

2º curso / 2º cuatr.  
Grado Ingeniería  
Informática

# Arquitectura de Computadores (AC)

## Cuaderno de prácticas.

### Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Pablo Olivares Martínez

Grupo de prácticas y profesor de prácticas: María Isabel García Arenas

Fecha de entrega: 17/03/2021

Fecha evaluación en clase: 18/03/2021

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 (PC = PC del aula de prácticas o su computador personal).

**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 `x` se debe usar con `sbatch/srun` la opción `--cpus-per-task=x` (`-cx`).
- En `slurm`, por defecto, `cpu` se refiere a cores lógicos (ej. en la opción `-c`), 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 `--ntasks=1` (`-n1`) en `sbatch/srun`.
- Para que no se ejecute más de un proceso en un nodo de cómputo 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 (partición `slurm`).
- Las opciones de `sbatch` se pueden especificar también dentro del *script* (usando `#SBATCH`, ver ejemplos en el script del seminario)

1. Ejecutar `lscpu` en el PC, en `atcgrid4` (usar `-p ac4`) y en uno de los restantes nodos de cómputo (`atcgrid1`, `atcgrid2` o `atcgrid3`, están en la cola `ac`). (Crear directorio `ejer1`)

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

**RESPUESTA:**

```

[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer2] 2021-03-11 jueves
$ lscpu
Arquitectura:                x86_64
modo(s) de operación de las CPUs:  32-bit, 64-bit
Orden de los bytes:          Little Endian
Address sizes:               39 bits physical, 48 bits virtual
CPU(s):                      8
Lista de la(s) CPU(s) en línea:  0-7
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»:       4
«Socket(s)»:                 1
Modo(s) NUMA:                1
ID de fabricante:            GenuineIntel
Familia de CPU:               6
Modelo:                      158
Nombre del modelo:            Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
Revisión:                    10
CPU MHz:                     800.016
CPU MHz máx.:                2300,0000
CPU MHz mín.:                800,0000
BogoMIPS:                    4599.93
Virtualización:              VT-x
Caché L1d:                   128 KiB
Caché L1i:                   128 KiB
Caché L2:                    1 MiB
Caché L3:                    8 MiB
CPU(s) del nodo NUMA 0:      0-7
Vulnerability Itlb multihit:  KVM: Mitigation: VMX disabled
Vulnerability L1tf:           Mitigation; PTE Inversion; VMX conditional
                              cache flushes, SMT vulnerable
Vulnerability Mds:            Mitigation; Clear CPU buffers; SMT vulnerab
                              le
Vulnerability Meltdown:       Mitigation; PTI
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabl
                              ed via prctl and seccomp
Vulnerability Spectre v1:     Mitigation; usercopy/swapgs barriers and __
                              user pointer sanitization
Vulnerability Spectre v2:     Mitigation; Full generic retpoline, IBPB co
                              nditional, IBRS_FW, STIBP conditional, RSB
                              filling
Vulnerability Srbds:          Mitigation; Microcode
Vulnerability Tsx async abort: Not affected
Indicadores:                  fpu vme de pse tsc msr pae mce cx8 apic sep
                              mtrr pge mca cmov pat pse36 clflush dts ac
                              pi mmx fxsr sse sse2 ss ht tm pbe syscall n
                              x pdpe1gb rdtscp lm constant_tsc art arch_p

```

Figura 1: Mi PC

```

e1estudiante21@atcgrid:~
[Pablo Olivares Martínez e1estudiante21@atcgrid:~] 2021-03-11 jueves
$ srun -p ac4 lscpu
Architecture:           x86_64
CPU op-mode(s):         32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 64
On-line CPU(s) list:    0-63
Thread(s) per core:     2
Core(s) per socket:     16
Socket(s):              2
NUMA node(s):          2
Vendor ID:              GenuineIntel
CPU family:             6
Model:                 85
Model name:             Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
Stepping:              7
CPU MHz:                1156.256
CPU max MHz:            3200,0000
CPU min MHz:            800,0000
BogoMIPS:               4200.00
Virtualization:         VT-x
L1d cache:              32K
L1i cache:              32K
L2 cache:               1024K
L3 cache:               22528K
NUMA node0 CPU(s):      0-15,32-47
NUMA node1 CPU(s):      16-31,48-63
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 art arch_perfmon pebs bts rep_good nopl xtopology nonst
op_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 s
sse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_dead
line_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb cat_l3 cdp_l3
invpcid_single intel_ppin intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_s
hadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erm
s invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb avx
512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_to
tal cqm_mbm_local dtherm ida arat pln pts pku ospke avx512_vnni md_clear spec_ct
rl intel stibp flush lld arch_capabilities

```

Figura 2: ATCGRID4

```

e1estudiante21@atcgrid:~
[Pablo Olivares Martínez e1estudiante21@atcgrid:~] 2021-03-11 jueves
$ srun lscpu
Architecture:           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:               4799.93
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 aperfmperf eagerfpu pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr
pdc_m pcid dca sse4_1 sse4_2 popcnt lahf_lm epb ssbd ibrs ibpb stibp tpr_shadow
vnmi flexpriority ept vpid dtherm ida arat spec ctrl intel stibp flush lld

```

Figura 3: ATCGRID1-3

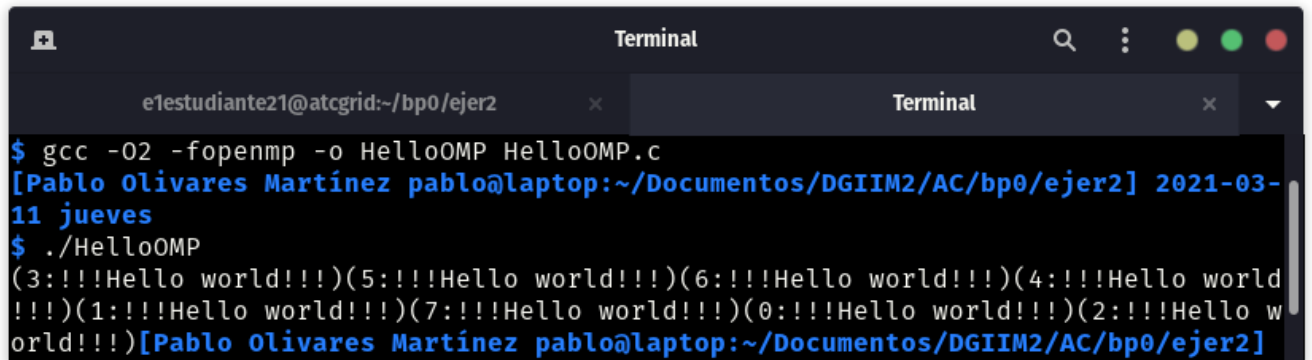
(b) ¿Cuántos cores físicos y cuántos cores lógicos tiene atcgrid4?, ¿cuántos tienen atcgrid1, atcgrid2 y atcgrid3? y ¿cuántos tiene el PC? Razonar las respuestas

**RESPUESTA:** Mi PC tiene 4 cores físicos (cuatro en un socket) y 8 lógicos (2 threads por núcleo); ATCGRID4 tiene 32 núcleos físicos (2 sockets y 16 por socket) y 64 lógicos, y ATCGRID1-3 tiene 12 cores físicos (2 sockets y 16 núcleos por socket) y 24 lógicos (2 hilos por núcleo).

2. Compilar y ejecutar en el PC el código `HelloOMP.c` del seminario (recordar que, como se indica en las normas de prácticas, 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`).

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

**RESPUESTA:**



```

Terminal
e1estudiante21@atcgrid:~/bp0/ejer2
$ gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer2] 2021-03-11 jueves
$ ./HelloOMP
(3:!!!Hello world!!!)(5:!!!Hello world!!!)(6:!!!Hello world!!!)(4:!!!Hello world!!!)(1:!!!Hello world!!!)(7:!!!Hello world!!!)(0:!!!Hello world!!!)(2:!!!Hello world!!!)[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer2]

```

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

**RESPUESTA:** Mi PC devuelve “Hello world!!!” 8 veces porque mi PC tiene 4 núcleos con dos hilos (hiperthreading), luego 8 cores lógicos.

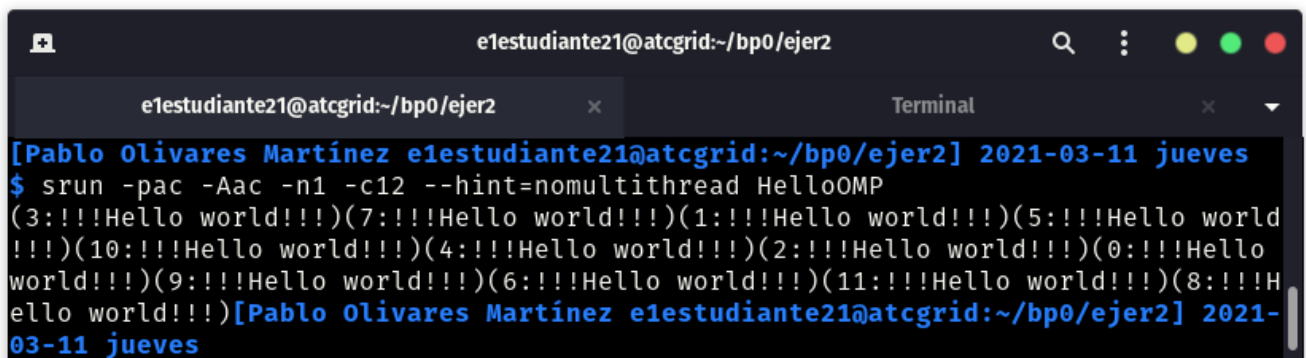
3. Copiar el ejecutable de `HelloOMP.c` que ha generado anteriormente y que se encuentra en el directorio `ejer2` del PC al directorio `ejer2` de su home en el *front-end* de `atcgrid`. Ejecutar este código en un nodo de cómputo de `atcgrid` (de 1 a 3) a través de cola `ac` del gestor de colas utilizando directamente en línea de comandos (no use ningún *script*):

(a) `srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP`

(Alternativa: `srun -pac -Aac -n1 -c12 --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:**



```

Terminal
e1estudiante21@atcgrid:~/bp0/ejer2
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer2] 2021-03-11 jueves
$ srun -pac -Aac -n1 -c12 --hint=nomultithread HelloOMP
(3:!!!Hello world!!!)(7:!!!Hello world!!!)(1:!!!Hello world!!!)(5:!!!Hello world!!!)(10:!!!Hello world!!!)(4:!!!Hello world!!!)(2:!!!Hello world!!!)(0:!!!Hello world!!!)(9:!!!Hello world!!!)(6:!!!Hello world!!!)(11:!!!Hello world!!!)(8:!!!Hello world!!!)[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer2] 2021-03-11 jueves

```

(b) `srun -pac -Aac -n1 -c24 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:**

```

e1estudiante21@atcgrid:~/bp0/ejer2
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer2] 2021-03-11 jueves
$ srun -pac -Aac -n1 -c24 HelloOMP
(0:!!!Hello world!!!)(23:!!!Hello world!!!)(18:!!!Hello world!!!)(11:!!!Hello wo
rld!!!)(4:!!!Hello world!!!)(19:!!!Hello world!!!)(21:!!!Hello world!!!)(17:!!!H
ello world!!!)(2:!!!Hello world!!!)(1:!!!Hello world!!!)(6:!!!Hello world!!!)(10
:!!!Hello world!!!)(13:!!!Hello world!!!)(16:!!!Hello world!!!)(14:!!!Hello worl
d!!!)(15:!!!Hello world!!!)(5:!!!Hello world!!!)(9:!!!Hello world!!!)(12:!!!Hell
o world!!!)(3:!!!Hello world!!!)(22:!!!Hello world!!!)(7:!!!Hello world!!!)(8:!!!
Hello world!!!)(20:!!!Hello world!!!)[Pablo Olivares Martínez e1estudiante21@at
cgrid:~/bp0/ejer2] 2021-03-11 jueves
$
  
```

(c) `srun -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. ¿Qué partición se está usando?

**RESPUESTA:** Está usando la partición por defecto ac.

```

e1estudiante21@atcgrid:~/bp0/ejer2
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer2] 2021-03-11 jueves
$ srun -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[Pablo Olivares Martínez e1estudiante2
1@atcgrid:~/bp0/ejer2] 2021-03-11 jueves
  
```

(d) ¿Qué orden `srun` usaría para que HelloOMP utilice todos los cores físicos de atcgrid4 (se debe imprimir un único mensaje desde cada uno de ellos)?

**RESPUESTA:** `srun -Aac -pac4 -cpus-per-task=32 -hint=nomultithread HelloOMP`



```
e1estudiante21@atcgrid:~/bp0/ejer2
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer2] 2021-03-11 jueves
$ srun -Aac -pac4 --cpus-per-task=32 --hint=nomultithread HelloOMP
(30:!!!Hello world!!!)(21:!!!Hello world!!!)(7:!!!Hello world!!!)(16:!!!Hello wo
rld!!!)(8:!!!Hello world!!!)(22:!!!Hello world!!!)(12:!!!Hello world!!!)(23:!!!H
ello world!!!)(27:!!!Hello world!!!)(18:!!!Hello world!!!)(9:!!!Hello world!!!)(
24:!!!Hello world!!!)(31:!!!Hello world!!!)(14:!!!Hello world!!!)(29:!!!Hello wo
rld!!!)(3:!!!Hello world!!!)(17:!!!Hello world!!!)(5:!!!Hello world!!!)(0:!!!Hel
lo world!!!)(10:!!!Hello world!!!)(1:!!!Hello world!!!)(28:!!!Hello world!!!)(15
:!!!Hello world!!!)(20:!!!Hello world!!!)(11:!!!Hello world!!!)(13:!!!Hello worl
d!!!)(26:!!!Hello world!!!)(2:!!!Hello world!!!)(6:!!!Hello world!!!)(25:!!!Hell
o world!!!)(19:!!!Hello world!!!)(4:!!!Hello world!!!)(5:!!!Hello world!!!)(0:!!
!Hello world!!!)(2:!!!Hello world!!!)(31:!!!Hello world!!!)(29:!!!Hello world!!!
)(25:!!!Hello world!!!)(28:!!!Hello world!!!)(4:!!!Hello world!!!)(17:!!!Hello w
orld!!!)(11:!!!Hello world!!!)(24:!!!Hello world!!!)(27:!!!Hello world!!!)(26:!!
!Hello world!!!)(20:!!!Hello world!!!)(21:!!!Hello world!!!)(30:!!!Hello world!!
!)(16:!!!Hello world!!!)(23:!!!Hello world!!!)(13:!!!Hello world!!!)(9:!!!Hello
world!!!)(15:!!!Hello world!!!)(19:!!!Hello world!!!)(10:!!!Hello world!!!)(12:!!
!Hello world!!!)(14:!!!Hello world!!!)(22:!!!Hello world!!!)(8:!!!Hello world!!
!)(7:!!!Hello world!!!)(18:!!!Hello world!!!)(6:!!!Hello world!!!)(3:!!!Hello wo
rld!!!)(1:!!!Hello world!!!)[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp
0/ejer2] 2021-03-11 jueves
```

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 -pac -n1 -c12 --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:**

```
Terminal
e1estudiante21@atcgrid:~/bp0/ejer2
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer4] 2021-03-11 jueves
$ gcc -O2 HelloOMP2.c -o HelloOMP2 -fopenmp
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer4] 2021-03-11 jueves
$ ./HelloOMP2
(2:Hello)(1:Hello)(0:Hello)(5:Hello)(4:Hello)(3:Hello)(7:Hello)(6:Hello)(0:world
!)[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer4] 2021-0
```

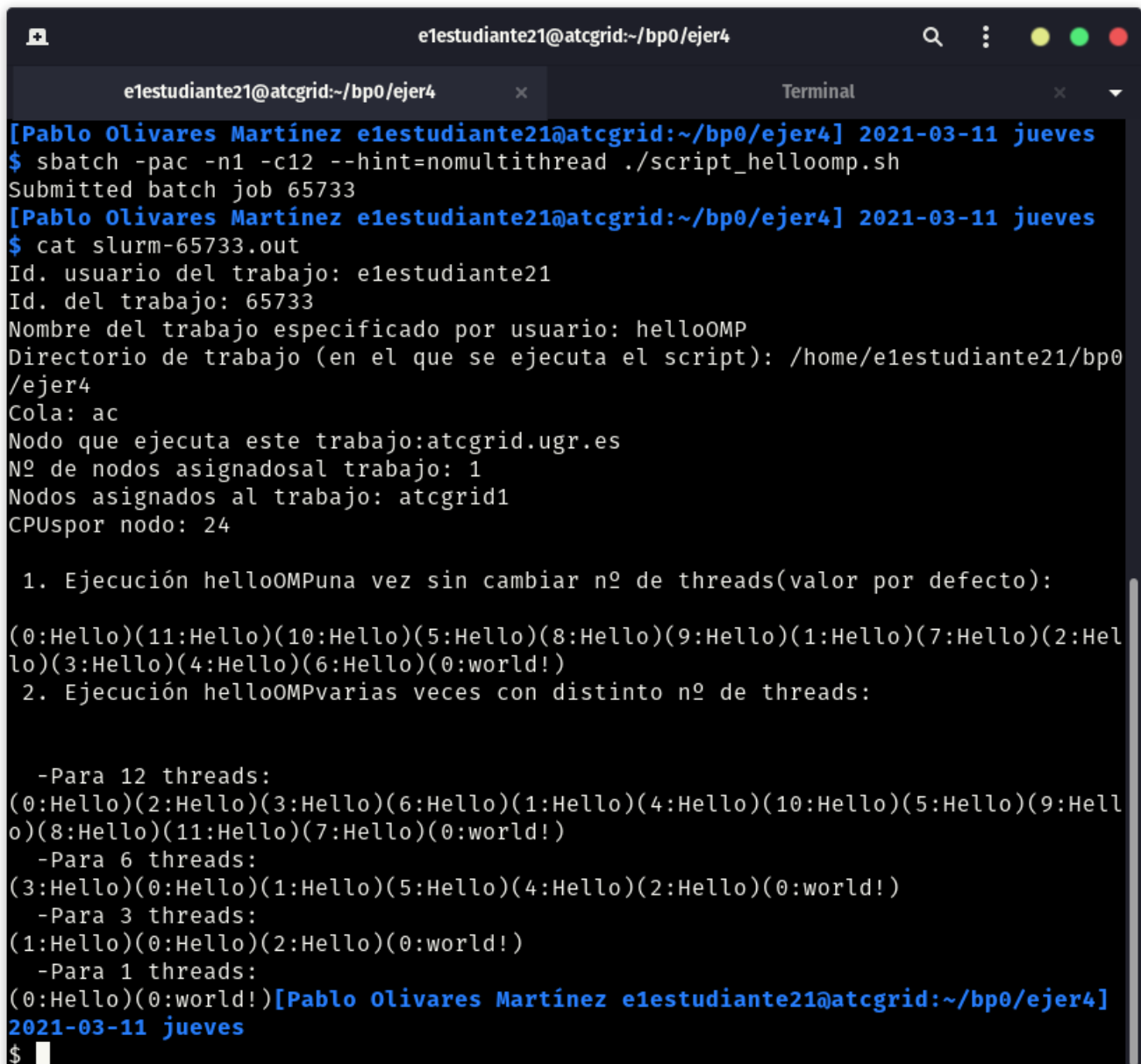


```

1 /* Compilar con: gcc -O2 -fopenmp -o HelloOMP HelloOMP.c */
2 #include<stdio.h>
3 #include<omp.h>
4
5 int main(void){
6     #pragma omp parallel
7     printf("(%d:Hello)",omp_get_thread_num());
8     printf("(%d:world!)",omp_get_thread_num());
9     return(0);
10 }

```

Figura 4: Código modificado para HelloOMP2



```

e1estudiante21@atcgrid:~/bp0/ejer4
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ sbatch -pac -n1 -c12 --hint=nomultithread ./script_helloomp.sh
Submitted batch job 65733
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ cat slurm-65733.out
Id. usuario del trabajo: e1estudiante21
Id. del trabajo: 65733
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/e1estudiante21/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
Nº de nodos asignadosal 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)(11:Hello)(10:Hello)(5:Hello)(8:Hello)(9:Hello)(1:Hello)(7:Hello)(2:Hello)(3:Hello)(4:Hello)(6:Hello)(0:world!)
2. Ejecución helloOMP varias veces con distinto nº de threads:

-Para 12 threads:
(0:Hello)(2:Hello)(3:Hello)(6:Hello)(1:Hello)(4:Hello)(10:Hello)(5:Hello)(9:Hello)(8:Hello)(11:Hello)(7:Hello)(0:world!)
-Para 6 threads:
(3:Hello)(0:Hello)(1:Hello)(5:Hello)(4:Hello)(2:Hello)(0:world!)
-Para 3 threads:
(1:Hello)(0:Hello)(2:Hello)(0:world!)
-Para 1 threads:
(0:Hello)(0:world!)[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer4]
2021-03-11 jueves
$

```





```

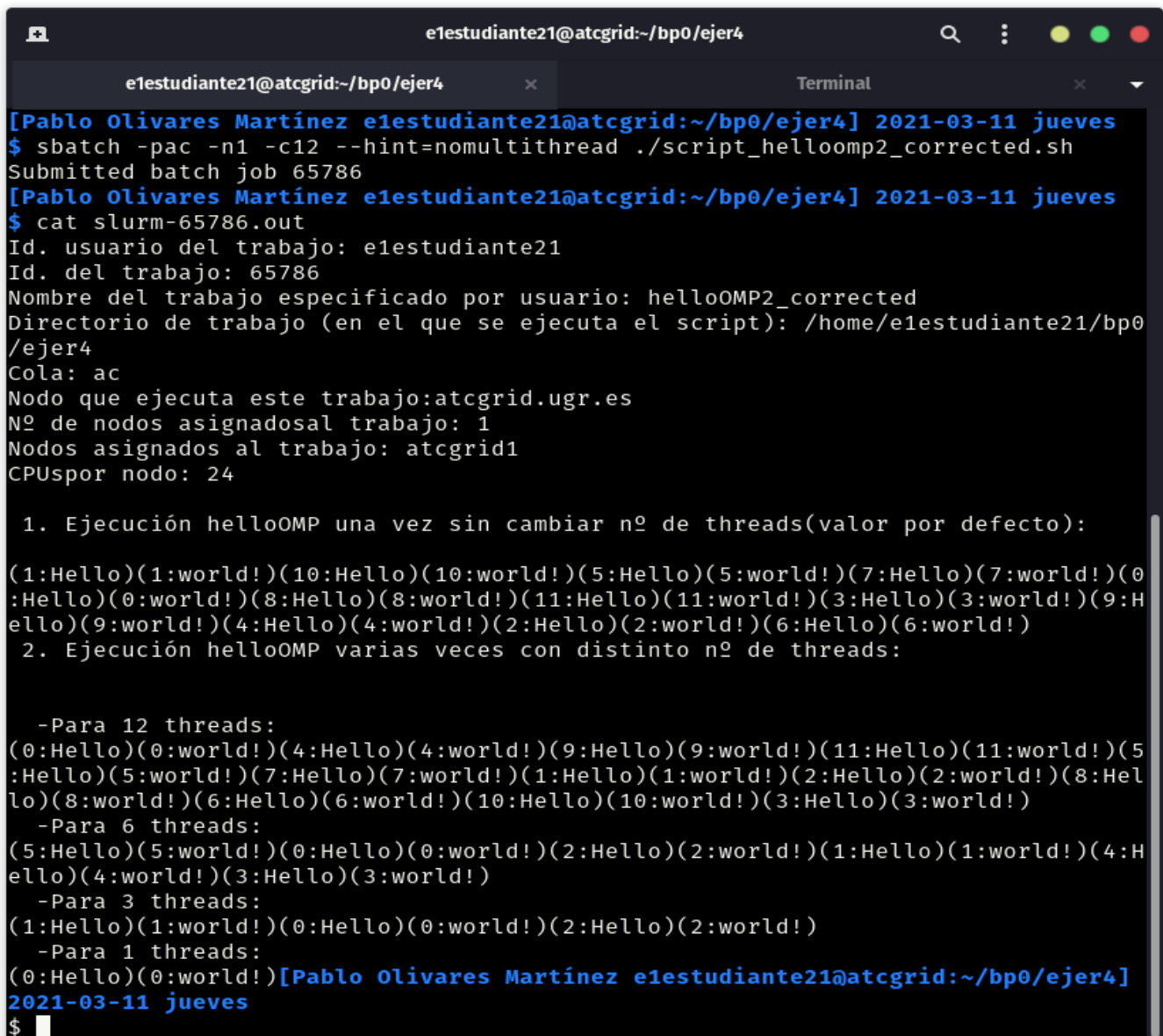
HelloOMP2_corrected.c
~/Documentos/DGIIIM2/AC/bp0/ejer4

1 /* Compilar con: gcc -O2 -fopenmp -o HelloOMP2_corrected
   HelloOMP2_corrected.c */
2 #include<stdio.h>
3 #include<omp.h>
4
5 int main(void){
6     #pragma omp parallel
7     {
8         printf("(%d:Hello)",omp_get_thread_num());
9         printf("(%d:world!)",omp_get_thread_num());
10    }
11    return(0);
12 }

```

Corchete coincidente encontrado en la línea: 7    C    Anchura del tabulador: 8    Ln 10, Col 10    INS

Vemos que al ejecutar el programa tan sólo nos devuelve un “world”. Esto se debe a que la directiva OMP sólo reconoce el primer printf, solucionándolo con dos llaves como he realizado en la imagen superior.



```

e1estudiante21@atcgrid:~/bp0/ejer4
e1estudiante21@atcgrid:~/bp0/ejer4
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ sbatch -pac -n1 -c12 --hint=nomultithread ./script_helloomp2_corrected.sh
Submitted batch job 65786
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ cat slurm-65786.out
Id. usuario del trabajo: e1estudiante21
Id. del trabajo: 65786
Nombre del trabajo especificado por usuario: helloOMP2_corrected
Directorio de trabajo (en el que se ejecuta el script): /home/e1estudiante21/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
Nº de nodos asignadosal trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

1. Ejecución helloOMP una vez sin cambiar nº de threads(valor por defecto):
(1:Hello)(1:world!)(10:Hello)(10:world!)(5:Hello)(5:world!)(7:Hello)(7:world!)(0:Hello)(0:world!)(8:Hello)(8:world!)(11:Hello)(11:world!)(3:Hello)(3:world!)(9:Hello)(9:world!)(4:Hello)(4:world!)(2:Hello)(2:world!)(6:Hello)(6:world!)
2. Ejecución helloOMP varias veces con distinto nº de threads:

-Para 12 threads:
(0:Hello)(0:world!)(4:Hello)(4:world!)(9:Hello)(9:world!)(11:Hello)(11:world!)(5:Hello)(5:world!)(7:Hello)(7:world!)(1:Hello)(1:world!)(2:Hello)(2:world!)(8:Hello)(8:world!)(6:Hello)(6:world!)(10:Hello)(10:world!)(3:Hello)(3:world!)
-Para 6 threads:
(5:Hello)(5:world!)(0:Hello)(0:world!)(2:Hello)(2:world!)(1:Hello)(1:world!)(4:Hello)(4:world!)(3:Hello)(3:world!)
-Para 3 threads:
(1:Hello)(1:world!)(0:Hello)(0:world!)(2:Hello)(2:world!)
-Para 1 threads:
(0:Hello)(0:world!)
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer4]
2021-03-11 jueves
$

```

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

**RESPUESTA:** Observando la información obtenida en el *script*, vemos que pone que es la partición “ac” la que se encarga de su ejecución, concretamente, dice que está asignado al nodo atcgrid1.

**NOTA:** Utilizar siempre con *sbatch* las opciones *-n1* y *-c*, *--exclusive* y, para usar cores físicos y no lógicos, no olvide incluir *--hint=nomultithread*. Utilizar siempre con *srunch*, si lo usa fuera de un *script*, las opciones *-n1* y *-c* y, para usar cores físicos y no lógicos, no olvide incluir *--hint=nomultithread*. Recordar que los *srunch* dentro de un *script* heredan las opciones incluidas en el *sbatch* que se usa para enviar el *script* a la cola *slurm*. Se recomienda usar *sbatch* en lugar de *srunch* 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:**

```

e1estudiante21@atcgrid:~/bp0/ejer5
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer5] 2021-03-16 martes
$ gcc -O2 SumaVectoresC.c -o SumaVectoresC
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer5] 2021-03-16 martes
$ ./SumaVectoresC 2021
Tiempo:0.000006313 / Tamaño Vectores:2021 / V1[0]+V2[0]=V3[0](0.333906+0.162237=0.496142) / / V1[2020]+V2[2020]=V3[2020](1.049666+1.248045=2.297711) /
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer5] 2021-03-16 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:** Esta variable es una variable de tipo *double* que almacena el tiempo que ha tardado en ejecutarse la suma de vectores.

(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:** Esta función devuelve una estructura del tipo *timespec* por referencia, la cual está compuesta por *tv\_sec* del tipo *time\_t*, el cual representa los segundos enteros y *tv\_nsec* del tipo *long*, que representa los nanosegundos.

(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 tiempo en el momento al que se llama la función, siendo un número que representa el tiempo UNIX (segundos pasados desde el 1 de enero de 2021).

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 (se pueden obtener errores en tiempo de ejecución o de compilación, ver ejercicio 9). Obtener estos resultados usando *scripts* (partir del *script* que hay en el seminario). Debe haber una tabla para un nodo de cómputo de atcgrid con procesador Intel Xeon E5645 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:**

**Tabla 1 .** RESULTADOS PC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000179524	0,000464989	0,000430703
131072	1048576	0,000258992	0,000829643	0,000722416
262144	2097152	0,000484678	0,001121265	0,001791194
524288	4194304	SegFault	0,002973524	0,002593601
1048576	8388608	SegFault	0,005471715	0,005010531
2097152	16777216	SegFault	0,009240331	0,008809842
4194304	33554432	SegFault	0,017531881	0,016548818
8388608	67108864	SegFault	0,033463767	0,032614521
16777216	134217728	SegFault	0,069094101	0,064153001
33554432	268435456	SegFault	0,132581874	0,128195669
67108864	536870912	SegFault	0,133457356	0,245463666

**Tabla 2 .** RESULTADOS ATCGRID

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000248408	0,000294563	0,000312157
131072	1048576	0,000297539	0,000603720	0,000611533
262144	2097152	0,000396926	0,001354235	0,001296049
524288	4194304	SegFault	0,002754188	0,002675715
1048576	8388608	SegFault	0,005615428	0,005098508
2097152	16777216	SegFault	0,010197679	0,010271567
4194304	33554432	SegFault	0,020258985	0,020775701
8388608	67108864	SegFault	0,040930825	0,041959131
16777216	134217728	SegFault	0,081606643	0,090140395
33554432	268435456	SegFault	0,185211161	0,169214996
67108864	536870912	SegFault	0,162616345	0,190367905

```
e1estudiante21@atcgrid:~/bp0/ejer7
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer7] 2021-03-16 martes
$ clear

[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer7] 2021-03-16 martes
$ sbatch ./script_sumavectoresc.sh
Submitted batch job 72878
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer7] 2021-03-16 martes
$ cat slurm-72878.out
Id. usuario del trabajo: e1estudiante21
Id. del trabajo: 72878
Nombre del trabajo especificado por usuario: SumaLocal
Directorio de trabajo (en el que se ejecuta el script): /home/e1estudiante21/bp0/
ejer7
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid3
CPUs por nodo: 2
VECTORES LOCALES
Tiempo:0.000179524 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.45
9415+0.616132=1.075547) / / V1[65535]+V2[65535]=V3[65535](0.032944+1.021359=1.05
4303) /
Tiempo:0.000258992 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.77
3733+0.150301=0.924033) / / V1[131071]+V2[131071]=V3[131071](0.619691+0.716525=1
.336216) /
Tiempo:0.000484678 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.77
3733+0.150301=0.924033) / / V1[262143]+V2[262143]=V3[262143](1.303283+2.428433=3
.731716) /
/var/spool/slurmd/job72878/slurm_script: línea 25: 16375 Violación de segmento
('core' generado) ./SumaVectoresC-local $P
/var/spool/slurmd/job72878/slurm_script: línea 25: 16377 Violación de segmento
('core' generado) ./SumaVectoresC-local $P
/var/spool/slurmd/job72878/slurm_script: línea 25: 16383 Violación de segmento
('core' generado) ./SumaVectoresC-local $P
/var/spool/slurmd/job72878/slurm_script: línea 25: 16385 Violación de segmento
('core' generado) ./SumaVectoresC-local $P
/var/spool/slurmd/job72878/slurm_script: línea 25: 16387 Violación de segmento
('core' generado) ./SumaVectoresC-local $P
/var/spool/slurmd/job72878/slurm_script: línea 25: 16389 Violación de segmento
('core' generado) ./SumaVectoresC-local $P
/var/spool/slurmd/job72878/slurm_script: línea 25: 16391 Violación de segmento
('core' generado) ./SumaVectoresC-local $P
/var/spool/slurmd/job72878/slurm_script: línea 25: 16393 Violación de segmento
('core' generado) ./SumaVectoresC-local $P
VECTORES GLOBALES
```

```
e1estudiante21@atcgrid:~/bp0/ejer7
e1estudiante21@atcgrid:~/bp0/ejer7 Terminal
VECTORES GLOBALES
Tiempo:0.000464989 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.53
6783+0.640538=1.177322) / / V1[65535]+V2[65535]=V3[65535](0.233386+3.029106=3.26
2492) /
Tiempo:0.000829643 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.53
6783+0.640538=1.177322) / / V1[131071]+V2[131071]=V3[131071](0.510579+1.901150=2
.411729) /
Tiempo:0.001121265 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.53
6783+0.640538=1.177322) / / V1[262143]+V2[262143]=V3[262143](0.561078+0.999669=1
.560747) /
Tiempo:0.002973524 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.53
6783+0.640538=1.177322) / / V1[524287]+V2[524287]=V3[524287](0.068006+0.591220=0
.659227) /
Tiempo:0.005471715 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](0.53
6783+0.640538=1.177322) / / V1[1048575]+V2[1048575]=V3[1048575](0.408231+3.12864
9=3.536880) /
Tiempo:0.009240331 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](0.53
6783+0.640538=1.177322) / / V1[2097151]+V2[2097151]=V3[2097151](0.346500+0.02100
6=0.367506) /
Tiempo:0.017531881 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.53
6783+0.640538=1.177322) / / V1[4194303]+V2[4194303]=V3[4194303](0.655910+2.21900
2=2.874912) /
Tiempo:0.033463767 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](0.72
9696+1.489720=2.219415) / / V1[8388607]+V2[8388607]=V3[8388607](0.270778+6.24571
1=6.516489) /
Tiempo:0.069094101 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](0.72
9696+1.489720=2.219415) / / V1[16777215]+V2[16777215]=V3[16777215](2.244156+1.83
8238=4.082394) /
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer7] 2021-03-16 martes
$
```

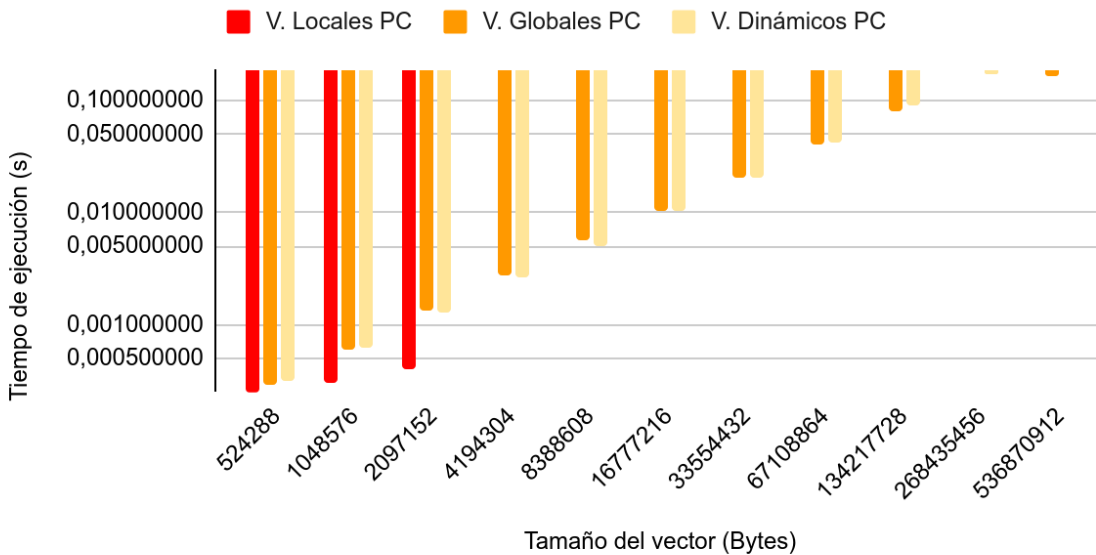
```
Terminal
e1estudiante21@atcgrid:~/bp0/ejer7 Terminal
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer7] 2021-03-
16 martes
$ ./script_sumavectoresc.sh | cat > ac0-7.txt
./script_sumavectoresc.sh: línea 25: 42503 Violación de segmento (`core' genera
do) ./SumaVectoresC-local $P
./script_sumavectoresc.sh: línea 25: 42505 Violación de segmento (`core' genera
do) ./SumaVectoresC-local $P
./script_sumavectoresc.sh: línea 25: 42507 Violación de segmento (`core' genera
do) ./SumaVectoresC-local $P
./script_sumavectoresc.sh: línea 25: 42509 Violación de segmento (`core' genera
do) ./SumaVectoresC-local $P
./script_sumavectoresc.sh: línea 25: 42511 Violación de segmento (`core' genera
do) ./SumaVectoresC-local $P
./script_sumavectoresc.sh: línea 25: 42513 Violación de segmento (`core' genera
do) ./SumaVectoresC-local $P
./script_sumavectoresc.sh: línea 25: 42515 Violación de segmento (`core' genera
do) ./SumaVectoresC-local $P
./script_sumavectoresc.sh: línea 25: 42517 Violación de segmento (`core' genera
do) ./SumaVectoresC-local $P
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer7] 2021-03-
16 martes
$
```

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:**

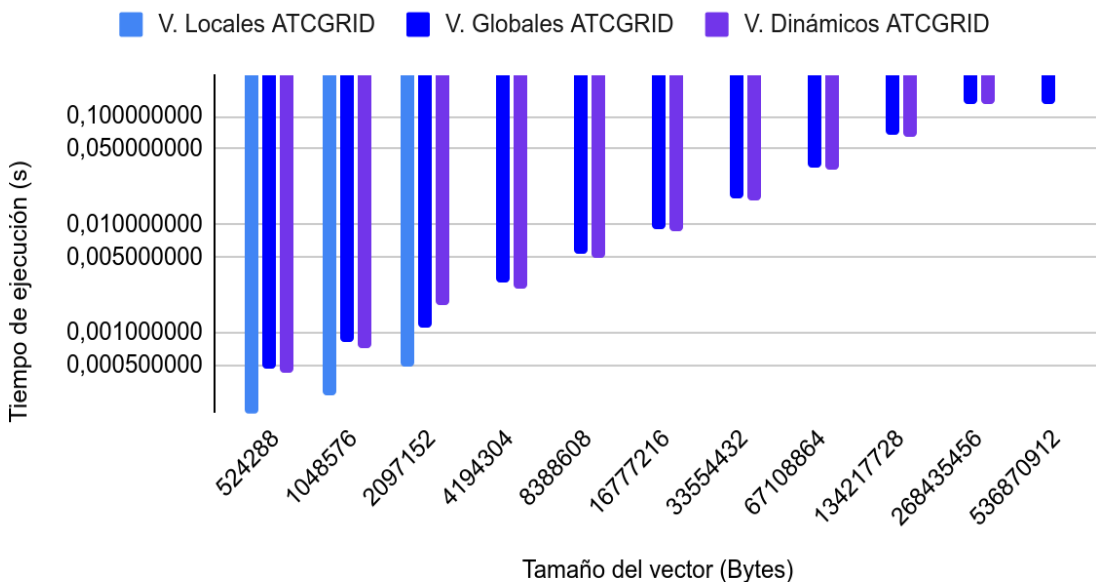
### Velocidad de ejecución por tamaño del vector (PC)

Escala logarítmica



### Velocidad de ejecución por tamaño del vector (ATCGRID)

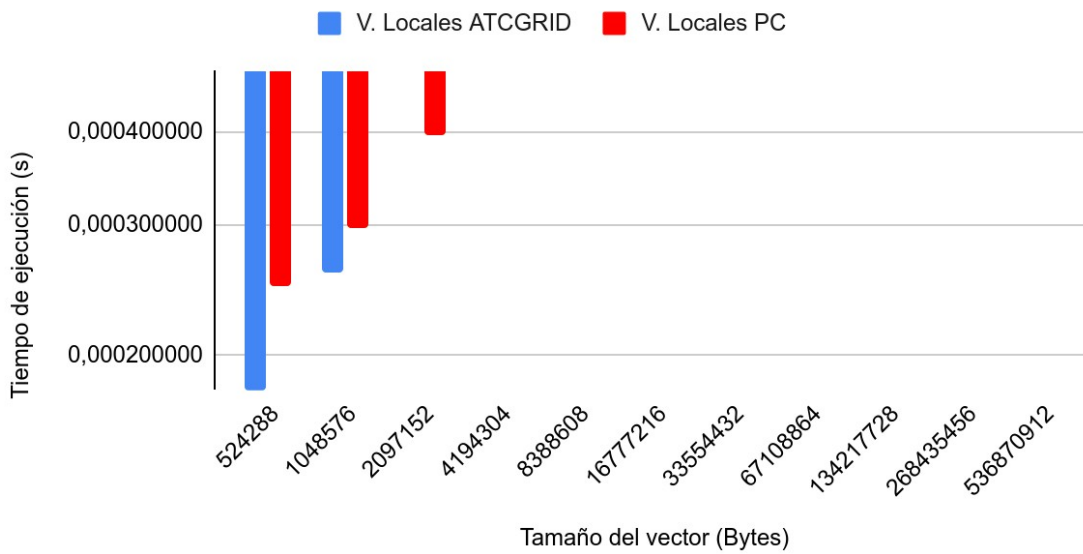
Escala logarítmica





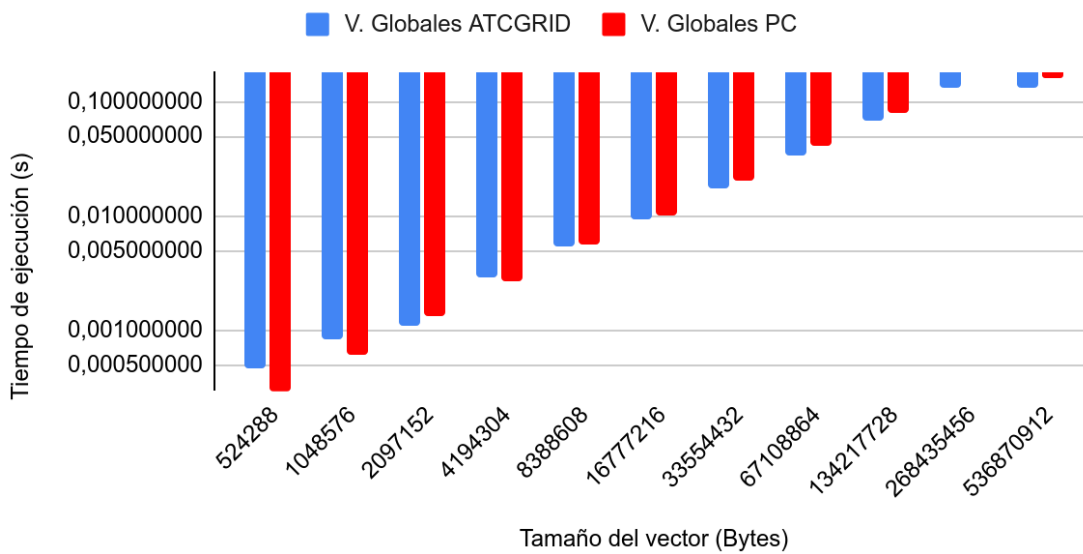
## Velocidad de ejecución por tamaño del vector local

Escala logarítmica



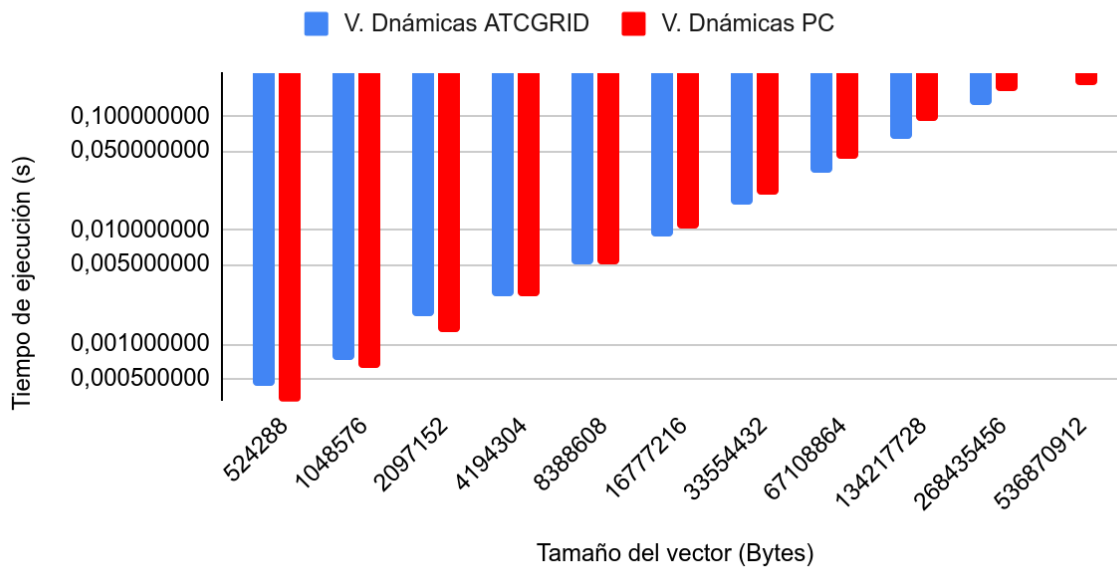
## Velocidad de ejecución por tamaño del vector global

Escala logarítmica



## Velocidad de ejecución por tamaño del vector dinámico

Escala logarítmica



Aquí podemos observar que mi PC es más veloz que ATCGRID de media.

### 2. Contestar a las siguientes preguntas:

(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:** A partir de los vectores de 524000 bytes o más, el programa da segmentation fault, ya que al ser variables locales, se reserva cierta cantidad de memoria en la pila, la cual es igual o menor a la cantidad mencionada.

```
Terminal
e1estudiante21@atcgrid:~/bp0/ejer7

[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer7] 2021-03-17 miércoles
$ ./SumaVectoresC-local 524000
Violación de segmento (`core' generado)
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer7] 2021-03-17 miércoles
$
```

```
Terminal
e1estudiante21@atcgrid:~/bp0/ejer7

[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer7] 2021-03-17 miércoles
$ srun SumaVectoresC-local 524000
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
[Pablo Olivares Martínez e1estudiante21@atcgrid:~/bp0/ejer7] 2021-03-17 miércoles
$
```

**(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:** En este caso, no se da ningún tipo de error, como se puede ver en las capturas del ejercicio 7. Esto se debe a que el array global se almacena en un espacio propio en memoria principal, pero no en la pila, por lo que no da ningún tipo de error.

( Esta conclusión se apoya principalmente en esta fuente: <https://eleceng.dit.ie/frank/IntroToC/Memory.html> )

**(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)

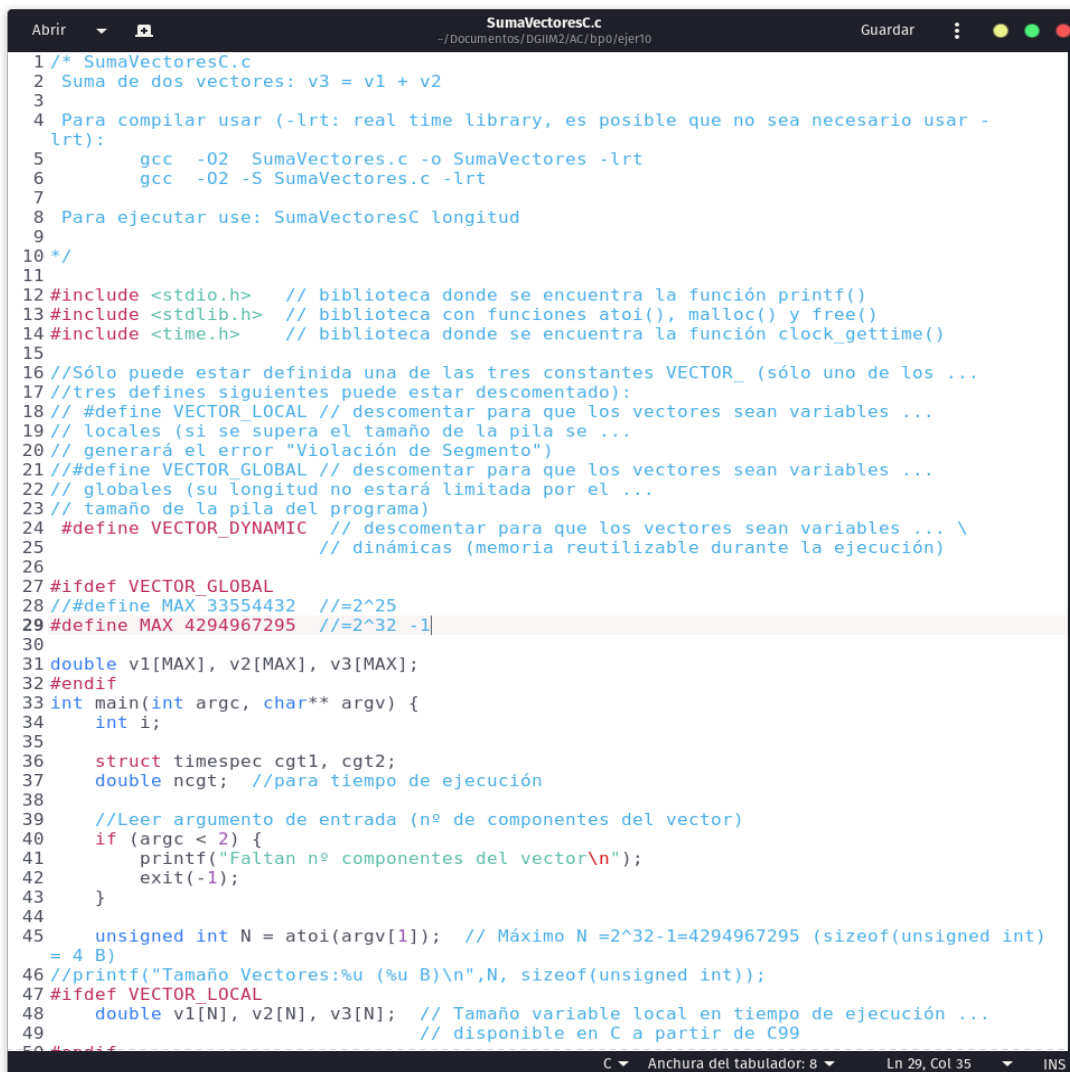
**RESPUESTA:** De nuevo, como vemos en las capturas del ejercicio 7, no da error debido a que la memoria dinámica se asigna en tiempo de ejecución, tomando todo el que necesite por lo que no hay problema.

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

**RESPUESTA:** Por ser unsigned int, ocupa 4 bytes y su rango de valores va de 0 a  $2^{32}-1$ .

**(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:**



```

1 /* SumaVectoresC.c
2 Suma de dos vectores: v3 = v1 + v2
3
4 Para compilar usar (-lrt: real time library, es posible que no sea necesario usar -
  lrt):
5     gcc -O2 SumaVectores.c -o SumaVectores -lrt
6     gcc -O2 -S SumaVectores.c -lrt
7
8 Para ejecutar use: SumaVectoresC longitud
9
10 */
11
12 #include <stdio.h> // biblioteca donde se encuentra la función printf()
13 #include <stdlib.h> // biblioteca con funciones atoi(), malloc() y free()
14 #include <time.h> // biblioteca donde se encuentra la función clock_gettime()
15
16 //Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
17 //tres defines siguientes puede estar descomentado):
18 // #define VECTOR_LOCAL // descomentar para que los vectores sean variables ...
19 // locales (si se supera el tamaño de la pila se ...
20 // generará el error "Violación de Segmento")
21 // #define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
22 // globales (su longitud no estará limitada por el ...
23 // tamaño de la pila del programa)
24 #define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ... \
25 // dinámicas (memoria reutilizable durante la ejecución)
26
27 #ifdef VECTOR_GLOBAL
28 // #define MAX 33554432 // =2^25
29 #define MAX 4294967295 // =2^32 -1|
30
31 double v1[MAX], v2[MAX], v3[MAX];
32 #endif
33 int main(int argc, char** argv) {
34     int i;
35
36     struct timespec cgt1, cgt2;
37     double ncgt; //para tiempo de ejecución
38
39     //Leer argumento de entrada (nº de componentes del vector)
40     if (argc < 2) {
41         printf("Faltan nº componentes del vector\n");
42         exit(-1);
43     }
44
45     unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned int)
46     = 4 B)
47     //printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
48 #ifdef VECTOR_LOCAL
49     double v1[N], v2[N], v3[N]; // Tamaño variable local en tiempo de ejecución ...
50 // disponible en C a partir de C99
51 #endif

```

```

[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer10] 2021-03-17 miércoles
$ gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
/tmp/ccZ3prxM.o: en la función `main':
SumaVectoresC.c:(.text.startup+0x61): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v1' definido en la sección .bss en /tmp/ccZ3prxM.o
SumaVectoresC.c:(.text.startup+0x68): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección .bss en /tmp/ccZ3prxM.o
SumaVectoresC.c:(.text.startup+0x1ff): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v1' definido en la sección .bss en /tmp/ccZ3prxM.o
SumaVectoresC.c:(.text.startup+0x206): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección .bss en /tmp/ccZ3prxM.o
collect2: error: ld returned 1 exit status
[Pablo Olivares Martínez pablo@laptop:~/Documentos/DGIIM2/AC/bp0/ejer10] 2021-03-17 miércoles
$

```

Este error de compilación se debe a que el enlazador lanza un error ya que el objetivo del esquema de direccionamiento relativo es mayor de lo que puede admitir con el desplazamiento de 32 bits del modo de direccionamiento relativo elegido.

## 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. Se generan aleatoriamente las componentes para vectores de tamaño mayor que 8 y se imprimen todas las componentes para vectores menores que 10.

```

/* 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(), rand(), srand(), 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)

#ifdef 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
#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
    if (N < 9)
        for (i = 0; i < N; i++)
        {
            v1[i] = N * 0.1 + i * 0.1;
            v2[i] = N * 0.1 - i * 0.1;
        }
    else
    {
        srand(time(0));
        for (i = 0; i < N; i++)
        {
            v1[i] = rand() / ((double) rand());
            v2[i] = rand() / ((double) rand()); // printf("%d:%f,%f/", i, v1[i], v2[i]);
        }
    }

    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;
}

```