

Grai2º curso / 2º
cuatr.
Grado Ing. Inform.
Doble Grado Ing.
Inform. y Mat.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 2. Programación paralela II: Cláusulas OpenMP

Estudiante (nombre y apellidos): Fco Javier Merchán Martín

Grupo de prácticas: B2

Fecha de entrega: 16-042-018

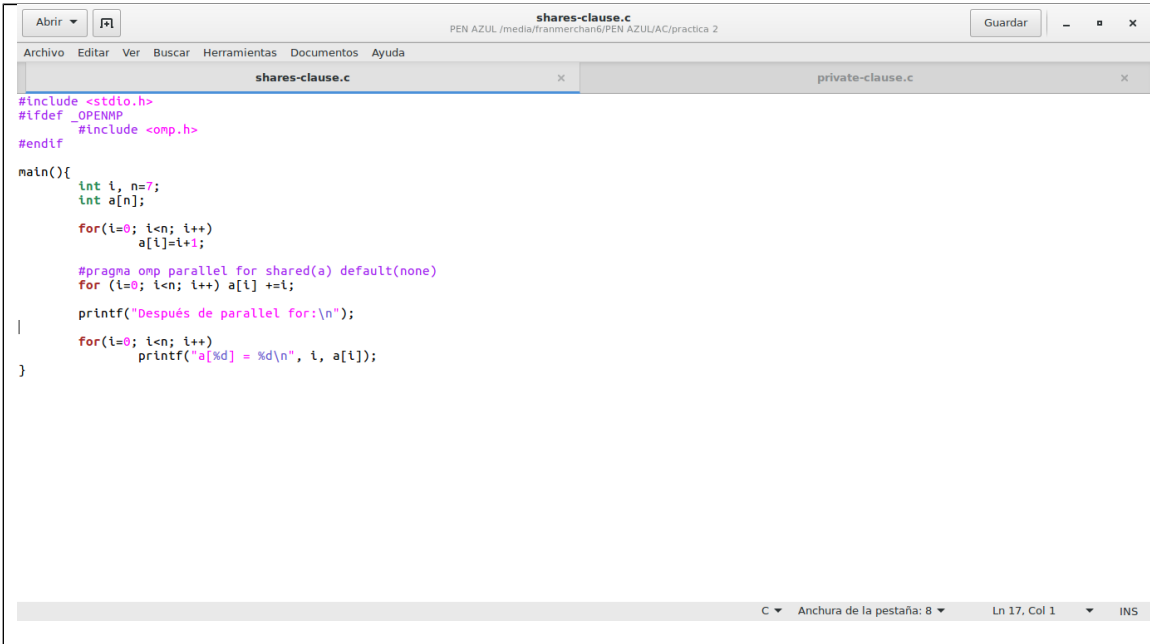
Fecha evaluación en clase:

Ejercicios basados en los ejemplos del seminario práctico

1. ¿Qué ocurre si en el ejemplo del seminario `shared-clause.c` se añade a la directiva `parallel` la cláusula `default(none)`? (añada una captura de pantalla que muestre lo que ocurre) **(b)** Resuelva el problema generado sin eliminar `default(none)`. Añada el código con la modificación al cuaderno de prácticas.

RESPUESTA: Que con el `none` se debe especificar el alcance de todas las variables, añadiendo las variables al `share` se soluciona

CAPTURA CÓDIGO FUENTE: `shared-clauseModificado.c`



```
#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) default(none)
    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]);
}
```

CAPTURAS

DE

PANTALLA:

```
#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, n) default(none)
        for (i=0; i<n; i++) a[i] +=i;

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

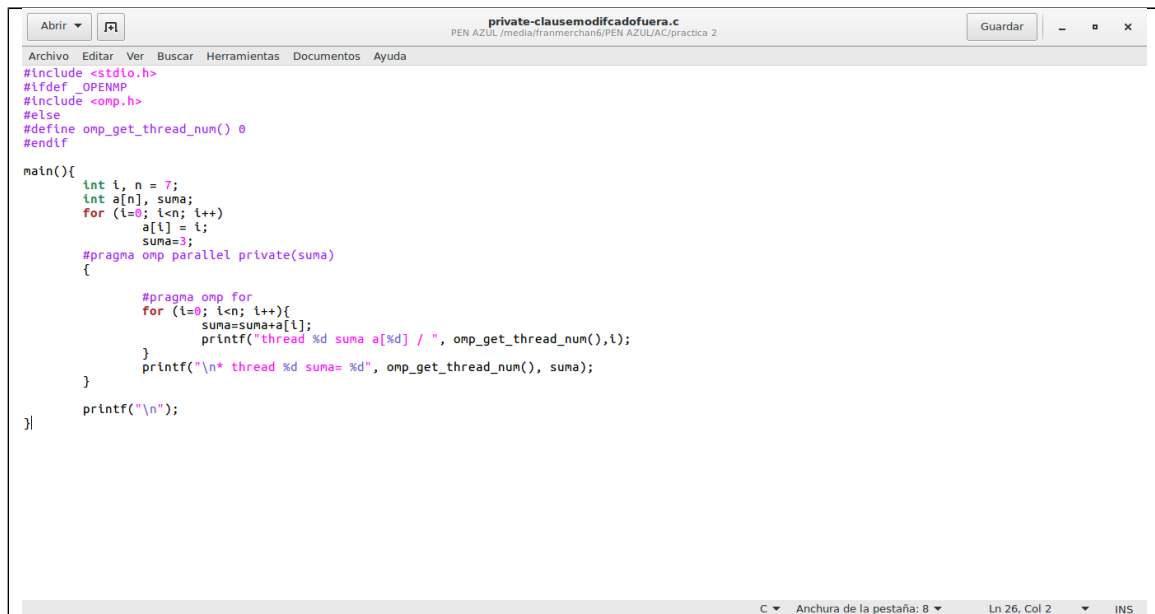
        for(i=0; i<n; i++)
            printf("a[%d] = %d\n", i, a[i]);
    }
}
```

```
[Fco Javier Merchan Martin franmerchan6@eil143080:/media/franmerchan6/PEN AZUL/AC /practica 2] 2018-04-10 martes
$gcc -O2 -o sharesmodificado1 shares-clause.c -fopenmp
shares-clause.c:6:1: warning: return type defaults to 'int' [-Wimplicit-int]
main(){
^
shares-clause.c: In function 'main':
shares-clause.c:13:10: error: 'n' not specified in enclosing parallel
    #pragma omp parallel for shared(a) default(none)
    ^
shares-clause.c:13:10: error: enclosing parallel
[Fco Javier Merchan Martin franmerchan6@eil143080:/media/franmerchan6/PEN AZUL/AC /practica 2] 2018-04-10 martes
$
```

2. ¿Qué ocurre si en private-clause.c se inicializa la variable suma fuera de la construcción parallel en lugar de dentro? (inicialice suma a un valor distinto de 0 dentro y fuera de parallel) Razone su respuesta. Añada el código con la modificación al cuaderno de prácticas.

RESPUESTA: Al no inicializarse dentro la variable coge un valor basura ya que la variable es privada en esta parte, por eso si la inicializamos dentro no. No pasa nada si la declaramos fuera siempre y cuando la inicializamos dentro.

CAPTURA CÓDIGO FUENTE: private-clauseModificado.c



```

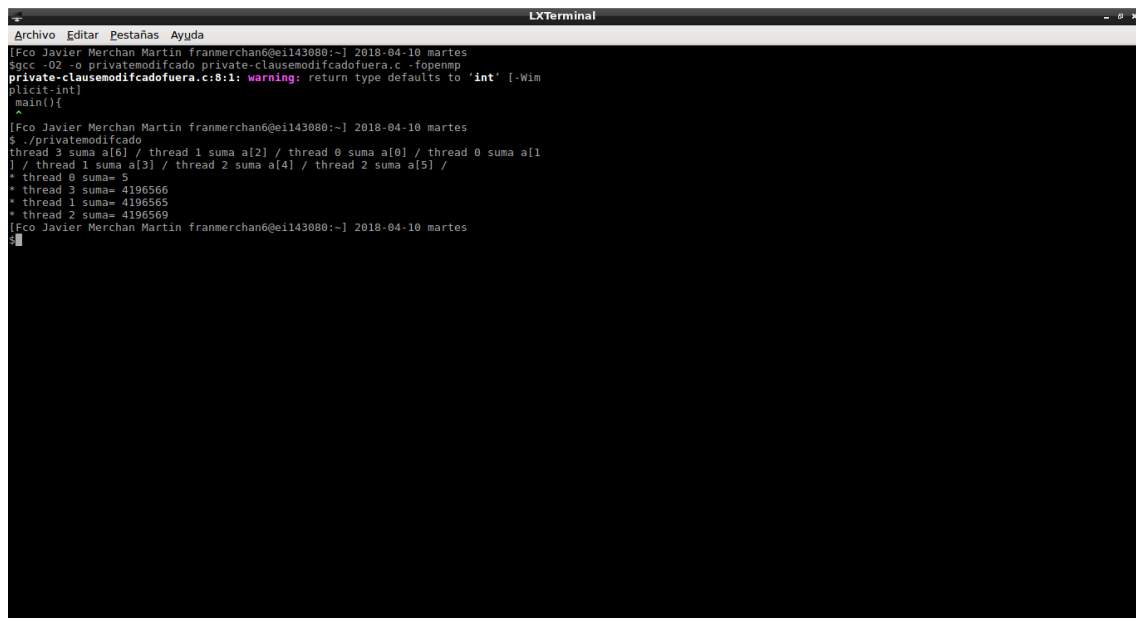
private-clausemodifcadorfuera.c
PEN AZUL /media/franmerchan6@e1143080:/PEN AZUL/AC/practica 2

Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda

#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;
        suma+=i;
        #pragma omp parallel private(suma)
        {
            #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");
    }
}
    
```

CAPTURAS DE PANTALLA: Ejecución fuera



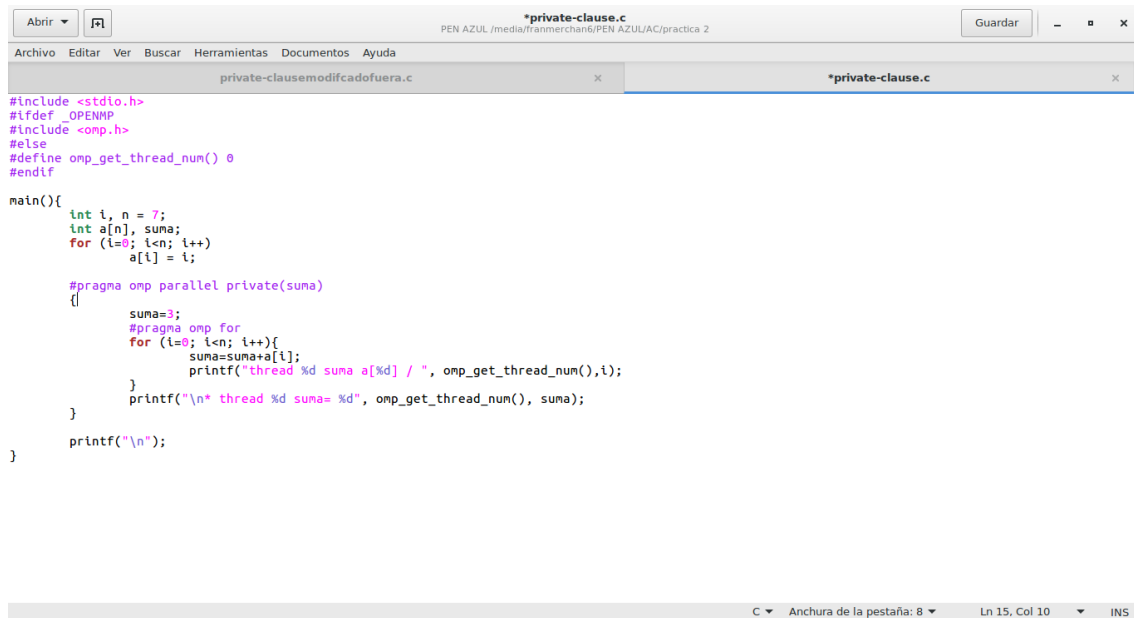
```

LXTerminal

Archivo  Editar  Pestañas  Ayuda

[fcjavier@merchan6:~]$ gcc -O2 -o privatemodifcadorfuera private-clausemodifcadorfuera.c -fopenmp
private-clausemodifcadorfuera.c:8:1: warning: return type defaults to 'int' [-Wimplicit-int]
main(){
^
[fcjavier@merchan6:~]$ ./privatemodifcadorfuera
thread 3 suma a[6] / thread 1 suma a[2] / thread 0 suma a[0] / thread 0 suma a[1]
/ thread 1 suma a[3] / thread 2 suma a[4] / thread 2 suma a[5] /
* thread 0 suma= 5
* thread 3 suma= 4196566
* thread 1 suma= 4196565
* thread 2 suma= 4196569
[fcjavier@merchan6:~]$
    
```

Código dentro



```

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

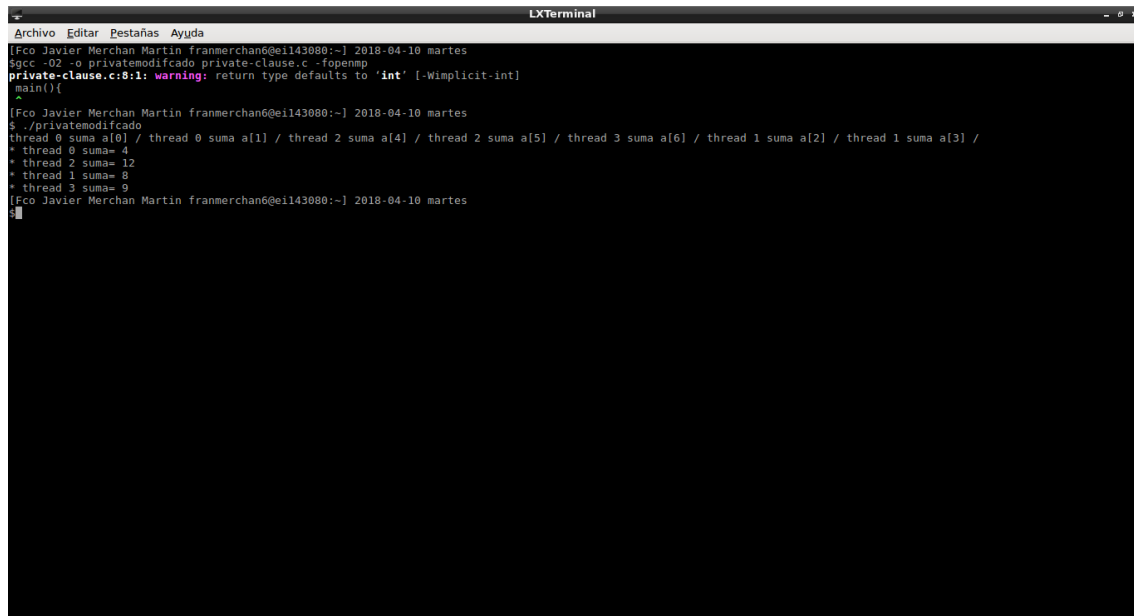
main(){
    int i, n = 7;
    int a[n], suma;
    for (i=0; i<n; i++){
        a[i] = i;
    }

    #pragma omp parallel private(suma)
    {
        suma=3;
        #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");
}

```

Ejecución dentro



```

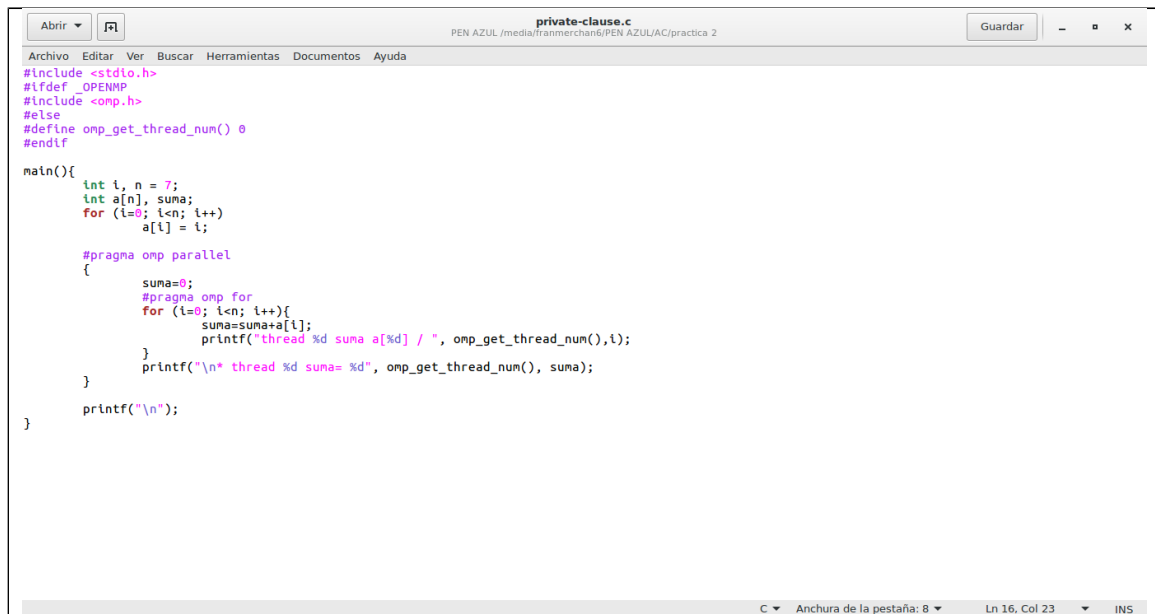
[FCo Javier Merchan Martin franmerchan6@eil43080:~] 2018-04-10 martes
$ gcc -O2 -o privatemodificado private-clause.c -fopenmp
private-clause.c:8:1: warning: return type defaults to 'int' [-Wimplicit-int]
main(){
^
[FCo Javier Merchan Martin franmerchan6@eil43080:~] 2018-04-10 martes
$ ./privatemodificado
thread 0 suma a[0] / thread 0 suma a[1] / thread 2 suma a[4] / thread 2 suma a[5] / thread 3 suma a[6] / thread 1 suma a[2] / thread 1 suma a[3] /
* thread 0 suma= 4
* thread 2 suma= 12
* thread 1 suma= 8
* thread 3 suma= 9
[FCo Javier Merchan Martin franmerchan6@eil43080:~] 2018-04-10 martes
$

```

3. ¿Qué ocurre si en `private-clause.c` se elimina la cláusula `private(suma)`? ¿A qué cree que es debido?

RESPUESTA: La variable pasaría a ser una variable compartida por todas las hebras por lo que se pisarían los datos unas a otras.

CAPTURA CÓDIGO FUENTE: `private-clauseModificado3.c`



```

private-clause.c
PEN AZUL /media/franmerchan6/PEN AZUL/AC/practica 2

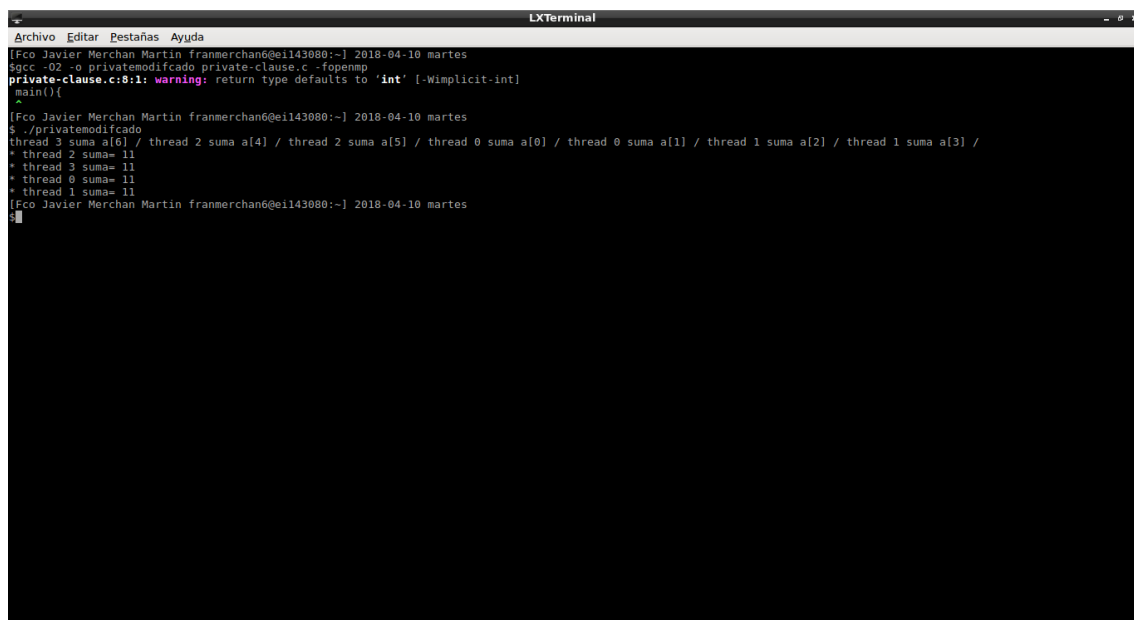
Archivo Editar Ver Buscar Herramientas Documentos Ayuda

#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
        {
            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");
    }
}
    
```

CAPTURAS DE PANTALLA:



```

LXTerminal

[FCo Javier Merchan Martin franmerchan6@ei143080:~] 2018-04-10 martes
$gcc -O2 -o privatemodificado private-clause.c -fopenmp
private-clause.c:8:1: warning: return type defaults to 'int' [-Wimplicit-int]
main(){
^
[FCo Javier Merchan Martin franmerchan6@ei143080:~] 2018-04-10 martes
$ ./privatemodificado
thread 3 suma a[6] / thread 2 suma a[4] / thread 2 suma a[5] / thread 0 suma a[0] / thread 0 suma a[1] / thread 1 suma a[2] / thread 1 suma a[3] /
* thread 2 suma= 11
* thread 3 suma= 11
* thread 0 suma= 11
* thread 1 suma= 11
[FCo Javier Merchan Martin franmerchan6@ei143080:~] 2018-04-10 martes
$
    
```

- En la ejecución de `firstlastprivate.c` de la pag. 21 del seminario se imprime un 6 fuera de la región `parallel`. ¿El código imprime siempre 6 fuera de la región `parallel`? Razone su respuesta.

RESPUESTA: Devolverá el valor de la última hebra que ha ejecutado el código

CAPTURAS DE PANTALLA:

```

Archivo Editar Pestañas Ayuda
[FCo Javier Merchan Martin franmerchan6@i143080:~] 2018-04-10 martes
$gcc -O2 -o firstmodificado firstprivate-clause.c -fopenmp
firstprivate-clause.c:8:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^
[FCo Javier Merchan Martin franmerchan6@i143080:~] 2018-04-10 martes
$ ./firstmodificado
thread 2 suma a[4] suma=4
thread 2 suma a[5] suma=9
thread 3 suma a[6] suma=6
thread 0 suma a[0] suma=0
thread 0 suma a[1] suma=1
thread 1 suma a[2] suma=2
thread 1 suma a[3] suma=5
Fuera de la construcción parallel suma=0
[FCo Javier Merchan Martin franmerchan6@i143080:~] 2018-04-10 martes
$

```

5. ¿Qué se observa en los resultados de ejecución de `copyprivate-clause.c` cuando se elimina la cláusula `copyprivate(a)` en la directiva `single`? ¿A qué cree que es debido?

RESPUESTA: No se copia el valor de `a` a los demás threads, por lo que algunos de ellos estarían con un valor no válido

CAPTURA CÓDIGO FUENTE: `copyprivate-clauseModificado.c`

```

Abrir [icon]
copyprivate-clauseModificado.c
PEN AZUL /media/franmerchan/PEN_AZUL/AC/practica 2
Guardar [icon] [icon] [icon]

Archivo Editar Ver Buscar Herramientas Documentos Ayuda
#include <stdio.h>
#include <omp.h>

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

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

    #pragma omp parallel
    {
        int a;
        #pragma omp single
        {
            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");
}

```

CAPTURAS DE PANTALLA:

```

LXTerminal
Archivo Editar Pestañas Ayuda
[FCo Javier Merchan Martin franmerchan6@ei143080:~] 2018-04-10 martes
$gcc -O2 -o copymodificado copyprivate-clauseModificado.c -fopenmp
[FCo Javier Merchan Martin franmerchan6@ei143080:~] 2018-04-10 martes
$ ./copymodificado

Introduce valor de inicialización a:5

Single ejecutada por el thread 1
Después de la región paralela:
b[0]=0 b[1]=0 b[2]=0 b[3]=5 b[4]=5 b[5]=0 b[6]=0 b[7]=0 b[8]=0
[FCo Javier Merchan Martin franmerchan6@ei143080:~] 2018-04-10 martes
$

```

6. En el ejemplo `reduction-clause.c` sustituya `suma=0` por `suma=10`. ¿Qué resultado se imprime ahora? Justifique el resultado

RESPUESTA: Se imprimirá 20 ya que el contador iba desde 0 hasta 10, sumado 10 al valor inicial daría los 20 que devuelve al final del código

CAPTURA CÓDIGO FUENTE: `reduction-clauseModificado.c`

```

reduction-clauseModificado.c
PEN AZUL /media/franmerchan6/PEN AZUL/AC/practica 2
Guardar
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 9
#endif

int main(int argc, char **argv){
    int i, n=20, a[n], suma=10;

    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);
}

```

CAPTURAS DE PANTALLA:

```

fco Javier Merchan Martin franmerchan6@ei143080:~$ 2018-04-10 martes
sgcc -O2 -o reductionmodificado reduction-clauseModificado.c -fopenmp
fco Javier Merchan Martin franmerchan6@ei143080:~$ 2018-04-10 martes
$ ./reductionmodificado 5
Tras 'parallel' suma=20
fco Javier Merchan Martin franmerchan6@ei143080:~$ 2018-04-10 martes

```

7. En el ejemplo `reduction-clause.c`, elimine `reduction()` de `#pragma omp parallel for reduction(+:suma)` y haga las modificaciones necesarias para que se siga realizando la suma de los componentes del vector `a` en paralelo sin usar directivas de trabajo compartido.

RESPUESTA: Al no usar la directivas de trabajo compartido, se debe tener una forma de separar el código para que cada thread pueda hacer su parte.

CAPTURA CÓDIGO FUENTE: `reduction-clauseModificado7.c`

```

#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#define omp_get_num_threads() 1
#endif

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

    if(argc < 2){
        fprintf(stderr, "Falta iteraciones\n");
        exit(-1);
    }

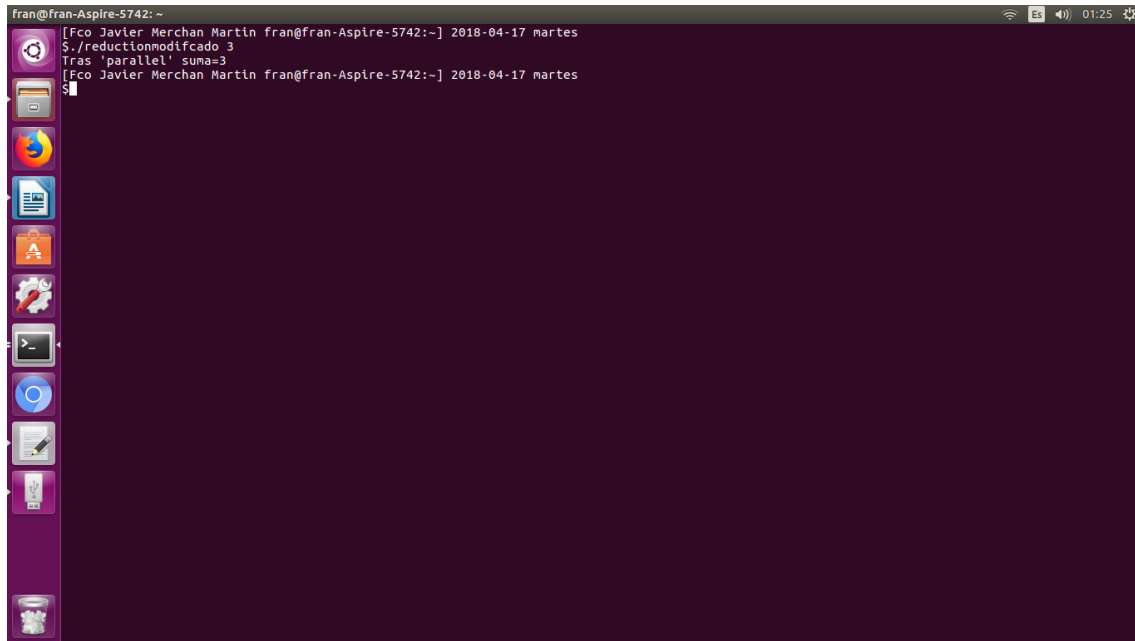
    n=atoi(argv[1]);
    if(n>20){
        n=20;
        printf("n=%d",n);
    }

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

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

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

```


CAPTURAS DE PANTALLA:**Resto de ejercicios**

8. Implementar un programa secuencial en C que calcule el producto de una matriz cuadrada, M, por un vector, v1 (implemente una versión para variables globales y otra para variables dinámicas, use una de estas versiones en los siguientes ejercicios):

$$v2 = M \cdot v1; \quad v2(i) = \sum_{k=0}^{N-1} M(i, k) \cdot v(k), \quad i = 0, \dots, N-1$$

NOTAS: (1) el número de filas /columnas N de la matriz deben ser argumentos de entrada al programa; (2) se debe inicializar la matriz y el vector antes del cálculo; (3) se debe asegurar que el programa calcula la suma correctamente imprimiendo todos los componentes del vector resultante, v3, para tamaños pequeños de los vectores (por ejemplo, N = 8 y N=11); (5) se debe imprimir sea cual sea el tamaño de los vectores el tiempo de ejecución del código paralelo que calcula el producto matriz vector y, al menos, el primer y último componente del resultado (esto último evita que las optimizaciones del compilador eliminen el código de la suma).

CAPTURA CÓDIGO FUENTE: pmv-secuencial.c

CAPTURAS DE PANTALLA:

9. Implementar en paralelo el producto matriz por vector con OpenMP a partir del código escrito en el ejercicio anterior usando la directiva `for`. Debe implementar dos versiones del código (consulte la lección 5/Tema 2):

- una primera que paralelice el bucle que recorre las filas de la matriz y
- una segunda que paralelice el bucle que recorre las columnas.

Use las directivas que estime oportunas y las cláusulas que sean necesarias **excepto la cláusula `reduction`**. Se debe paralelizar también la inicialización de las matrices. Respecto a este ejercicio:

- Anote en su cuaderno de prácticas todos los errores de compilación que se han generado durante la realización del ejercicio y explique cómo los ha resuelto (especifique qué ayudas externas ha usado o recibido).

- Anote todos los errores en tiempo de ejecución que se han generado durante la realización del ejercicio y explique cómo los ha resuelto (especifique qué ayudas externas ha usado o recibido).

NOTAS: (1) el número de filas /columnas N de la matriz deben ser argumentos de entrada; (2) se debe inicializar la matriz y el vector antes del cálculo; (3) se debe asegurar que el programa calcula la suma correctamente imprimiendo todos los componentes del vector resultante, v3, para tamaños pequeños de los vectores (por ejemplo, N = 8 y N=11); (5) se debe imprimir sea cual sea el tamaño de los vectores el tiempo de ejecución del código que calcula el producto matriz vector y, al menos, el primer y último componente del resultado (esto último evita que las optimizaciones del compilador eliminen el código de la suma).

CAPTURA CÓDIGO FUENTE : pmv-OpenMP-a.c

CAPTURA CÓDIGO FUENTE: pmv-OpenMP-b.c

RESPUESTA:

CAPTURAS DE PANTALLA:

10. A partir de la segunda versión de código paralelo desarrollado en el ejercicio anterior, implementar una versión paralela del producto matriz por vector con OpenMP que use para comunicación/sincronización la cláusula `reduction`. Respecto a este ejercicio:

- Anote en su cuaderno de prácticas todos los errores de compilación que se han generado durante la realización del ejercicio y explique cómo los ha resuelto (especifique qué ayudas externas ha usado o recibido).
- Anote todos los errores en tiempo de ejecución que se han generado durante la realización del ejercicio y explique cómo los ha resuelto (especifique qué ayudas externas ha usado o recibido).

CAPTURA CÓDIGO FUENTE: pmv-OpenmMP-reduction.c

RESPUESTA:

CAPTURAS DE PANTALLA:

11. Ayudándose de una hoja de cálculo (recuerde que en las aulas está instalado OpenOffice) realice una tabla y una gráfica que permitan comparar la escalabilidad (ganancia en velocidad en función del número de cores) en atcgrid y en su PC del mejor código paralelo de los tres implementados en los ejercicios anteriores para dos tamaños (N) distintos (consulte la Lección 6/Tema 2). Usar `-O2` al compilar. Justificar por qué el código escogido es el mejor. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

CAPTURAS DE PANTALLA (que justifique el código elegido):

TABLA Y GRÁFICA (por ejemplo para 1-4 threads PC local, y para 1-12 threads en atcgrid, tamaños-N-: un N entre 30000 y 100000, y otro entre 5000 y 30000):

COMENTARIOS SOBRE LOS RESULTADOS: