

<p>2º curso / 2º cuatr. Grado Ing. Inform.</p>	<p>Arquitectura de Computadores (AC)</p> <p>Cuaderno de prácticas.</p> <p>Bloque Práctico 3. Programación paralela III:</p> <p>Interacción con el entorno en OpenMP</p> <p>Estudiante (nombre y apellidos): Amador Carmona Méndez</p> <p>Grupo de prácticas: B1</p> <p>Fecha de entrega:</p> <p>Fecha evaluación en clase:</p>
--	--

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  int main(int argc, char **argv){
5      int i, n=20, tid;
6      int a[n], suma=0, sumalocal;
7      if(argc< 2){
8          fprintf(stderr,"[ERROR]-Falta iteraciones\n");
9          exit(-1);
10     }
11     n=atoi(argv[1]);
12     if (n>20)
13         n=20;
14     for(i=0; i<n; i++){
15         a[i] = i;
16     }
17     #pragma omp parallel if(n>4) default(none) private(sumalocal,tid) shared(a,suma,n)
18     {
19         sumalocal=0;
20         tid=omp_get_thread_num();
21         #pragma omp for private(i) schedule(static) nowait
22         for(i=0; i<n; i++)
23             sumalocal+=a[i];
24         #pragma omp for private(i) schedule(static) nowait
25         for (i=0; i<n; i++) {
26             sumalocal += a[i];
27             printf(" thread %d suma de a[%d]=%d sumalocal=%d \n", tid,i,a[i],sumalocal);
28         }
29         #pragma omp atomic
30         suma += sumalocal;
31         #pragma omp barrier
32         #pragma omp master
33         printf("thread master=%d imprime suma=%d\n",tid,suma);
34     }
35 }

```

CAPTURAS DE PANTALLA:

```

[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3] 2021-05-22 sábado
$cd ejer1
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer1] 2021-05-22 sábado
$ls
Captura de pantalla de 2021-05-22 02-58-25.png  if-clauseModificado.c
if-clause
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer1] 2021-05-22 sábado
$./if-clause
[ERROR]-Falta iteraciones
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer1] 2021-05-22 sábado
$srn ./if-clause 3 5
thread 0 suma de a[0]=0 sumalocal=3
thread 0 suma de a[1]=1 sumalocal=4
thread 0 suma de a[2]=2 sumalocal=6
thread master=0 imprime suma=6
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer1] 2021-05-22 sábado
$srn ./if-clause 6 2
thread 0 suma de a[0]=0 sumalocal=3
thread 0 suma de a[1]=1 sumalocal=4
thread 0 suma de a[2]=2 sumalocal=6
thread 1 suma de a[3]=3 sumalocal=15
thread 1 suma de a[4]=4 sumalocal=19
thread 1 suma de a[5]=5 sumalocal=24
thread master=0 imprime suma=30
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer1] 2021-05-22 sábado
$srn ./if-clause 8 4
thread 0 suma de a[0]=0 sumalocal=6
thread 0 suma de a[1]=1 sumalocal=7
thread 0 suma de a[2]=2 sumalocal=9
thread 0 suma de a[3]=3 sumalocal=12
thread 1 suma de a[4]=4 sumalocal=26
thread 1 suma de a[5]=5 sumalocal=31
thread 1 suma de a[6]=6 sumalocal=37
thread 1 suma de a[7]=7 sumalocal=44
thread master=0 imprime suma=56
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer1] 2021-05-22 sábado
$

```

RESPUESTA: Indica el tamaño del vector y las hebras que trabajan en paralelo y calcula de forma paralela la suma.

2. Rellenar la Tabla 1 (se debe poner en la tabla el id del *thread* que ejecuta cada iteración) usando scheduler-clause.c con tres *threads* (0,1,2) y un número de iteraciones de 16 (0 a 15 en la tabla). Con este ejercicio se pretende comparar distintas alternativas de planificación de bucles. Se van a usar distintos tipos (static, dynamic, guided), modificadores (monotonic y nonmonotonic) y tamaños de chunk (x = 1, 2 y 4).

Tabla 1 . Tabla schedule. Rellenar esta tabla ejecutando scheduler-clause.c asignando previamente a la variable de entorno OMP_SCHEDULE los valores que se indican en la tabla (por ej.: export OMP_SCHEDULE="nonmonotonic:static,2). En la segunda fila, 1, 2 4 representan el tamaño del chunk

Iteración	"monotonic:static,x"		"nonmonotonic:static,x"		"monotonic:dynamic,x"		"monotonic:guided,x"	
	x=1	x=2	x=1	x=2	x=1	x=2	x=1	x=2
0	0	0	1	1	0	0	1	0
1	0	0	1	1	0	0	1	0
2	0	0	1	1	1	0	1	0
3	0	0	1	1	1	0	1	0
4	0	0	0	1	1	1	1	0
5	1	0	0	1	1	1	0	0
6	1	1	0	0	0	1	0	1
7	1	1	0	0	0	1	0	1
8	1	1	0	0	0	0	0	1
9	1	1	0	0	0	0	0	1
10	1	1	1	0	0	0	0	1
11	0	1	1	0	0	1	1	1
12	0	0	0	1	1	1	1	1
13	0	0	0	1	1	1	1	1
14	0	0	0	1	1	1	1	1
15	0	0	0	1	1	1	1	1

Destacar las diferencias entre las 4 alternativas de planificación de la tabla, en particular, las que hay entre static, dynamic y guided y las diferencias entre usar monotonic y nonmonotonic.

RESPUESTA: Par empezar, debo tener un error no se si al ejecutar por que aunque lo haya ejecutado en el atcgrid, solo me salen dos hebras 0 y 1 y entonces tengo el ejercicio erróneo ya que me deberían salir las hebras 0,1 y 2, por el seminario le puedo decir que static divide y distribuye las interacciones a las hebras por round-robin. Dynamic hace que los chunk más veloces ejecuten más y guided lo distribuye en un bloque largo y este va disminuyendo pero no se hace más pequeño que el chunk salvo el ultimo bloque

3. ¿Qué valor por defecto usa OpenMP para chunk y modifier con static, dynamic y guided? Explicar qué ha hecho para contestar a esta pregunta.

4. 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  #ifdef _OPENMP
4      #include <omp.h>
5  #else
6      #define omp_get_thread_num() 0
7  #endif
8  int main(int argc, char ** argv){
9      int i, n = 200, chunk, a[n], suma = 0, mod;
10     omp_sched_t tipo;
11     if(argc < 3){
12         fprintf(stderr, "\nFalta iteraciones o chunk \n");
13         exit(-1);
14     }
15     n = atoi(argv[1]);
16     if(n>200)
17         n=200;
18     chunk = atoi(argv[2]);
19     int dyn = omp_get_dynamic(), nthreads = omp_get_max_threads(), limit = omp_get_thread_limit();
20
21     omp_get_schedule(&tipo, &mod);
22
23     for(i=0; i<n; i++) a[i]=i;
24     #pragma omp parallel for firstprivate(suma) lastprivate(suma) schedule(dynamic, chunk)
25     for(i=0; i<n; i++){
26         suma = suma + a[i];
27         printf("thread %d suma a[%d] suma=%d \n", omp_get_thread_num(), i, suma);
28     }
29     #pragma omp single
30     printf("\nthread %d dynamic %d \n", omp_get_thread_num(), dyn);
31     printf("thread %d nthreads %d \n", omp_get_thread_num(), nthreads);
32     printf("thread %d limit %d \n", omp_get_thread_num(), limit);
33     printf("thread %d mod %d \n", omp_get_thread_num(), mod);
34     if(tipo == omp_sched_static)
35         printf("Es estático \n");
36     if(tipo == omp_sched_dynamic)
37         printf("Es dinámico \n");
38     if(tipo == omp_sched_guided)
39         printf("Es guided \n");
40
41     printf("\nFuera de 'parallel for' suma %d\n dynamic %d\n nthreads %d\n limit %d\n mod %d\n", suma, dyn,
42     if(tipo == omp_sched_static)
43         printf("Es estático \n");
44     if(tipo == omp_sched_dynamic)
45         printf("Es dinámico \n");
46     if(tipo == omp_sched_guided)
47         printf("Es guided \n");
48
49

```

CAPTURAS DE PANTALLA:

```

$cd ../ejer4
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer4] 2021-05-22 sábado
$ls
Captura de pantalla de 2021-05-22 03-00-01.png scheduled-clauseModificado.c
scheduled
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer4] 2021-05-22 sábado
$srn ./scheduled 10 5
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 1 suma a[5] suma=5
thread 1 suma a[6] suma=11
thread 1 suma a[7] suma=18
thread 1 suma a[8] suma=26
thread 1 suma a[9] suma=35

thread 0 dynamic 0
thread 0 nthreads 2
thread 0 limit 2147483647
thread 0 mod 1
Es dinámico

Fuera de 'parallel for'
suma 35
dynamic 0
nthreads 2
limit 2147483647
mod 1
Es dinámico
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer4] 2021-05-22 sábado
$

```

RESPUESTA: Podemos observar que dentro y fuera del parallel obtenemos el mismo resultado.

5. 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  #ifdef _OPENMP
4      #include <omp.h>
5  #else
6      #define omp_get_thread_num() 0
7  #endif
8  int main(int argc, char ** argv){
9      int i, n = 200, chunk, a[n], suma = 0, mod;
10     omp_sched t tipo;
11     if(argc < 3){
12         fprintf(stderr, "\nFalta iteraciones o chunk \n");
13         exit(-1);
14     }
15     n = atoi(argv[1]);
16     if(n>200)
17         n=200;
18     chunk = atoi(argv[2]);
19     int dyn = omp_get_dynamic(), nthreads = omp_get_max_threads(), limit = omp_get_thread_limit();
20
21     omp_get_schedule(&tipo, &mod);
22
23     for(i=0; i<n; i++) a[i]=i;
24     #pragma omp parallel for firstprivate(suma) lastprivate(suma) schedule(dynamic, chunk)
25     for(i=0; i<n; i++){
26         suma = suma + a[i];
27         printf("\nthread %d suma a[%d] suma=%d \n", omp_get_thread_num(), i, suma);
28         printf("omp_get_num_threads() = %d \n", omp_get_num_threads());
29     }
30
31     printf("\nDentro de la región parallel:\nomp_get_num_threads() = %d \nomp_get_num_procs() = %d \n");
32
33     printf("\nFuera de la región parallel: \nomp_get_num_threads()= %d \n omp_get_num_procs() = %d \n omp");
34 }

```

CAPTURAS DE PANTALLA:

```

$cd ../ejer5
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer5] 2021-05-22 sábado
$run ./scheduled 8 3

thread 1 suma a[3] suma=3
omp_get_num_threads() = 2

thread 1 suma a[4] suma=7
omp_get_num_threads() = 2

thread 1 suma a[5] suma=12
omp_get_num_threads() = 2

thread 0 suma a[0] suma=0

thread 1 suma a[6] suma=18
omp_get_num_threads() = 2
omp_get_num_threads() = 2

thread 1 suma a[7] suma=25

thread 0 suma a[1] suma=1
omp_get_num_threads() = 2
omp_get_num_threads() = 2

thread 0 suma a[2] suma=3
omp_get_num_threads() = 2

Dentro de la región parallel:
omp_get_num_threads() = 1
omp_get_num_procs() = 2
omp_in_parallel() = 0

Fuera de la región parallel:
omp_get_num_threads()= 1
omp_get_num_procs() = 2
omp_in_parallel() = 0
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer5] 2021-05-22 sábado
$

```

RESPUESTA: podemos observar que el num threads cambia a una fuera del parallel, esto lo he observado con mas de una ejecución puesto con una solo podría ser casualidad.

6. Añadir al programa scheduled-clause.c lo necesario para, usando funciones, modificar las variables de control dyn-var, nthreads-var y run-sched-var dentro de la región paralela y fuera de la región paralela. En la modificación de run-sched-var se debe usar un valor de kind distinto al utilizado en la cláusula schedule(). Añadir lo necesario para imprimir el contenido de estas variables antes y después de cada una de las dos modificaciones. Comentar los resultados.

CAPTURA CÓDIGO FUENTE: scheduled-clauseModificado5.c


```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #ifdef _OPENMP
4      #include <omp.h>
5  #else
6      #define omp_get_thread_num() 0
7  #endif
8  int main(int argc, char ** argv){
9      int i, n = 200, chunk, a[n], suma = 0, mod;
10     omp_sched_t tipo;
11     if(argc < 3){
12         fprintf(stderr, "\nFalta iteraciones o chunk \n");
13         exit(-1);
14     }
15     n = atoi(argv[1]);
16     if(n>200)
17         n=200;
18     chunk = atoi(argv[2]);
19     int dyn = omp_get_dynamic(), nthreads = omp_get_max_threads(), limit = omp_get_thread_limit();
20
21     omp_get_schedule(&tipo, &mod);
22
23     for(i=0; i<n; i++) a[i]=i;
24     #pragma omp parallel for firstprivate(suma) lastprivate(suma) schedule(dynamic, chunk)
25     for(i=0; i<n; i++){
26         suma = suma + a[i];
27         printf("thread %d suma a[%d] suma=%d \n", omp_get_thread_num(), i, suma);
28     }
29     #pragma omp single
30     omp_get_schedule(&tipo, &mod);
31     printf("\nValores actuales: dynamic: %d, nthreads: %d, tipo: %d, mod: %d\n", omp_get_dynamic(), omp_get_thread_num(), tipo, mod);
32     omp_set_dynamic(9);
33     omp_set_num_threads(2);
34     omp_set_schedule(omp_sched_guided, chunk + 3);
35     omp_get_schedule(&tipo, &mod);
36     printf("\nValores modicados: dynamic: %d, nthreads: %d, tipo: %d, mod: %d\n", omp_get_dynamic(), omp_get_thread_num(), tipo, mod);
37     printf("Fuera de 'parallel for' suma=%d\n", suma);
38 }

```

CAPTURAS DE PANTALLA:

```

[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer4] 2021-05-22 sábado
$cd ../ejer6
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer6] 2021-05-22 sábado
$run ./scheduled 9 3
thread 1 suma a[3] suma=3
thread 1 suma a[4] suma=7
thread 1 suma a[5] suma=12
thread 0 suma a[0] suma=0
thread 0 suma a[1] suma=1
thread 0 suma a[2] suma=3
thread 1 suma a[6] suma=18
thread 1 suma a[7] suma=25
thread 1 suma a[8] suma=33

Valores actuales: dynamic: 0, nthreads: 2, tipo: 2, mod: 1]

Valores modicados: dynamic: 1, nthreads: 2, tipo: 3, mod: 6]
Fuera de 'parallel for' suma=33
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer6] 2021-05-22 sábado
$

```

RESPUESTA: Vemos que el resultado sigue sin variar, y con cambia el valor de dynamic antes de modificarlo y después así como el tipo (tipo de planificación que tiene) y el mod que es el

chunk_size.

Resto de ejercicios (usar en atcgrid la cola ac a no ser que se tenga que usar atcgrid4)

7. Implementar un programa secuencial en C que multiplique una matriz triangular inferior por un vector (use variables dinámicas y tipo de datos double). Comparar el orden de complejidad y el número total de operaciones (sumas y productos) de este código respecto al que implementó para el producto matriz por vector.

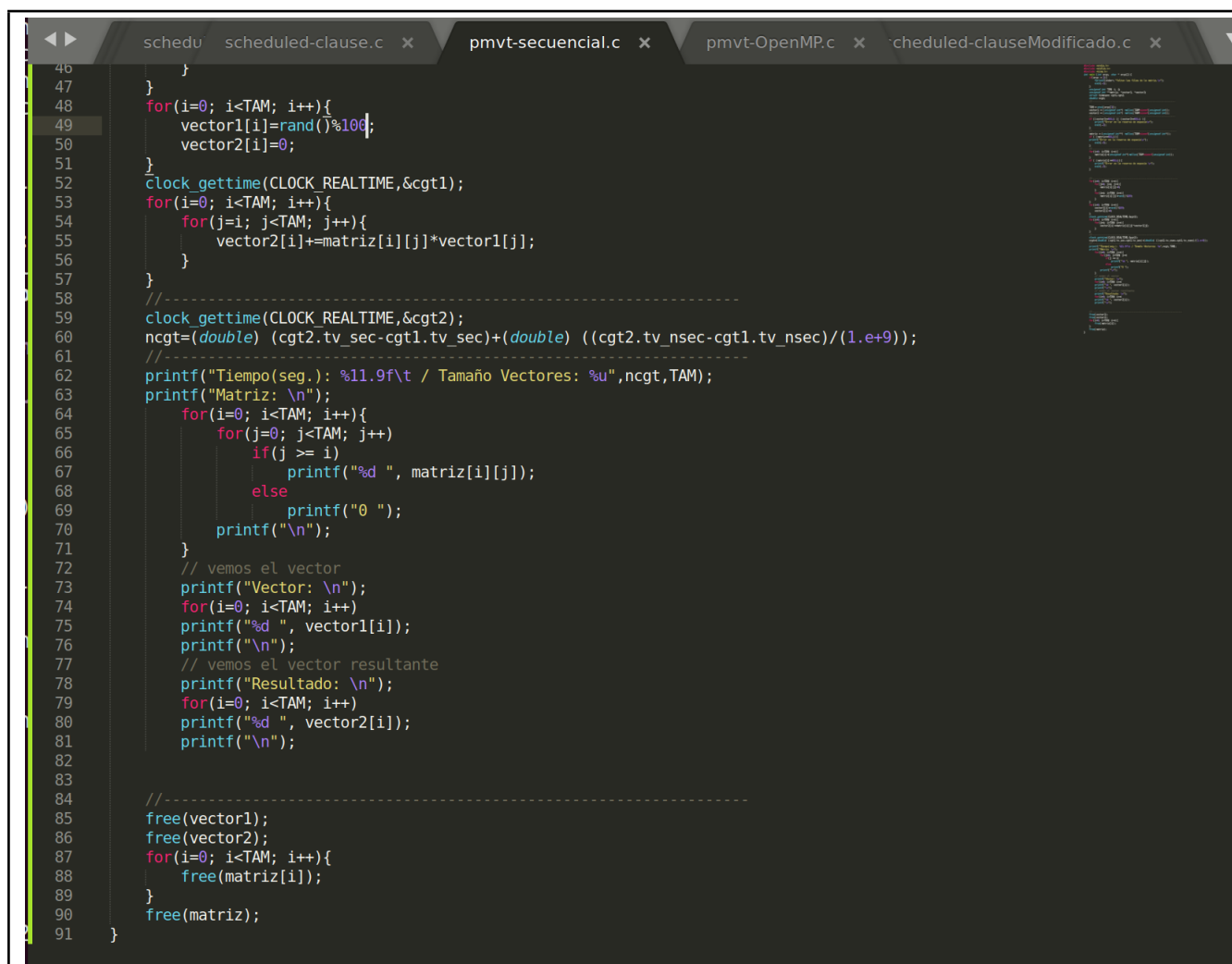
NOTAS: (1) el número de filas/columnas debe ser un argumento de entrada; (2) se debe inicializar las matrices antes del cálculo; (3) se debe imprimir siempre la primera y última componente del resultado antes de que termine el programa.

CAPTURA CÓDIGO FUENTE: pmtv-secuencial.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  int main (int argc, char * argv[]){
5      if(argc < 2){
6          fprintf(stderr,"faltan las filas de la matriz.\n");
7          exit(-1);
8      }
9      unsigned int TAM, i, j;
10     unsigned int **matriz, *vector1, *vector2;
11     struct timespec cgt1,cgt2;
12     double ncgt;
13     //-----
14
15     TAM = atoi(argv[1]);
16     vector1 = (unsigned int*) malloc(TAM*sizeof(unsigned int));
17     vector2 = (unsigned int*) malloc(TAM*sizeof(unsigned int));
18     //-----
19     if ((vector1==NULL) || (vector2==NULL) ){
20         printf("Error en la reserva de espacio\n");
21         exit(-2);
22     }
23     //-----
24     matriz = (unsigned int**) malloc(TAM*sizeof(unsigned int**));
25     if ( (matriz==NULL)){
26         printf("Error en la reserva de espacio\n");
27         exit(-3);
28     }
29     //-----
30     for(i=0; i<TAM; i++){
31         matriz[i]=(unsigned int*)malloc(TAM*sizeof(unsigned int));
32     }
33     if ( (matriz[i]==NULL)){
34         printf("Error en la reserva de espacio \n");
35         exit(-3);
36     }
37
38
39     //-----
40     for(i=0; i<TAM; i++){
41         for(j=0; j<i; j++){
42             matriz[i][j]=0;
43         }
44         for(j=i; j<TAM; j++){
45             matriz[i][j]=rand()%100;
46         }
47     }
48     for(i=0; i<TAM; i++){
49         vector1[i]=rand()%100;
50         vector2[i]=0;
51     }

```



```
46     }
47 }
48 for(i=0; i<TAM; i++){
49     vector1[i]=rand()%100;
50     vector2[i]=0;
51 }
52 clock_gettime(CLOCK_REALTIME,&cgt1);
53 for(i=0; i<TAM; i++){
54     for(j=i; j<TAM; j++){
55         vector2[i]+=matriz[i][j]*vector1[j];
56     }
57 }
58 //-----
59 clock_gettime(CLOCK_REALTIME,&cgt2);
60 ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+(double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
61 //-----
62 printf("Tiempo(seg.): %11.9f\t / Tamaño Vectores: %u",ncgt,TAM);
63 printf("Matriz: \n");
64 for(i=0; i<TAM; i++){
65     for(j=0; j<TAM; j++){
66         if(j >= i)
67             printf("%d ", matriz[i][j]);
68         else
69             printf("0 ");
70     }
71     printf("\n");
72     // vemos el vector
73     printf("Vector: \n");
74     for(i=0; i<TAM; i++)
75         printf("%d ", vector1[i]);
76     printf("\n");
77     // vemos el vector resultante
78     printf("Resultado: \n");
79     for(i=0; i<TAM; i++)
80         printf("%d ", vector2[i]);
81     printf("\n");
82 }
83
84 //-----
85 free(vector1);
86 free(vector2);
87 for(i=0; i<TAM; i++){
88     free(matriz[i]);
89 }
90 free(matriz);
91 }
```

CAPTURAS DE PANTALLA:

```

[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer6] 2021-05-22 sábado
$cd ../ejer7
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer7] 2021-05-22 sábado
$ls
Captura de pantalla de 2021-05-22 03-01-01.png  pmvt
Captura de pantalla de 2021-05-22 03-01-17.png  pmvt-secuencial.c
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer7] 2021-05-22 sábado
$batch -p ac --wrap "./pmvt 5"
Submitted batch job 107227
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer7] 2021-05-22 sábado
$ls
Captura de pantalla de 2021-05-22 03-01-01.png  pmvt-secuencial.c
Captura de pantalla de 2021-05-22 03-01-17.png  slurm-107227.out
pmvt
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer7] 2021-05-22 sábado
$cat slurm-107227.out
Tiempo(seg.):0.000000272          / Tamaño Vectores:5Matriz:
1 2 3 4 5
0 3 4 5 6
0 0 5 6 7
0 0 0 7 8
0 0 0 0 9
Vector:
0 1 2 3 4
Resultado:
40 50 56 53 36
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer7] 2021-05-22 sábado
$

```

8. Implementar en paralelo la multiplicación de una matriz triangular inferior 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. La inicialización de los datos la debe hacer el thread 0. 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`. Mostrar en una captura de pantalla que el código resultante funciona correctamente. NOTA: usar para generar los valores aleatorios, por ejemplo, `drand48_r()`.

CAPTURA CÓDIGO FUENTE: pmtv-OpenMP.c

```

31 }
32 if ( (matriz[i]==NULL)){
33     printf("Error en la reserva de espacio \n");
34     exit(-3);
35 }
36
37
38 //-----
39 #pragma omp parallel for private(j)
40     for(i=0; i<TAM; i++){
41         for(j=0; j<i; j++){
42             matriz[i][j]=0;
43         }
44         for(j=i; j<TAM; j++){
45             matriz[i][j]=rand()%100;
46         }
47     }
48     for(i=0; i<TAM; i++){
49         vector1[i]=rand()%100;
50         vector2[i]=0;
51     }
52     t1 = omp_get_wtime();
53     #pragma omp parallel for private(j) schedule(runtime)
54         for(i=0; i<TAM; i++){
55             for(j=i; j<TAM; j++){
56                 vector2[i]+=matriz[i][j]*vector1[j];
57             }
58         }
59 //-----
60 t2=omp_get_wtime();
61 tiempo=t2-t1;
62 //-----
63 //printf("t1 %f, t2 %f",t1,t2);
64 printf("Tiempo(seg.): %11.9f\t / Tamaño Vectores: %u",tiempo,TAM);
65 printf("Matriz: \n");
66     for(i=0; i<TAM; i++){
67         for(j=0; j<TAM; j++){
68             if(j >= i)
69                 printf("%d ", matriz[i][j]);
70             else
71                 printf("0 ");
72         }
73         printf("\n");
74     }
75     printf("Vector: \n");
76     for(i=0; i<TAM; i++)
77         printf("%d ", vector1[i]);
78     printf("\n");
79     printf("Resultado: \n");
80     for(i=0; i<TAM; i++)
81         printf("%d ", vector2[i]);
82     printf("\n");

```

DESCOMPOSICIÓN DE DOMINIO: Esta descomposición la he sacado de una diapositiva de teoría que explica lo mismo.

Asignación. Ej.: multiplicación matriz por vector II



$$c = A \cdot b; \quad c_i = \sum_{k=0}^{M-1} a_{ik} \cdot b_k = a_i^T \cdot b, \quad c(i) = \sum_{k=0}^{M-1} A(i, k) \cdot b(k), \quad i = 0, \dots, N-1$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} \end{pmatrix}_{8 \times 6} \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{pmatrix}_{M=6} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{pmatrix}_{N=8}$$

$c_1 = a_{11}b_1 + a_{12}b_2 + a_{13}b_3 + a_{14}b_4 + a_{15}b_5 + a_{16}b_6$
 $c_2 = a_{21}b_1 + a_{22}b_2 + a_{23}b_3 + a_{24}b_4 + a_{25}b_5 + a_{26}b_6$
 $c_3 = a_{31}b_1 + a_{32}b_2 + a_{33}b_3 + a_{34}b_4 + a_{35}b_5 + a_{36}b_6$
 $c_4 = a_{41}b_1 + a_{42}b_2 + a_{43}b_3 + a_{44}b_4 + a_{45}b_5 + a_{46}b_6$
 $c_5 = a_{51}b_1 + a_{52}b_2 + a_{53}b_3 + a_{54}b_4 + a_{55}b_5 + a_{56}b_6$
 $c_6 = a_{61}b_1 + a_{62}b_2 + a_{63}b_3 + a_{64}b_4 + a_{65}b_5 + a_{66}b_6$
 $c_7 = a_{71}b_1 + a_{72}b_2 + a_{73}b_3 + a_{74}b_4 + a_{75}b_5 + a_{76}b_6$
 $c_8 = a_{81}b_1 + a_{82}b_2 + a_{83}b_3 + a_{84}b_4 + a_{85}b_5 + a_{86}b_6$

CAPTURAS DE PANTALLA:

```

[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer7] 2021-05-22 sábado
$cd ../ejer8
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer8] 2021-05-22 sábado
$batch -p ac --wrap "./pmvt 6"
Submitted batch job 107231
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer8] 2021-05-22 sábado
$cat slurm-107231.out
t1 0.000000, t2 0.000000Tiempo(seg.): 0.000000000 / Tamaño Vectores: 6Matr
iz:
83 86 77 15 93 35
0 86 92 49 21 62
0 0 27 90 59 63
0 0 0 26 40 26
0 0 0 0 72 36
0 0 0 0 0 11
Vector:
68 67 29 82 30 62
Resultado:
19829 16922 13839 4944 4392 682
[AmadorCarmonaMendez b1estudiante4@atcgrid:~/BP3/ejer8] 2021-05-22 sábado
$

```

9. Contestar a las siguientes preguntas sobre el código del ejercicio anterior:

(a) ¿Qué número de operaciones de multiplicación y qué número de operaciones de suma realizan cada uno de los threads en la asignación static con monotonic y un chunk de 1?

RESPUESTA: Dependerá de la longitud de la matriz puesto que cada fila, al ser una matriz triangular hace una operación menos que la anterior.

(b) 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: dependerá de los threads disponibles

(c) ¿Qué alternativa ofrece mejores prestaciones? Razonar la respuesta.

RESPUESTA: Desde mi punto de vista la planificación guided es la que más reduce el tiempo de ejecución, aunque si sabes que vas a tener mucha sobrecarga mejor la static, y teóricamente la dynamic, debería ser la mejor puesto que es la que reparte equitativamente según se ejecute.

10. 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 (con monotonic en todos los casos). Usar un tamaño de vector N múltiplo del número de cores y de 64 que esté entre 11520 y 23040. El número de threads en las ejecuciones debe coincidir con el número de núcleos del computador. 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 (representar los valores de las dos tablas). Incluir los scripts utilizado en el cuaderno de prácticas. **NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.**

CAPTURA CÓDIGO FUENTE: pmtv-OpenMP.c

DESCOMPOSICIÓN DE DOMINIO:

CAPTURAS DE PANTALLA:

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid

SCRIPT: pmvt-OpenMP_atcgrid.sh

Tabla 2 . Tiempos de ejecución de la versión paralela del producto de una matriz triangular por un vector **para vectores de tamaño N=** (solo se ha paralelizado el producto, no la inicialización de los datos).

Chunk	Static	Dynamic	Guided
por defecto			
1			
64			

Chunk	Static	Dynamic	Guided
por defecto			
1			
64			