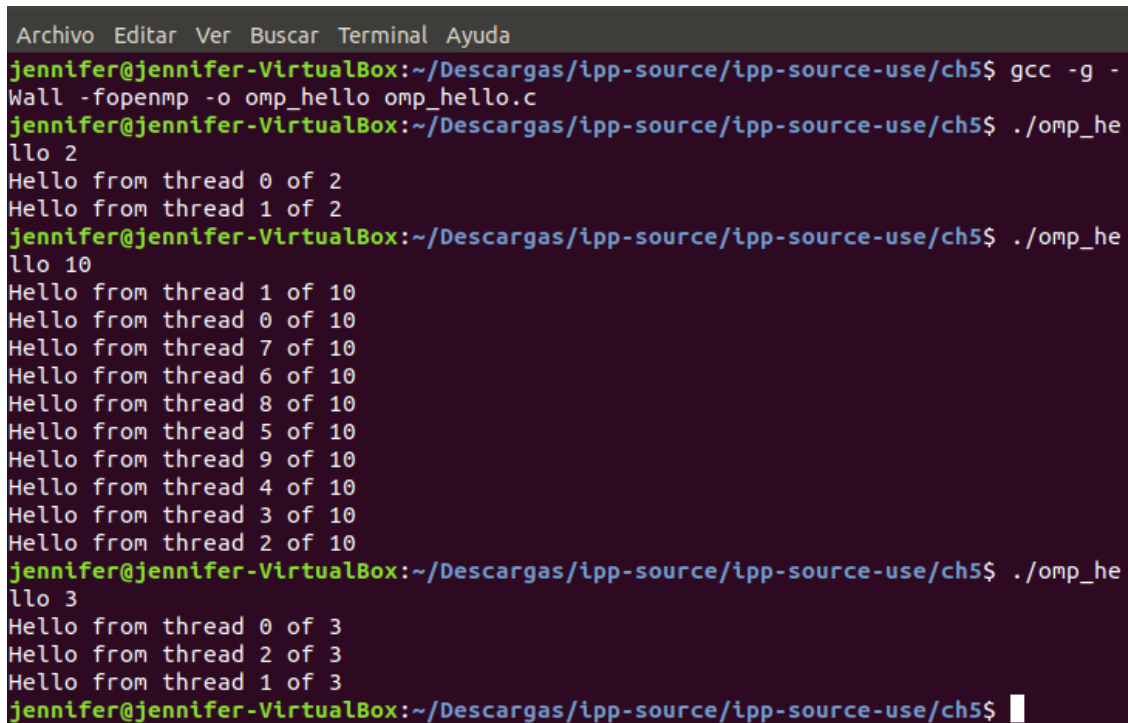


1. Compilar y correr el programa "omp hello.c".

- Correr el programa varias veces.



```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ gcc -g -Wall -fopenmp -o omp_hello omp_hello.c
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_hello 2
Hello from thread 0 of 2
Hello from thread 1 of 2
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_hello 10
Hello from thread 1 of 10
Hello from thread 0 of 10
Hello from thread 7 of 10
Hello from thread 6 of 10
Hello from thread 8 of 10
Hello from thread 5 of 10
Hello from thread 9 of 10
Hello from thread 4 of 10
Hello from thread 3 of 10
Hello from thread 2 of 10
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_hello 3
Hello from thread 0 of 3
Hello from thread 2 of 3
Hello from thread 1 of 3
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(a) Ejecución del programa "omp hello.c".

- Explicar por qué la salida del programa varía: varía ya que no existe una barrera que espere por la finalización de todos los hilos, para luego imprimirlos en consola. Así que cada uno de ellos muestra su resultado apenas culmina su función. Entre más hilos se creen, se notará más la condición de carrera.

2. Compilar y correr el programa "omp fibo.c".

```
Archivo Editar Ver Buscar Terminal Ayuda
0      1
1      1
2      2
3     38
4     40
5    288230376923725886
6    288230376923725857
7     33605093
8    288230574525826548
9    288230574492222006
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_fi
bo 5 10
The first n Fibonacci numbers:
0      1
1      1
2      2
3      3
4     38
5     39
6     77
7    116
8    288230574525826548
9    288230574492222006
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(b) Ejecución del programa "omp fibo.c".

- Anotar si el programa despliega el resultado correctamente: Despliega los números iniciales correctamente. Los demás no.
- Explique la razón por lo que usted piensa que el programa no funciona: el programa, al igual que el anterior presenta una condición de carrera. El diseño del código hace que solo se inicialicen los primeros 2 "campos" del long array, así se toman el número 2 como dígito inicial y se realiza la suma de sus dos anteriores para continuar con la sucesión de Fibonacci. El problema ocurre cuando, por ejemplo, un hilo "i" ingresa antes de que el hilo "i-1" haya hecho su suma y por ende, la inicialización de su "campo". Este hilo no encontraría un anterior inicializado, solo encontraría "basura" y seguiría su proceso.
- Anote el respaldo teórico de su razonamiento: ya que C y C++ no inicializan las variables con algún valor por defecto, cuando a estas se les asigna una ubicación de memoria, el valor predeterminado de esa variable es cualquier valor (basura) que ya esté en esa ubicación de memoria. Por eso todos los "campos" no inicializados del long array tendrán como valor esa "basura" hasta que algún hilo lo inicialice.

3. Compilar y correr el programa "omp private.c".

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
omp_private.c: In function 'main._omp_fn.0':
omp_private.c:27:7: warning: 'x' is used uninitialized in this function [-Wunini
tialized]
    printf("Thread %d > before initialization, x = %d\n",
           ^
           my_rank, x);
           ^
omp_private.c:20:8: note: 'x' was declared here
    int x = 5;
    ^
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_pr
ivate 5
Thread 1 > before initialization, x = 0
Thread 1 > after initialization, x = 4
Thread 4 > before initialization, x = 0
Thread 4 > after initialization, x = 10
Thread 3 > before initialization, x = 0
Thread 3 > after initialization, x = 8
Thread 2 > before initialization, x = 0
Thread 2 > after initialization, x = 6
Thread 0 > before initialization, x = 0
Thread 0 > after initialization, x = 2
After parallel block, x = 5
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$

```

(c) Ejecución del programa "omp private.c".

- Explique el funcionamiento de hacer la variable x privada: al hacer x privada, se logra que cada hilo tenga "su propia x". Esto significa que la variable no está siendo compartida, su valor no es "comunicado" a los demás hilos, permitiendo que estos la utilicen a conveniencia, como por ejemplo, realizar un calculo cada uno, sin preocuparse por que los otros cambien ese valor.

#### 4. Compilar y correr el programa "omp trap1.c".

- Correr el programa para comprobar el resultado.

```

jennifer@jennifer-VirtualBox:~$ cd ~/Descargas/ipp-source/ipp-source-use/ch5
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ gcc -g -
Wall -fopenmp -o omp_trap1 omp_trap1.c
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_tr
ap1 50
Enter a, b, and n
32 64 150
With n = 150 trapezoids, our estimate
of the integral from 32.000000 to 64.000000 = 7.64589093925926e+04

```

(d) Ejecución del programa "omp trap1.c".

- Escoja valores para los datos y anótelos para las demás pruebas: número de hilos = 10. A = 3, B = 8, N = 10.
- Anote el valor del resultado del programa: 1.61875000000000e+02.
- Elimine "pragma omp critical"
- Compruebe si el resultado es el mismo: no.

- Comente sobre el problema encontrado: el pragma omp critical especifica un bloque de código en el cual el acceso está restringido por solo un hilo a la vez. En el programa al quitar el critical se pone en riesgo la variable global que almacena el resultado, ya que puede ser aumentada más de una vez por la condición de carrera de los hilos.

5. Compilar y correr el programa "omp trap2a.c".

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_trap2a 10
Enter a, b, and n
3 8 10
With n = 10 trapezoids, our estimate
of the integral from 3.000000 to 8.000000 = 1.61875000000000e+02
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$

```

(e) Ejecución del programa "omp trap2a.c".

- Utilice los mismos datos que en el caso anterior: número de hilos = 10.  $A = 3$ ,  $B = 8$ ,  $N = 10$ .
- Anote si el resultado es correcto: sí.

6. Compilar y correr el programa "omp trap2b.c".

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ gcc -g -Wall -fopenmp -o omp_trap2b omp_trap2b.c
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_trap2b 10
Enter a, b, and n
3 8 10
With n = 10 trapezoids, our estimate
of the integral from 3.000000 to 8.000000 = 1.61875000000000e+02
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$

```

(f) Ejecución del programa "omp trap2b.c".

- Utilice los mismos datos que en el caso anterior: número de hilos = 10.  $A = 3$ ,  $B = 8$ ,  $N = 10$ .
- Anote si el resultado es correcto: sí.

7. Compilar y correr el programa "omp trap3.c".

```
Archivo Editar Ver Buscar Terminal Ayuda
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ gcc -g -
Wall -fopenmp -o omp_trap3 omp_trap3.c
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_tr
ap3 10
Enter a, b, and n
3 8 10
With n = 10 trapezoids, our estimate
of the integral from 3.000000 to 8.000000 = 1.618750000000000e+02
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(g) Ejecución del programa "omp trap3.c".

- Utilice los mismos datos que en el caso anterior: número de hilos = 10.  $A = 3$ ,  $B = 8$ ,  $N = 10$ .
- Anote si el resultado es correcto: sí.

8. Compilar y correr el programa "omp pi.c".

- Correr el programa y comprobar el resultado.

```
Archivo Editar Ver Buscar Terminal Ayuda
jennifer@jennifer-VirtualBox:~$ cd ~/Descargas/ipp-source/ipp-source-use/ch5
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ gcc -g -
Wall -fopenmp -o omp_pi omp_pi.c -lm
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_pi
5 60
With n = 60 terms and 5 threads,
  Our estimate of pi = 3.12492714392900
                    pi = 3.14159265358979
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(h) Ejecución del programa "omp pi.c".

- Explicar la manera que se realiza la acumulación de la suma: para realizar la suma acumulativa se declara una variable global sum, la cual tomará el resultado del reduction. Al declarar el pragma omp parallel for, también se introduce el reduction, con la variable sum. Este comando hace que dentro de las iteraciones del bucle for entre los hilos, cada hilo tenga su propia copia local de la variable de reducción, en este caso sum. Cada hilo modifica solo la copia local de esta variable. Por lo tanto, no hay race condition. Cuando los hilos se unen, todas las copias locales de la variable de reducción se combinan en la variable global sum.

9. Compilar y correr el programa "bubble.c".

- Correr el programa.
- Agregue datos y compruebe el resultado:

```
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ gcc -g -Wall -o bubble bubble.c
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./bubble
usage:  ./bubble <n> <g|i>
      n:  number of elements in list
      'g': generate list using a random number generator
      'i': user input list
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./bubble
50 g
Before sort:
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62
23 67 35 29 2 22 58 69 67 93 56 11 42 29 73 21 19 84 37 98 24 15 70

After sort:
2 11 11 15 15 19 21 21 22 23 24 26 26 27 29 29 29 30 35 35 36 37 40 42 49 56 58
59 62 62 63 67 67 67 68 69 70 72 73 77 82 83 84 86 86 90 92 93 93 98

jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(i) Ejecución del programa "bubble.c".

- Utilice el comando "time" para estimar el tiempo que le toma para realizar el cálculo:

```
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ gcc -g -Wall -fopenmp -o bubble bubble.c
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./bubble
50 g
Before sort:
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62
23 67 35 29 2 22 58 69 67 93 56 11 42 29 73 21 19 84 37 98 24 15 70

After sort:
2 11 11 15 15 19 21 21 22 23 24 26 26 27 29 29 29 30 35 35 36 37 40 42 49 56 58
59 62 62 63 67 67 67 68 69 70 72 73 77 82 83 84 86 86 90 92 93 93 98

Tiempo transcurrido: 0.000353 jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(j) Ejecución del programa "bubblec" con la función "omp\_get\_wtime".

- Corra el comando "valgrind ./bubble".

```
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ valgrind
./bubble
==2585== Memcheck, a memory error detector
==2585== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2585== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==2585== Command: ./bubble
==2585==
usage:  ./bubble <n> <g|i>
      n:  number of elements in list
      'g': generate list using a random number generator
      'i': user input list
==2585==
==2585== HEAP SUMMARY:
==2585==    in use at exit: 8 bytes in 1 blocks
==2585==  total heap usage: 2 allocs, 1 frees, 32,824 bytes allocated
==2585==
==2585== LEAK SUMMARY:
==2585==    definitely lost: 0 bytes in 0 blocks
==2585==    indirectly lost: 0 bytes in 0 blocks
==2585==    possibly lost: 0 bytes in 0 blocks
==2585==    still reachable: 8 bytes in 1 blocks
==2585==    suppressed: 0 bytes in 0 blocks
==2585== Rerun with --leak-check=full to see details of leaked memory
==2585==
```

(k) Ejecución del comando "valgrind" con el programa "omp bubble.c".

- Interprete la salida y anote los errores que encuentre: el comando revisa la memoria ocupada por el programa, desde los bytes ocupados hasta la información que puede ser perdida. Se encontraron 0 errores.

#### 10. Compilar y correr el programa "omp odd even1.c".

- Correr el programa.
- Agregue datos y compruebe la salida:

```
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_od
d_even1 5 60 g
Before sort:
4289383 6930886 1692777 4636915 7747793 4238335 9885386 9760492 6516649 9641421
5202362 490027 3368690 2520059 4897763 7513926 5180540 383426 4089172 3455736 50
05211 1595368 4702567 6956429 6465782 1021530 8722862 3665123 5174067 8703135 15
13929 1979802 5634022 5723058 9133069 5898167 9961393 9018456 8175011 6478042 11
76229 3377373 9484421 4544919 8413784 6898537 4575198 3594324 9798315 8664370 95
66413 4803526 2776091 4268980 1759956 9241873 7806862 2999170 2906996 5497281

After sort:
383426 490027 1021530 1176229 1513929 1595368 1692777 1759956 1979802 2520059 27
76091 2906996 2999170 3368690 3377373 3455736 3594324 3665123 4089172 4238335 42
68980 4289383 4544919 4575198 4636915 4702567 4803526 4897763 5005211 5174067 51
80540 5202362 5497281 5634022 5723058 5898167 6465782 6478042 6516649 6898537 69
30886 6956429 7513926 7747793 7806862 8175011 8413784 8664370 8703135 8722862 90
18456 9133069 9241873 9484421 9566413 9641421 9760492 9798315 9885386 9961393
```

(l) Ejecución del programa "omp odd even1.c".

- Utilice el comando "time" para estimar el tiempo que le toma para realizar el cálculo, haga una comparación con "bubble.c" y anote los resultados: el resultado de ordenar una lista de



50 elementos con "bubble.c", se realiza en 0.000353. El mismo proceso con "odd even1" se realiza en 3.733422e-03.

```
jennifer@jennifer-VirtualBox:~$ cd ~/Descargas/ipp-source/ipp-source-use/ch5
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_odd_even1 5 50 g
Before sort:
4289383 6930886 1692777 4636915 7747793 4238335 9885386 9760492 6516649 9641421
5202362 490027 3368690 2520059 4897763 7513926 5180540 383426 4089172 3455736 50
05211 1595368 4702567 6956429 6465782 1021530 8722862 3665123 5174067 8703135 15
13929 1979802 5634022 5723058 9133069 5898167 9961393 9018456 8175011 6478042 11
76229 3377373 9484421 4544919 8413784 6898537 4575198 3594324 9798315 8664370

After sort:
383426 490027 1021530 1176229 1513929 1595368 1692777 1979802 2520059 3368690 33
77373 3455736 3594324 3665123 4089172 4238335 4289383 4544919 4575198 4636915 47
02567 4897763 5005211 5174067 5180540 5202362 5634022 5723058 5898167 6465782 64
78042 6516649 6898537 6930886 6956429 7513926 7747793 8175011 8413784 8664370 87
03135 8722862 9018456 9133069 9484421 9641421 9760492 9798315 9885386 9961393

Elapsed time = 3.733422e-03 seconds
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(m) Ejecución del programa "omp odd even1.c" con el cálculo del tiempo.

# 11. Compilar y correr el programa "omp sin sum.c".

- Correr el programa con 1 hilo y 10 000 términos:

```
jennifer@jennifer-VirtualBox:~/Descargas$ ./omp_sin_sum 1 10000
Result = 1.88892327977904e+00
Check = 1.88892327977966e+00
With n = 10000 terms, the error is 6.23945339839338e-13
Elapsed time = 4.590527e+00 seconds
jennifer@jennifer-VirtualBox:~/Descargas$
```

(n) Ejecución del programa "omp sin sum.c" con 1 hilo.

- Anote el tiempo: 4.590527e+00 segundos
- Correr el programa con 2 hilos y 10 000 términos:

```
jennifer@jennifer-VirtualBox:~/Descargas$ ./omp_sin_sum 2 10000
Result = 1.88892327977902e+00
Check = 1.88892327977905e+00
With n = 10000 terms, the error is 2.30926389122033e-14
Elapsed time = 4.802735e+00 seconds
jennifer@jennifer-VirtualBox:~/Descargas$
```

(o) Ejecución del programa "omp sin sum.c" con 2 hilos.

- Anote el tiempo: 4.802735e+00 segundos.
- Calcule el "speedup": Tiempo serial / tiempo paralelo:  $4.590527 / 4.802735 = 0.9558151761$ .



- Note que el programa tiene "schedule(runtime)". Escriba en el shell, antes de correr el programa: `OMP_SCHEDULE="dynamic"` (pruebe con "auto" y "static" también)
- Correr el programa de nuevo y anote las diferencias en los tres casos indicados:

```
jennifer@jennifer-VirtualBox:~/Descargas$ OMP_SCHEDULE=dynamic
jennifer@jennifer-VirtualBox:~/Descargas$ ./omp_sin_sum 2 10000
Result = 1.88892327977889e+00
Check = 1.88892327977905e+00
With n = 10000 terms, the error is 1.54543045027822e-13
Elapsed time = 4.573496e+00 seconds
jennifer@jennifer-VirtualBox:~/Descargas$
```

(p) Ejecución del programa "omp sin sum.c" con el comando `OMP_SCHEDULE=dynamic`.

```
jennifer@jennifer-VirtualBox:~/Descargas$ OMP_SCHEDULE=auto
jennifer@jennifer-VirtualBox:~/Descargas$ ./omp_sin_sum 2 10000
Result = 1.88892327977898e+00
Check = 1.88892327977905e+00
With n = 10000 terms, the error is 6.57252030578093e-14
Elapsed time = 4.673449e+00 seconds
jennifer@jennifer-VirtualBox:~/Descargas$
```

(q) Ejecución del programa "omp sin sum.c" con el comando `OMP_SCHEDULE=auto`.

```
jennifer@jennifer-VirtualBox:~/Descargas$ OMP_SCHEDULE=static
jennifer@jennifer-VirtualBox:~/Descargas$ ./omp_sin_sum 2 10000
Result = 1.88892327977908e+00
Check = 1.88892327977905e+00
With n = 10000 terms, the error is 3.37507799486048e-14
Elapsed time = 4.311205e+00 seconds
jennifer@jennifer-VirtualBox:~/Descargas$
```

(r) Ejecución del programa "omp sin sum.c" con el comando `OMP_SCHEDULE=static`.

## 12 Cambiar al directorio "omp msg".

- Compilar y correr "omp msgps.c"
- Analice la salida y describa que está sucediendo: cada hilo creado envía un mensaje, con un valor de -(cantidad de mensajes por enviar) a 0. Esta acción la repite las veces que se ingresó como parámetro. Cada hilo envía esta cantidad de mensajes a algún hilo existente, es decir, de los hilos creados, elige un destinatario al azar.

```
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5/omp_msg$
./omp_msgps 3 8
Thread 2 > received -2 from 2
Thread 2 > received 0 from 0
Thread 0 > received 0 from 2
Thread 0 > received -3 from 2
Thread 0 > received -4 from 2
Thread 0 > received -5 from 2
Thread 0 > received -6 from 2
Thread 0 > received -1 from 0
Thread 0 > received -3 from 0
Thread 0 > received -4 from 0
Thread 0 > received -7 from 0
Thread 1 > received -1 from 2
Thread 1 > received -7 from 2
Thread 1 > received -2 from 0
Thread 0 > received -1 from 1
Thread 0 > received -2 from 1
Thread 2 > received -5 from 0
Thread 2 > received -6 from 0
Thread 2 > received 0 from 1
Thread 2 > received -3 from 1
```

(s) Ejecución del programa "omp msgps.c".

- Busque e indique la funcionalidad de "barrier" y "atomic" en este programa: pragma omp barrier especifica un punto en el código donde cada hilo debe esperar hasta que lleguen todos los hilos creados, en este caso, todos deben esperar hasta que todas las colas de mensajes de cada uno de ellos sea creada. Pragma omp atomic se asegura de que una ubicación de memoria específica se actualice atómicamente, evitando múltiples y simultáneas lecturas y escrituras de hilos, en este caso se utiliza para que la variable que actualiza si un hilo terminó o no, sea aumentada un hilo a la vez (apenas terminen de enviar los mensajes).

13. Compilar y correr el programa "omp mat vect.c".

- Correr el programa y anotar los datos de corrida:

```
Archivo Editar Ver Buscar Terminal Ayuda
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_ma
t_vect 2 8 8
Elapsed time = 1.331370e-04 seconds
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ gcc -g -
Wall -o omp_mat_vect omp_mat_vect.c -lpthread -fopenmp
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_ma
t_vect 2 8 8
Elapsed time = 6.286600e-05 seconds
The product is
2.1 2.1 1.3 1.4 2.0 2.2 1.8 1.5
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(t) Primera ejecución del programa "omp mat vect.c".

- Cambiar alguno de los parámetros y anote los resultados: primera ejecución con 2 hilos, m y n con valor 8. Segunda ejecución con 5 hilos, m y n con valor 10.

```
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$ ./omp_ma  
t_vect 5 10 10  
Elapsed time = 2.508000e-04 seconds  
The product is  
3.2 3.1 2.1 3.0 3.2 1.9 3.6 3.3 3.0 4.2  
jennifer@jennifer-VirtualBox:~/Descargas/ipp-source/ipp-source-use/ch5$
```

(u) Segunda ejecución del programa "omp mat vect.c".