

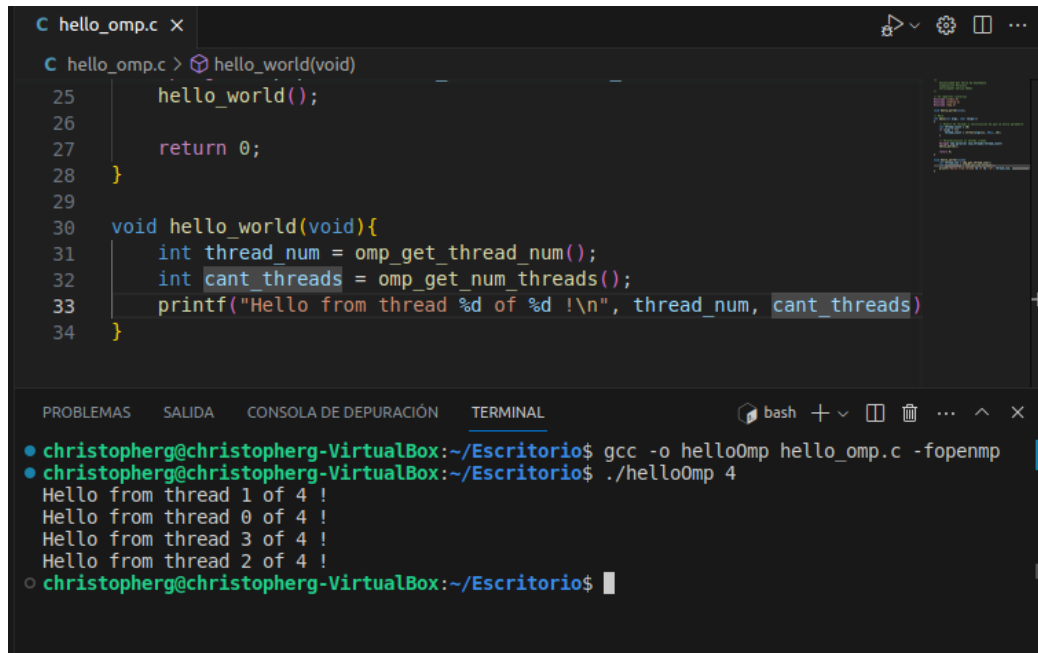
Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Computación Paralela y Distribuida
Docente: Sebastián Galindo
Hoja de trabajo#1



Christopher García 20541
Agosto, 2023

- Ejercicio 1 (10 puntos)



The screenshot shows a code editor with a file named `hello_omp.c`. The code defines a `hello_world(void)` function that uses OpenMP to create 4 threads. Each thread prints a message indicating its thread number and the total number of threads. The terminal output shows the program being compiled with `gcc -o helloOmp hello_omp.c -fopenmp` and then executed with `./helloOmp 4`. The output messages are interleaved, demonstrating non-deterministic execution order due to parallelism.

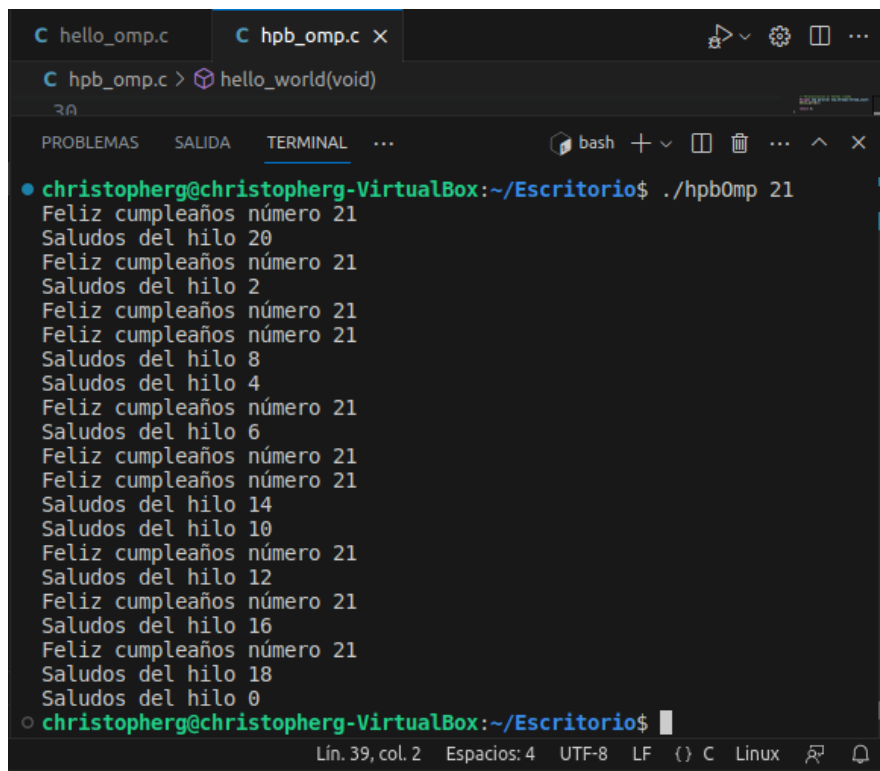
```
C hello_omp.c x
C hello_omp.c > hello_world(void)
25     hello_world();
26
27     return 0;
28 }
29
30 void hello_world(void){
31     int thread_num = omp_get_thread_num();
32     int cant_threads = omp_get_num_threads();
33     printf("Hello from thread %d of %d !\n", thread_num, cant_threads)
34 }
```

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
christopherg@christopherg-VirtualBox:~/Escritorio$ gcc -o helloOmp hello_omp.c -fopenmp
christopherg@christopherg-VirtualBox:~/Escritorio$ ./helloOmp 4
Hello from thread 1 of 4 !
Hello from thread 0 of 4 !
Hello from thread 3 of 4 !
Hello from thread 2 of 4 !
christopherg@christopherg-VirtualBox:~/Escritorio$
```

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

- Esto debido a la funcionalidad de paralelismo en sí, el hecho de correr el programa con múltiples hilos provoca una independencia en la ejecución de las tareas y como no existen métodos de sincronización aplicados al código es coherente que los mensajes no se desplieguen en orden.

- Ejercicio 2 (10 puntos)

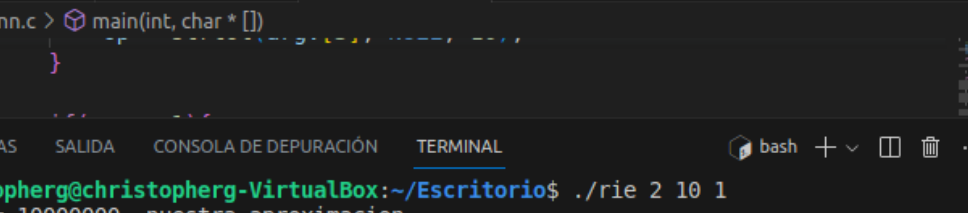


The screenshot shows a code editor with a file named `hpb_omp.c`. The code defines a `hello_world(void)` function that uses OpenMP to create 21 threads. Each thread prints a birthday greeting for a specific thread number. The terminal output shows the program being executed with `./hpbOmp 21`. The output messages are interleaved, demonstrating non-deterministic execution order due to parallelism.

```
C hello_omp.c  C hpb_omp.c x
C hpb_omp.c > hello_world(void)
25     hello_world();
26
27     return 0;
28 }
```

```
PROBLEMAS  SALIDA  TERMINAL  ...
christopherg@christopherg-VirtualBox:~/Escritorio$ ./hpbOmp 21
Feliz cumpleaños número 21
Saludos del hilo 20
Feliz cumpleaños número 21
Saludos del hilo 2
Feliz cumpleaños número 21
Feliz cumpleaños número 21
Saludos del hilo 8
Saludos del hilo 4
Feliz cumpleaños número 21
Saludos del hilo 6
Feliz cumpleaños número 21
Feliz cumpleaños número 21
Saludos del hilo 14
Saludos del hilo 10
Feliz cumpleaños número 21
Saludos del hilo 12
Feliz cumpleaños número 21
Saludos del hilo 16
Feliz cumpleaños número 21
Saludos del hilo 18
Saludos del hilo 0
christopherg@christopherg-VirtualBox:~/Escritorio$
```

Lín. 39, col. 2 Espacios: 4 UTF-8 LF {} C Linux

- 
- The screenshot shows a code editor with three tabs: `hello_omp.c`, `hpb_omp.c`, and `riemann.c`. The `riemann.c` tab is active, showing the `main` function. The code defines a function `rie` that calculates the Riemann sum for the integral of $f(x) = x^2$ from a to b using n rectangles. The `main` function calls `rie` with three different sets of parameters: $n=2, a=2, b=10$; $n=3, a=3, b=7$; and $n=0, a=0, b=1$. Below the code editor, the terminal window shows the output of these three calculations.
- ```
C riemann.c > main(int, char * [])
35 }
36
```
- PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL
- ```
christopherg@christopherg-VirtualBox:~/Escritorio$ ./rie 2 10 1
Con n = 10000000, nuestra aproximacion
de la integral de 2.000000 a 10.000000 es = 330.666667
christopherg@christopherg-VirtualBox:~/Escritorio$ ./rie 3 7 2
Con n = 10000000, nuestra aproximacion
de la integral de 3.000000 a 7.000000 es = 1160.000000
christopherg@christopherg-VirtualBox:~/Escritorio$ ./rie 0 1 3
Con n = 10000000, nuestra aproximacion
de la integral de 0.000000 a 1.000000 es = 0.459698
christopherg@christopherg-VirtualBox:~/Escritorio$
```

(Para este ejercicio yo envié un tercer parámetro que especifica la función a utilizar, 1, 2 y 3 representan los ejercicios solicitados respectivamente)

- ```
C hello_omp.c C hpb_omp.c C riemann.c C riemann_omp2.c X
```

```
C riemann_omp2.c > main(int, char * [])
38 if(n % num_threads != 0){
39 fprintf(stderr, "Número de trapezoides no es múltiplo de n
40 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
christopherg@christopherg-VirtualBox:~/Escritorio$./rie2 2 10 1 1
Con n = 10000000, nuestra aproximacion
de la integral de 2.000000 a 10.000000 es = 330.666667
christopherg@christopherg-VirtualBox:~/Escritorio$./rie2 3 7 1 2
Con n = 10000000, nuestra aproximacion
de la integral de 3.000000 a 7.000000 es = 1160.000000
christopherg@christopherg-VirtualBox:~/Escritorio$./rie2 0 1 1 3
Con n = 10000000, nuestra aproximacion
de la integral de 0.000000 a 1.000000 es = 0.459698
christopherg@christopherg-VirtualBox:~/Escritorio$./rie2 2 10 10 1
Con n = 10000000, nuestra aproximacion
de la integral de 2.000000 a 10.000000 es = 1.024793
christopherg@christopherg-VirtualBox:~/Escritorio$./rie2 3 7 10 2
Con n = 10000000, nuestra aproximacion
de la integral de 3.000000 a 7.000000 es = 9.151886
christopherg@christopherg-VirtualBox:~/Escritorio$./rie2 0 1 10 3
Con n = 10000000, nuestra aproximacion
de la integral de 0.000000 a 1.000000 es = 0.162512
christopherg@christopherg-VirtualBox:~/Escritorio$
```

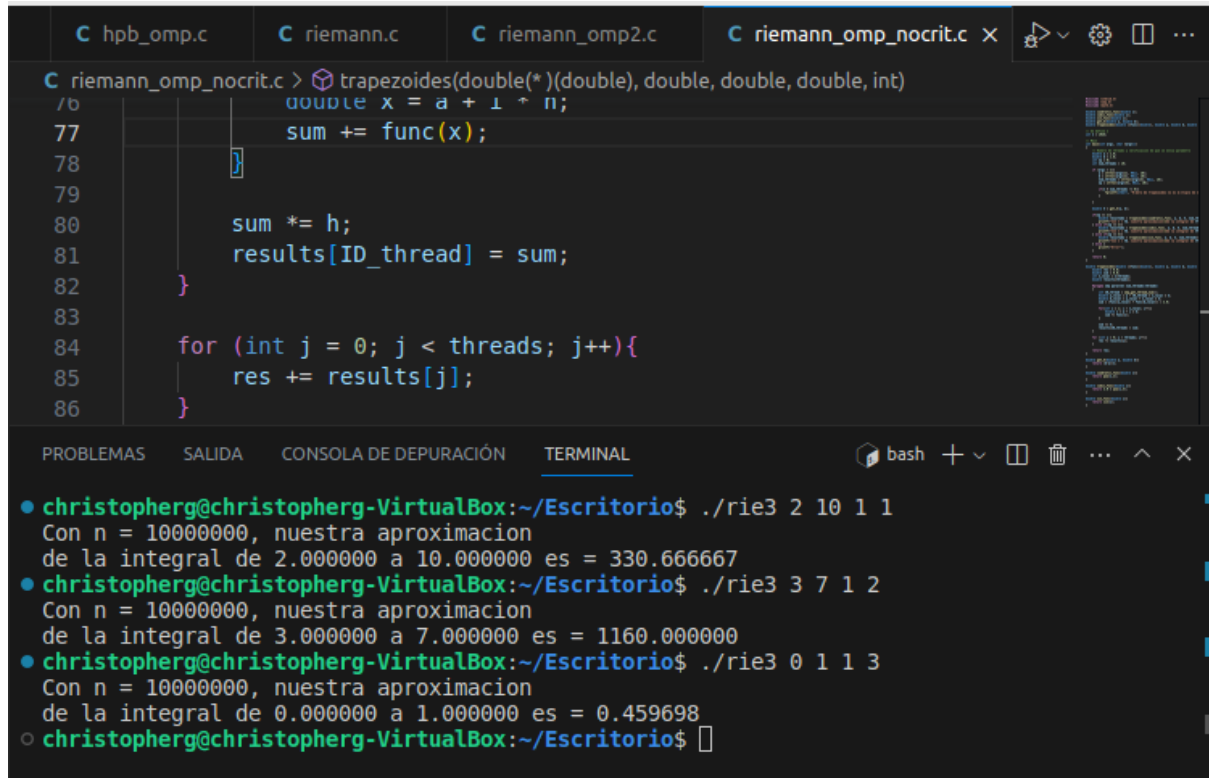
0 0 0 Lin. 46, col. 24 Espacios: 4 UTF-8 LF {} C Linux

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

- El uso de esta directiva evita los “race conditions” ya que el punto donde se coloca es cuando se agrega parte de la integral a la variable global y esto en el ambiente paralelo se convierte en un punto donde múltiples hilos podrían intentar acceder a dicha variable y modificarla causando incongruencias en el resultado final. Es por

esto que es tan importante el uso de esta directiva en este punto ya que permite que cada hilo acceda a la parte crítica uno por uno mientras los demás esperaban agregando un nivel de sincronización que, como ya mencioné, evita errores en el resultado por la concurrencia a la variable global.

#### Ejercicio 5



```
C riemann_omp_nocrit.c > trapezoides(double (*)(double), double, double, double, int)
76 double x = a + 1 * h;
77 sum += func(x);
78
79
80 sum *= h;
81 results[ID_thread] = sum;
82 }
83
84 for (int j = 0; j < threads; j++){
85 res += results[j];
86 }
```

```
christopherg@christopherg-VirtualBox:~/Escritorio$./rie3 2 10 1 1
Con n = 10000000, nuestra aproximacion
de la integral de 2.000000 a 10.000000 es = 330.666667
christopherg@christopherg-VirtualBox:~/Escritorio$./rie3 3 7 1 2
Con n = 10000000, nuestra aproximacion
de la integral de 3.000000 a 7.000000 es = 1160.000000
christopherg@christopherg-VirtualBox:~/Escritorio$./rie3 0 1 1 3
Con n = 10000000, nuestra aproximacion
de la integral de 0.000000 a 1.000000 es = 0.459698
christopherg@christopherg-VirtualBox:~/Escritorio$
```

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

- La gestión de los resultados de cada hilo. Como ya mencioné en la anterior pregunta, la directiva evita la concurrencia a la variable global y asigna un turno a cada hilo para ir operando los datos sin que ocurran inconsistencias. El utilizar un arreglo de cierta manera mitiga esta situación ya que se genera una estructura donde cada hilo posee un espacio destinado para almacenar su resultado evitando que un hilo pueda interferir en el resultado del otro y al final de todo esto se realiza la suma de los resultados del arreglo asegurando una consistencia en los datos y el resultado final.

Repositorio de Github: [https://github.com/ChristopherG19/UVG\\_Paralela\\_HT1.git](https://github.com/ChristopherG19/UVG_Paralela_HT1.git)