

# Arquitectura de Computadores (AC)

## Cuaderno de prácticas.

### Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Jorge Gangoso Klöck

Grupo de prácticas y profesor de prácticas: D2

Fecha de entrega: 02/03/2020

Fecha evaluación en clase: 03/03/2020

Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con los normas de prácticas que se encuentra en SWAD

## Parte I. Ejercicios basados en los ejemplos del seminario práctico

Crear el directorio con nombre `bp0` en `atcgrid` y en el PC local.

**NOTA:** En las prácticas se usa `slurm` como gestor de colas. Consideraciones a tener en cuenta:

- Slurm está configurado para asignar recursos a los procesos (llamados *tasks* en slurm) a nivel de core físico. Esto significa que por defecto slurm asigna un core a un proceso, para asignar más de uno se debe usar con `sbatch/srun` la opción `--cpus-per-task`.
- En slurm, por defecto, `cpu` se refiere a cores lógicos (ej. en la opción `--cpus-per-task`), si no se quieren usar cores lógicos hay que añadir la opción `--hint=nomultithread` a `sbatch/srun`.
- Para asegurar que solo se crea un proceso hay que incluir `-n1` en `sbatch/srun`.
- Para que no se ejecute más de un proceso en un nodo de `atcgrid` hay que usar `--exclusive` con `sbatch/srun` (se recomienda no utilizarlo en los `srun` dentro de un script).
- Los `srun` dentro de un script heredan las opciones fijadas en el `sbatch` que se usa para enviar el script a la cola slurm.

1. Ejecutar `lscpu` en el PC y en un nodo de cómputo de `atcgrid`. (Crear directorio `ejer1`)

(a) Mostrar con capturas de pantalla el resultado de estas ejecuciones.

### RESPUESTA:

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0] 2020-02-25 martes
lscpu
Architectura: x86_64
modo(s) de operación de las CPUs: 32-bit, 64-bit
Orden de los bytes: Little Endian
CPU(s): 4
Lista de la(s) CPU(s) en línea: 0-3
Hilo(s) de procesamiento por núcleo: 1
Núcleo(s) por «socket»: 4
«Socket(s)»: 1
Modo(s) NUMA: 1
ID de fabricante: GenuineIntel
Familia de CPU: 60
Modelo: 60
Nombre del modelo: Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz
Revisión: 3
CPU MHz: 798.700
CPU MHz máx.: 3400,0000
CPU MHz mín.: 800,0000
BogoMIPS: 6385.64
Virtualización: VT-x
Cache L1d: 32K
Cache L1i: 32K
Cache L2: 256K
Cache L3: 6144K
CPU(s) del nodo NUMA 0: 0-3
Indicadores: fpu vme de pse tsc msr pae mce cx8 apic sep mtr
r pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop
_tsc cpuid aperfperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma
cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx f16c rdrand lahf
f_lm abm cpuid_fault epb invpcid_single pti tpr_shadow vnni flexpriority ept vpid fs
gsbase tsc_adjust bmt1 avx2 smep bmt2 erms invpcid xsaveopt dtherm ida arat pln pts
[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0] 2020-02-25 martes

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[d2estudiante8@atcgrid:~] 2020-02-25 martes
srun -p ac -n1 lscpu
Architectura: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 24
On-line CPU(s) list: 0-23
Thread(s) per core: 2
Core(s) per socket: 6
Socket(s): 2
NUMA node(s): 2
Vendor ID: GenuineIntel
CPU family: 6
Model: 44
Model name: Intel(R) Xeon(R) CPU E5645 @ 2.40GHz
Stepping: 2
CPU MHz: 1600.000
CPU max MHz: 2401,0000
CPU min MHz: 1600,0000
BogoMIPS: 4800.38
Virtualization: VT-x
L1d cache: 32K
L1i cache: 32K
L2 cache: 256K
L3 cache: 12288K
NUMA node0 CPU(s): 0-5,12-17
NUMA node1 CPU(s): 6-11,18-23
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1g
b rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_t
sc aperfperf eagerfpu pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr
pdcm pcid dca sse4_1 sse4_2 popcnt lahf_lm epb ssbd ibrs lbrp stibp tpr_shadow
vnni flexpriority ept vpid dtherm ida arat spec_ctrl intel_stibp flush_lid
[JorgeGangosoKlock d2estudiante8@atcgrid:~] 2020-02-25 martes
```

**(b)** ¿Cuántos cores físicos y cuántos cores lógicos tienen los nodos de cómputo de atcgrid y el PC? Razonar las respuestas

**RESPUESTA:**

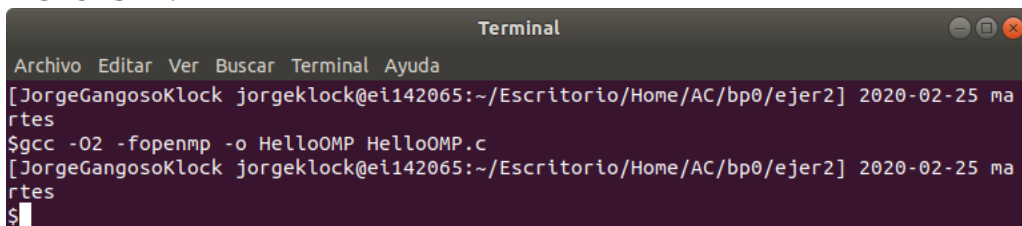
El Pc utilizado tiene 1 core físico (cantidad de sockets) y por cada socket tiene 4 cores lógicos (núcleos por socket). Es decir 4 lógicos.

El nodo de cómputo de atcgrid tiene 2 cores físicos y 24 lógicos, ya que cada core físico tiene 6 lógicos y estos a su vez utilizan multithreading duplicando cada core lógico ( $12 \text{ cores} * 2 = 24$ )

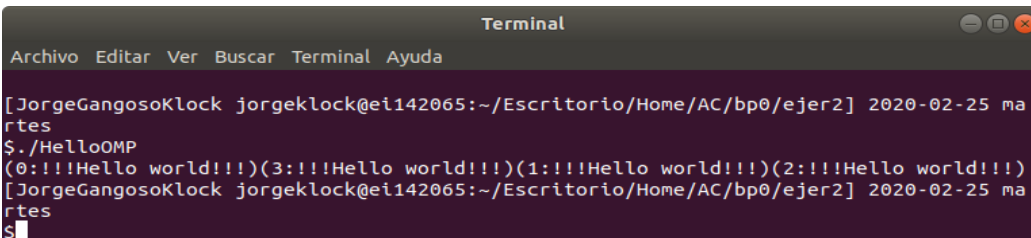
2. Compilar y ejecutar en el PC el código HelloOMP.c del seminario (recordar que se debe usar un directorio independiente para cada ejercicio dentro de bp0 que contenga todo lo utilizado, implementado o generado durante el desarrollo del mismo, para el presente ejercicio el directorio sería **ejer2**, como se indica en las normas de prácticas).

**(a)** Adjuntar capturas de pantalla que muestren la compilación y ejecución en el PC.

**RESPUESTA:**



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer2] 2020-02-25 ma
rtes
$gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer2] 2020-02-25 ma
rtes
$
```



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer2] 2020-02-25 ma
rtes
$./HelloOMP
(0:!!!Hello world!!!)(3:!!!Hello world!!!)(1:!!!Hello world!!!)(2:!!!Hello world!!!)
[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer2] 2020-02-25 ma
rtes
$
```

**(b)** Justificar el número de “Hello world” que se imprimen en pantalla teniendo en cuenta la salida que devuelve lscpu.

**RESPUESTA:**

Aparecen 4 Hello World porque como vimos en el apartado 1, el PC tiene 4 cores lógicos, así que es capaz de ejecutar el programa paralelamente 4 veces.

3. Copiar el ejecutable de HelloOMP.c que ha generado anteriormente y que se encuentra en el directorio ejer2 del PC al directorio ejer2 de su home en el *front-end* de atcgrid. Ejecutar este código en un nodo de cómputo de atcgrid a través de cola ac del gestor de colas (no use ningún *script*) utilizando directamente en línea de comandos:

**(a)** `srun -p ac -n1 --cpus-per-task=12 --hint=nomultithread HelloOMP`

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

**RESPUESTA:**

```
d2estudiante8@atcgrid:~/bp0/ejer2
Archivo Editar Ver Buscar Terminal Ayuda
[JorgeGangosoKlock d2estudiante8@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$ srun -p ac -n1 --cpus-per-task=12 --hint=nomultithread HelloOMP
srun: job 10531 queued and waiting for resources
srun: job 10531 has been allocated resources
(4:!!!Hello world!!!)(3:!!!Hello world!!!)(1:!!!Hello world!!!)(7:!!!Hello world
!!!)(6:!!!Hello world!!!)(10:!!!Hello world!!!)(0:!!!Hello world!!!)(9:!!!Hello
world!!!)(8:!!!Hello world!!!)(2:!!!Hello world!!!)(5:!!!Hello world!!!)(11:!!!H
ello world!!!)[JorgeGangosoKlock d2estudiante8@atcgrid:~/bp0/ejer2] 2020-02-25 m
artes
$
```

(b) `srun -p ac -n1 --cpus-per-task=24 HelloOMP`

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

**RESPUESTA:**

```
d2estudiante8@atcgrid:~/bp0/ejer2
Archivo Editar Ver Buscar Terminal Ayuda
[JorgeGangosoKlock d2estudiante8@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$ srun -p ac -n1 --cpus-per-task=24 HelloOMP
(19:!!!Hello world!!!)(1:!!!Hello world!!!)(3:!!!Hello world!!!)(13:!!!Hello wor
ld!!!)(8:!!!Hello world!!!)(23:!!!Hello world!!!)(10:!!!Hello world!!!)(6:!!!Hel
lo world!!!)(15:!!!Hello world!!!)(2:!!!Hello world!!!)(14:!!!Hello world!!!)(12
:!!!Hello world!!!)(4:!!!Hello world!!!)(9:!!!Hello world!!!)(22:!!!Hello world!
!!)(0:!!!Hello world!!!)(17:!!!Hello world!!!)(5:!!!Hello world!!!)(11:!!!Hello
world!!!)(20:!!!Hello world!!!)(16:!!!Hello world!!!)(7:!!!Hello world!!!)(18:!!
!Hello world!!!)(21:!!!Hello world!!!)[JorgeGangosoKlock d2estudiante8@atcgrid:~
/bp0/ejer2] 2020-02-25 martes
$
```

(c) `srun -p ac -n1 HelloOMP`

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

**RESPUESTA:**

```
d2estudiante8@atcgrid:~/bp0/ejer2
Archivo Editar Ver Buscar Terminal Ayuda
[JorgeGangosoKlock d2estudiante8@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$ srun -p ac -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[JorgeGangosoKlock d2estudiante8@atcgr
id:~/bp0/ejer2] 2020-02-25 martes
$
```

(d) ¿Qué orden `srun` usaría para que HelloOMP utilice los 12 cores físicos de un nodo de cómputo de atcgrid (se debe imprimir un único mensaje desde cada uno de ellos, en total, 12)?

**RESPUESTA:**

La orden utilizada en el apartado (a) `srun -p ac -n1 - -cpus-per-task=12 - -hint=nomultithread HelloOMP`, ya que creamos 12 threads pero obligamos a que cada uno se ejecute en un core distinto (bloqueando el multithread)

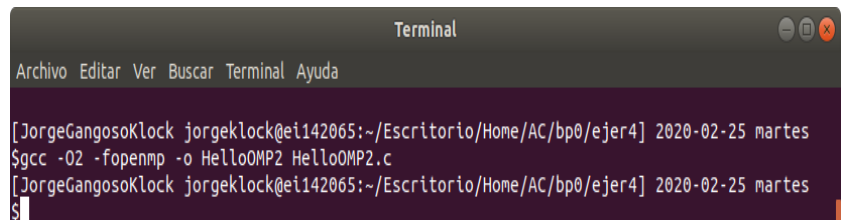
4. Modificar en su PC `HelloOMP.c` para que se imprima “world” en un `printf` distinto al usado para “Hello”, en ambos `printf` se debe imprimir el identificador del thread que escribe en pantalla. Nombrar al código resultante `HelloOMP2.c`. Compilar este nuevo código en el PC y ejecutarlo. Copiar el fichero ejecutable resultante al front-end de `atcgrid` (directorio `ejer4`). Ejecutar el código en un nodo de cómputo de `atcgrid` usando el script `script_helloomp.sh` del seminario (el nombre del ejecutable en el script debe ser `HelloOMP2`).

(a) Utilizar: `sbatch -p ac -n1 --cpus-per-task=12 --hint=nomultithread script_helloomp.sh`. Adjuntar capturas de pantalla que muestren el nuevo código, la compilación, el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

### RESPUESTA:

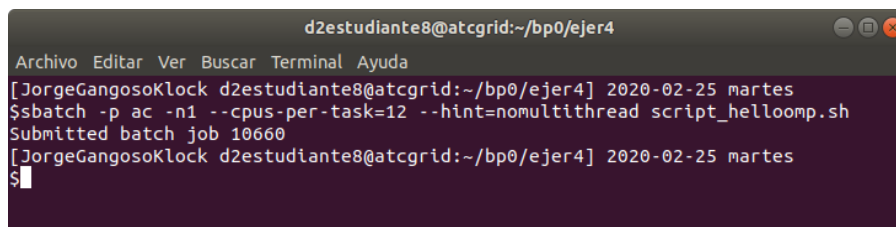
```
/* Compilar con: gcc -O2 -fopenmp -o HelloOMPHelloOMP.c */
#include<stdio.h>
#include<omp.h>

int main(void)
{
    #pragma omp parallel
    {
        printf("(%d:!!!Hello", omp_get_thread_num());
        printf("(%d: world!!!)", omp_get_thread_num());
    }
    return(0);
}
```



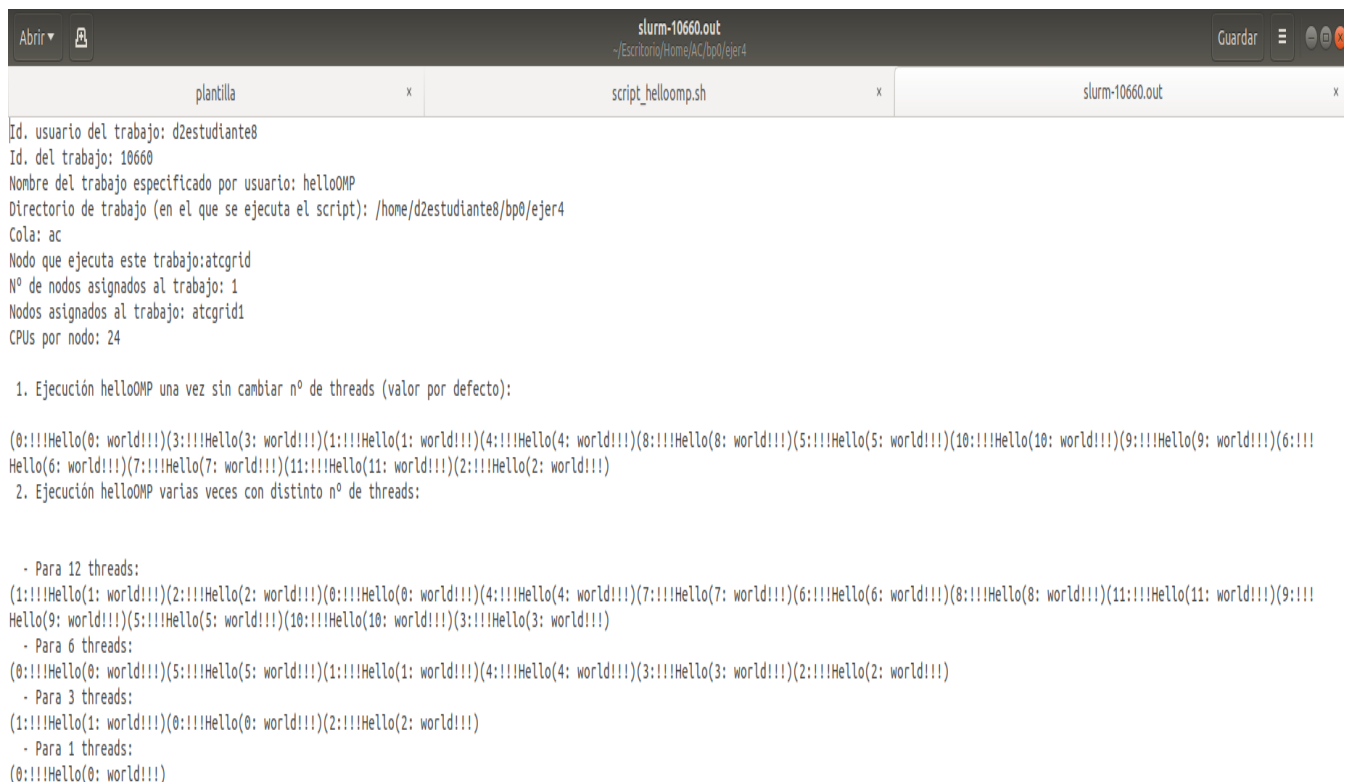
```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer4] 2020-02-25 martes
$gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer4] 2020-02-25 martes
$
```



```
d2estudiante8@atcgrid:~/bp0/ejer4
Archivo Editar Ver Buscar Terminal Ayuda

[JorgeGangosoKlock d2estudiante8@atcgrid:~/bp0/ejer4] 2020-02-25 martes
$sbatch -p ac -n1 --cpus-per-task=12 --hint=nomultithread script_helloomp.sh
Submitted batch job 10660
[JorgeGangosoKlock d2estudiante8@atcgrid:~/bp0/ejer4] 2020-02-25 martes
$
```



```
slurm-10660.out
~/Escritorio/Home/AC/bp0/ejer4
Guardar

plantilla x script_helloomp.sh x slurm-10660.out x

Id. usuario del trabajo: d2estudiante8
Id. del trabajo: 10660
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/d2estudiante8/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

1. Ejecución helloOMP una vez sin cambiar nº de threads (valor por defecto):

(0:!!!Hello(0: world!!!)(3:!!!Hello(3: world!!!)(1:!!!Hello(1: world!!!)(4:!!!Hello(4: world!!!)(8:!!!Hello(8: world!!!)(5:!!!Hello(5: world!!!)(10:!!!Hello(10: world!!!)(9:!!!Hello(9: world!!!)(6:!!!Hello(6: world!!!)(7:!!!Hello(7: world!!!)(11:!!!Hello(11: world!!!)(2:!!!Hello(2: world!!!)

2. Ejecución helloOMP varias veces con distinto nº de threads:

- Para 12 threads:
(1:!!!Hello(1: world!!!)(2:!!!Hello(2: world!!!)(0:!!!Hello(0: world!!!)(4:!!!Hello(4: world!!!)(7:!!!Hello(7: world!!!)(6:!!!Hello(6: world!!!)(8:!!!Hello(8: world!!!)(11:!!!Hello(11: world!!!)(9:!!!Hello(9: world!!!)(5:!!!Hello(5: world!!!)(10:!!!Hello(10: world!!!)(3:!!!Hello(3: world!!!)

- Para 6 threads:
(0:!!!Hello(0: world!!!)(5:!!!Hello(5: world!!!)(1:!!!Hello(1: world!!!)(4:!!!Hello(4: world!!!)(3:!!!Hello(3: world!!!)(2:!!!Hello(2: world!!!)

- Para 3 threads:
(1:!!!Hello(1: world!!!)(0:!!!Hello(0: world!!!)(2:!!!Hello(2: world!!!)

- Para 1 threads:
(0:!!!Hello(0: world!!!)
```

(b) ¿Qué nodo de cómputo de atcgrid ha ejecutado el script? Explicar cómo ha obtenido esta información.

**RESPUESTA:**

Viene como una de las líneas de información generado (slurm) y el nodo que lo ha ejecutado es el propio atcgrid.

**NOTA:** Utilizar siempre con `sbatch` las opciones `-n1` y `--cpus-per-task`, `--exclusive` y, para usar cores físicos y no lógicos, no olvide incluir `--hint=nomultithread`. Utilizar siempre con `srun`, si lo usa fuera de un script, las opciones `-n1` y `--cpus-per-task` y, para usar cores físicos y no lógicos, no olvide incluir `--hint=nomultithread`. Recordar que los `srun` dentro de un script heredan las opciones utilizadas en el `sbatch` que se usa para enviar el script a la cola slurm. Se recomienda usar `sbatch` en lugar de `srun` para enviar trabajos a ejecutar a través slurm porque éste último deja bloqueada la ventana hasta que termina la ejecución, mientras que usando `sbatch` la ejecución se realiza en segundo plano.

## Parte II. Resto de ejercicios

5. Generar en el PC el ejecutable del código fuente C del Listado 1 para vectores locales (para ello antes de compilar debe descomentar la definición de `VECTOR_LOCAL` y comentar las definiciones de `VECTOR_GLOBAL` y `VECTOR_DYNAMIC`). El comentario inicial del código muestra la orden para compilar (siempre hay que usar `-O2` al compilar como se indica en las normas de prácticas). Incorporar volcados de pantalla que demuestren la compilación y la ejecución correcta del código en el PC (leer lo indicado al respecto en las normas de prácticas).

**RESPUESTA:**

```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda

[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer5] 2020-02-25 martes
$gcc -O2 SumaVectores.c -o SumaVectores -lrt
SumaVectores.c: In function 'main':
SumaVectores.c:45:32: warning: format '%u' expects argument of type 'unsigned int', but argument 3 has type 'long unsigned int' [-Wformat=]
    printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                               ~^
                               %lu

[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer5] 2020-02-25 martes
$gcc -O2 -S SumaVectores.c -lrt
SumaVectores.c: In function 'main':
SumaVectores.c:45:32: warning: format '%u' expects argument of type 'unsigned int', but argument 3 has type 'long unsigned int' [-Wformat=]
    printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                               ~^
                               %lu

[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer5] 2020-02-25 martes
$

Terminal
Archivo Editar Ver Buscar Terminal Ayuda

[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer5] 2020-02-25 martes
$./SumaVectores 200
Tamaño Vectores:200 (4 B)
Tiempo:0.000001062 / Tamaño Vectores:200 / V1[0]+V2[0]=V3[0](20.000000+20.000000=40.000000) / / V1[199]+V2[199]=V3[199](39.900000+0.100000=40.000000) /
[JorgeGangosoKlock jorgeklock@ei142065:~/Escritorio/Home/AC/bp0/ejer5] 2020-02-25 martes
$

```

6. En el código del Listado 1 se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. El código se imprime la variable `ncgt`,

(a) ¿qué contiene esta variable?

**RESPUESTA:**

La variable `ncgt` formatea los segundos de ejecución concatenados/sumados con los nanosegundos de ejecución del segmento calculado

(b) ¿en qué estructura de datos devuelve `clock_gettime()` la información de tiempo (indicar el tipo de estructura de datos, describir la estructura de datos, e indicar los tipos de datos que usa)?

**RESPUESTA:**

Devuelve un struct `timespec` que usa dos datos: uno de tipo `time_t` y uno de tipo `long`.

(c) ¿qué información devuelve exactamente la función `clock_gettime()` en la estructura de datos descrita en el apartado (b)? ¿qué representan los valores numéricos que devuelve?

**RESPUESTA:**

Devuelve el instante de tiempo en que se ejecuta en segundos y nanosegundos almacenados en `tv_sec` y `tv_nsec` respectivamente.

7. Rellenar una tabla como la Tabla 1 en una hoja de cálculo con los tiempos de ejecución del código del Listado 1 para vectores locales, globales y dinámicos. Obtener estos resultados usando scripts (partir del script que hay en el seminario). Debe haber una tabla para `atcgrid` y otra para su PC en la hoja de cálculo. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. (NOTA: Se recomienda usar en la hoja de cálculo el mismo separador para decimales que usan los códigos al imprimir. Este separador se puede modificar en la hoja de cálculo.)

**RESPUESTA:**

La primera tabla es la calculada en el PC y la segunda ejecutada en el grid de la universidad.

**Tabla 1 .** Copiar la tabla de la hoja de cálculo utilizada

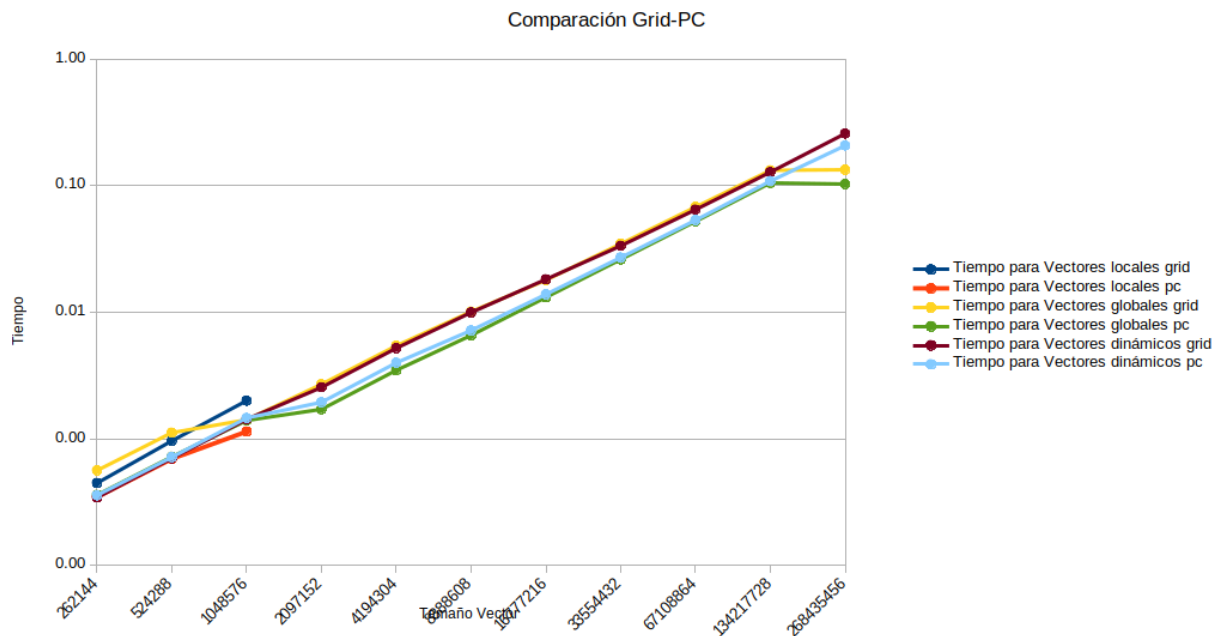
N.º Componentes	Bytes de un Vector	Tiempo para Vectores locales	Tiempo para Vectores globales	Tiempo para Vectores dinámicos
65536	262144	0.000346733	0.000360239	0.000357781
131072	524288	0.000695221	0.000721298	0.000715588
262144	1048576	0.001141344	0.001396334	0.001460844
524288	2097152	N/A	0.001711256	0.001944502
1048576	4194304	N/A	0.003479026	0.003987225
2097152	8388608	N/A	0.006563110	0.007180900
4194304	16777216	N/A	0.013116175	0.013838647
8388608	33554432	N/A	0.026122222	0.027064511
16777216	67108864	N/A	0.052149479	0.053344810
33554432	134217728	N/A	0.105018997	0.108556102
67108864	268435456	N/A	0.102864405	0.207190026



N.º Componentes	Bytes de un Vector	Tiempo para Vectores locales	Tiempo para Vectores globales	Tiempo para Vectores dinámicos
65536	262144	0.000447150	0.000562505	0.000344461
131072	524288	0.000963547	0.001118939	0.000702400
262144	1048576	0.002001805	0.001413712	0.001426301
524288	2097152	N/A	0.002698962	0.002548354
1048576	4194304	N/A	0.005438035	0.005201763
2097152	8388608	N/A	0.010115369	0.009955579
4194304	16777216	N/A	0.017947163	0.018195891
8388608	33554432	N/A	0.034695709	0.033436367
16777216	67108864	N/A	0.067864967	0.064637573
33554432	134217728	N/A	0.132096048	0.127798938
67108864	268435456	N/A	0.133336707	0.257655516

1. Con ayuda de la hoja de cálculo representar **en una misma gráfica** los tiempos de ejecución obtenidos en atcgrid y en su PC para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (por tanto, los valores de la segunda columna de la tabla, que están en escala logarítmica, deben estar en el eje x). Utilizar escala logarítmica en el eje de ordenadas (eje y). ¿Hay diferencias en los tiempos de ejecución?

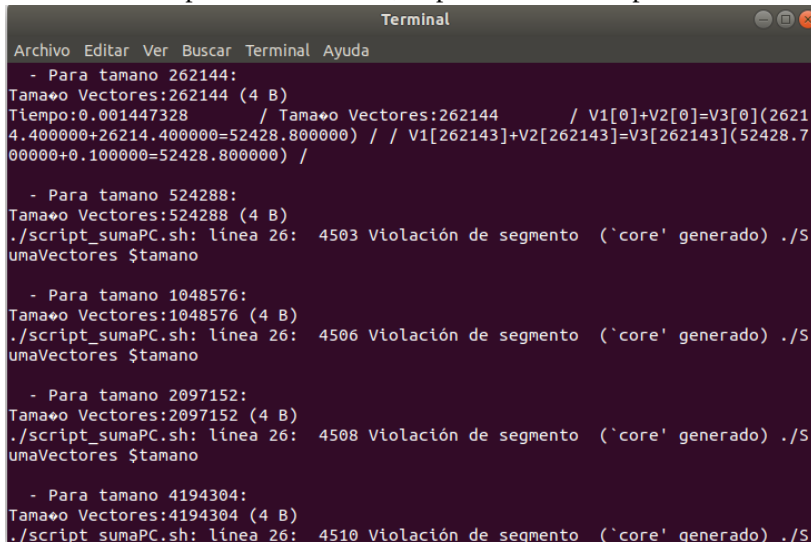
**RESPUESTA:** Las diferencias apenas se aprecian pero utilizando el PC del aula de estudios de la UGR y el nodo maestro de atcgrid los tiempos son algo mejores en el PC, además podemos observar como los tiempos con vectores globales son mayores que los locales pero menores que los dinámicos. ( $T_{local} < T_{global} < T_{dinamico}$ )



2. (a) Cuando se usan vectores locales, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

### RESPUESTA:

Nos aparece error por violación de segmento. Puesto que estas variables se almacenan en la pila, a partir de la cantidad de componentes: 262144, la pila se desborda por exceso de tamaño.



```

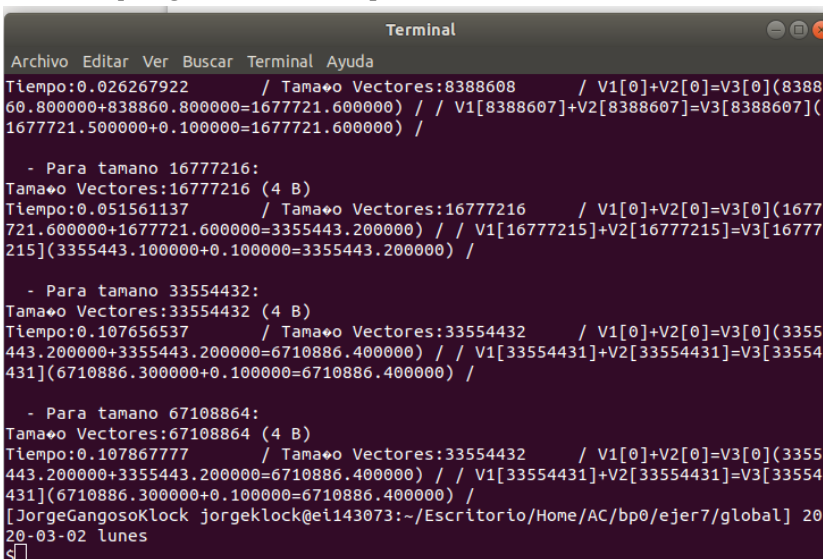
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
- Para tamaño 262144:
Tamaño Vectores:262144 (4 B)
Tiempo:0.001447328 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](2621
4.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143](52428.7
00000+0.100000=52428.800000) /
- Para tamaño 524288:
Tamaño Vectores:524288 (4 B)
./script_sumaPC.sh: línea 26: 4503 Violación de segmento ('core' generado) ./S
umaVectores $tamano
- Para tamaño 1048576:
Tamaño Vectores:1048576 (4 B)
./script_sumaPC.sh: línea 26: 4506 Violación de segmento ('core' generado) ./S
umaVectores $tamano
- Para tamaño 2097152:
Tamaño Vectores:2097152 (4 B)
./script_sumaPC.sh: línea 26: 4508 Violación de segmento ('core' generado) ./S
umaVectores $tamano
- Para tamaño 4194304:
Tamaño Vectores:4194304 (4 B)
./script_sumaPC.sh: línea 26: 4510 Violación de segmento ('core' generado) ./S

```

- (b) Cuando se usan vectores globales, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

### RESPUESTA:

No aparece ningún error pero el tamaño del vector se capta en el penúltimo valor puesto que en el código hay una sentencia que iguala tamaños superiores a ese valor al mismo.



```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
Tiempo:0.026267922 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](8388
60.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](
1677721.500000+0.100000=1677721.600000) /
- Para tamaño 16777216:
Tamaño Vectores:16777216 (4 B)
Tiempo:0.051561137 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677
721.600000+1677721.600000=3355443.200000) / / V1[16777215]+V2[16777215]=V3[16777
215](3355443.100000+0.100000=3355443.200000) /
- Para tamaño 33554432:
Tamaño Vectores:33554432 (4 B)
Tiempo:0.107656537 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355
443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[33554431]=V3[33554
431](6710886.300000+0.100000=6710886.400000) /
- Para tamaño 67108864:
Tamaño Vectores:67108864 (4 B)
Tiempo:0.107867777 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355
443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[33554431]=V3[33554
431](6710886.300000+0.100000=6710886.400000) /
[JorgeGangosoKlock jorgeklock@ei143073:~/Escritorio/Home/AC/bp0/ejer7/global] 20
20-03-02 Lunes
$

```

- (c) Cuando se usan vectores dinámicos, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

### RESPUESTA:

No aparece ningún error usando vectores dinámicos, pero hay que tener cuidado de realizar correctamente las sentencias de reserva de espacio y la limpieza posterior. Puesto que no hacerlo conlleva errores.



```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
Tiempo:0.027267274 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](8388
60.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](
1677721.500000+0.100000=1677721.600000) /

- Para tamaño 16777216:
Tamaño Vectores:16777216 (4 B)
Tiempo:0.053388640 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677
721.600000+1677721.600000=3355443.200000) / / V1[16777215]+V2[16777215]=V3[16777
215](3355443.100000+0.100000=3355443.200000) /

- Para tamaño 33554432:
Tamaño Vectores:33554432 (4 B)
Tiempo:0.107411642 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355
443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[33554431]=V3[33554
431](6710886.300000+0.100000=6710886.400000) /

- Para tamaño 67108864:
Tamaño Vectores:67108864 (4 B)
Tiempo:0.216076552 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](6710
886.400000+6710886.400000=13421772.800000) / / V1[67108863]+V2[67108863]=V3[6710
8863](13421772.700000+0.100000=13421772.800000) /
[JorgeGangosoKlock jorgeklock@ei143073:~/Escritorio/Home/AC/bp0/ejer7/dinamico]
2020-03-02 lunes
$

```

3. (a) ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.

**RESPUESTA:**

Se usan unsigned int (enteros de 32 bits = 4bytes) así que el máximo es  $2^{32}-1$  (32 Bits puestos a 1): 4294967295

- (b) Modificar el código fuente C (en el PC) para que el límite de los vectores cuando se declaran como variables globales sea igual al máximo número que se puede almacenar en la variable N y generar el ejecutable. ¿Qué ocurre? ¿A qué es debido? (Incorporar volcados de pantalla que muestren lo que ocurre)

**RESPUESTA:**

Se muestra un error en la propia compilación, debido a que la cantidad de espacio necesaria para alojar los tres arrays excede la memoria disponible para ello, puesto que en un espacio de 32 bits solo podemos direccionar esos  $2^{32}$  bits, y no el triple de esa cantidad.

```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
grafica_comparativa.png SumaVectores_globalmodificado.c
local tabla_sumavectorcomparativa.ods
script_helloomp.sh tabla_sumavectorGRID.ods
script_sumaPC.sh tabla_sumavectorPC.ods
[JorgeGangosoKlock jorgeklock@ei143073:~/Escritorio/Home/AC/bp0/ejer7] 2020-03-0
2 lunes
$gcc -O2 SumaVectores_globalmodificado.c -o SumaVectores_globalmodificado -lrt
SumaVectores_globalmodificado.c: In function 'main':
SumaVectores_globalmodificado.c:45:32: warning: format '%u' expects argument of
type 'unsigned int', but argument 3 has type 'long unsigned int' [-Wformat=]
printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
    ~^
    %lu
/tmp/cc6QTWS0.o: En la función 'main':
SumaVectores_globalmodificado.c:(.text.startup+0x76): reubicación truncada para
ajustar: R_X86_64_PC32 contra el símbolo 'v2' definido en la sección COMMON en /
tmp/cc6QTWS0.o
SumaVectores_globalmodificado.c:(.text.startup+0xc9): reubicación truncada para
ajustar: R_X86_64_PC32 contra el símbolo 'v3' definido en la sección COMMON en /
tmp/cc6QTWS0.o
collect2: error: ld returned 1 exit status
[JorgeGangosoKlock jorgeklock@ei143073:~/Escritorio/Home/AC/bp0/ejer7] 2020-03-0
2 lunes
$gcc

```

## Entrega del trabajo

Leer lo indicado en las normas de prácticas sobre la entrega del trabajo del bloque práctico en SWAD.

**Listado 1.** Código C que suma dos vectores

```

/* SumaVectoresC.c
   Suma de dos vectores: v3 = v1 + v2

   Para compilar usar (-lrt: real time library, no todas las versiones de gcc necesitan que se incluya
   -lrt):
       gcc -O2 SumaVectores.c -o SumaVectores -lrt
       gcc -O2 -S SumaVectores.c -lrt //para generar el código ensamblador

   Para ejecutar use: SumaVectoresC longitud
*/

#include <stdlib.h> // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h> // biblioteca donde se encuentra la función printf()
#include <time.h> // biblioteca donde se encuentra la función clock_gettime()

//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
//define VECTOR_LOCAL // descomentar para que los vectores sean variables ...
// locales (si se supera el tamaño de la pila se ...
// generará el error "Violación de Segmento")
//define VECTOR_GLOBAL// descomentar para que los vectores sean variables ...
// globales (su longitud no estará limitada por el ...
// tamaño de la pila del programa)
#define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ...
// dinámicas (memoria reutilizable durante la ejecución)

#ifndef VECTOR_GLOBAL
#define MAX 33554432 //2^25
double v1[MAX], v2[MAX], v3[MAX];
#endif

int main(int argc, char** argv){

    int i;
    struct timespec cgt1,cgt2; double ncgt; //para tiempo de ejecución

    //Leer argumento de entrada (nº de componentes del vector)
    if (argc<2){
        printf("Faltan nº componentes del vector\n");
        exit(-1);
    }

    unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
    #ifdef VECTOR_LOCAL
        double v1[N], v2[N], v3[N]; // Tamaño variable local en tiempo de ejecución ...
        // disponible en C a partir de actualización C99
    #endif
    #ifdef VECTOR_GLOBAL
        if (N>MAX) N=MAX;
    #endif

```

```

#endif
#ifdef VECTOR_DYNAMIC
double *v1, *v2, *v3;
v1 = (double*) malloc(N*sizeof(double)); // malloc necesita el tamaño en bytes
v2 = (double*) malloc(N*sizeof(double)); //si no hay espacio suficiente malloc devuelve NULL
v3 = (double*) malloc(N*sizeof(double));
    if ( (v1==NULL) || (v2==NULL) || (v3==NULL) ){
        printf("Error en la reserva de espacio para los vectores\n");
        exit(-2);
    }
#endif

//Inicializar vectores
for(i=0; i<N; i++){
    v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
}

clock_gettime(CLOCK_REALTIME,&cgt1);
//Calcular suma de vectores
for(i=0; i<N; i++)
    v3[i] = v1[i] + v2[i];

clock_gettime(CLOCK_REALTIME,&cgt2);
ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
        (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

//Imprimir resultado de la suma y el tiempo de ejecución
if (N<10) {
    printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n",ncgt,N);
    for(i=0; i<N; i++)
        printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
            i,i,i,v1[i],v2[i],v3[i]);
}
else
    printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f=%8.6f) / /
        V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
        ncgt,N,v1[0],v2[0],v3[0],N-1,N-1,N-1,v1[N-1],v2[N-1],v3[N-1]);

#ifdef VECTOR_DYNAMIC
free(v1); // libera el espacio reservado para v1
free(v2); // libera el espacio reservado para v2
free(v3); // libera el espacio reservado para v3
#endif
return 0;
}

```