

2º curso / 2º cuatr.

Grado en
Ing. Informática

Arquitectura de Computadores

Seminario 2. Herramientas de programación paralela II: Cláusulas OpenMP

Material elaborado por los profesores responsables de la asignatura:

Mancia Anguita – Julio Ortega

Licencia Creative Commons



ugr

Universidad
de Granada

ETSIIT

Escuela Técnica Superior
de Ingenierías Informática
y de Telecomunicación



ATC

Departamento de Arquitectura
y Tecnología de Computadores
UNIVERSIDAD DE GRANADA



Contenidos

- Cláusulas
- Ámbito de los datos por defecto. Cláusulas relacionadas con la compartición de datos
- Cláusulas relacionadas con comunicación/sincronización

Contenidos

- Cláusulas
- Ámbito de los datos por defecto. Cláusulas relacionadas con la compartición de datos
- Cláusulas relacionadas con comunicación/sincronización

Cláusulas

- Ajustan el comportamiento de las directivas

```
#pragma omp parallel num_threads(8) if(N>20)
```

- Las directivas con cláusulas son (v2.5):

- `parallel`

- Directivas de trabajo compartido:

- `DO/for`, `sections`, `single` y `workshare`

- Directivas que combinan `parallel` y directivas de trabajo compartido:

- aceptan cláusulas de las dos directivas que combinan excepto `nowait`

- `parallel DO/for`, `parallel sections`, `parallel workshare`

- Directivas que no aceptan cláusulas (v2.5):

- `master`

- Sincronización/consistencia:

- `critical`, `barrier`, `atomic`, `flush`

- `ordered`

- `threadprivate`

Directivas con cláusulas (v2.5)

DIRECTIVA	ejecutable	declarativa	
con bloque estructurado	<i><u>parallel</u></i> <i><u>sections, worksharing, single</u></i> master critical ordered		Con sentencias
bucle	<i><u>DO/for</u></i>		
simple (una sentencia)	atomic		
autónoma (sin código asociado)	barrier, flush	threadprivate	sin

- Las directivas que aceptan cláusulas aparecen en cursiva

Cláusulas relacionadas con compartición de datos

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	if (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

Se han destacado en color las cláusulas que se van a comentar en este seminario

Contenidos

- Cláusulas
- **Ámbito de los datos por defecto. Cláusulas relacionadas con la compartición de datos**
- Cláusulas relacionadas con comunicación/sincronización

Ámbito de los datos o atributos de compartición

- Regla general para regiones paralelas (para variables que no se usan en cláusulas o directivas de ámbito):

- Las variables declaradas fuera de una región y las dinámicas son compartidas por las threads de la región
- Las variables declaradas dentro son privadas

- Excepciones:

- índice de bucles `for`: ámbito predeterminado de privado
- Variables declaradas `static`

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

main(int argc, char **argv) {
    int i, n=20, a[n], suma=0;
    if(argc < 2) {
        fprintf(stderr, "\nFalta iteraciones\n");
        exit(-1);
    }
    n = atoi(argv[1]); if (n>20) n=20;

    for (i=0; i<n; i++)    a[i] = i;

    #pragma omp parallel
    { int sumalocal=0;
      #pragma omp for schedule(static)
      for (i=0; i<n; i++)
      { sumalocal += a[i];
        printf(" thread %d suma de a[%d]=%d sumalocal=%d \n",
              omp_get_thread_num(), i, a[i], sumalocal);
      }
      #pragma omp atomic
      suma += sumalocal;
    }
    printf("Fuera de 'parallel' suma=%d\n", suma); return(0);
}
```


Cláusulas relacionadas con compartición de datos

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	if (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

Cláusulas

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	if (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

Cláusula `shared`

`shared-clause.c` [Chapman 2008]



- Sintaxis:
 - `shared(list)`
- Se comparten las variables de *list* por todas los threads
- Precauciones:
 - Cuando al menos un thread lee lo que otro escribe en alguna variable de la lista

```
#include <stdio.h>
#ifdef _OPENMP
#include <omp.h>
#endif

main()
{
    int i, n = 7;
    int a[n];

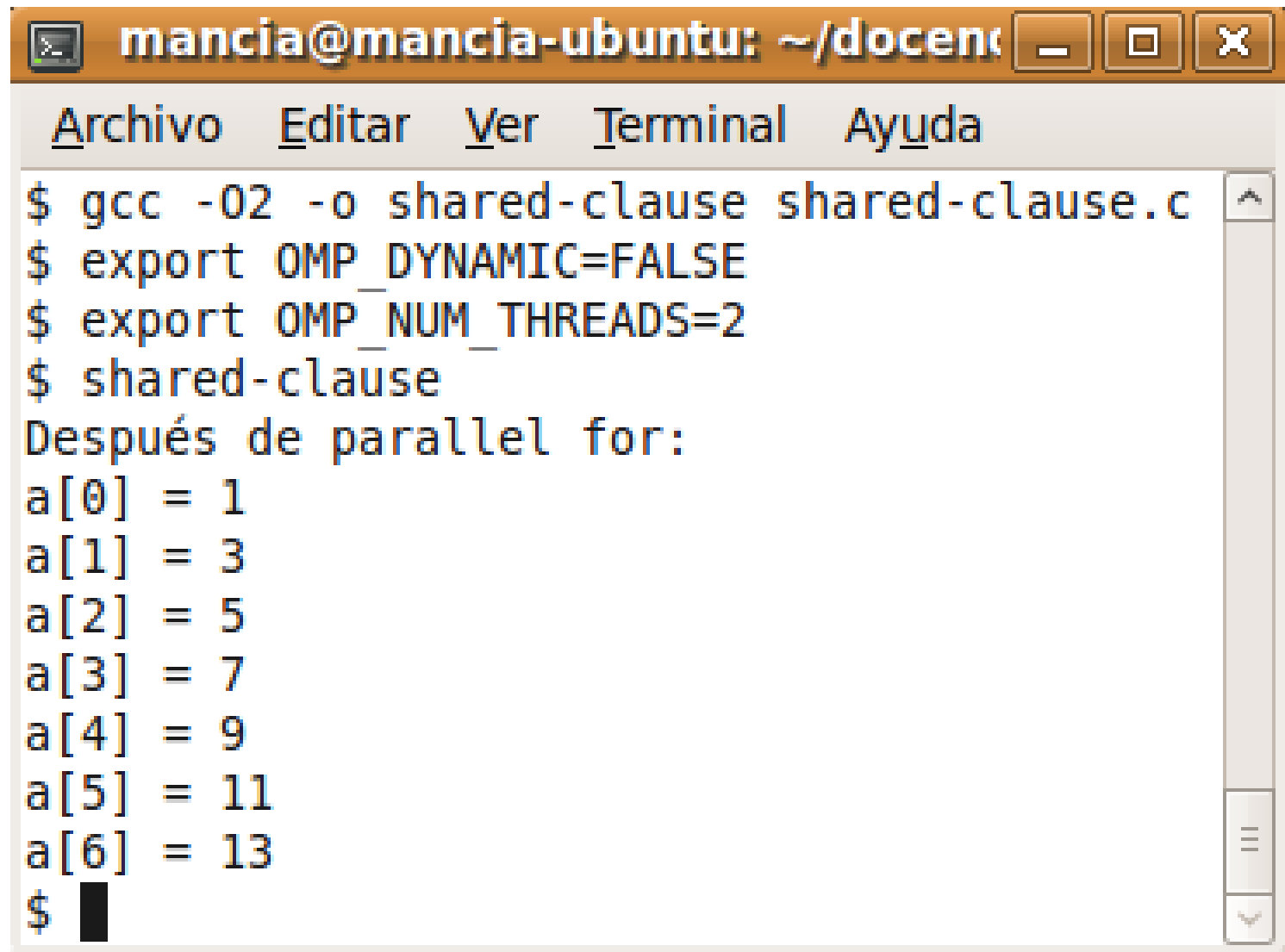
    for (i=0; i<n; i++)
        a[i] = i+1;

    #pragma omp parallel for shared(a)
    for (i=0; i<n; i++)    a[i] += i;

    printf("Después de parallel for:\n");

    for (i=0; i<n; i++)
        printf("a[%d] = %d\n",i,a[i]);
}
```

Cláusula shared. Salida



```
mancia@mancia-ubuntu: ~/docenc
Archivo  Editar  Ver  Terminal  Ayuda
$ gcc -O2 -o shared-clause shared-clause.c
$ export OMP_DYNAMIC=FALSE
$ export OMP_NUM_THREADS=2
$ shared-clause
Después de parallel for:
a[0] = 1
a[1] = 3
a[2] = 5
a[3] = 7
a[4] = 9
a[5] = 11
a[6] = 13
$
```

Cláusulas

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	If (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

Cláusula private

private-clause.c



- Sintaxis:
 - `private(list)`
- Precauciones:
 - El valor de **entrada** y de **salida** está **indefinido** aunque la variable esté declarada fuera de la construcción
- Predeterminado:
 - Los índices de un bucle tienen un ámbito **predeterminado** de privado si se usa para ese bucle la directiva `for`

```
#include <stdio.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif

main()
{
    int i, n = 7;
    int a[n], suma;

    for (i=0; i<n; i++)
        a[i] = i;

    #pragma omp parallel private(suma)
    {
        suma=0;
        #pragma omp for
        for (i=0; i<n; i++)
        {
            suma = suma + a[i];
            printf(
                "thread %d suma a[%d] / ", omp_get_thread_num(), i);
        }
        printf(
            "\n* thread %d suma= %d", omp_get_thread_num(), suma);
    }

    printf("\n");
}
```

Cláusula private. Salida

```
mancia@mancia-ubuntu: ~/docencia/OpenMP/
Archivo  Editar  Ver  Terminal  Ayuda
$ gcc -O2 -fopenmp -o private-clause private-clause.c
$ export OMP_DYNAMIC=FALSE
$ export OMP_NUM_THREADS=4
$ private-clause
Hebra 3 suma a[6] / Hebra 1 suma a[2] / Hebra 1 suma a
[3] / Hebra 2 suma a[4] / Hebra 2 suma a[5] / Hebra 0
suma a[0] / Hebra 0 suma a[1] /
* Hebra 2 suma= 9
* Hebra 1 suma= 5
* Hebra 3 suma= 6
* Hebra 0 suma= 1
$ export OMP_NUM_THREADS=3
$ private-clause
Hebra 2 suma a[6] / Hebra 1 suma a[3] / Hebra 1 suma a
[4] / Hebra 1 suma a[5] / Hebra 0 suma a[0] / Hebra 0
suma a[1] / Hebra 0 suma a[2] /
* Hebra 1 suma= 12
* Hebra 2 suma= 6
* Hebra 0 suma= 3
$ private-clause
Hebra 2 suma a[6] / Hebra 1 suma a[3] / Hebra 1 suma a
[4] / Hebra 1 suma a[5] / Hebra 0 suma a[0] / Hebra 0
suma a[1] / Hebra 0 suma a[2] /
* Hebra 1 suma= 12
* Hebra 2 suma= 6
* Hebra 0 suma= 3
```

Cláusulas

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	If (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

lastprivate



➤ Sintaxis:

➤ `lastprivate(list)`

➤ Combina

- la acción de `private` (con)
- la copia (al salir de región paralela) del último valor (en una ejecución secuencial) de las variables de la lista:
 - Construcción bucle: el valor en la última iteración
 - Construcción `sections`: el valor que tenga tras la última sección

```
#include <stdio.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif

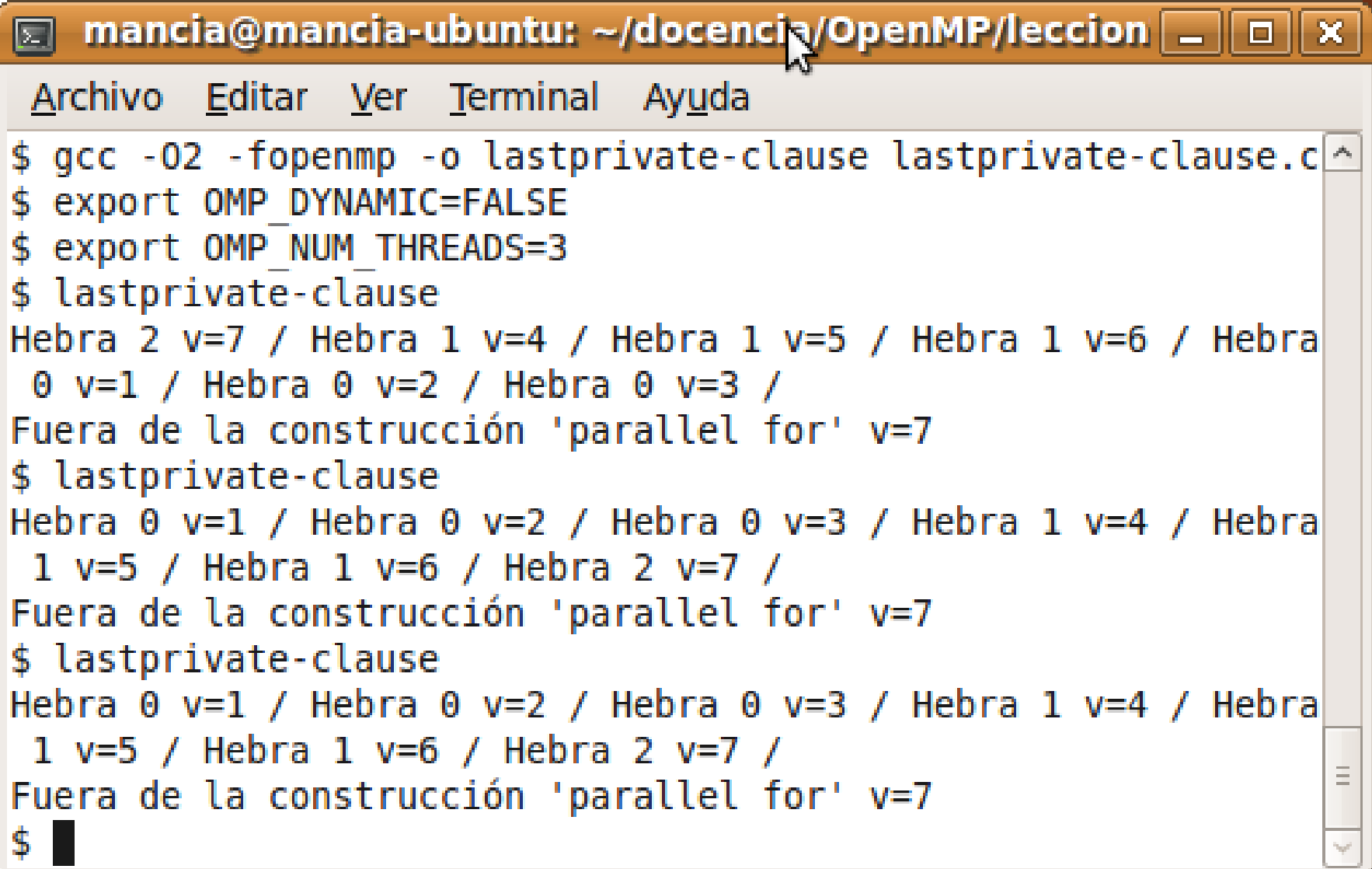
main() {
    int i, n = 7;
    int a[n], v;

    for (i=0; i<n; i++)    a[i] = i+1;

    #pragma omp parallel for lastprivate(v)
    for (i=0; i<n; i++)
    {
        v = a[i];
        printf("thread %d v=%d / ",
               omp_get_thread_num(), v);
    }

    printf("\nFuera de la construcción'parallel for' v=%d\n",
           v);
}
```

Cláusula lastprivate. Salida



A terminal window titled "mancia@mancia-ubuntu: ~/docencia/OpenMP/leccion" with standard window controls. The terminal shows the compilation and execution of a program using the OpenMP 'lastprivate' clause. The output displays the state of three threads (Hebra 0, 1, 2) at different points in the execution, showing how the 'lastprivate' clause updates the shared variable 'v'.

```
mancia@mancia-ubuntu: ~/docencia/OpenMP/leccion
Archivo  Editar  Ver  Terminal  Ayuda
$ gcc -O2 -fopenmp -o lastprivate-clause lastprivate-clause.c
$ export OMP_DYNAMIC=FALSE
$ export OMP_NUM_THREADS=3
$ lastprivate-clause
Hebra 2 v=7 / Hebra 1 v=4 / Hebra 1 v=5 / Hebra 1 v=6 / Hebra
 0 v=1 / Hebra 0 v=2 / Hebra 0 v=3 /
Fuera de la construcción 'parallel for' v=7
$ lastprivate-clause
Hebra 0 v=1 / Hebra 0 v=2 / Hebra 0 v=3 / Hebra 1 v=4 / Hebra
 1 v=5 / Hebra 1 v=6 / Hebra 2 v=7 /
Fuera de la construcción 'parallel for' v=7
$ lastprivate-clause
Hebra 0 v=1 / Hebra 0 v=2 / Hebra 0 v=3 / Hebra 1 v=4 / Hebra
 1 v=5 / Hebra 1 v=6 / Hebra 2 v=7 /
Fuera de la construcción 'parallel for' v=7
$
```

Cláusulas

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	If (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

firstprivate



➤ Sintaxis:

➤ `firstprivate (list)`

➤ Combina

- la acción de `private` (con)
- la inicialización de las variables de la lista (al entrar en región paralela)

```
#include <stdio.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif

main() {
    int i, n = 7;
    int a[n], suma=0;

    for (i=0; i<n; i++)    a[i] = i;

    #pragma omp parallel for firstprivate(suma)
    for (i=0; i<n; i++)
    {
        suma = suma + a[i];
        printf(" thread %d suma a[%d] suma=%d \n",
            omp_get_thread_num(),i,suma);
    }

    printf("\nFuera de la construcción parallel suma=%d\n",
        suma);
}
```

Cláusula `firstprivate`. Salida

```
mancia@mancia-ubuntu: ~/docencia/OpenMP/leccion2/clausulas
Archivo  Editar  Ver  Terminal  Ayuda

$ export OMP_DYNAMIC=FALSE
$ export OMP_NUM_THREADS=3
$ gcc -O2 -fopenmp -o firstprivate-clause firstprivate-clause.c
$ firstprivate-clause
Hebra 1 suma a[3] suma=3
Hebra 1 suma a[4] suma=7
Hebra 1 suma a[5] suma=12
Hebra 0 suma a[0] suma=0
Hebra 0 suma a[1] suma=1
Hebra 0 suma a[2] suma=3
Hebra 2 suma a[6] suma=6

Fuera de la construcción parallel suma=0
$ gcc -O2 -fopenmp -o firstlastprivate-clause firstlastprivate-clause.c
$ firstlastprivate-clause
Hebra 2 suma a[6] suma=6
Hebra 1 suma a[3] suma=3
Hebra 1 suma a[4] suma=7
Hebra 1 suma a[5] suma=12
Hebra 0 suma a[0] suma=0
Hebra 0 suma a[1] suma=1
Hebra 0 suma a[2] suma=3

Fuera de la construcción parallel suma=6
$
```

➤ El segundo código usa también `lastprivate()`

Cláusulas

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	if (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

Cláusula default

```
default (none | shared)
```

- Con `none` el programador debe especificar el alcance de todas las variables usadas en la construcción
 - excepción variables `threadprivate` y los índices en Directivas `con for`
- Se pueden excluir variables del ámbito especificado en `default` con:
 - `shared, private, firstprivate, lastprivate, reduction`
- Restricción:
 - Sólo puede haber una cláusula `default`

Contenidos

- Cláusulas
- Ámbito de los datos por defecto. Cláusulas relacionadas con la compartición de datos
- Cláusulas relacionadas con comunicación/sincronización

Cláusulas relacionadas con comunicación/sincronización

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	If (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

Cláusula reduction

➤ Sintaxis:

➤ `reduction`
(operator: list)

➤ Operadores de reducción (v3.0):

C/C++	
tipo	Valor inicial variables locales
+	0
-	0
*	1
&	~0
	0
^	0
&&	1
	0

```
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
    #include <omp.h>
#else
    #define omp_get_thread_num() 0
#endif

main(int argc, char **argv) {
    int i, n=20, a[n], suma=0;

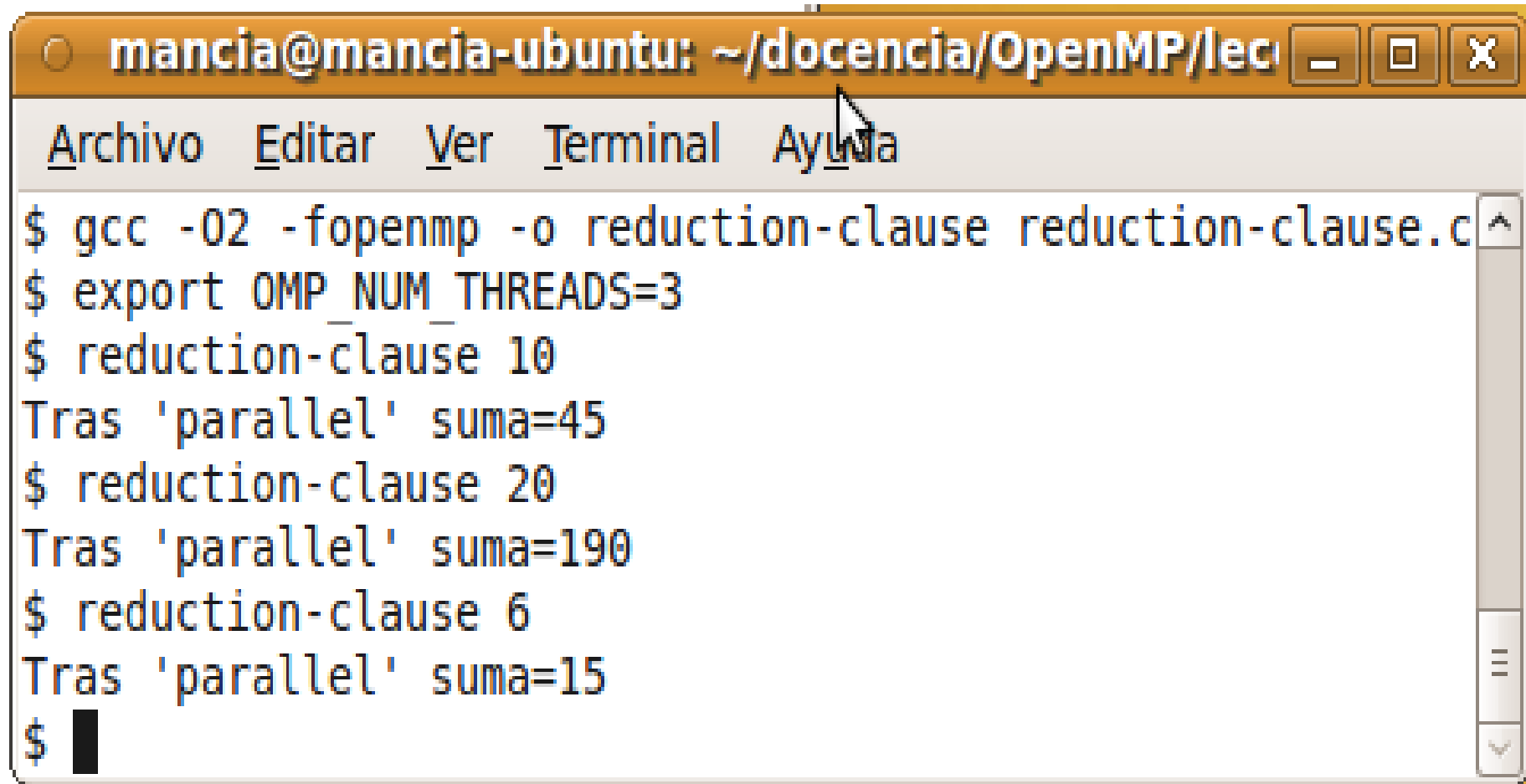
    if(argc < 2) {
        fprintf(stderr, "Falta iteraciones\n");
        exit(-1);
    }
    n = atoi(argv[1]); if (n>20) {n=20; printf("n=%d",n);}

    for (i=0; i<n; i++)    a[i] = i;

    #pragma omp parallel for reduction(+:suma)
    for (i=0; i<n; i++)    suma += a[i];

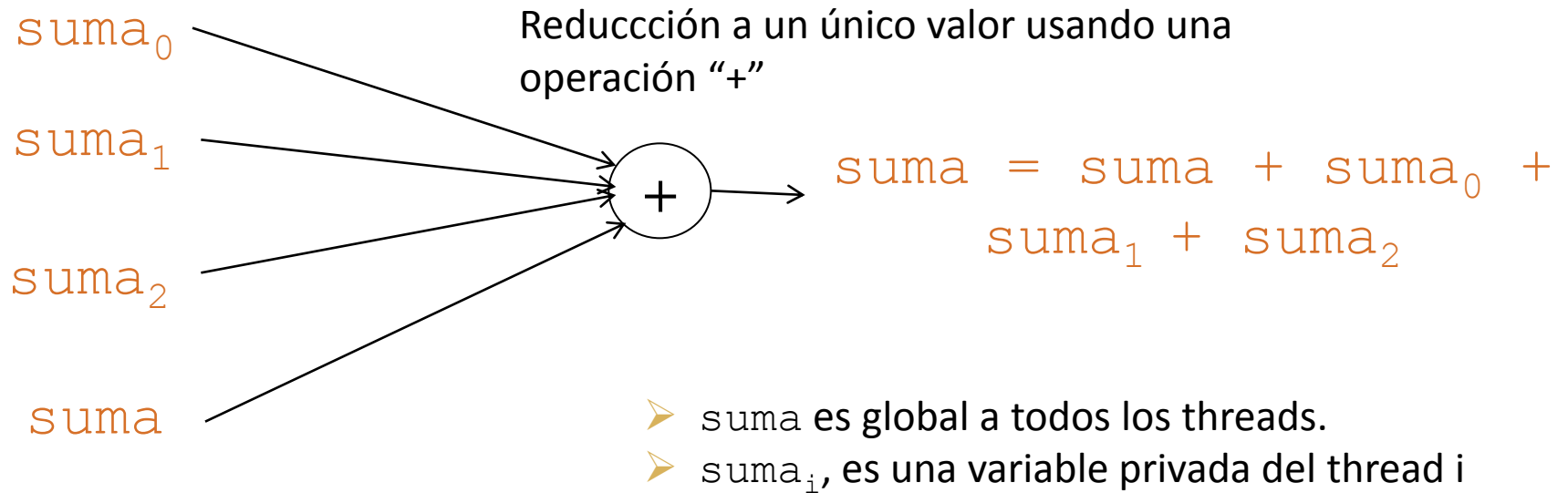
    printf("Tras 'parallel' suma=%d\n", suma);
}
```

Cláusula reduction. Salida

A terminal window titled 'mancia@mancia-ubuntu: ~/docencia/OpenMP/lec' with standard window controls. The menu bar includes 'Archivo', 'Editar', 'Ver', 'Terminal', and 'Ayuda'. The terminal shows the compilation of 'reduction-clause.c' with gcc -O2 -fopenmp, setting OMP_NUM_THREADS=3, and running the program three times with different reduction clauses (10, 20, 6). Each run outputs the sum after a parallel section. A mouse cursor is hovering over the 'Ayuda' menu item.

```
mancia@mancia-ubuntu: ~/docencia/OpenMP/lec
Archivo  Editar  Ver  Terminal  Ayuda
$ gcc -O2 -fopenmp -o reduction-clause reduction-clause.c
$ export OMP_NUM_THREADS=3
$ reduction-clause 10
Tras 'parallel' suma=45
$ reduction-clause 20
Tras 'parallel' suma=190
$ reduction-clause 6
Tras 'parallel' suma=15
$
```

Cláusula reduction



- Comunicación colectiva todos a uno (Lección 4/Tema2)

Cláusula `copyprivate` y `directiva single`

copyprivate-clause.c



- Sintaxis:
 - `copyprivate(list)`
- Sólo se puede usar con `single`
- Permite que una variable privada de un thread se copie a las variables privadas del mismo nombre del resto de threads (difusión)
- Útil para lectura (entrada) de variables

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

main() {
    int n = 9, i, b[n];

    for (i=0; i<n; i++)    b[i] = -1;

    #pragma omp parallel
    { int a;
      #pragma omp single copyprivate(a)
      {
          printf("\nIntroduce valor de inicialización a: ");
          scanf("%d", &a );
          printf("\nSingle ejecutada por el thread %d\n",
                omp_get_thread_num());
      }
      #pragma omp for
      for (i=0; i<n; i++) b[i] = a;
    }

    printf("Después de la región parallel:\n");
    for (i=0; i<n; i++) printf("b[%d] = %d\t",i,b[i]);
    printf("\n");
}
```