David Martos
Mariona Farré
Grupo: paco1208
Curso: 2022-23

PACO:
Laboratorio 2

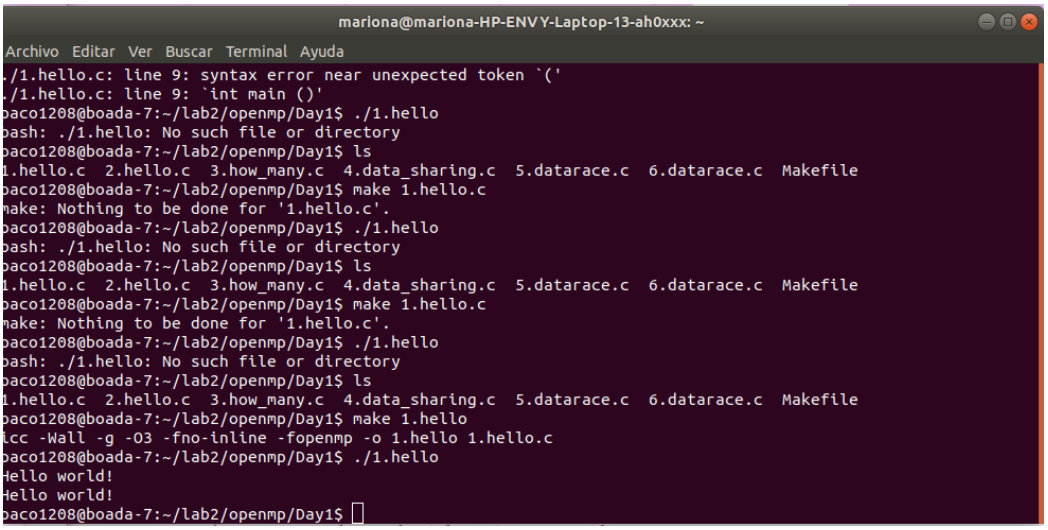
3.1 OpenMP questionnaire

When answering to the questions in this questionnaire, please DO NOT simply answer with yes, no or a number; try to minimally justify all your answers and if necessary include any code fragment you need to support your answer. Sometimes you may need to execute several times in order to see the effect of data races in the parallel execution.

Day 1: Parallel regions and implicit tasks

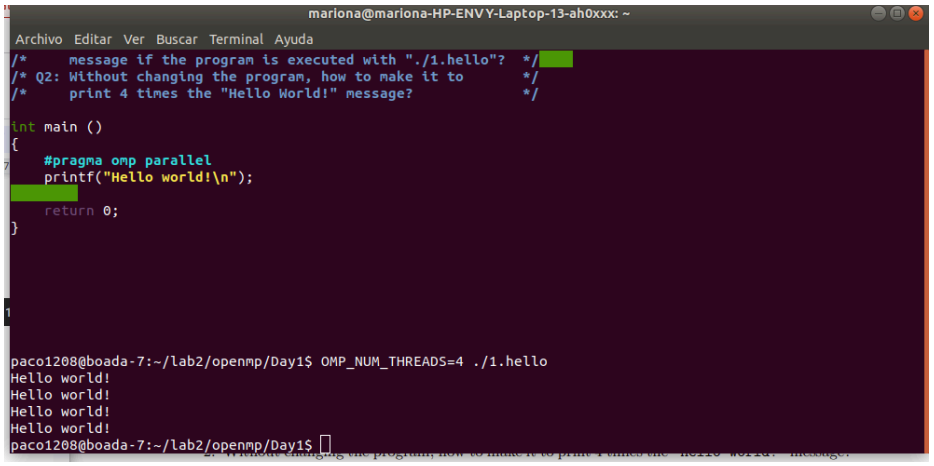
1.hello.c

1. How many times will you see the "Hello world!" message if the program is executed with "./1.hello"?



Por defecto el numero de hiles es 2 por lo que vemos 2 salidas.

2. Without changing the program, how to make it to print 4 times the "Hello World!" message?



Si modificamos el número por defecto de hilos, podemos cambiarlo a nuestro gusto, esto lo haremos con OMP_NUM_THREADS=XXXXX donde XXXXX es el número de hilos que vamos a usar y por ende el número de Hello Worlds que veremos por pantalla.

2.hello.c:

1. Is the execution of the program correct? (i.e., prints a sequence of "(Thid) Hello (Thid) world!" being Thid the thread identifier). If not, add a data sharing clause to make it correct?

No se ejecuta de manera correcta, ya que el thread identifier de cada secuencia “Hello” y “world!” no coinciden la mayoría de las veces y en cada ejecución el número de threads va variando.

```
paco1208@boada-7:~/lab2/openmp/Day1$ make 2.hello
icc -Wall -g -O3 -fno-inline -fopenmp -o 2.hello 2.hello.c
paco1208@boada-7:~/lab2/openmp/Day1$ ./2.hello
(3) Hello (0) world!
(1) Hello (1) world!
(0) Hello (1) world!
(2) Hello (2) world!
(4) Hello (4) world!
(5) Hello (5) world!
(6) Hello (6) world!
(7) Hello (7) world!
paco1208@boada-7:~/lab2/openmp/Day1$
```

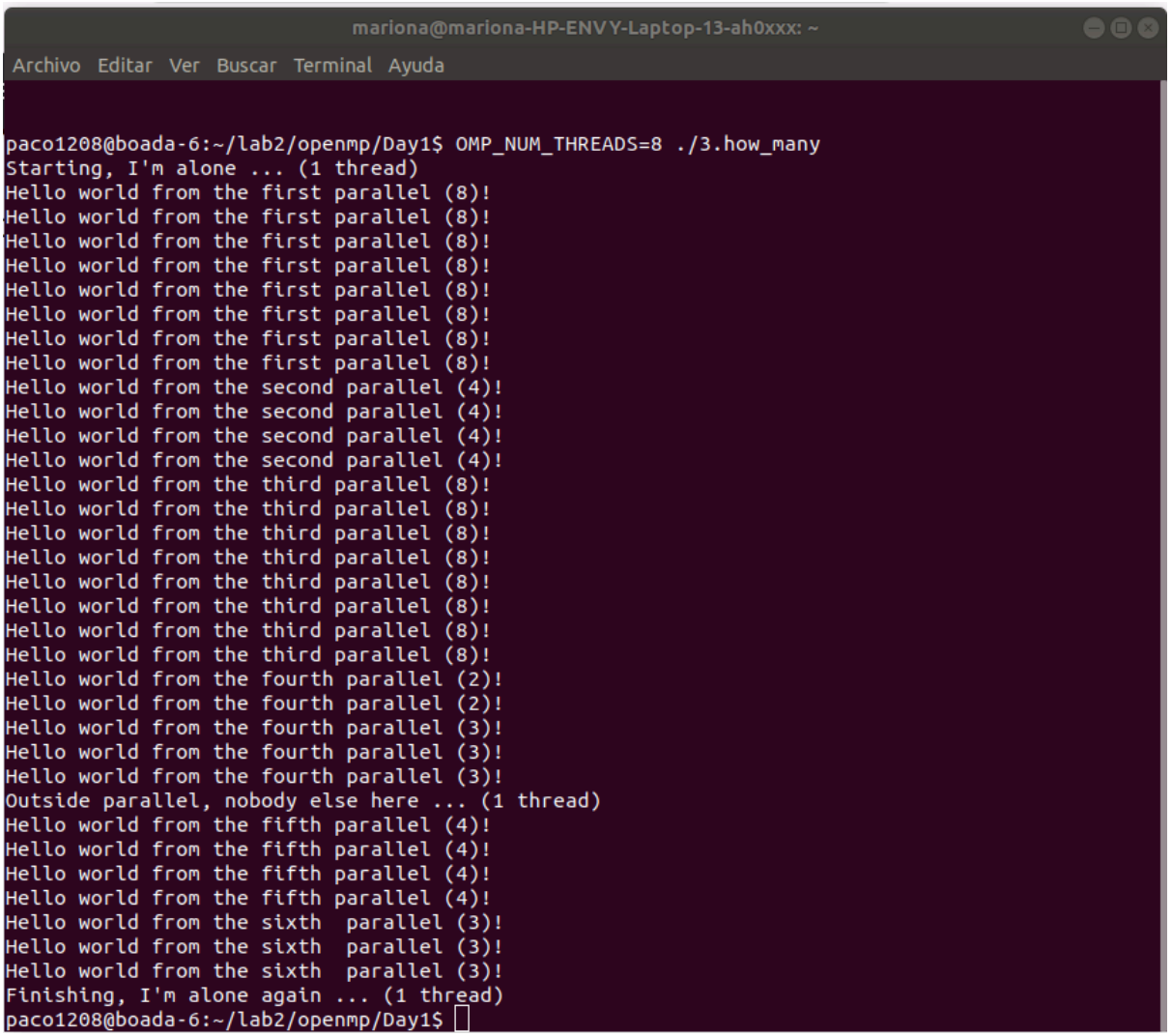
```
paco1208@boada-7:~/lab2/openmp/Day1$ ./2.hello
(0) Hello (1) world!
(1) Hello (1) world!
(2) Hello (2) world!
(3) Hello (3) world!
(7) Hello (7) world!
(6) Hello (6) world!
(4) Hello (4) world!
(5) Hello (5) world!
paco1208@boada-7:~/lab2/openmp/Day1$
```

Para que se ejecute bien se debe modificar el código:

2. Are the lines always printed in the same order? Why the messages sometimes appear intermixed? (Execute several times in order to see this).

Eso es debido a que la carga del procesador no es siempre la misma, por lo que la posición de cada mensaje puede variar.

3.how many.c:
Assuming the OMP NUM THREADS variable is set to 8 with "OMP NUM THREADS=8 ./3.how many"



1. What does omp get num threads return when invoked outside and inside a parallel region?

Cuando es invocado a fuera de la región paralelizada retorna el número de threads que se están usando que es 1 thread, mientras que cuando está en la zona paralelizada este número va variando por su ejecución y también porque se inicializa varias veces a 4 threads.

2. Indicate the two alternatives to supersede the number of threads that is specified by the OMP NUM THREADS environment variable.

Incluir en el código la línea:
pragma omp parallel num_threads(XXXX)

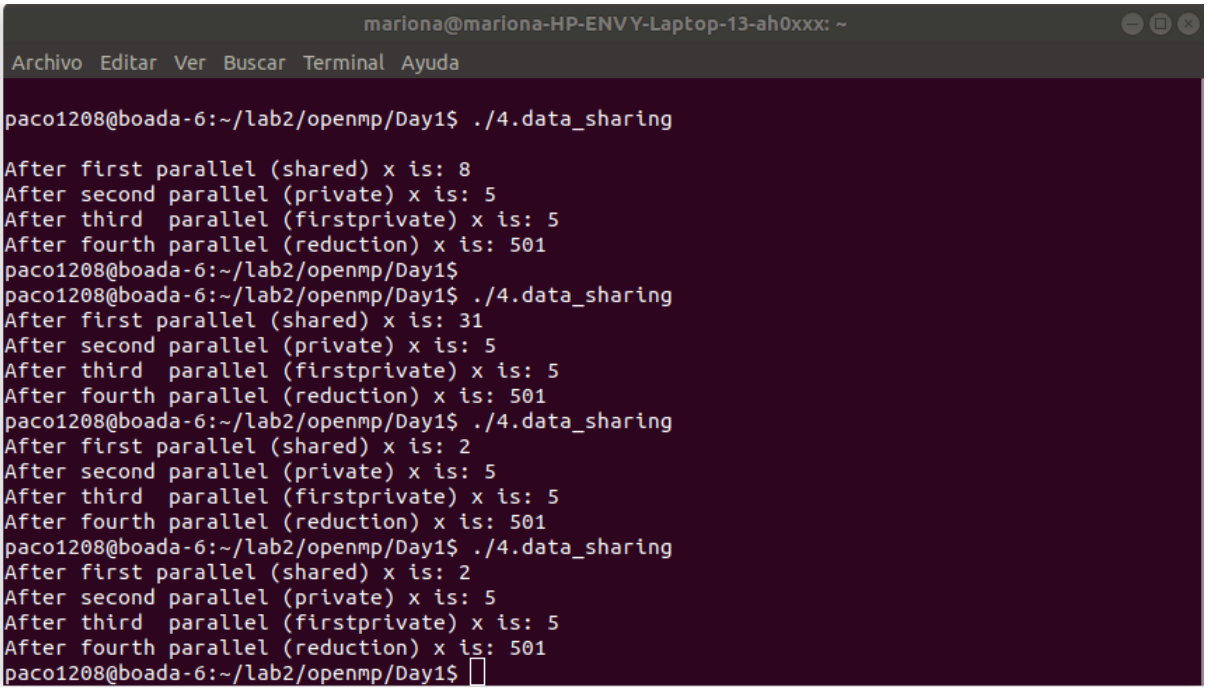
O bien realizar la llamada al ejecutable pasandole la variable OMP_NUM_THREADS=XXXX (tal y como podemos observar en la captura de pantalla).

3. Which is the lifespan for each way of defining the number of threads to be used?

En el caso de la variable global, afecta a todo el código, excepto en aquellas regiones donde se le aplique la línea de pragma omp parallel num_threads(XXX). En cambio, al aplicarlo sobre el código, solamente afectará al tramo de código que esté entre los corchetes.

4.data sharing.c

1. Which is the value of variable x after the execution of each parallel region with different data sharing attribute (shared, private, firstprivate and reduction)? Is that the value you would expect? (Execute several times if necessary)



Si la variable es shared, es una variable que dentro de la constructora es la misma que la de fuera de la constructora.
Si la variable es private quiere decir que la variable dentro de la constructora es una nueva variable del mismo tipo con un valor indefinido.
Si la variable es firstprivate es una variable que dentro de la constructora es una variable nueva del mismo tipos però inicializada con el valor original de la variable.
Si la variable es reduction es donde todos los threads se acumulan los valores dentro de una sola variable, al final de una región, el compilador asegura que las variables shared son actualizadas correctamente con los valores parciales de cada thread.
Entonces es normal que con diferentes ejecuciones las variables shared puedan cambiar de valor, mientras que las variables private, firstprivate y reduction mantengan el mismo valor.

5.datarace.c

1. Should this program always return a correct result? Reason either your positive or negative answer.

```
mariona@mariona-HP-ENVY-Laptop-13-ah0xxx: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
paco1208@boada-6:~/lab2/openmp/Day1$ make 5.datarace
make: '5.datarace' is up to date.
paco1208@boada-6:~/lab2/openmp/Day1$ ./5.datarace
Sorry, something went wrong - incorrect maxvalue=10 found
paco1208@boada-6:~/lab2/openmp/Day1$
```

No es factible que el programa de un resultado correcto, ya que los threads no comparten la memoria y el valor de i no será correcto.

2. Propose two alternative solutions to make it correct, without changing the structure of the code (just add directives or clauses). Explain why they make the execution correct.

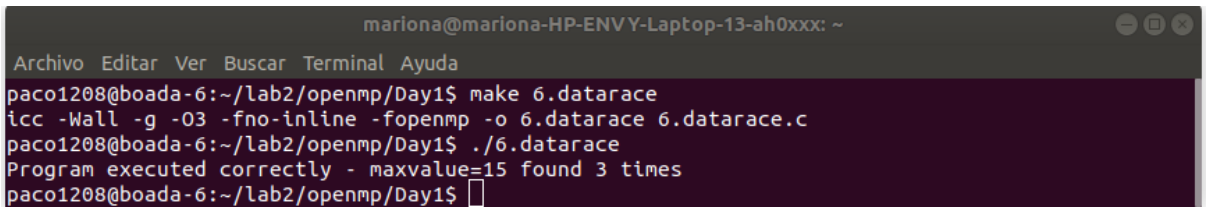
```
mariona@mariona-HP-ENVY-Laptop-13-ah0xxx: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
/*     execution correct.                                    */
/* Q3: Write an alternative distribution of iterations to    */
/*     implicit tasks (threads) so that each of them executes */
/*     only one block of consecutive iterations (i.e. N      */
/*     divided by the number of threads.                     */

#define N 1 << 20
int vector[N]={0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 15, 14, 13, 12, 11, 10, 9, 8, 15, 15};

int main()
{
    int i, maxvalue=3;

    omp_set_num_threads(8);
    #pragma omp parallel private(i)
    {
       int id = omp_get_thread_num();
       int howmany = omp_get_num_threads();

       for (i=id; i < N; i+=howmany) {
      #pragma omp critical
           if (vector[i] > maxvalue)
          {
             sleep(1); // this is just to force problems
             maxvalue = vector[i];


paco1208@boada-6:~/lab2/openmp/Day1$ make 5.datarace
icc -Wall -g -O3 -fno-inline -fopenmp -o 5.datarace 5.datarace.c
paco1208@boada-6:~/lab2/openmp/Day1$ ./5.datarace
Program executed correctly - maxvalue=15 found
paco1208@boada-6:~/lab2/openmp/Day1$
```

Si ponemos un pragma omp critical después del for, podemos crear una región de exclusión mutua donde solo un thread puede estar siendo ejecutado a ese tiempo, dando así el espacio necesario para que el programa funcione.

3. Write an alternative distribution of iterations to implicit tasks (threads) so that each of them executes only one block of consecutive iterations (i.e. N divided by the number of threads).

6.datarace.c

1. Should this program always return a correct result? Reason either your positive or negative answer.



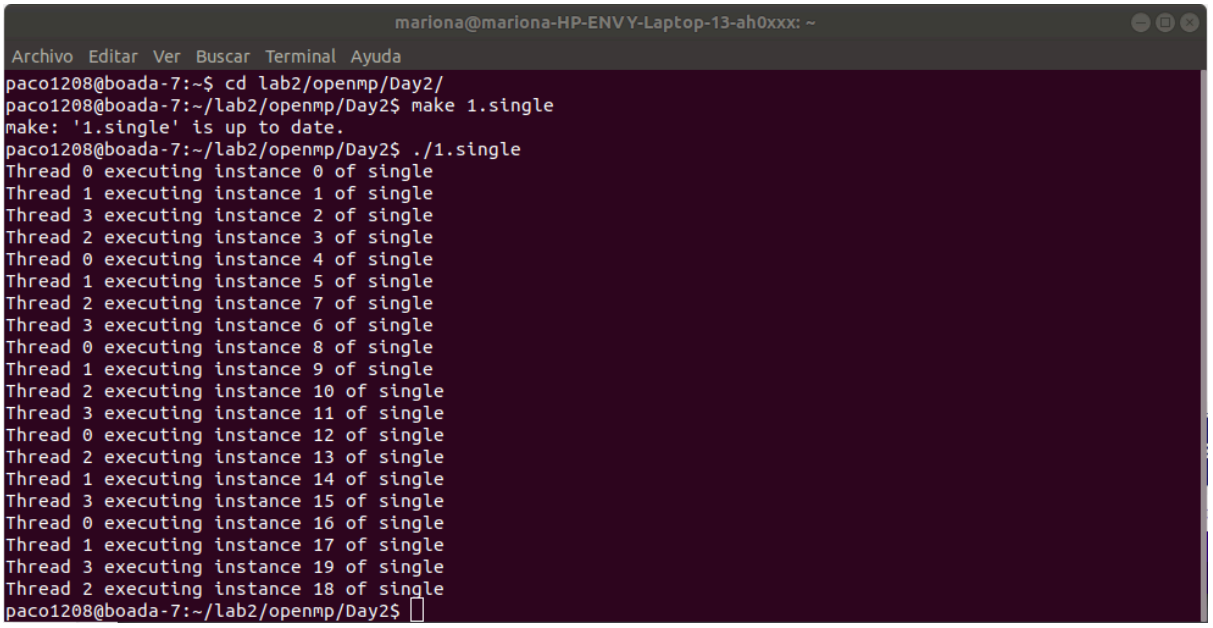
En este programa sí que en cada ejecución se hará correctamente, ya que el programa irá mirando por to

2. Propose two alternative solutions to make it correct, without changing the structure of the program (just using directives or clauses) and never making use of critical. Explain why they make the execution correct.

Day 2: explicit tasks

1.single.c

1. What is the nowait clause doing when associated to single?



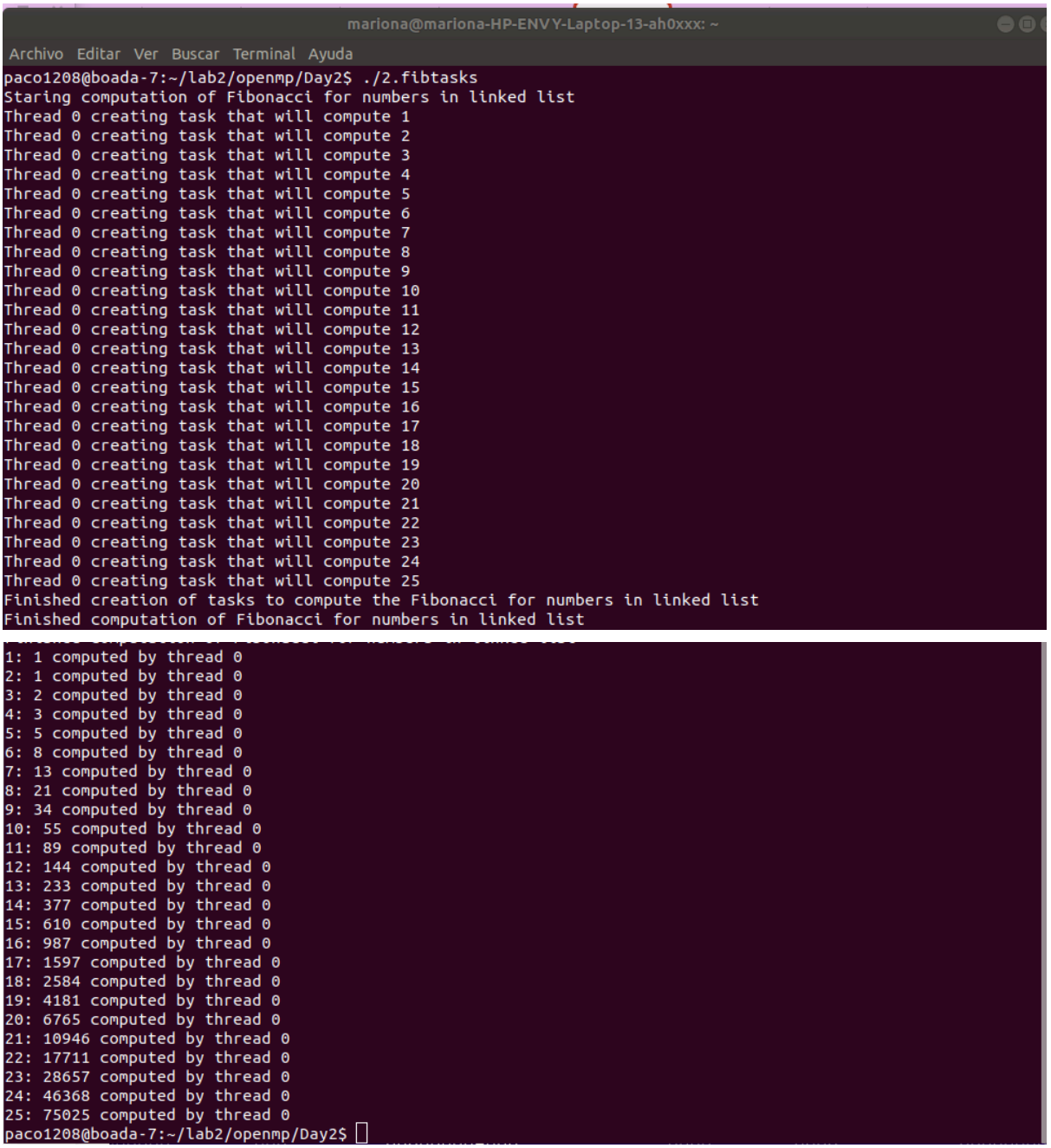
Indica que la región es independiente, por lo que se puede sobreponer a otra region single (de un solo hilo).

2. Then, can you explain why all threads contribute to the execution of the multiple instances of single? Why those instances appear to be executed in bursts?

Porque tenemos solo 4 hilos disponibles, con lo que se ejecutan a la vez solo 4 procesos single threaded, si cambiáramos a N procesadores, se ejecutarán todos a la vez, ya que se generarían N hilos single threated.

2.fibtasks.c

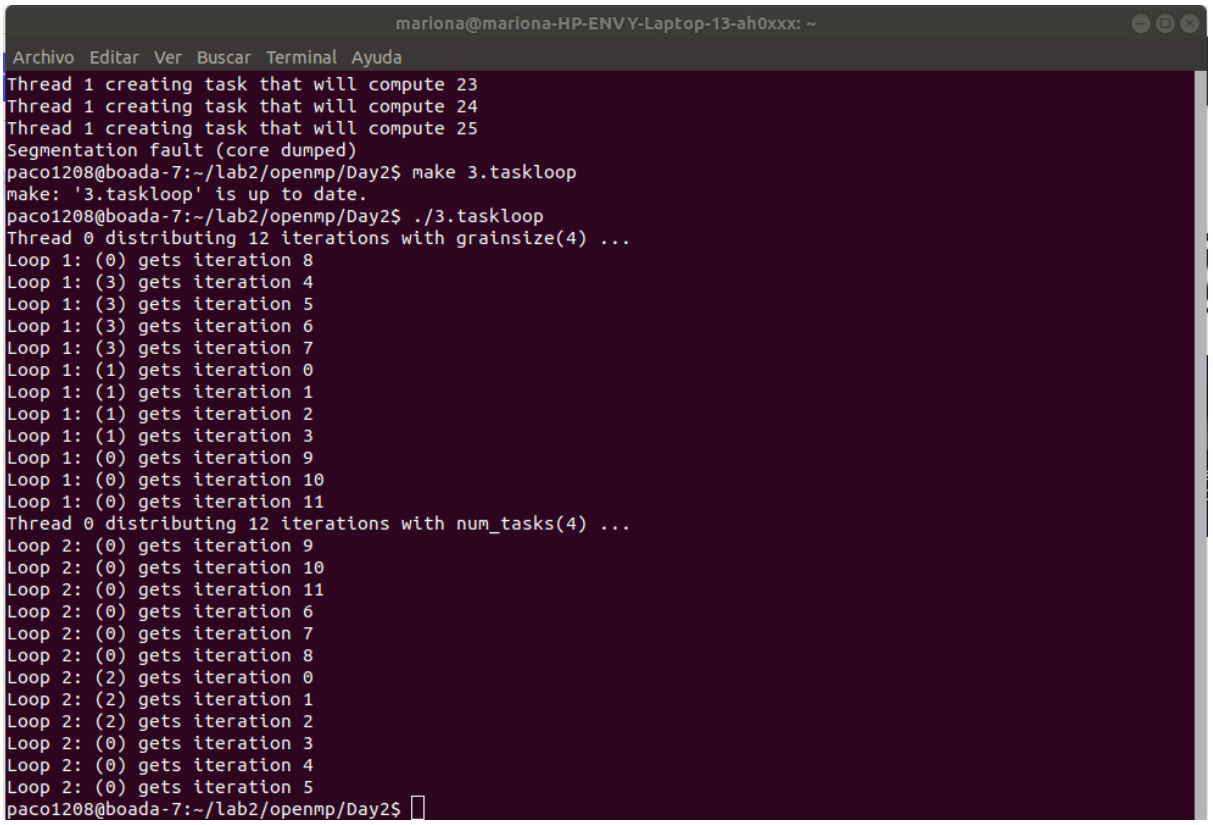
1. Why all tasks are created and executed by the same thread? In other words, why the program is not executing in parallel?



En el código no puede hacer ejecución en paralelo ya que no tiene ninguna instrucción de paralelización en el código, así que su ejecución es secuencial con un solo thread.

3.taskloop.c

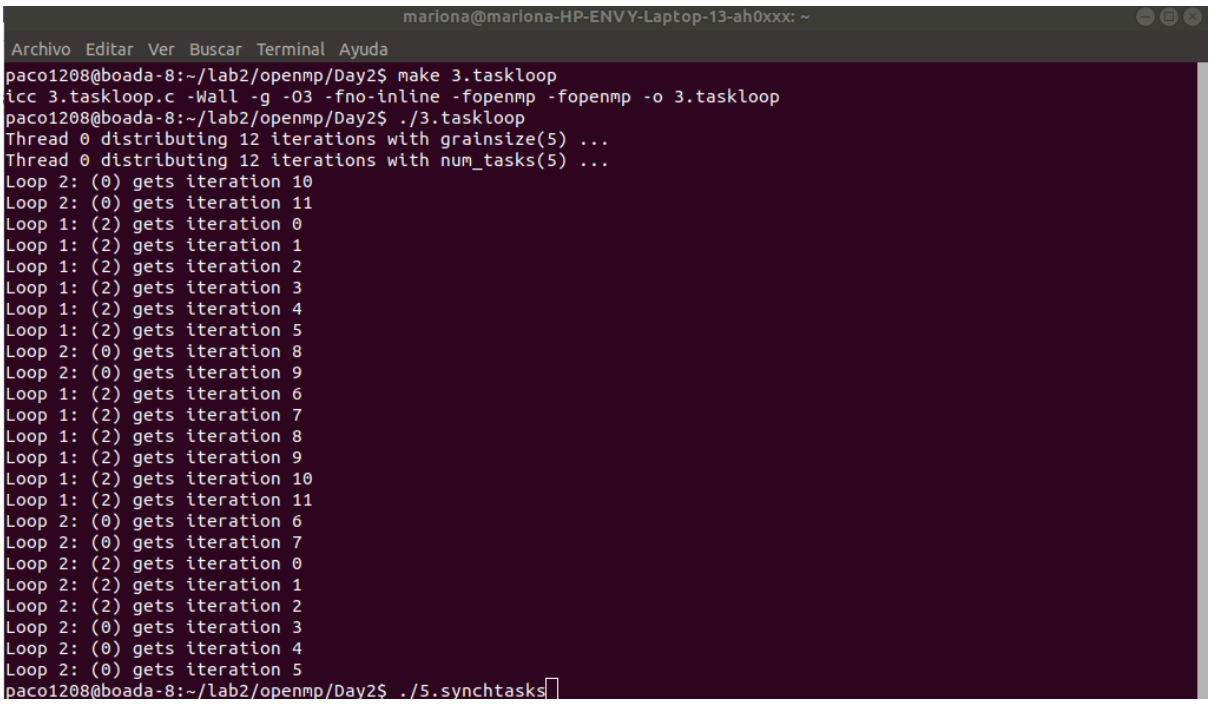
1. Which iterations of the loops are executed by each thread for each task grainsize or num tasks specified?



El hilo que ejecuta el código viene determinado por el grain size, no el número de tareas.

2. Change the value for grainsize and num tasks to 5. How many iterations is now each thread executing? How is the number of iterations decided in each case?

```c
#define VALUE 5

int main()
{
    int i;

    omp_set_num_threads(4);
    #pragma omp parallel
    #pragma omp single
    {
        printf("Thread %d distributing %d iterations with grainsize(%d) ...\n", omp_get_thread_num(), N, VALUE);
        #pragma omp taskloop grainsize(VALUE) nogroup
        for (i=0; i < N; i++) {
                printf("Loop 1: (%d) gets iteration %d\n", omp_get_thread_num(), i);
        }

        printf("Thread %d distributing %d iterations with num_tasks(%d) ...\n", omp_get_thread_num(), N, VALUE);
        #pragma omp taskloop num_tasks(VALUE)
        for (i=0; i < N; i++) {
                printf("Loop 2: (%d) gets iteration %d\n", omp_get_thread_num(), i);
        }
    }

    return 0;
}
```



3. Can grain size and num tasks be used at the same time in the same loop?

Si, ya que el grain size no afecta en el número de el bucle.

4.reduction.c

1. Complete the parallelisation of the program so that the correct value for variable sum is returned in each printf statement. Note: in each part of the 3 parts of the program, all tasks generated should potentially execute in parallel.

Antes de paralelizar el código: