

2º curso / 2º cuatr.
Grado Ing. Inform.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 3. Programación paralela III: Interacción con el entorno en OpenMP

Estudiante (nombre y apellidos): Guillermo Sandoval Schmidt

Grupo de prácticas: C3

Fecha de entrega:

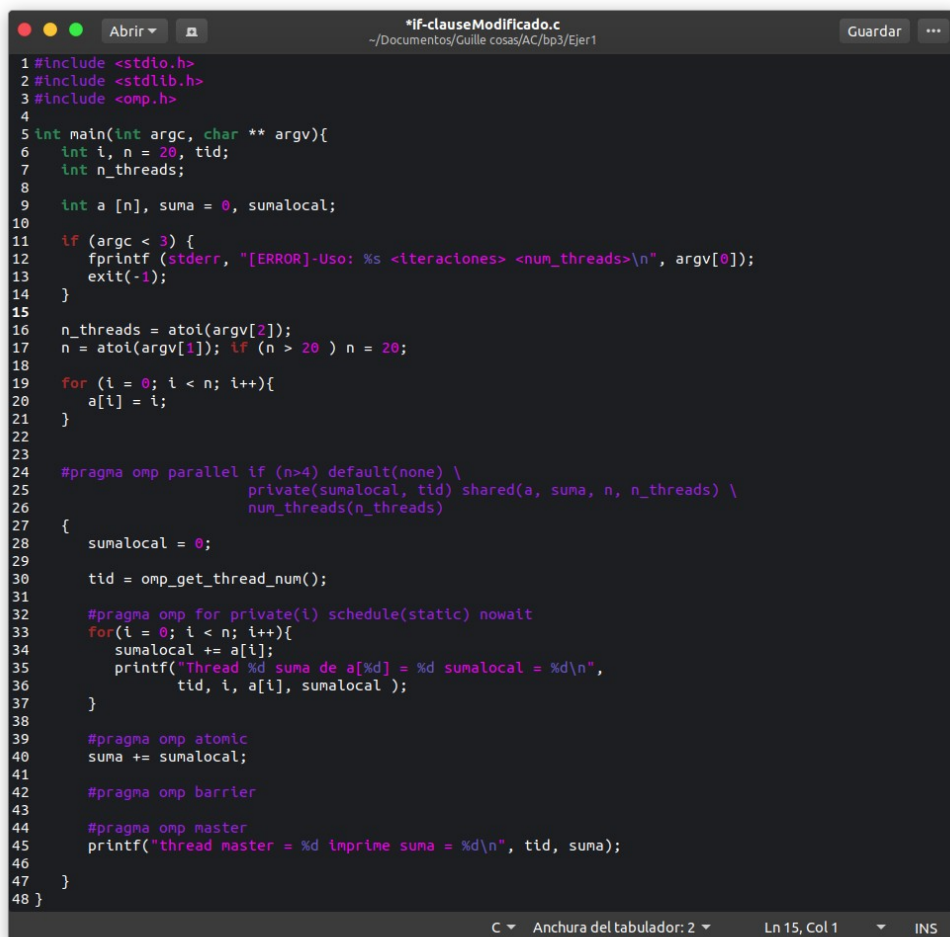
Fecha evaluación en clase:

Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con los normas de prácticas que se encuentra en SWAD

Ejercicios basados en los ejemplos del seminario práctico

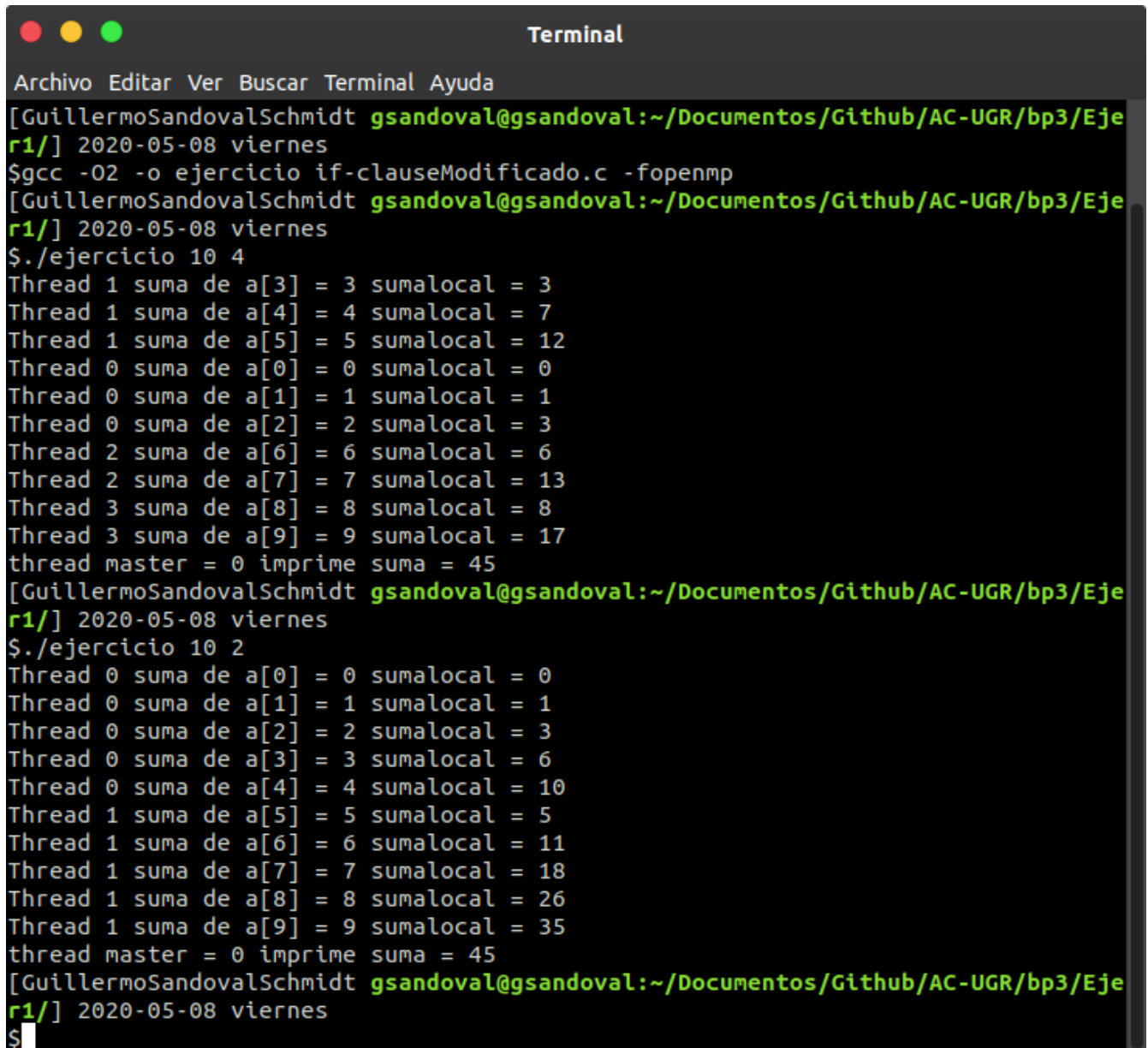
1. Usar la cláusula `num_threads(x)` en el ejemplo del seminario `if_clause.c`, y añadir un parámetro de entrada al programa que fije el valor `x` que se va a usar en la cláusula. Incorporar en el cuaderno de trabajo de esta práctica volcados de pantalla con ejemplos de ejecución que ilustren la funcionalidad de esta cláusula y explicar por qué lo ilustran.

CAPTURA CÓDIGO FUENTE: `if-clauseModificado.c`



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 int main(int argc, char ** argv){
6     int i, n = 20, tid;
7     int n_threads;
8
9     int a [n], suma = 0, sumalocal;
10
11     if (argc < 3) {
12         fprintf (stderr, "[ERROR]-Uso: %s <iteraciones> <num_threads>\n", argv[0]);
13         exit(-1);
14     }
15
16     n_threads = atoi(argv[2]);
17     n = atoi(argv[1]); if (n > 20 ) n = 20;
18
19     for (i = 0; i < n; i++){
20         a[i] = i;
21     }
22
23
24     #pragma omp parallel if (n>4) default(none) \
25         private(sumalocal, tid) shared(a, suma, n, n_threads) \
26         num_threads(n_threads)
27     {
28         sumalocal = 0;
29
30         tid = omp_get_thread_num();
31
32         #pragma omp for private(i) schedule(static) nowait
33         for(i = 0; i < n; i++){
34             sumalocal += a[i];
35             printf("Thread %d suma de a[%d] = %d sumalocal = %d\n",
36                 tid, i, a[i], sumalocal );
37         }
38
39         #pragma omp atomic
40         suma += sumalocal;
41
42         #pragma omp barrier
43
44         #pragma omp master
45         printf("thread master = %d imprime suma = %d\n", tid, suma);
46     }
47 }
48 }
```

CAPTURAS DE PANTALLA:



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r1/] 2020-05-08 viernes
$gcc -O2 -o ejercicio if-clauseModificado.c -fopenmp
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r1/] 2020-05-08 viernes
$./ejercicio 10 4
Thread 1 suma de a[3] = 3 sumalocal = 3
Thread 1 suma de a[4] = 4 sumalocal = 7
Thread 1 suma de a[5] = 5 sumalocal = 12
Thread 0 suma de a[0] = 0 sumalocal = 0
Thread 0 suma de a[1] = 1 sumalocal = 1
Thread 0 suma de a[2] = 2 sumalocal = 3
Thread 2 suma de a[6] = 6 sumalocal = 6
Thread 2 suma de a[7] = 7 sumalocal = 13
Thread 3 suma de a[8] = 8 sumalocal = 8
Thread 3 suma de a[9] = 9 sumalocal = 17
thread master = 0 imprime suma = 45
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r1/] 2020-05-08 viernes
$./ejercicio 10 2
Thread 0 suma de a[0] = 0 sumalocal = 0
Thread 0 suma de a[1] = 1 sumalocal = 1
Thread 0 suma de a[2] = 2 sumalocal = 3
Thread 0 suma de a[3] = 3 sumalocal = 6
Thread 0 suma de a[4] = 4 sumalocal = 10
Thread 1 suma de a[5] = 5 sumalocal = 5
Thread 1 suma de a[6] = 6 sumalocal = 11
Thread 1 suma de a[7] = 7 sumalocal = 18
Thread 1 suma de a[8] = 8 sumalocal = 26
Thread 1 suma de a[9] = 9 sumalocal = 35
thread master = 0 imprime suma = 45
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r1/] 2020-05-08 viernes
$
```

RESPUESTA:

Podemos observar como al utilizar la clausula, utiliza un número de hebras en función del parámetro introducido.

2. (a) Rellenar la Tabla 1 (se debe poner en la tabla el id del *thread* que ejecuta cada iteración) ejecutando los ejemplos del seminario `schedule-clause.c`, `scheduled-clause.c` y `scheduleg-clause.c` con dos *threads* (0,1) y unas entradas de:

- iteraciones: 16 (0,...15)
- chunk= 1, 2 y 4

Tabla 1 . Tabla schedule. En la segunda fila, 1, 2 4 representan el tamaño del chunk (consulte seminario)

Iteración	schedule-clause.c			schedule-clause.d.c			schedule-clauseg.c		
	1	2	4	1	2	4	1	2	4
0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0
2	0	1	0	0	1	0	0	0	0
3	1	1	0	0	1	0	0	0	0
4	0	0	1	0	0	1	0	0	0
5	1	0	1	0	0	1	0	0	0
6	0	1	1	0	0	1	0	0	0
7	1	1	1	0	0	1	0	0	0
8	0	0	0	0	0	0	1	1	1
9	1	0	0	0	0	0	1	1	1
10	0	1	0	0	0	0	1	1	1
11	1	1	0	0	0	0	1	1	1
12	0	0	1	0	0	0	1	0	0
13	1	0	1	0	0	0	1	0	0
14	0	1	1	0	0	0	1	0	0
15	1	1	1	0	0	0	1	0	0

(b) Rellenar otra tabla como la de la figura pero esta vez usando cuatro *threads* (0,1,2,3).

Tabla 2 . Tabla schedule. En la segunda fila, 1, 2 4 representan el tamaño del chunk (consulte seminario)

Iteración	schedule- clause.c			schedule- claused.c			schedule- clauseg.c		
	1	2	4	1	2	4	1	2	4
0	0	0	0	2	0	2	0	1	3
1	1	0	0	1	0	2	0	1	3
2	2	1	0	3	1	2	0	1	3
3	3	1	0	0	1	2	0	1	3
4	0	2	1	0	2	3	1	2	1
5	1	2	1	0	2	3	1	2	1
6	2	3	1	0	3	3	1	2	1
7	3	3	1	0	3	3	2	3	1
8	0	0	2	0	0	1	2	3	2
9	1	0	2	0	0	1	2	3	2
10	2	1	2	3	0	1	3	0	2
11	3	1	2	3	0	1	3	0	2
12	0	2	3	3	0	0	0	0	0
13	1	2	3	3	0	0	0	0	0
14	2	3	3	0	0	0	0	0	0
15	3	3	3	1	0	0	0	0	0

Escriba en el cuaderno de prácticas las diferencias en el comportamiento de `schedule()` con `static`, `dynamic` y `guided`.

RESPUESTA:

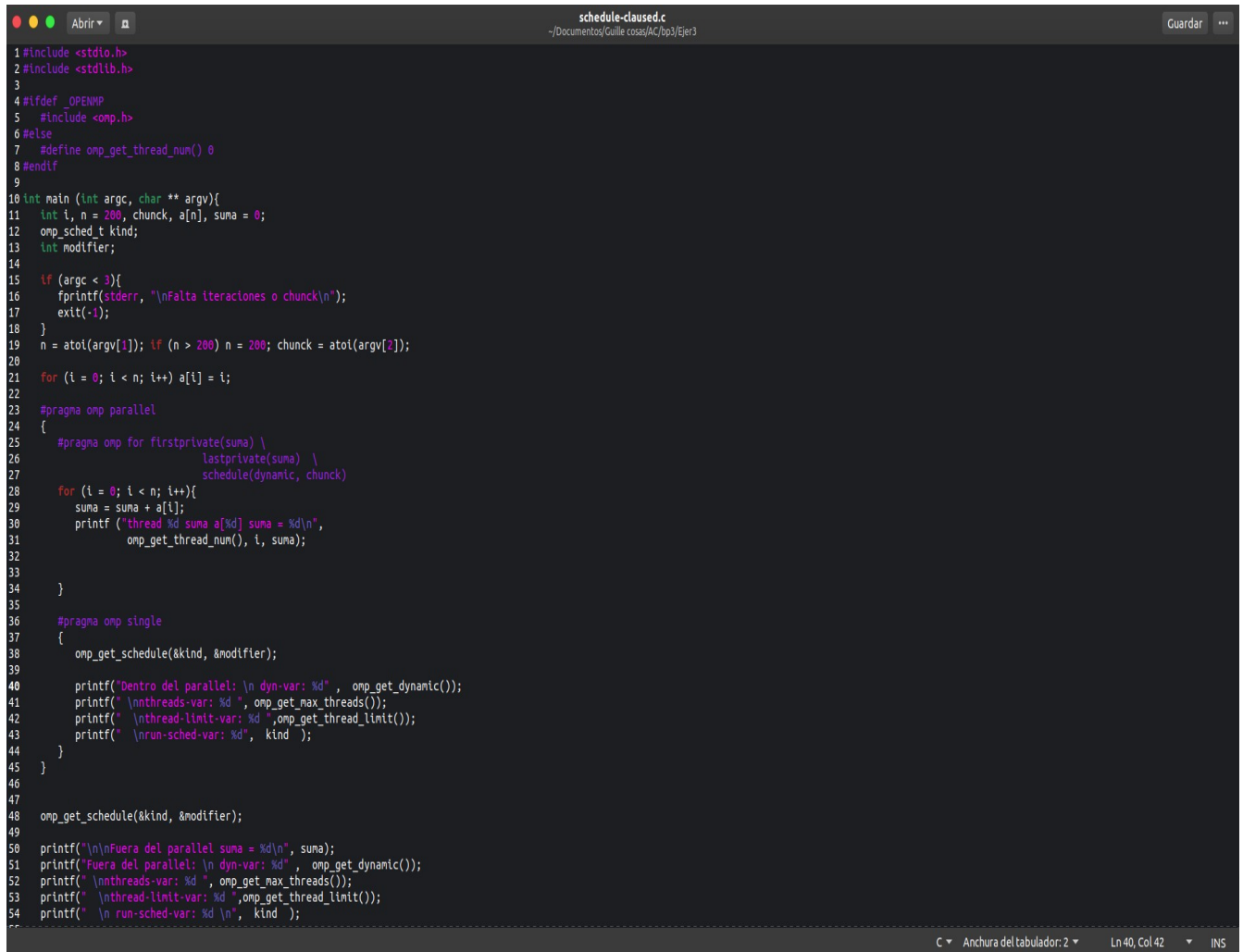
Static: reparte en tiempo de compilación el trabajo realizado entre los núcleos

Dynamic: reparte en tiempo de ejecución en función del trabajo que estén realizando los cores

Guided: se hace el reparto de manera dinamica pero el chunk es de tamaño variable, siempre siendo mayor o igual al valor dado

3. Añadir al programa `scheduled-clause.c` lo necesario para que imprima el valor de las variables de control `dyn-var`, `nthreads-var`, `thread-limit-var` y `run-sched-var` dentro (debe imprimir sólo un thread) y fuera de la región paralela. Realizar varias ejecuciones usando variables de entorno para modificar estas variables de control antes de la ejecución. Incorporar en su cuaderno de prácticas volcados de pantalla de estas ejecuciones. ¿Se imprimen valores distintos dentro y fuera de la región paralela?

CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado.c`

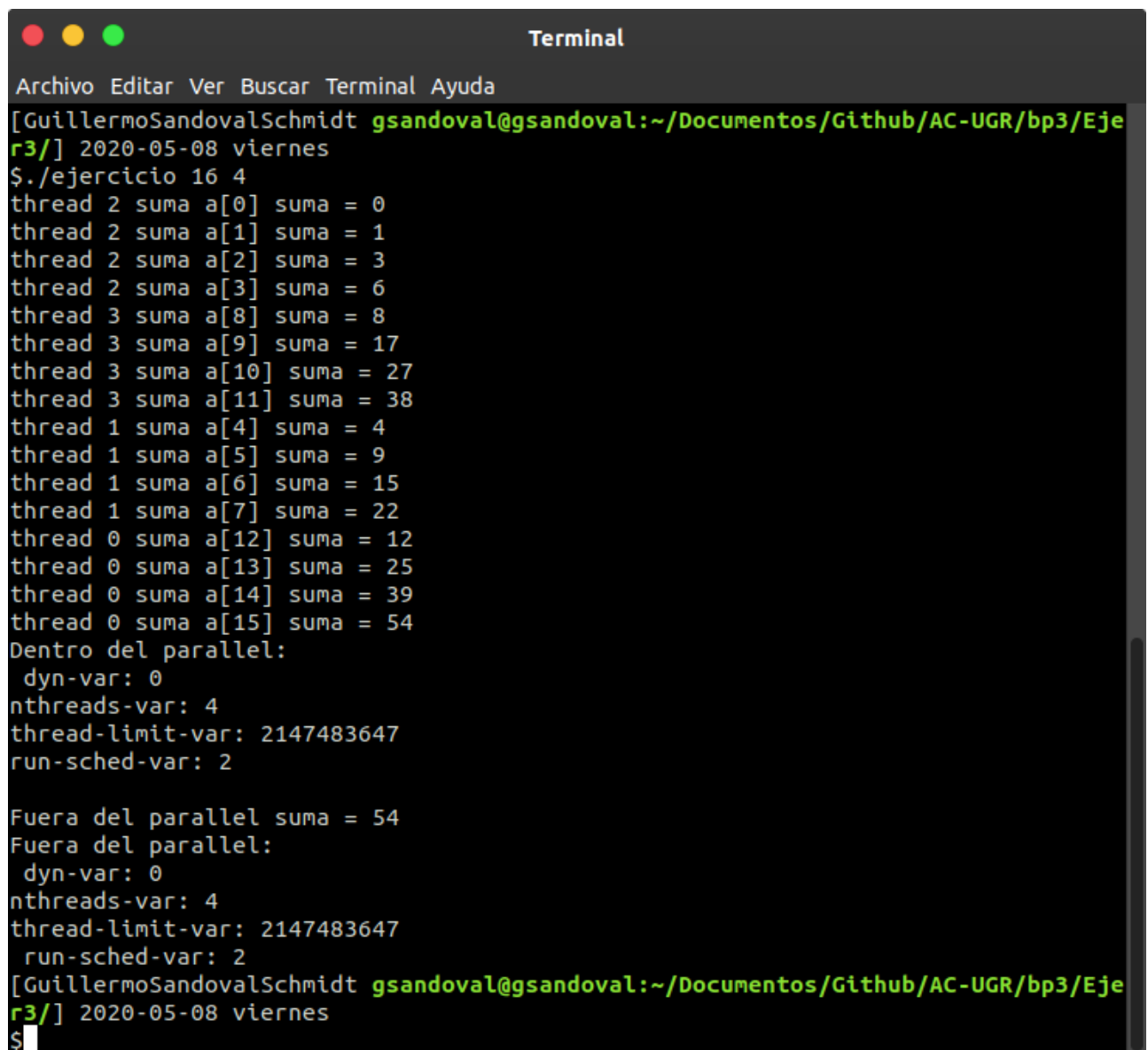


```

1#include <stdio.h>
2#include <stdlib.h>
3
4#ifdef _OPENMP
5#include <omp.h>
6#else
7#define omp_get_thread_num() 0
8#endif
9
10int main (int argc, char ** argv){
11    int i, n = 200, chunk, a[n], suma = 0;
12    omp_sched_t kind;
13    int modifier;
14
15    if (argc < 3){
16        fprintf(stderr, "\nFalta iteraciones o chunk\n");
17        exit(-1);
18    }
19    n = atoi(argv[1]); if (n > 200) n = 200; chunk = atoi(argv[2]);
20
21    for (i = 0; i < n; i++) a[i] = i;
22
23    #pragma omp parallel
24    {
25        #pragma omp for firstprivate(suma) \
26                                lastprivate(suma) \
27                                schedule(dynamic, chunk)
28        for (i = 0; i < n; i++){
29            suma = suma + a[i];
30            printf ("thread %d suma a[%d] suma = %d\n",
31                omp_get_thread_num(), i, suma);
32        }
33
34        #pragma omp single
35        {
36            omp_get_schedule(&kind, &modifier);
37
38            printf("Dentro del parallel: \n dyn-var: %d", omp_get_dynamic());
39            printf(" \nnthreads-var: %d ", omp_get_max_threads());
40            printf(" \nthread-limit-var: %d ",omp_get_thread_limit());
41            printf(" \nrun-sched-var: %d", kind );
42        }
43    }
44
45    omp_get_schedule(&kind, &modifier);
46
47    printf("\n\nFuera del parallel suma = %d\n", suma);
48    printf("Fuera del parallel: \n dyn-var: %d", omp_get_dynamic());
49    printf(" \nnthreads-var: %d ", omp_get_max_threads());
50    printf(" \nthread-limit-var: %d ",omp_get_thread_limit());
51    printf(" \n run-sched-var: %d \n", kind );
52
53    return 0;
54}

```

CAPTURAS DE PANTALLA:

A screenshot of a terminal window titled "Terminal" with a dark background. The window shows the execution of a program using OpenMP. The output displays the state of various OpenMP variables at different points in the execution. The prompt is a green user@host:~/Documents/Github/AC-UGR/bp3/Ejer3/. The program runs a loop with 16 iterations, each with a thread ID and a 'suma' value. The output shows the state of 'dyn-var', 'nthreads-var', 'thread-limit-var', and 'run-sched-var' both inside and outside a parallel region. The final output shows the state of these variables after the parallel region has completed.

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Ejer3/] 2020-05-08 viernes
$./ejercicio 16 4
thread 2 suma a[0] suma = 0
thread 2 suma a[1] suma = 1
thread 2 suma a[2] suma = 3
thread 2 suma a[3] suma = 6
thread 3 suma a[8] suma = 8
thread 3 suma a[9] suma = 17
thread 3 suma a[10] suma = 27
thread 3 suma a[11] suma = 38
thread 1 suma a[4] suma = 4
thread 1 suma a[5] suma = 9
thread 1 suma a[6] suma = 15
thread 1 suma a[7] suma = 22
thread 0 suma a[12] suma = 12
thread 0 suma a[13] suma = 25
thread 0 suma a[14] suma = 39
thread 0 suma a[15] suma = 54
Dentro del parallel:
  dyn-var: 0
nthreads-var: 4
thread-limit-var: 2147483647
run-sched-var: 2

Fuera del parallel suma = 54
Fuera del parallel:
  dyn-var: 0
nthreads-var: 4
thread-limit-var: 2147483647
run-sched-var: 2
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Ejer3/] 2020-05-08 viernes
$
```

RESPUESTA: Se imprimen los mismos valores, ya que lo que estamos haciendo es consultar las variables internas de estado de OpenMP, que siempre tendrán el mismo valor aunque las modifiquemos en el código.

4. Usar en el ejemplo anterior las funciones `omp_get_num_threads()`, `omp_get_num_procs()` y `omp_in_parallel()` dentro y fuera de la región paralela. Imprimir los valores que obtienen estas funciones dentro (lo debe imprimir sólo uno de los threads) y fuera de la región paralela. Incorporar en su cuaderno de prácticas volcados de pantalla con los resultados de ejecución obtenidos. Indicar en qué funciones se obtienen valores distintos dentro y fuera de la región paralela.

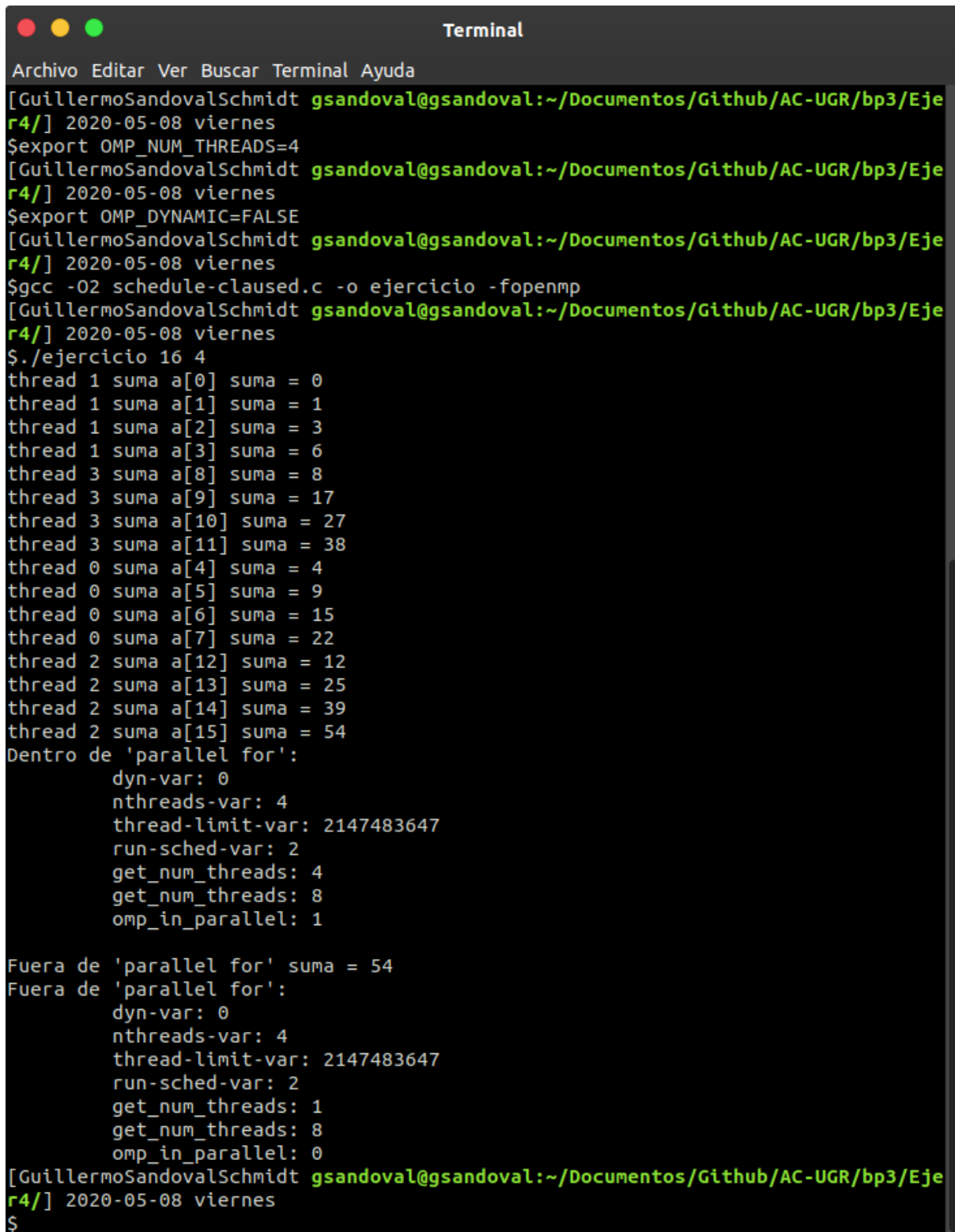
CAPTURA CÓDIGO FUENTE: scheduled-clauseModificado4.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #if __cplusplus
5 #include <omp.h>
6 #else
7 #define omp_get_thread_num() 0
8 #endif
9
10 int main (int argc, char ** argv){
11     int i, n = 200, chunk, a[n], suma = 0;
12     omp_sched_t kind;
13     int modifier;
14
15     if (argc < 3){
16         fprintf(stderr, "¡nFalta iteraciones o chunk!\n");
17         exit(-1);
18     }
19     n = atoi(argv[1]); if (n > 200) n = 200; chunk = atoi(argv[2]);
20
21     for (i = 0; i < n; i++) a[i] = i;
22
23     #pragma omp parallel
24     {
25         #pragma omp for firstprivate(suma) \
26             lastprivate(suma) \
27             schedule(dynamic, chunk)
28         for (i = 0; i < n; i++){
29             suma = suma + a[i];
30             printf ("thread %d suma a[%d] suma = %d\n",
31                 omp_get_thread_num(), i, suma);
32         }
33     }
34
35     #pragma omp master
36     {
37         omp_get_schedule(&kind, &modifier);
38
39         printf("Dentro de 'parallel for': %s\n", omp_get_dynamic());
40         printf(" %d threads-var: %d", omp_get_max_threads());
41         printf(" %d thread-limit-var: %d", omp_get_thread_limit());
42         printf(" %d run-sched-var: %d", kind);
43         printf(" %d get num threads: %d", omp_get_num_threads());
44         printf(" %d get num threads: %d", omp_get_num_procs());
45         printf(" %d omp_in_parallel: %d", omp_in_parallel());
46     }
47
48     omp_get_schedule(&kind, &modifier);
49
50     printf("\nFuera de 'parallel for' suma = %d\n", suma);
51     printf("Fuera de 'parallel for': %s\n", omp_get_dynamic());
52     printf(" %d threads-var: %d", omp_get_max_threads());
53     printf(" %d thread-limit-var: %d", omp_get_thread_limit());
54     printf(" %d run-sched-var: %d", kind);
55     printf(" %d get num threads: %d", omp_get_num_threads());
56     printf(" %d get num threads: %d", omp_get_num_procs());
57     printf(" %d omp_in_parallel: %d", omp_in_parallel());
58 }

```

CAPTURAS DE PANTALLA:



```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r4/] 2020-05-08 viernes
$export OMP_NUM_THREADS=4
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r4/] 2020-05-08 viernes
$export OMP_DYNAMIC=FALSE
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r4/] 2020-05-08 viernes
$gcc -O2 schedule-claused.c -o ejercicio -fopenmp
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r4/] 2020-05-08 viernes
$./ejercicio 16 4
thread 1 suma a[0] suma = 0
thread 1 suma a[1] suma = 1
thread 1 suma a[2] suma = 3
thread 1 suma a[3] suma = 6
thread 3 suma a[8] suma = 8
thread 3 suma a[9] suma = 17
thread 3 suma a[10] suma = 27
thread 3 suma a[11] suma = 38
thread 0 suma a[4] suma = 4
thread 0 suma a[5] suma = 9
thread 0 suma a[6] suma = 15
thread 0 suma a[7] suma = 22
thread 2 suma a[12] suma = 12
thread 2 suma a[13] suma = 25
thread 2 suma a[14] suma = 39
thread 2 suma a[15] suma = 54
Dentro de 'parallel for':
    dyn-var: 0
    nthreads-var: 4
    thread-limit-var: 2147483647
    run-sched-var: 2
    get_num_threads: 4
    get_num_threads: 8
    omp_in_parallel: 1

Fuera de 'parallel for' suma = 54
Fuera de 'parallel for':
    dyn-var: 0
    nthreads-var: 4
    thread-limit-var: 2147483647
    run-sched-var: 2
    get_num_threads: 1
    get_num_threads: 8
    omp_in_parallel: 0
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Eje
r4/] 2020-05-08 viernes
$
    
```

RESPUESTA:

Vemos como estas últimas si que se modifican, ya que son variables sobre como se esta usando en ese momento la región parallel.

5. Añadir al programa `scheduled-clause.c` lo necesario para modificar las variables de control `dyn-var`, `nthreads-var` y `run-sched-var` y para poder imprimir el valor de estas variables antes y después de dicha modificación. Incorporar en su cuaderno de prácticas volcados de pantalla con los resultados de ejecución obtenidos.

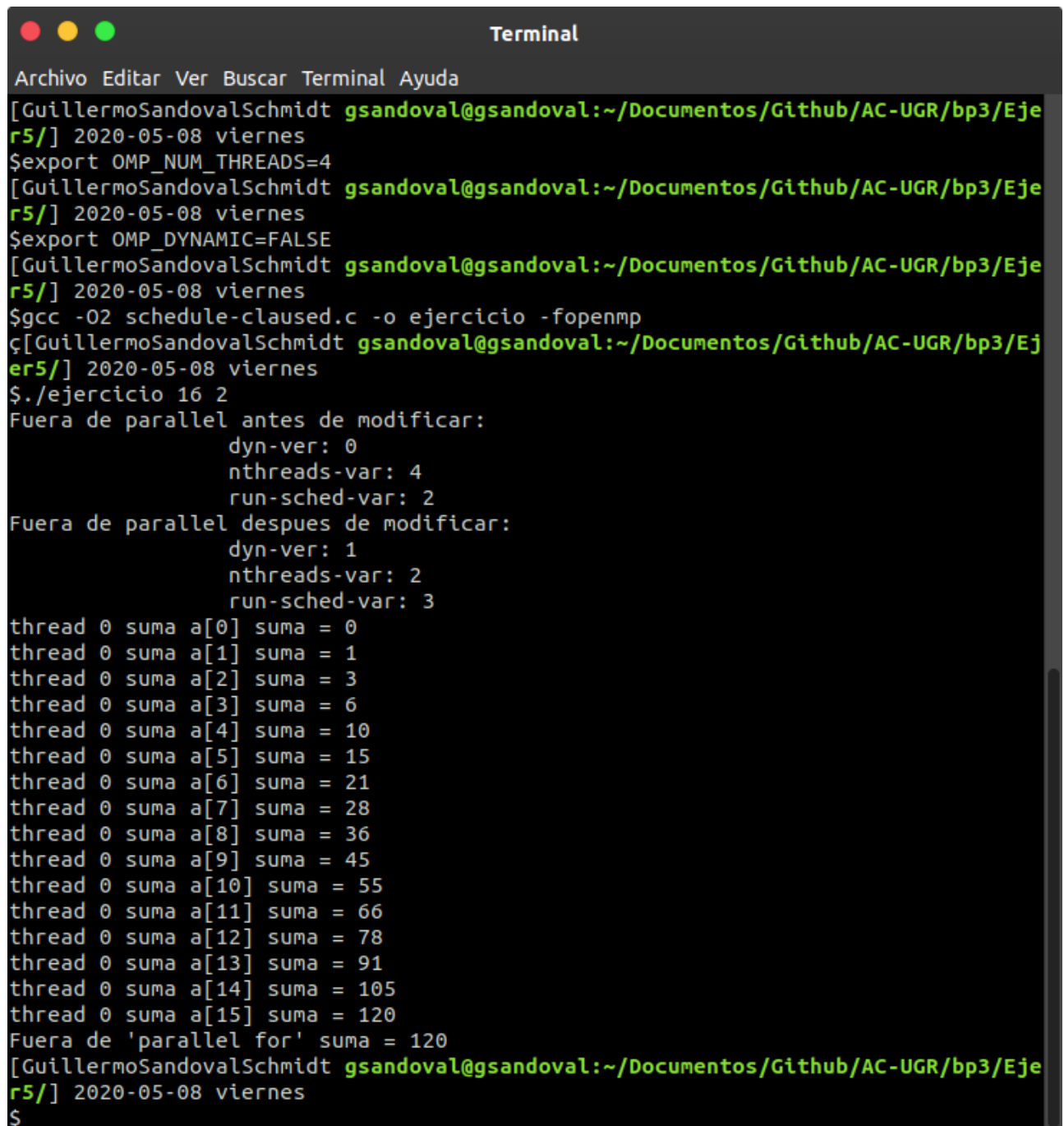
CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado5.c`

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #ifdef _OPENMP
5  #include <omp.h>
6  #else
7  #define omp_get_thread_num() 0
8  #endif
9
10 int main (int argc, char ** argv){
11     int i, n = 200, chunk, a[n], suma = 0;
12
13     omp_sched_t kind;
14     int modifier;
15
16     if (argc < 3){
17         fprintf(stderr, "nFalta iteraciones o chunk\n");
18         exit(-1);
19     }
20     n = atoi(argv[1]); if (n > 200) n = 200; chunk = atoi(argv[2]);
21
22     for (i = 0; i < n; i++) a[i] = i;
23
24     omp_get_schedule(&kind, &modifier);
25
26     printf("Fuera de parallel antes de modificar: \n");
27     printf("\t\t dyn-ver: %d\n", omp_get_dynamic());
28     printf("\t\t nthreads-var: %d\n", omp_get_max_threads());
29     printf("\t\t run-sched-var: %d\n", kind);
30
31     kind = 3;
32
33     omp_set_dynamic(1);
34     omp_set_num_threads(2);
35     omp_set_schedule(kind, modifier);
36
37     printf("Fuera de parallel despues de modificar: \n");
38     printf("\t\t dyn-ver: %d\n", omp_get_dynamic());
39     printf("\t\t nthreads-var: %d\n", omp_get_max_threads());
40     printf("\t\t run-sched-var: %d\n", kind);
41
42     #pragma omp parallel for firstprivate(suma) \
43                             lastprivate(suma) \
44                             schedule(dynamic, chunk)
45     for (i = 0; i < n; i++){
46         suma = suma + a[i];
47         printf ("thread %d suma a[%d] suma = %d\n",
48                omp_get_thread_num(), i, suma);
49     }
50
51     printf("Fuera de 'parallel for' suma = %d\n", suma);
52
53     return 0;
54 }
55
56
57

```

CAPTURAS DE PANTALLA:



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Ejer5/] 2020-05-08 viernes
$export OMP_NUM_THREADS=4
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Ejer5/] 2020-05-08 viernes
$export OMP_DYNAMIC=FALSE
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Ejer5/] 2020-05-08 viernes
$gcc -O2 schedule-claused.c -o ejercicio -fopenmp
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Ejer5/] 2020-05-08 viernes
$./ejercicio 16 2
Fuera de parallel antes de modificar:
    dyn-ver: 0
    nthreads-var: 4
    run-sched-var: 2
Fuera de parallel despues de modificar:
    dyn-ver: 1
    nthreads-var: 2
    run-sched-var: 3
thread 0 suma a[0] suma = 0
thread 0 suma a[1] suma = 1
thread 0 suma a[2] suma = 3
thread 0 suma a[3] suma = 6
thread 0 suma a[4] suma = 10
thread 0 suma a[5] suma = 15
thread 0 suma a[6] suma = 21
thread 0 suma a[7] suma = 28
thread 0 suma a[8] suma = 36
thread 0 suma a[9] suma = 45
thread 0 suma a[10] suma = 55
thread 0 suma a[11] suma = 66
thread 0 suma a[12] suma = 78
thread 0 suma a[13] suma = 91
thread 0 suma a[14] suma = 105
thread 0 suma a[15] suma = 120
Fuera de 'parallel for' suma = 120
[GuillermoSandovalSchmidt gsandoval@gsandoval:~/Documentos/Github/AC-UGR/bp3/Ejer5/] 2020-05-08 viernes
$
```

Resto de ejercicios

6. Implementar un programa secuencial en C que multiplique una matriz triangular por un vector (use variables dinámicas). Compare el orden de complejidad del código que ha implementado con el código que implementó para el producto matriz por vector.

NOTAS: (1) el número de filas/columnas debe ser un argumento de entrada; (2) se debe inicializar las matrices antes del cálculo; (3) se debe imprimir siempre la primera y última componente del resultado antes de que termine el programa.

CAPTURA CÓDIGO FUENTE: pmtv-secuencial.c

CAPTURAS DE PANTALLA:

7. Implementar en paralelo la multiplicación de una matriz triangular por un vector a partir del código secuencial realizado para el ejercicio anterior utilizando la directiva `for` de OpenMP. El código debe repartir entre los threads las iteraciones del bucle que recorre las filas. Dibujar en el cuaderno de prácticas la descomposición de dominio utilizada (Lección 4/Tema 2) en el código paralelo implementado para asignar tareas a los threads (Lección 5/Tema 2). Añadir lo necesario para que el usuario pueda fijar la planificación de tareas usando la variable de entorno `OMP_SCHEDULE`. Obtener en atcgrid los tiempos de ejecución del código paralelo (usando, como siempre, `-O2` al compilar) que multiplica una matriz triangular por un vector con las alternativas de planificación `static`, `dynamic` y `guided` para chunk de 1, 64 y el chunk por defecto para la alternativa. Use un tamaño de vector N múltiplo del número de cores y de 64 que no sea inferior a 15360. El número de threads en las ejecuciones debe coincidir con el número de cores. Rellenar la Tabla 3 dos veces con los tiempos obtenidos. Representar el tiempo para `static`, `dynamic` y `guided` en función del tamaño del chunk en una gráfica. ¿Qué alternativa ofrece mejores prestaciones? Razone por qué. Incluya los scripts utilizado en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

Conteste a las siguientes preguntas: (a) ¿Qué valor por defecto usa OpenMP para chunk con `static`, `dynamic` y `guided`? Indique qué ha hecho para obtener este valor por defecto para cada alternativa. (b) ¿Qué número de operaciones de multiplicación y suma realizan cada uno de los threads en la asignación `static` para cada uno de los chunks? (c) Con la asignación `dynamic` y `guided`, ¿qué cree que debe ocurrir con el número de operaciones de multiplicación y suma que realizan cada uno de los threads?

RESPUESTA:

CAPTURA CÓDIGO FUENTE: pmtv-OpenMP.c

DESCOMPOSICIÓN DE DOMINIO:

CAPTURAS DE PANTALLA:

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid**SCRIPT:** pmvt-OpenMP_atcgrid.sh**Tabla 3.** Tiempos de ejecución de la versión paralela del producto de una matriz triangular por un vector r **para vectores de tamaño N= , 12 threads**

Chunk	Static	Dynamic	Guided
por defecto			
1			
64			
Chunk	Static	Dynamic	Guided
por defecto			
1			
64			

8. Implementar un programa secuencial en C que calcule la multiplicación de matrices cuadradas, B y C:

$$A = B \bullet C; A(i, j) = \sum_{k=0}^{N-1} B(i, k) \bullet C(k, j), i, j = 0, \dots, N-1$$

NOTAS: (1) el número de filas/columnas debe ser un argumento de entrada; (2) se deben inicializar las matrices antes del cálculo; (3) se debe imprimir siempre las componentes (0,0) y (N-1, N-1) del resultado antes de que termine el programa.

CAPTURA CÓDIGO FUENTE: pmm-secuencial.c**CAPTURAS DE PANTALLA:**

9. Implementar en paralelo la multiplicación de matrices cuadradas con OpenMP a partir del código escrito en el ejercicio anterior. Use las directivas, las cláusulas y las funciones de entorno que considere oportunas. Se debe paralelizar también la inicialización de las matrices. Dibuje en su cuaderno de prácticas la descomposición de dominio que ha utilizado en el código paralelo implementado para asignar tareas a los threads (Lección 4/Tema 2, Lección 5/Tema 2).

10.

DESCOMPOSICIÓN DE DOMINIO:**CAPTURA CÓDIGO FUENTE:** pmm-OpenMP.c**CAPTURAS DE PANTALLA:**

10. Hacer un estudio de escalabilidad (ganancia en velocidad en función del número de cores) en atcgrid y en su PC del código paralelo implementado para dos tamaños de las matrices. Debe recordar usar `-O2` al compilar. El número de núcleos máximo en este estudio debe ser el igual al de núcleos físicos del computador. Presente los resultados del estudio en tablas de valores y en gráficas. Escoger los tamaños de manera que se observe diferentes curvas de escalabilidad en las gráficas que entregue en su cuaderno de prácticas (pruebe con valores de N entre 100 y 1500). Consulte la Lección 6/Tema 2. Incluya los scripts utilizado en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

ESTUDIO DE ESCALABILIDAD EN atcgrid:

SCRIPT: pmm-OpenMP_atcgrid.sh

ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

SCRIPT: pmm-OpenMP_pclocal.sh