Francisco Joaquín Murcia Gómez

48734281H

Grado en ingeniería informática UA

**Ingeniería de los computadores**

Guía OpenMP



**índice**

[Directivas 3](#_Toc56464190)

[Parallel 3](#_Toc56464191)

[For 3](#_Toc56464192)

[Sections 4](#_Toc56464193)

[Single 4](#_Toc56464194)

[Paralellel for/sections 5](#_Toc56464195)

[Task 5](#_Toc56464196)

[Master 6](#_Toc56464197)

[Critical 6](#_Toc56464198)

[Barrier 6](#_Toc56464199)

[Taskwait 7](#_Toc56464200)

[Atomic 7](#_Toc56464201)

[Flush 7](#_Toc56464202)

[Ordered 7](#_Toc56464203)

[Threadprivate 8](#_Toc56464204)

[Clausulas 8](#_Toc56464205)

[Cláusula de compartición de datos 8](#_Toc56464206)

[default(shared|none) 8](#_Toc56464207)

[shared(list) 8](#_Toc56464208)

[private(list) 8](#_Toc56464209)

[firstprivate(list) 9](#_Toc56464210)

[lastprivate(list) 9](#_Toc56464211)

[reduction(operator:list) 9](#_Toc56464212)

[nowait 9](#_Toc56464213)

[Cláusulas de copia de datos 9](#_Toc56464214)

[copyin(list) 9](#_Toc56464215)

[copyprivate(list) 9](#_Toc56464216)

[Schedule 9](#_Toc56464217)

[Static 9](#_Toc56464218)

[Dynamic 9](#_Toc56464219)

[Guided 10](#_Toc56464220)

[Auto 10](#_Toc56464221)

[Runtime 10](#_Toc56464222)

[Rutinas 10](#_Toc56464223)

[Rutinas de ejecución 10](#_Toc56464224)

[set\_num\_threads 10](#_Toc56464225)

[get\_num\_threads 10](#_Toc56464226)

[get\_max\_threads 10](#_Toc56464227)

[get\_thread\_num 10](#_Toc56464228)

[get\_num\_procs 10](#_Toc56464229)

[omp\_in\_parallel(void); 11](#_Toc56464230)

[set\_dynamic 11](#_Toc56464231)

[omp\_get\_dynamic(void); 11](#_Toc56464232)

[set\_nested 11](#_Toc56464233)

[get\_nested 11](#_Toc56464234)

[set\_schedule 11](#_Toc56464235)

[get\_schedule 11](#_Toc56464236)

[get\_thread\_limit 11](#_Toc56464237)

[set\_max\_active\_levels 12](#_Toc56464238)

[get\_level 12](#_Toc56464239)

[get\_ancestor\_thread\_num 12](#_Toc56464240)

[get\_team\_size 12](#_Toc56464241)

[get\_active\_level 12](#_Toc56464242)

[Cerrojos 12](#_Toc56464243)

[init\_[nest]\_lock 12](#_Toc56464244)

[destroy\_[nest]\_lock 12](#_Toc56464245)

[set\_[nest]\_lock 13](#_Toc56464246)

[unset\_[nest]\_lock 13](#_Toc56464247)

[test\_[nest]\_lock 13](#_Toc56464248)

[Rutinas de tiempo 13](#_Toc56464249)

[get\_wtime 13](#_Toc56464250)

[get\_wtick 13](#_Toc56464251)

# Directivas

## Parallel

Se utiliza para iniciar la programación paralela

#pragma omp parallel [clause[ [, ]clause] ...] new-line

structured-block

**Clausulas:**

* if(scalar-expression)
* num\_threads(integer-expression)
* default(shared | none)
* private(list)
* firstprivate(list)
* shared(list)
* copyin(list)
* reduction(operator: list)

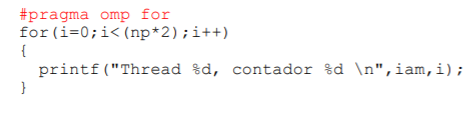
## For

Se utiliza para los bucles, especifica las iteracionas y como se ejecutara en cada hilo

#pragma omp for [clause[[,] clause] ... ] new-line

for-loops

**Clausulas:**

* private(list)
* firstprivate(list)
* lastprivate(list)
* reduction(operator: list)
* schedule(kind[, chunk\_size])
* collapse(n)
* ordered
* nowait

## Sections

Un conjunto de bloques que tienen que ser distribuidos y ejecutados por un equipo de hilos.

#pragma omp sections [clause[[,] clause] ...] new-line

{

[#pragma omp section new-line]

structured-block

[#pragma omp section new-line

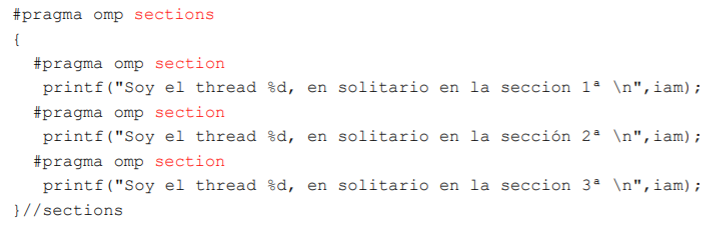
structured-block ]

...

}

**Clausulas:**

* private(list)
* firstprivate(list)
* lastprivate(list)
* reduction(operator: list)
* nowait



## Single

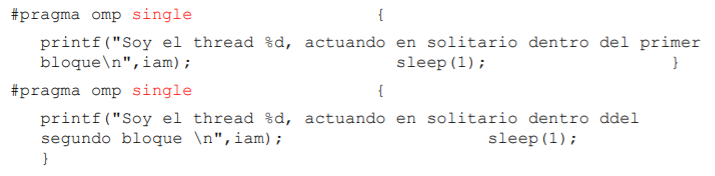
Especifica que el bloque asociado ha de ser ejecutado por un solo hilo (no tiene por qué ser la maestra)

#pragma omp single [clause[[,] clause] ...] new-line

structured-block

**Clausula:**

* private(list)
* firstprivate(list)
* copyprivate(list)
* nowait



## Paralellel for/sections

Funcionan igual que parallel y for o sections pero en una sola línea:

#pragma omp parallel for [clause[[,] clause] ...] new-line

for-loop

#pragma omp parallel sections [clause[ [, ]clause] ...]

new-line

{

[#pragma omp section new-line]

structured-block

[#pragma omp section new-line

structured-block ]

...

}

## Task

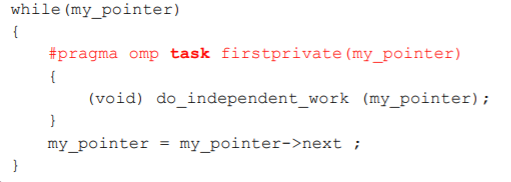
Define una tarea especifica

#pragma omp task [clause[ [, ]clause] ...] new-line

structured-block

**Clausulas:**

* if(scalar-expression)
* untied
* default(shared | none)
* private(list)
* firstprivate(list)
* shared(list)

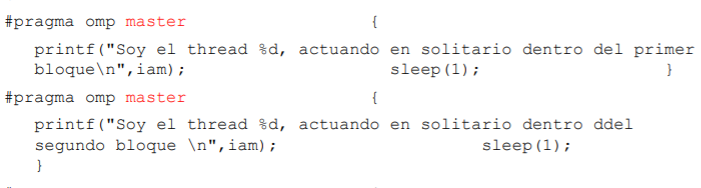


## Master

Similar al single, pero lo ejecuta la rama principal

#pragma omp master new-line

structured-block

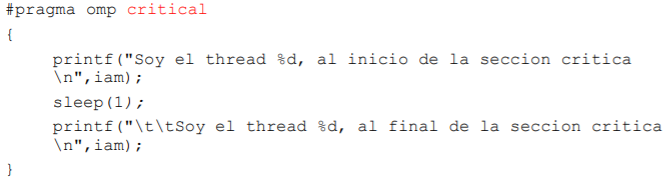


## Critical

Restringe la ejecución a un solo hilo

#pragma omp critical [(name)] new-line

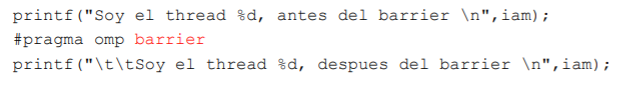
structured-block



## Barrier

Especifica un punto de unión de los hilos

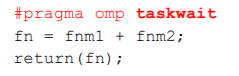
#pragma omp barrier new-line



## Taskwait

Especifica el lugar de la finalización de las tareas secundarias

#pragma omp taskwait newline



## Atomic

Hace que un lugar de almacenamiento se actualiza automáticamente con lecturas y escrituras simultaneas

#pragma omp atomic new-line

expression-stmt

expression-stmt: one of the following forms:

x binop = expr

x++

++x

x--

--x

## Flush

Crea un hilo temporal

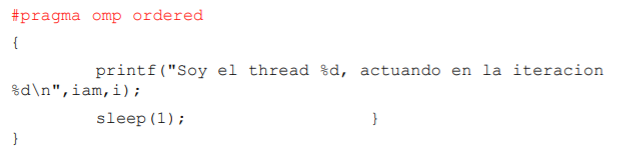
#pragma omp flush [(list)] new-line

## Ordered

El bloque se ejecuta en el orden en que se ejecutaría de manera secuencial

#pragma omp ordered new-line

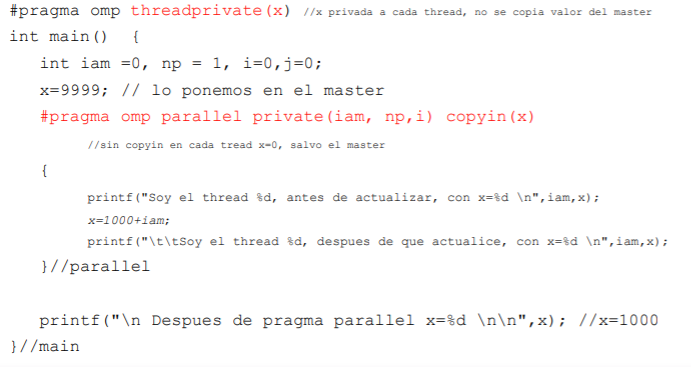
structured-block



## Threadprivate

Especifica las variables que se replican, teniendo cada hilo una copia

#pragma omp threadprivate(list) new-line



# Clausulas

## Cláusula de compartición de datos

### default(shared|none)

Controla el atributo por defecto de comparticion de datos para variables referenciasdas en el contructor de la región paralela o tarea

### shared(list)

Declara uno o mas elementos que se comparten por las tareas generadas por el contructor de la región paralela

### private(list)

Declaro uno o mas elementos privados para la tarea

### firstprivate(list)

Declara uno o más elementos privados para la tarea, e inicializados cada una a su correspondiente valor original en el momento que se encuentra el constructor

### lastprivate(list)

Declara uno o mas elementos privados implícitos a la tarea, y causa que el elemento original se actualiza después de terminar la región

### reduction(operator:list)

Declara una acumulación a partir de una lista de elementos usando un operador asociativo indicado.

### nowait

Evita la sincronización implícita de las hebras al terminar el bloque de una directiva

## Cláusulas de copia de datos

### copyin(list)

Copia los valores de las variables privadas del hilo maestro en cada una de las otras pertenecientes al equipo de ejecución de la región paralela

### copyprivate(list)

Transmite el valor de un entorno de datos de una tarea a otro entorno de datos de las tareas pertenecientes a la región paralela

# Schedule

## Static

Se dividen las iteraciones en partes, esta se asigna a los hilos siguiendo el método de planificación round-robin sehun en los hilos

## Dynamic

Cada hilo ejecuta una parte de las iteraciones, después pide otra parte así hasta que se acabe

## Guided

Como la dinámica, pero las partes comienzan siendo grandes y se van reduciendo con forme se avanza

## Auto

La planificación del reparto de tareas es decidida por el compilador

## Runtime

El planificador y el tamaño de las partes las toma de run-sched-var

# Rutinas

## Rutinas de ejecución

### set\_num\_threads

Indica el numero de hilos a utilizar una región paralela

void omp\_set\_num\_threads(int num\_threads);

### get\_num\_threads

Obtiene el número de hilos que se están usando en una región paralela

int omp\_get\_num\_threads(void)

### get\_max\_threads

Obtiene la máxima cantidad posible de hilos

int omp\_get\_max\_threads(void);

### get\_thread\_num

Devuelve el número del hilo

int omp\_get\_thread\_num(void);

### get\_num\_procs

Devuelve el máximo número de procesadores que se pueden asignar al programa

int omp\_get\_num\_procs(void);

### omp\_in\_parallel(void);

Devuelve valor distinto de cero si se ejecuta dentro de una región paralela

int omp\_in\_parallel(void);

### set\_dynamic

Permite poner o quitar el que el número de threads se pueda ajustar dinámicamente en las regiones paralelas.

void omp\_set\_dynamic(void);

### omp\_get\_dynamic(void);

Devuelve un valor distinto de cero si está permitido el ajuste dinámico del número de hilo

int omp\_get\_dynamic(void);

### set\_nested

Para permitir o desautorizar el paralelismo anidado

void omp\_set\_nested(int nested);

### get\_nested

Devuelve un valor distinto de cero si está permitido el paralelismo anidado

int omp\_get\_nested(void);

### set\_schedule

Afecta al planificador por defecto usado en las rutinas, modifica la variable run-sched-var

void omp\_set\_schedule(omp\_sched\_t kind, int modifier);

### get\_schedule

Devuelve el planificador aplicado en las rutinas de planificación

void [omp\_get\_schedule](https://lsi.ugr.es/jmantas/ppr/ayuda/omp_ayuda.php?ayuda=omp_rutinas)(omp\_sched\_t \*kind,int \*modifier);

### get\_thread\_limit

Devuelve el número máximo de hilos disponibles para el programa

int [omp\_get\_thread\_limit](https://lsi.ugr.es/jmantas/ppr/ayuda/omp_ayuda.php?ayuda=omp_rutinas)(void)

### set\_max\_active\_levels

Limita el numero regiones paralelas anidadas, modifica la variable max-active-levels-var (VIC).

int omp\_get\_max\_active\_levels(void);

### get\_level

Devuelve el número de regiones paralelas anidadas que contiene la llamada.

### get\_ancestor\_thread\_num

Devuelve, para un determinado nivel anidado de la hebra actual, el numero de hebra padre o al que actualmente pertenece

int omp\_get\_ancestor\_thread\_num(int level);

### get\_team\_size

Devuelve, para un determinado nivel anidado de la hebra actual, el tamaño del equipo de la hebra padre o al que actualmente pertenece

int omp\_get\_team\_size(int level);

### get\_active\_level

Devuelve, para un determinado nivel anidado de la hebra actual, el numero de hebra del padre.

int omp\_get\_active\_level(void);

## Cerrojos

### init\_[nest]\_lock

Esta rutina inicializa los cerrojos de OpenMP

void omp\_init\_lock(omp\_lock\_t \*lock);

void omp\_init\_nest\_lock(omp\_nest\_lock\_t \*lock);

### destroy\_[nest]\_lock

Estas rutinas de asegurar que el bloqueo de OpenMP no está inicializado.

void omp\_destroy\_lock(omp\_lock\_t \*lock);

void omp\_destroy\_nest\_lock(omp\_nest\_lock\_t \*lock);

### set\_[nest]\_lock

Estas rutinas proporcionan un medio de establecer un bloqueo de OpenMP.

void omp\_set\_lock(omp\_lock\_t \*lock);

void omp\_set\_nest\_lock(omp\_nest\_lock\_t \*lock);

### unset\_[nest]\_lock

Estas rutinas proporcionan un medio de establecer un bloqueo de OpenMP.

void omp\_unset\_lock(omp\_lock\_t \*lock);

void omp\_unset\_nest\_lock(omp\_nest\_lock\_t \*lock);

### test\_[nest]\_lock

Estas rutinas comprueban si el cerrojo esta bloqueado, no suspendiendo la ejecución de la tarea.

int omp\_test\_lock(omp\_lock\_t \*lock);

int omp\_test\_nest\_lock(omp\_nest\_lock\_t \*lock);

## Rutinas de tiempo

### get\_wtime

Devuelve el valor del reloj en segundos.

double omp\_get\_wtime(void);

### get\_wtick

Devuelve la precisión del reloj

double omp\_get\_wtick(void);