

1. V o F. Las variables de entorno (OMP\_NUM\_THREADS, OMP\_SCHEDULE...) y variables de control (nthreads-var, run-sched-var...) pueden ser modificadas desde dentro de un programa OMP
2. Escribir la sentencia que habría que escribir dentro de un programa en C para hacer que parallel se ejecute 20 veces
3. V o F. El resultado del siguiente código no es nada porque no se cumple la condición del if
4. V o F. La granularidad (chunk) por defecto de schedule es 2
5. V o F. La forma de asignación de schedule es dinámica por defecto. Es decir, la distribución de las hebras para cada iteración se hace en tiempo de ejecución.
6. V o F. Solo se puede usar if con bucles
7. V o F. Si hemos usado la cláusula schedule(static,1) y el valor de n-threadsvar es 3, la quinta iteración será ejecutada por la hebra número 2
8. V o F. Si hemos usado la cláusula schedule(static,1) y el valor de n-threadsvar es 3, la quinta iteración será ejecutada por la hebra número 1
9. V o F. Usando schedule(runtime) se puede elegir haciendo

1. V o F. Las variables de entorno (OMP\_NUM\_THREADS, OMP\_SCHEDULE...) y variables de control (nthreads-var, run-sched-var...) pueden ser modificadas desde dentro de un programa OMP

f

```
#include <stdio.h>
#include <omp.h>

int main () {
    int n=5;
    OMP_NUM_THREADS=20;
    #pragma omp parallel if(n>4)
    printf("x");
    return 0;
}
~
~
~
```

```
cristina@mipc:~/Uni/AC/practicas/bp3$ gcc -fopenmp testIf.c
testIf.c: In function 'main':
testIf.c:6:2: error: 'OMP_NUM_THREADS' undeclared (first use in
this function); did you mean '_IO_NO_READS'?
    OMP_NUM_THREADS=20;
    ^
    _IO_NO_READS
testIf.c:6:2: note: each undeclared identifier is reported only
once for each function it appears in
cristina@mipc:~/Uni/AC/practicas/bp3$
```

```
omp_set_num_threads(20);
```

[illegible]

3. V o F. El resultado del siguiente código no es nada porque no se cumple la condición del if

```
#include <stdio.h>
#include <omp.h>

int main () {
    int n=5;
    omp_set_num_threads(20);
    #pragma omp parallel if(n<4)
    printf("x");
    return 0;
}
```

f

x

```
cristina@mipc:~/Uni/AC/practicas/bp3$ ./a.out
xcristina@mipc:~/Uni/AC/practicas/bp3$
```

4. V o F. La granularidad (chunk) por defecto de schedule es 2

f

- Mejor no asumir una granularidad de distribución por defecto

✓

5. V o F. La forma de asignación de schedule es dinámica por defecto. Es decir, la distribución de las hebras para cada iteración se hace en tiempo de ejecución.

f

- Por defecto tipo `static` (distribución en tiempo de compilación) en la mayor parte de las implementaciones.

- - - .

6. V o F. Solo se puede usar if con bucles

f

# Cláusula schedule

AC



ATC

- Sintaxis:
  - `schedule (kind[, chunk])`
  - `kind`: forma de asignación
    - `static`
    - `dynamic`
    - `guided`
    - `auto`
    - `runtime`
  - `chunk`: granularidad de la distribución
- Precauciones:
  - Sólo bucles
  - Por defecto tipo `static` (distribución en tiempo de compilación) en la mayor parte de las implementaciones.
  - Mejor no asumir una granularidad de distribución por defecto

V3.0 en gris

7. V o F. Si hemos usado la cláusula `schedule(static,1)` y el valor de `n-threadsvar` es 3, la quinta iteración será ejecutada por la hebra número 2

f

## Cláusula `schedule. static`



### ➤ Usa

- `schedule(static, chunk)`
- Las iteraciones se dividen en unidades de chunk iteraciones.
- Las unidades se asignan en round-robin

```
mancia@mancia-ubuntu: ~/docencia/OpenMP/lec
Archivo Editar Ver Terminal Ayuda
$ gcc -O2 -fopenmp -o schedule-clause schedule-clause.c
$ ./schedule-clause 1
Hebra 1 suma a[1] suma=1
Hebra 1 suma a[4] suma=5
Hebra 2 suma a[2] suma=2
Hebra 2 suma a[5] suma=7
Hebra 0 suma a[0] suma=0
Hebra 0 suma a[3] suma=3
Hebra 0 suma a[6] suma=9
Fuera de 'parallel for' suma=9
$ ./schedule-clause 2
```



8. V o F. Si hemos usado la cláusula `schedule(static,1)` y el valor de `n-threadsvar` es 3, la quinta iteración será ejecutada por la hebra número 1

V

## Cláusula `schedule. static`



### ➤ Usa

- `schedule(static, chunk)`
- Las iteraciones se dividen en unidades de `chunk` iteraciones.
- Las unidades se asignan en round-robin

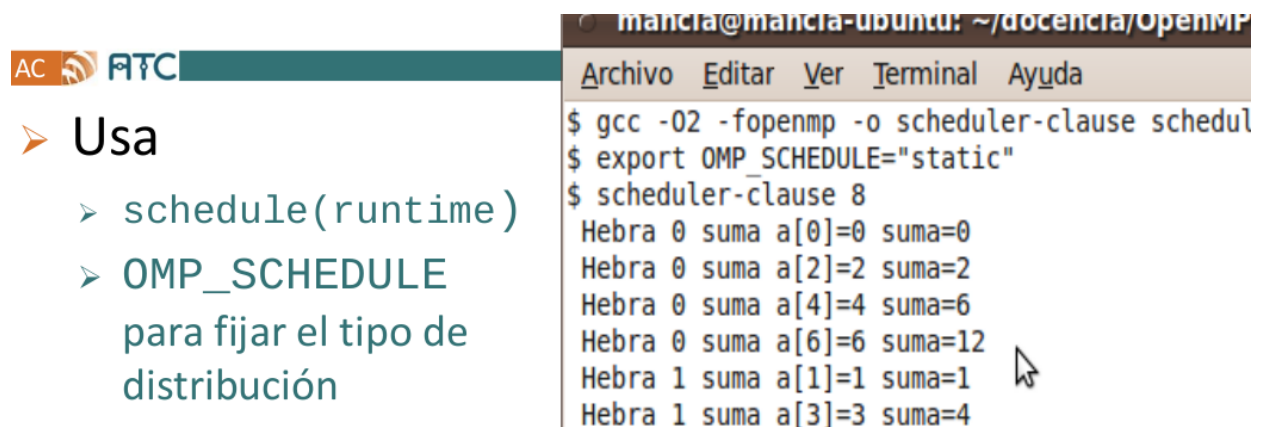
```
mancia@mancia-ubuntu: ~/docencia/OpenMP/lec
Archivo Editar Ver Terminal Ayuda
$ gcc -O2 -fopenmp -o schedule-clause schedule-clause.c
$ ./schedule-clause 1
Hebra 1 suma a[1] suma=1
Hebra 1 suma a[4] suma=5
Hebra 2 suma a[2] suma=2
Hebra 2 suma a[5] suma=7
Hebra 0 suma a[0] suma=0
Hebra 0 suma a[3] suma=3
Hebra 0 suma a[6] suma=9
Fuera de 'parallel for' suma=9
$ ./schedule-clause 2
```

9. V o F. Usando `schedule(runtime)` se puede elegir haciendo

`export OMP_SCHEDULE="static/dynamic/guided"`

Se puede elegir un tipo de ejecución diferente para cada ejecución

v



```
mancha@mancha-ubuntu: ~/docencia/OpenMP
Archivo  Editar  Ver  Terminal  Ayuda
$ gcc -O2 -fopenmp -o scheduler-clause scheduler-clause.c
$ export OMP_SCHEDULE="static"
$ scheduler-clause 8
Hebra 0 suma a[0]=0 suma=0
Hebra 0 suma a[2]=2 suma=2
Hebra 0 suma a[4]=4 suma=6
Hebra 0 suma a[6]=6 suma=12
Hebra 1 suma a[1]=1 suma=1
Hebra 1 suma a[3]=3 suma=4
```

- Usa
  - `schedule(runtime)`
  - `OMP_SCHEDULE` para fijar el tipo de distribución