

UNIVERSIDAD DEL VALLE DE GUATEMALA

FACULTAD DE INGENIERÍA
CC3069 COMPUTACIÓN PARALELA Y
DISTRIBUIDA

Hoja de Trabajo 1

Introducción al uso y conceptos de OpenMP

Por: Samuel A. Chamalé
Email: cha21881@uvg.edu.gt

Fecha de entrega: 11/08/2024

Índice

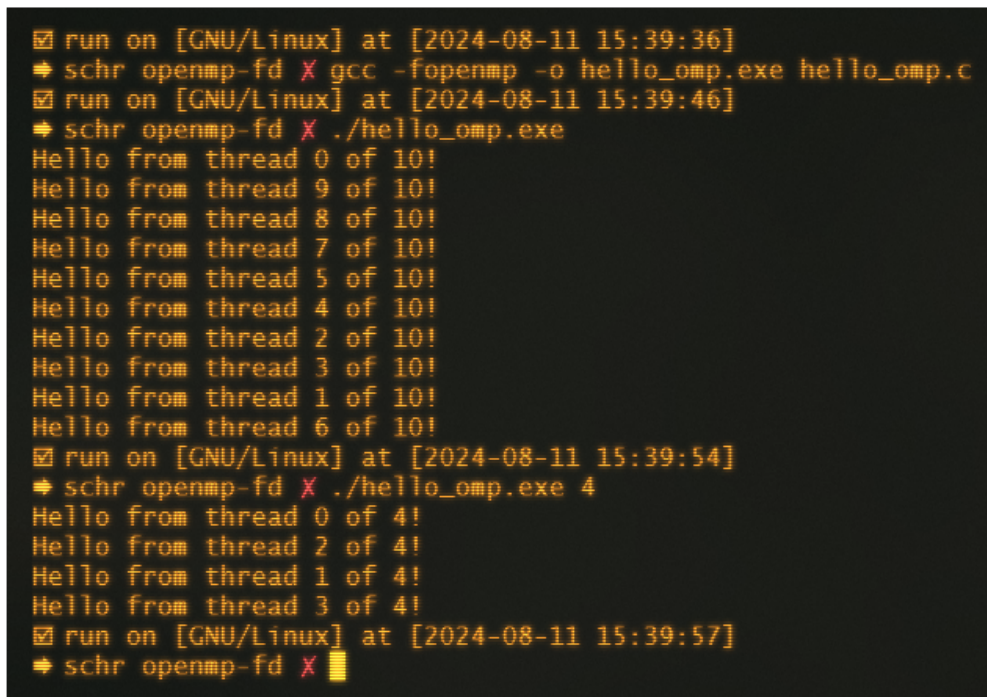
1. Ejercicio 1 [hello_omp.c]	1
1.1. ¿Por qué al ejecutar su código los mensajes no están desplegados en orden?	1
1.2. Evidencias de ejecución y resultados	1
2. Ejercicio 2 [hbd_omp.c]	2
2.1. Evidencias de ejecución y resultados	2
3. Ejercicio 3 [riemann.c]	3
3.1. Evidencias de ejecución y resultados	3
4. Ejercicio 4 [remann_omp2.c]	3
4.1. ¿Por qué es necesario el uso de la directiva <code>#pragma omp critical</code> ?	3
4.2. Evidencias de ejecución y resultados	4
5. Ejercicio 5 [riemann_omp_nocrit.c]	4
5.1. ¿Qué diferencia hay entre usar una variable global para añadir los resultados a un arreglo?	4
5.2. Evidencias de ejecución y resultados	5
6. Repositorio	5

1. Ejercicio 1 [hello_omp.c]

1.1. ¿Por qué al ejecutar su código los mensajes no están desplegados en orden?

Los mensajes no se despliegan en orden porque los *threads* se ejecutan de manera concurrente y no se garantiza un orden de ejecución. En sistemas *multithreading*, la planificación de los *threads* la realiza el sistema operativo, y diferentes threads pueden imprimir sus mensajes en cualquier orden basado en la disponibilidad de recursos y la gestión del tiempo de ejecución por parte del sistema.

1.2. Evidencias de ejecución y resultados



```
❑ run on [GNU/Linux] at [2024-08-11 15:39:36]
➡ schr openmp-fd X gcc -fopenmp -o hello_omp.exe hello_omp.c
❑ run on [GNU/Linux] at [2024-08-11 15:39:46]
➡ schr openmp-fd X ./hello_omp.exe
Hello from thread 0 of 10!
Hello from thread 9 of 10!
Hello from thread 8 of 10!
Hello from thread 7 of 10!
Hello from thread 5 of 10!
Hello from thread 4 of 10!
Hello from thread 2 of 10!
Hello from thread 3 of 10!
Hello from thread 1 of 10!
Hello from thread 6 of 10!
❑ run on [GNU/Linux] at [2024-08-11 15:39:54]
➡ schr openmp-fd X ./hello_omp.exe 4
Hello from thread 0 of 4!
Hello from thread 2 of 4!
Hello from thread 1 of 4!
Hello from thread 3 of 4!
❑ run on [GNU/Linux] at [2024-08-11 15:39:57]
➡ schr openmp-fd X
```

Figura 1: Ejecución de hello_omp.c

2. Ejercicio 2 [hbd_omp.c]

2.1. Evidencias de ejecución y resultados

```

[✓] run on [GNU/Linux] at [2024-08-11 15:57:47]
[✚] schr openmp-fd ✚ gcc -fopenmp -o hbd_omp.exe hbd_omp.c
[✓] run on [GNU/Linux] at [2024-08-11 15:57:48]
[✚] schr openmp-fd ✚ ./hbd_omp.exe 21
Feliz cumpleaños número 21!
Saludos del hilo 6
Feliz cumpleaños número 21!
Feliz cumpleaños número 21!
Saludos del hilo 10
Feliz cumpleaños número 21!
Saludos del hilo 4
Saludos del hilo 14
Saludos del hilo 16
Saludos del hilo 8
Feliz cumpleaños número 21!
Saludos del hilo 12
Feliz cumpleaños número 21!
Saludos del hilo 2
Feliz cumpleaños número 21!
Feliz cumpleaños número 21!
Saludos del hilo 18
Feliz cumpleaños número 21!
Feliz cumpleaños número 21!
Saludos del hilo 20
Saludos del hilo 0
[✓] run on [GNU/Linux] at [2024-08-11 15:57:50]
[✚] schr openmp-fd ✚
```

Figura 2: Ejecución de hbd_omp.c

3. Ejercicio 3 [riemann.c]

3.1. Evidencias de ejecución y resultados

```

❑ run on [GNU/Linux] at [2024-08-11 16:23:05]
➤ schr openmp-fd ✗ gcc -o riemann.exe riemann.c -lm
❑ run on [GNU/Linux] at [2024-08-11 16:23:20]
➤ schr openmp-fd ✗ ./riemann.exe 2 10
Con n = 1000000 trapezoides, nuestra aproximación de la integral de 2.000000 a 10.000000 es:
Integral de x^2: 330.666667
Integral de 2x^3: 4992.000000
Integral de sin(x): 0.422925
❑ run on [GNU/Linux] at [2024-08-11 16:23:25]
➤ schr openmp-fd ✗ ./riemann.exe 3 7
Con n = 1000000 trapezoides, nuestra aproximación de la integral de 3.000000 a 7.000000 es:
Integral de x^2: 105.333333
Integral de 2x^3: 1160.000000
Integral de sin(x): -1.743895
❑ run on [GNU/Linux] at [2024-08-11 16:23:29]
➤ schr openmp-fd ✗ ./riemann.exe 0 1
Con n = 1000000 trapezoides, nuestra aproximación de la integral de 0.000000 a 1.000000 es:
Integral de x^2: 0.333333
Integral de 2x^3: 0.500000
Integral de sin(x): 0.459698
❑ run on [GNU/Linux] at [2024-08-11 16:23:33]
➤ schr openmp-fd ✗
```

Figura 3: Ejecución de riemann.c

4. Ejercicio 4 [remann_omp2.c]

4.1. ¿Por qué es necesario el uso de la directiva #pragma omp critical?

La directiva `#pragma omp critical` es necesaria para evitar condiciones de carrera. Si varios threads intentan actualizar la variable `global_result` simultáneamente sin ninguna forma de sincronización, puede ocurrir que los valores escritos por un thread sean sobrescritos por otro thread, lo que lleva a resultados incorrectos. La directiva `critical` asegura que la actualización de `global_result` sea atómica, es decir, que un solo thread puede ejecutar la sección crítica a la vez, garantizando la consistencia de los datos.

4.2. Evidencias de ejecución y resultados

```

[✓] run on [GNU/Linux] at [2024-08-11 16:39:37]
[✚] schr openmp-fd X gcc -fopenmp -o remann_omp2.exe remann_omp2.c -lm
[✓] run on [GNU/Linux] at [2024-08-11 16:39:40]
[✚] schr openmp-fd X ./remann_omp2.exe 2 10 4
Con n = 1000000 trapezoides y 4 threads, la aproximación de la integral de 2.000000 a 10.000000 es:
Integral de x^2: 330.666667
Integral de 2x^3: 4992.000000
Integral de sin(x): 0.422925
[✓] run on [GNU/Linux] at [2024-08-11 16:39:49]
[✚] schr openmp-fd X ./remann_omp2.exe 3 7 8
Con n = 1000000 trapezoides y 8 threads, la aproximación de la integral de 3.000000 a 7.000000 es:
Integral de x^2: 105.333333
Integral de 2x^3: 1160.000000
Integral de sin(x): -1.743895
[✓] run on [GNU/Linux] at [2024-08-11 16:39:56]
[✚] schr openmp-fd X ./remann_omp2.exe 0 1 2
Con n = 1000000 trapezoides y 2 threads, la aproximación de la integral de 0.000000 a 1.000000 es:
Integral de x^2: 0.333333
Integral de 2x^3: 0.500000
Integral de sin(x): 0.459698
[✓] run on [GNU/Linux] at [2024-08-11 16:40:04]
[✚] schr openmp-fd X [ ]

```

Figura 4: Ejecución de remann_omp2.c

5. Ejercicio 5 [riemann_omp_nocrit.c]

5.1. ¿Qué diferencia hay entre usar una variable global para añadir los resultados a un arreglo?

Diferencia principal:

Usar `#pragma omp critical`: Cada thread entra a una sección crítica para sumar su resultado parcial a una variable global compartida. Esto asegura que no ocurran condiciones de carrera, pero puede reducir la eficiencia si muchos threads tienen que esperar para entrar en la sección crítica.

Usar un arreglo global: Cada thread almacena su resultado parcial en un índice único de un arreglo global. Al final, un solo thread recorre el arreglo y suma los resultados parciales. Esto evita el uso de secciones críticas y puede ser más eficiente, ya que elimina la necesidad de sincronización entre los threads durante la escritura de los resultados parciales.

Ventajas del Enfoque sin `#pragma omp critical`:

- **Mayor rendimiento:** Al eliminar la necesidad de una sección crítica, se reduce la espera entre threads, lo que puede mejorar el rendimiento en sistemas con muchos threads.
- **Simplicidad en la suma de resultados:** Al sumar los resultados fuera de la sección paralela, se evita la sobrecarga asociada con la sincronización.

Este enfoque es útil cuando queremos maximizar la eficiencia en cálculos paralelos, especialmente en sistemas con un gran número de núcleos de procesamiento.

5.2. Evidencias de ejecución y resultados

```
☑ run on [GNU/Linux] at [2024-08-11 16:50:08]
➤ schr openmp-fd X gcc -fopenmp -o riemann_omp_nocrit.exe riemann_omp_nocrit.c -lm
☑ run on [GNU/Linux] at [2024-08-11 16:50:14]
➤ schr openmp-fd X ./riemann_omp_nocrit.exe 2 10 4
Con n = 1000000 trapezoides y 4 threads, la aproximación de la integral de 2.000000 a 10.000000 es:
Integral de x^2: 330.666667
Integral de 2x^3: 4992.000000
Integral de sin(x): 0.422925
☑ run on [GNU/Linux] at [2024-08-11 16:50:58]
➤ schr openmp-fd X ./riemann_omp_nocrit.exe 3 7 8
Con n = 1000000 trapezoides y 8 threads, la aproximación de la integral de 3.000000 a 7.000000 es:
Integral de x^2: 105.333333
Integral de 2x^3: 1160.000000
Integral de sin(x): -1.743895
☑ run on [GNU/Linux] at [2024-08-11 16:51:23]
➤ schr openmp-fd X ./riemann_omp_nocrit.exe 0 1 2
Con n = 1000000 trapezoides y 2 threads, la aproximación de la integral de 0.000000 a 1.000000 es:
Integral de x^2: 0.333333
Integral de 2x^3: 0.500000
Integral de sin(x): 0.459698
☑ run on [GNU/Linux] at [2024-08-11 16:51:27]
➤ schr openmp-fd X
```

Figura 5: Ejecución de riemann_omp_nocrit.c

6. Repositorio

<https://github.com/chamale-rac/movies-xd>

Referencias

[Ope] *Using OpenMP with C — Research Computing, University of Colorado Boulder documentation.* <https://curc.readthedocs.io/en/latest/programming/OpenMP-C.html>. Accessed: 2024-08-11. n.d.

Índice de figuras

1.	Ejecución de <code>hello_omp.c</code>	1
2.	Ejecución de <code>hbd_omp.c</code>	2
3.	Ejecución de <code>riemann.c</code>	3
4.	Ejecución de <code>remann_omp2.c</code>	4
5.	Ejecución de <code>riemann_omp_nocrit.c</code>	5