

# Arquitectura de Computadores (AC)

## Cuaderno de prácticas.

### Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Jesús García León

Grupo de prácticas y profesor de prácticas: A2

Fecha de entrega: 18/03/2021

Fecha evaluación en clase: 19/03/2021

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)

- 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 `ej1`)

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

### RESPUESTA:

-PC:

```
usuario@usuario-VivoBook-ASUSLaptop-X580GD-N580GD: ~/Escritorio/AC
Archivo Editar Ver Buscar Terminal Ayuda
[JesusGarciaLeon :~/Escritorio/AC] 2021-03-13 sábado
lscpu
Arquitectura: x86_64
modo(s) de operación de las CPUs: 32-bit, 64-bit
Orden de los bytes: Little Endian
CPU(s): 12
Lista de la(s) CPU(s) en línea: 0-11
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»: 6
«Socket(s)»: 1
Modo(s) NUMA: 1
ID de fabricante: GenuineIntel
Familia de CPU: 6
Modelo: 158
Nombre del modelo: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
Revisión: 10
CPU MHz: 800.198
CPU MHz máx.: 4100.0000
CPU MHz mín.: 800.0000
BogoMIPS: 4399.99
Virtualización: VT-x
Caché L1d: 32K
Caché L1i: 32K
Caché L2: 256K
Caché L3: 9216K
CPU(s) del nodo NUMA 0: 0-11
Indicadores: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat p
se36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_
perfmnon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfperf pni pclmulqdq dtes64 monitor ds_cp
l vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer ae
s xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs lbrp stib
p tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx
rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notif
y hwp_act_window hwp_epp md_clear flush_l1d
```

**-AC:**

```

e2estudiante7@atcgrid:~
Archivo Editar Ver Buscar Terminal Ayuda
[JesusGarciaLeon e2estudiante7@atcgrid:~] 2021-03-13 sábado
$srunc -p ac -A ac 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 pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts
                        rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni dtes64 monitor ds_cpl vmx smx est tm2 ssse
                        3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 popcnt lahf_lm epb ssbd ibrs ibpb stibp tpr_shadow vnmi flexpri
                        ority ept vpid dtherm ida arat spec_ctrl intel_stibp flush_l1d

```

**-AC4**

```

e2estudiante7@atcgrid:~
Archivo Editar Ver Buscar Terminal Ayuda
[JesusGarciaLeon e2estudiante7@atcgrid:~] 2021-03-13 sábado
$srunc -p ac4 -A ac 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:               1197.399
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 pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopo
                        logy nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr p
                        dcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefet
                        ch epb cat_l3 cdp_l3 invpcid_single intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flex
                        priority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdsee
                        d adx snap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total c
                        qm_mbm_local dtherm ida arat pln pts pku ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities

```

**(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:**

-atcgrid4 tiene 32 cores físicos (2 sockets\*16 cores per socket) y 64 cores lógicos (2 sockets\*16 cores per socket \* 2 threads per core).

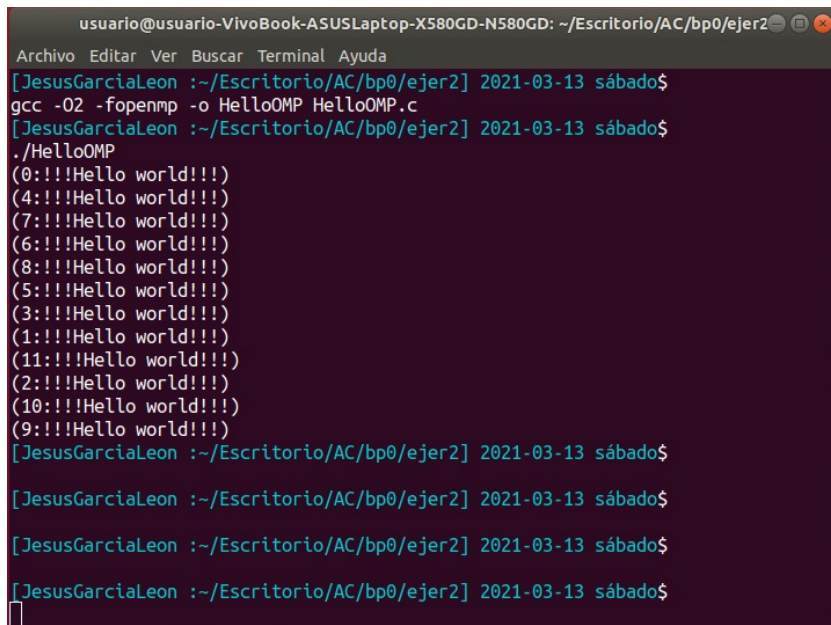
-atcgrid4 tiene 12 cores físicos (2 sockets\*6 cores per socket) y 24 cores lógicos (2 sockets\*6 cores per socket \* 2 threads per core).

-Mi PC tiene 6 cores físicos (1 socket\*6 cores per socket) y 12cores lógicos (1 socket\*6 cores per socket \* 2 threads per core).

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 `ej2`).

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

**RESPUESTA:**



```

usuario@usuario-VivoBook-ASUSLaptop-X580GD-N580GD: ~/Escritorio/AC/bp0/ej2
Archivo Editar Ver Buscar Terminal Ayuda
[JesusGarciaLeon :~/Escritorio/AC/bp0/ej2] 2021-03-13 sábado$
gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
[JesusGarciaLeon :~/Escritorio/AC/bp0/ej2] 2021-03-13 sábado$
./HelloOMP
(0:!!!Hello world!!!)
(4:!!!Hello world!!!)
(7:!!!Hello world!!!)
(6:!!!Hello world!!!)
(8:!!!Hello world!!!)
(5:!!!Hello world!!!)
(3:!!!Hello world!!!)
(1:!!!Hello world!!!)
(11:!!!Hello world!!!)
(2:!!!Hello world!!!)
(10:!!!Hello world!!!)
(9:!!!Hello world!!!)
[JesusGarciaLeon :~/Escritorio/AC/bp0/ej2] 2021-03-13 sábado$
[JesusGarciaLeon :~/Escritorio/AC/bp0/ej2] 2021-03-13 sábado$
[JesusGarciaLeon :~/Escritorio/AC/bp0/ej2] 2021-03-13 sábado$
[JesusGarciaLeon :~/Escritorio/AC/bp0/ej2] 2021-03-13 sábado$

```

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

**RESPUESTA:** Se imprime 12 veces porque mi PC tiene 12 cores lógicos (Nota: Se ha incluido un salto de línea para visualizar mejor la salida)

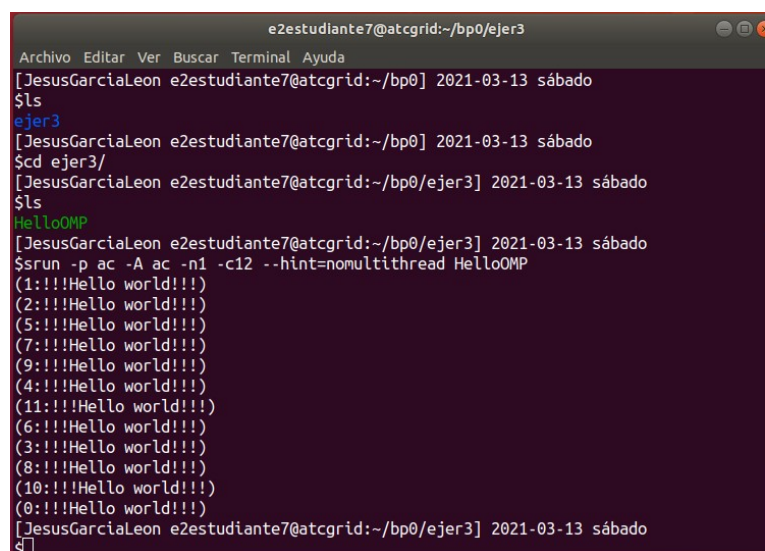
3. Copiar el ejecutable de `HelloOMP.c` que ha generado anteriormente y que se encuentra en el directorio `ej2` del PC al directorio `ej2` de su home en el *front-end* de `atcgrid`. Ejecutar este código en un nodo de cómputo de `atcgrid` (de 1 a 3) a través de `cola` ac del gestor de colas utilizando 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 -p ac -A ac -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:**



```

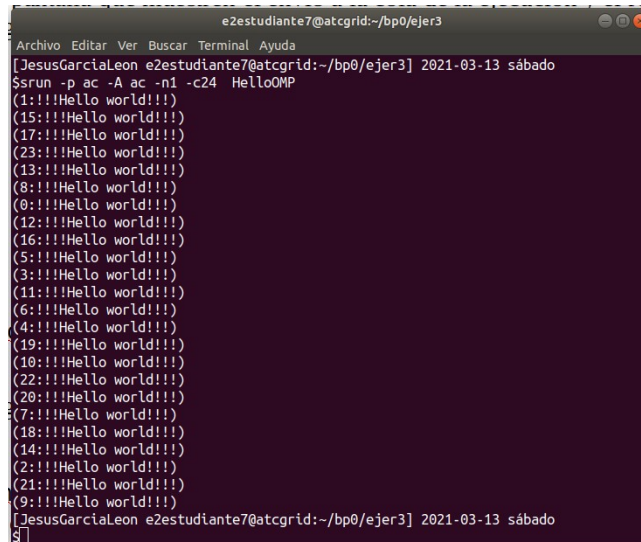
e2estudiante7@atcgrid:~/bp0/ej3
Archivo Editar Ver Buscar Terminal Ayuda
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0] 2021-03-13 sábado
$ls
ej2
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0] 2021-03-13 sábado
$cd ej2/
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ej2] 2021-03-13 sábado
$ls
HelloOMP
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ej2] 2021-03-13 sábado
$srun -p ac -A ac -n1 -c12 --hint=nomultithread HelloOMP
(1:!!!Hello world!!!)
(2:!!!Hello world!!!)
(5:!!!Hello world!!!)
(7:!!!Hello world!!!)
(9:!!!Hello world!!!)
(4:!!!Hello world!!!)
(11:!!!Hello world!!!)
(6:!!!Hello world!!!)
(3:!!!Hello world!!!)
(8:!!!Hello world!!!)
(10:!!!Hello world!!!)
(0:!!!Hello world!!!)
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ej2] 2021-03-13 sábado
$

```

**(b) `srun -p ac -A ac -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:**

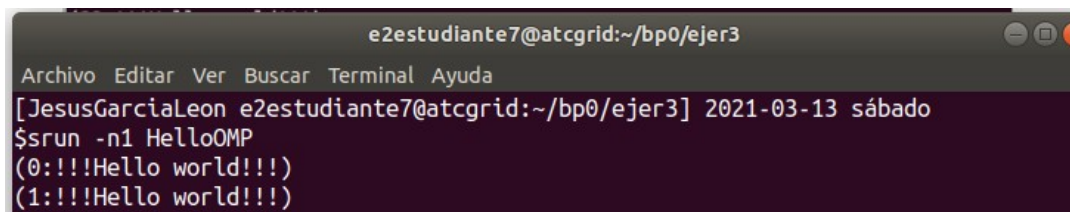


```
e2estudiante7@atcgrid:~/bp0/ejer3
Archivo Editar Ver Buscar Terminal Ayuda
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer3] 2021-03-13 sábado
$ srun -p ac -A ac -n1 -c24 HelloOMP
(1:!!!Hello world!!!)
(15:!!!Hello world!!!)
(17:!!!Hello world!!!)
(23:!!!Hello world!!!)
(13:!!!Hello world!!!)
(8:!!!Hello world!!!)
(0:!!!Hello world!!!)
(12:!!!Hello world!!!)
(16:!!!Hello world!!!)
(5:!!!Hello world!!!)
(3:!!!Hello world!!!)
(11:!!!Hello world!!!)
(6:!!!Hello world!!!)
(4:!!!Hello world!!!)
(19:!!!Hello world!!!)
(10:!!!Hello world!!!)
(22:!!!Hello world!!!)
(20:!!!Hello world!!!)
(7:!!!Hello world!!!)
(18:!!!Hello world!!!)
(14:!!!Hello world!!!)
(2:!!!Hello world!!!)
(21:!!!Hello world!!!)
(9:!!!Hello world!!!)
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer3] 2021-03-13 sábado
$
```

**(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:** Se está usando la partición ac (atcgrid 1-3) que es la cola que se usa por defecto



```
e2estudiante7@atcgrid:~/bp0/ejer3
Archivo Editar Ver Buscar Terminal Ayuda
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer3] 2021-03-13 sábado
$ srun -n1 HelloOMP
(0:!!!Hello world!!!)
(1:!!!Hello world!!!)
```

**(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)?

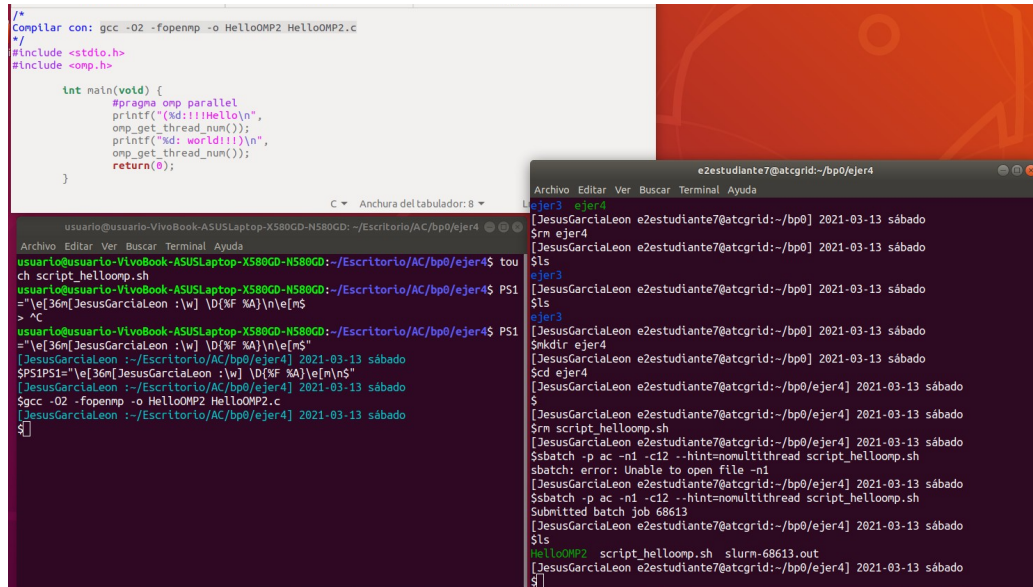
`srun -p ac4 -A ac -n1 -c32 --hint=nomultithread HelloOMP`



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:**



```

/*
Compilar con: gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
*/
#include <stdio.h>
#include <omp.h>

int main(void) {
    #pragma omp parallel
    printf("Hd:!!Hello\n",
        omp_get_thread_num());
    printf("Hd: world!!\n",
        omp_get_thread_num());
    return(0);
}

```

```

usuario@usuario-VivoBook-ASUSLaptop-X580GD-NS80GD: ~/Escritorio/AC/bp0/ejer4$
ch script_helloomp.sh
usuario@usuario-VivoBook-ASUSLaptop-X580GD-NS80GD: ~/Escritorio/AC/bp0/ejer4$ PS1
="[\36m[JesusGarciaLeon : \w] \D[\%F %A])\ne\ns
> ./c
usuario@usuario-VivoBook-ASUSLaptop-X580GD-NS80GD: ~/Escritorio/AC/bp0/ejer4$ PS1
="[\36m[JesusGarciaLeon : \w] \D[\%F %A])\ne\ns"
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer4] 2021-03-13 sábado
$PS1PS1="[\36m[JesusGarciaLeon : \w] \D[\%F %A])\ne\ns"
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer4] 2021-03-13 sábado
$gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer4] 2021-03-13 sábado
$

```

```

e2estudiante7@atcgrid:~/bp0/ejer4
ejer3 ejer4
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0] 2021-03-13 sábado
$rm ejer4
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0] 2021-03-13 sábado
$ls
ejer3
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0] 2021-03-13 sábado
$mkdir ejer4
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0] 2021-03-13 sábado
$cd ejer4
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer4] 2021-03-13 sábado
$
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer4] 2021-03-13 sábado
$rm script_helloomp.sh
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer4] 2021-03-13 sábado
$sbatch -p ac -n1 -c12 --hint=nomultithread script_helloomp.sh
sbatch: error: Unable to open file -n1
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer4] 2021-03-13 sábado
$sbatch -p ac -n1 -c12 --hint=nomultithread script_helloomp.sh
Submitted batch job 68613
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer4] 2021-03-13 sábado
$ls
HelloOMP2 script_helloomp.sh slurm-68613.out
[JesusGarciaLeon e2estudiante7@atcgrid:~/bp0/ejer4] 2021-03-13 sábado
$

```

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

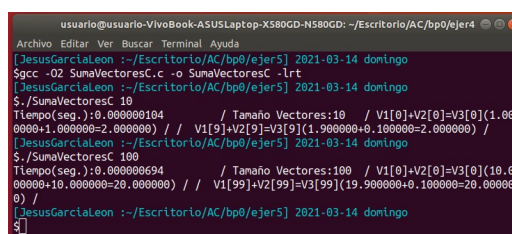
**RESPUESTA:** Se ejecuta en `atcgrid1`. Examinando `slurm-68613.out` (resultado de la ejecución del `script`) podemos ver esta información.

**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 `srun`, 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 `srun` 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 `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:**



```

usuario@usuario-VivoBook-ASUSLaptop-X580GD-NS80GD: ~/Escritorio/AC/bp0/ejer4$
Archivo Editar Ver Buscar Terminal Ayuda
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer5] 2021-03-14 domingo
$gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer5] 2021-03-14 domingo
$./SumaVectoresC 10
Tiempo(seg.):0.000000184 / Tamaño Vectores:10 / V1[0]+V2[0]=V3[0](1.00
0000+1.000000=2.000000) // V1[9]+V2[9]=V3[9](1.900000+0.100000=2.000000) /
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer5] 2021-03-14 domingo
$./SumaVectoresC 100
Tiempo(seg.):0.000000694 / Tamaño Vectores:100 / V1[0]+V2[0]=V3[0](10.0
00000+10.000000=20.000000) // V1[99]+V2[99]=V3[99](19.900000+0.100000=20.0000
00) /
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer5] 2021-03-14 domingo
$

```

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:** El tiempo en segundos que tardan en sumarse los 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:** Se devuelve en la estructura de datos `timespec`. Contiene 2 estructuras de datos, una para almacenar los segundos y otra para los nanosegundos.

```
__time_t tv_sec;           /* Seconds. */
__syscall_slong_t tv_nsec; /* Nanoseconds. */
```

(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:** Obtiene el valor actual de `CLOCK_ID` y lo guarda en `timespec`. Una para almacenar los segundos y otra para los nanosegundos.

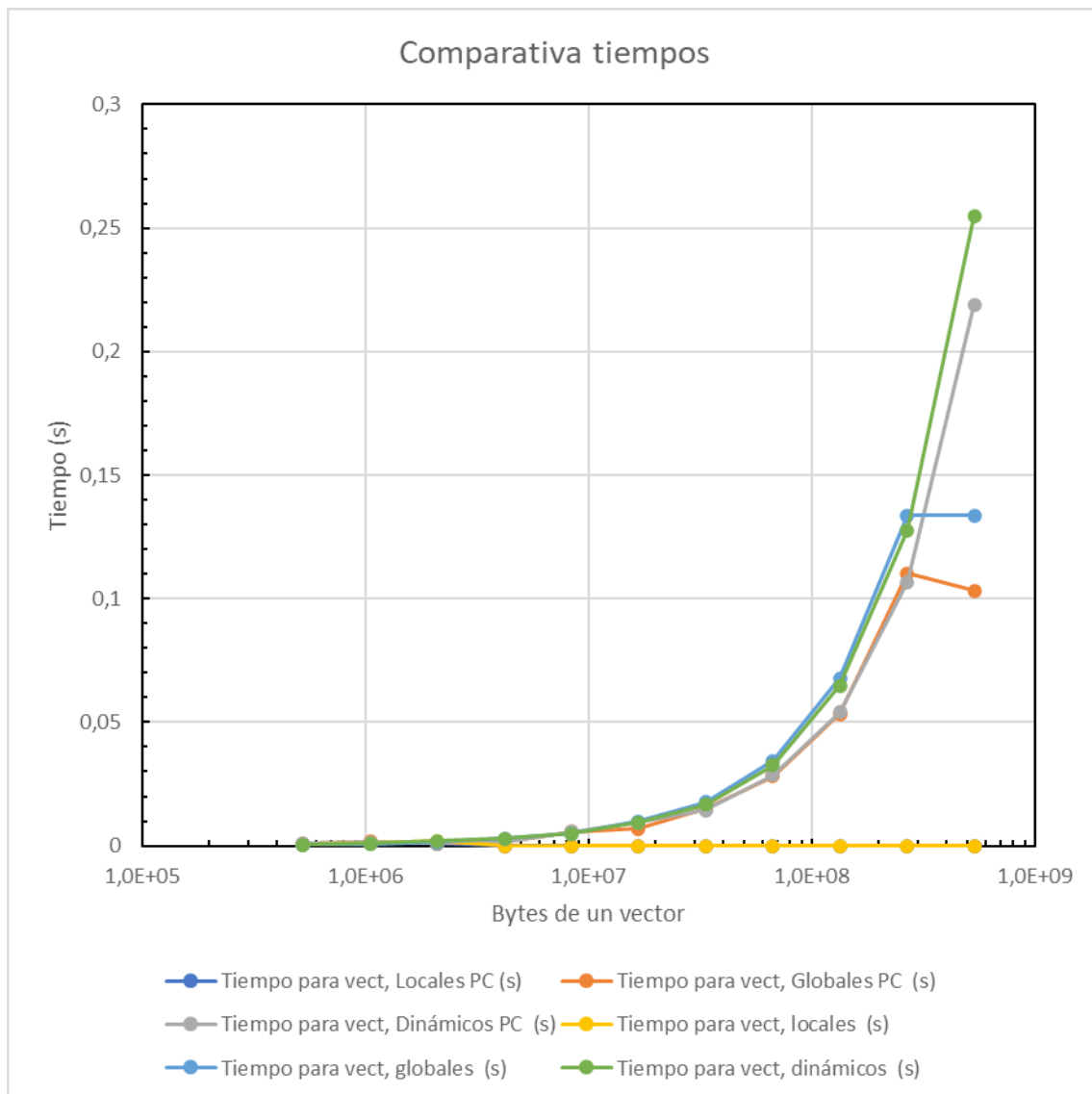
7. Rellenar una tabla como la Tabla 1 en una hoja de cálculo con los tiempos de ejecución del código del Listado 1 para vectores locales, globales y dinámicos (se pueden obtener errores en tiempo de ejecución o de compilación, ver ejercicio 9). Obtener estos resultados usando *scripts* (partir del *script* 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:**

(Tiempo en segundos)

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos	PC
65536	524288	0.000797437	0.000869347	0.000908855	PC
131072	1048576	0.001110429	0.001603057	0.001328175	PC
262144	2097152	0.000982893	0.000888058	0.000825374	PC
524288	4194304	core	0.001633004	0.001630787	PC
1048576	8388608	core	0.005618774	0.005594533	PC
2097152	16777216	core	0.006958186	0.009435875	PC
4194304	33554432	core	0.014827774	0.014338049	PC
8388608	67108864	core	0.028083511	0.028733471	PC
16777216	134217728	core	0.053237686	0.054166124	PC
33554432	268435456	core	0.110096492	0.106656211	PC
67108864	536870912	core	0.103239499	0.219065546	PC
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos	ATCGRID
65536	524288	0.000463820	0.000544942	0.000475042	ATCGRID
131072	1048576	0.000940383	0.000612399	0.000968464	ATCGRID
262144	2097152	0.001923704	0.001538290	0.001806859	ATCGRID
524288	4194304	core	0.002853266	0.002849471	ATCGRID
1048576	8388608	core	0.005015443	0.005010355	ATCGRID
2097152	16777216	core	0.009960953	0.009312012	ATCGRID
4194304	33554432	core	0.017716707	0.016616999	ATCGRID
8388608	67108864	core	0.034264406	0.032643437	ATCGRID
16777216	134217728	core	0.067759919	0.064846766	ATCGRID
33554432	268435456	core	0.133768365	0.127620560	ATCGRID
67108864	536870912	core	0.133666920	0.254988878	ATCGRID

8. 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:** Prácticamente no hay diferencia, ya que no se usa ningún tipo de paralelismo en el código.

## 9. 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:** Para tamaños a partir de 524288 se produce error debido a que se supera el tamaño de la pila.

```
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer7] 2021-03-14 domingo
$bash script_Local.sh

Vector local

- Para tamaño del vector = 65536:
Tiempo(seg.):0.000797437 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /

- Para tamaño del vector = 131072:
Tiempo(seg.):0.001110429 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /

- Para tamaño del vector = 262144:
Tiempo(seg.):0.000982893 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /

- Para tamaño del vector = 524288:
script_Local.sh: línea 10: 611 Violación de segmento ('core' generado) ./SumaVectoresLocales $N

- Para tamaño del vector = 1048576:
script_Local.sh: línea 10: 613 Violación de segmento ('core' generado) ./SumaVectoresLocales $N

- Para tamaño del vector = 2097152:
script_Local.sh: línea 10: 615 Violación de segmento ('core' generado) ./SumaVectoresLocales $N

- Para tamaño del vector = 4194304:
script_Local.sh: línea 10: 617 Violación de segmento ('core' generado) ./SumaVectoresLocales $N

- Para tamaño del vector = 8388608:
script_Local.sh: línea 10: 619 Violación de segmento ('core' generado) ./SumaVectoresLocales $N

- Para tamaño del vector = 16777216:
script_Local.sh: línea 10: 621 Violación de segmento ('core' generado) ./SumaVectoresLocales $N

- Para tamaño del vector = 33554432:
script_Local.sh: línea 10: 623 Violación de segmento ('core' generado) ./SumaVectoresLocales $N

- Para tamaño del vector = 67108864:
script_Local.sh: línea 10: 625 Violación de segmento ('core' generado) ./SumaVectoresLocales $N
```

(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 porque al ser variables globales no esta limitado por el tamaño de la pila

```
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer7] 2021-03-14 domingo
$bash script_Global.sh

Vector global

- Para tamaño del vector = 65536:
Tiempo(seg.):0.000869347 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /

- Para tamaño del vector = 131072:
Tiempo(seg.):0.001603057 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /

- Para tamaño del vector = 262144:
Tiempo(seg.):0.000880058 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /

- Para tamaño del vector = 524288:
Tiempo(seg.):0.001633004 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /

- Para tamaño del vector = 1048576:
Tiempo(seg.):0.005618774 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /

- Para tamaño del vector = 2097152:
Tiempo(seg.):0.006958186 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /

- Para tamaño del vector = 4194304:
Tiempo(seg.):0.014827774 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /

- Para tamaño del vector = 8388608:
Tiempo(seg.):0.028083511 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /

- Para tamaño del vector = 16777216:
Tiempo(seg.):0.053237686 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / / V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /

- Para tamaño del vector = 33554432:
Tiempo(seg.):0.110096492 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /

- Para tamaño del vector = 67108864:
Tiempo(seg.):0.103239499 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) / / V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000) /
```



(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 porque al ser memoria dinámica no se almacena en la pila.

```
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer7] 2021-03-14 domingo
$bash script_Dinamico.sh

Vector dinamico
Tiempo(seg.):0.000908855 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.001328175 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.000825374 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.001630787 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tiempo(seg.):0.005594533 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.009435875 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.014338049 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.028733471 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo(seg.):0.054166124 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / / V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
Tiempo(seg.):0.106656211 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
Tiempo(seg.):0.219065546 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) / / V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000) /
```

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

**RESPUESTA:** El tamaño máximo que puede almacenar es 4294967295 por ser un unsigned int.

(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:** No compila por ser demasiado grande el tamaño del vector

```
[JesusGarciaLeon :~/Escritorio/AC/bp0/ejer10] 2021-03-14 domingo$
gcc -O2 SumaVectoresGlobalesLimite.c -o SumaVectoresGlobalesLimite -lrt
/tmp/ccKw7Qew.o: En la función `main':
SumaVectoresGlobalesLimite.c:(.text.startup+0x5a): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/ccKw7Qew.o
SumaVectoresGlobalesLimite.c:(.text.startup+0xb1): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/ccKw7Qew.o
collect2: error: ld returned 1 exit status
```

## 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 SumaVectoresC.c -o SumaVectores -lrt
       gcc -O2 -S SumaVectoresC.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)

#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
    #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;
}

```