number of threads: %d\n", count);
}
```

Output:

```
Number of threads: 10
```

Master + for + barrier
Master: especificara el subproceso a ejecutar
For: divide en subprocesos
Barrier:sincronizara los procesos

```
// compile with: /openmp
#include <omp.h>
#include <stdio.h>

int main( )
{
    int a[5], i;

    #pragma omp parallel
    {
        // Perform some computation.
        #pragma omp for
        for (i = 0; i < 5; i++)
            a[i] = i * i;

        // Print intermediate results.
        #pragma omp master
            for (i = 0; i < 5; i++)
                printf_s("a[%d] = %d\n", i, a[i]);

        // Wait.
        #pragma omp barrier

        // Continue with the computation.
        #pragma omp for
        for (i = 0; i < 5; i++)
            a[i] += i;
    }
}
```

Output:

```
a[0] = 0
a[1] = 1
a[2] = 4
a[3] = 9
a[4] = 16
```