2º curso / 2º cuatr.

Grado Ingeniería Informática

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Elena Ortiz Moreno
Grupo de prácticas y profesor de prácticas: Niceto Rafael Luque Sola
Fecha de entrega: 11-03-21
Fecha evaluación en clase: 19-03-21

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:

Ejecución en atcgrid4:

```
[b2estudiante27@atcgrid ~]$ 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:               1159.332
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 dt
s acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts
 rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est
tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_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_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 sme
p bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pln pt
s pku ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities
[b2estudiante27@atcgrid ~]$
```

Ejecución en atcgrid restante:

```
elena@elena97om:~/Escritorio/6AÑO/AC/practicas/BP0$ ssh -X b2estudiante27@atcgrid.ugr.es
b2estudiante27@atcgrid.ugr.es's password:
Last login: Tue Jul 21 12:26:03 2020
[b2estudiante27@atcgrid ~]$ srun -p 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 dt
s 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 ssse3 cx16
 xtpr pdcm pcid dca sse4_1 sse4_2 popcnt lahf_lm epb ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ep
t vpid dtherm ida arat spec_ctrl intel_stibp flush_l1d
[b2estudiante27@atcgrid ~]$
```

Ejecución en mi PC:

```
elena@elena97om:~$ lscpu
Arquitectura:                    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: 2
Núcleo(s) por «socket»:          2
«Socket(s)»                      1
Modo(s) NUMA:                    1
ID de fabricante:                GenuineIntel
Familia de CPU:                  6
Modelo:                          142
Nombre del modelo:               Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
Revisión:                        9
CPU MHz:                         874.426
CPU MHz máx.:                    3100,0000
CPU MHz mín.:                    400,0000
BogoMIPS:                        5399.81
Virtualización:                  VT-x
Caché L1d:                       32K
Caché L1i:                       32K
Caché L2:                        256K
Caché L3:                        3072K
CPU(s) del nodo NUMA 0:          0-3
Indicadores:                     fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca c
mov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm cons
tant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pcl
mulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2api
c movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fau
lt epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid ept_ad fsg
sbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsave
opt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp md_clea
r flush_l1d
```

(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 16 cores físicos por socket, y como tenemos 2, son 32. Cada uno de ellos puede ejecutar 2 hebras a la vez, por lo que los núcleos lógicos son 64 → 16 * 2 * 2

Atcgrid1, 2 y 3: Tiene 6 cores físicos por socket, y como tenemos 2, son 12. Cada uno de ellos puede ejecutar 2 hebras a la vez, por lo que los núcleos lógicos son 24 → 6 * 2 * 2

PC: Se calcula de la misma manera. Tiene 2 cores físicos, los cuales ejecutan 2 hebras a la vez y por lo tanto tenemos 4 cores lógicos. 4 → 2 * 1 * 2

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:

```
elena@elena97om:~/Escritorio/6AÑO/AC/practicas/BP0/ejer2$ gcc -O2 -fopenmp -o He
lloOMP HelloOMP.c
elena@elena97om:~/Escritorio/6AÑO/AC/practicas/BP0/ejer2$ ./HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)(2:!!!Hello world!!!)(3:!!!Hello world
```

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

RESPUESTA:

Como mi PC tiene 4 núcleos lógicos, (2 físicos ejecutando 2 hebras cada uno), el programa se ejecuta 4 veces en las 4 hebras (hebra0, hebra1, hebra2 y hebra3)

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:

```
[b2estudiante27@atcgrid ejer3]$ srun --partition=ac --account=ac --ntasks=1 --cp
us-per-task=12 --hint=nomultithread HelloOMP
(11:!!!Hello world!!!)(1:!!!Hello world!!!)(2:!!!Hello world!!!)(7:!!!Hello worl
d!!!)(0:!!!Hello world!!!)(9:!!!Hello world!!!)(3:!!!Hello world!!!)(4:!!!Hello
world!!!)(10:!!!Hello world!!!)(6:!!!Hello world!!!)(8:!!!Hello world!!!)(5:!!!H
ello world!!!)[b2estudiante27@atcgrid ejer3]$
```

(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:

```
[b2estudiante27@atcgrid ejer3]$ srun -pac -Aac -n1 -c24  HelloOMP
(22:!!!Hello world!!!)(2:!!!Hello world!!!)(10:!!!Hello world!!!)(3:!!!Hello world!!!)(1
4:!!!Hello world!!!)(19:!!!Hello world!!!)(4:!!!Hello world!!!)(21:!!!Hello world!!!)(20
:!!!Hello world!!!)(11:!!!Hello world!!!)(13:!!!Hello world!!!)(7:!!!Hello world!!!)(17:
!!!Hello world!!!)(18:!!!Hello world!!!)(16:!!!Hello world!!!)(1:!!!Hello world!!!)(5:!!
!Hello world!!!)(23:!!!Hello world!!!)(12:!!!Hello world!!!)(0:!!!Hello world!!!)(8:!!!H
ello world!!!)(6:!!!Hello world!!!)(15:!!!Hello world!!!)(9:!!!Hello world!!!)[b2estudia
```

(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:

```
[b2estudiante27@atcgrid ejer3]$ srun -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[b2estudiante27@atcgrid ejer3]$
```

(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)?

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:

Nuevo código:

```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);
}
```

Compilación y ejecución:

```
elena@elena97om:~/Escritorio/6AÑO/AC/practicas/BP0/ejer4$ gcc -O2 -fopenmp -o He
lloOMP2 HelloOMP2.c
elena@elena97om:~/Escritorio/6AÑO/AC/practicas/BP0/ejer4$ ./HelloOMP2
(0:!!!Hello)(2:!!!Hello)(1:!!!Hello)(3:!!!Hello)(0:world!!!)elena@elena97om:~/Es
```

Copia del fichero del PC al atcgrid:

```
sftp> put /home/elena/Escritorio/6AÑO/AC/practicas/BP0/ejer4/HelloOMP2
Uploading /home/elena/Escritorio/6AÑO/AC/practicas/BP0/ejer4/HelloOMP2 to /home/
b2estudiante27/ejer4/HelloOMP2
/home/elena/Escritorio/6AÑO/AC/practicas/BP0/ 100% 8688   163.1KB/s   00:00
sftp> ls
HelloOMP2
```

Envío a la cola de ejecución del atcgrid con el script:

```
[b2estudiante27@atcgrid ejer4]$ sbatch -pac -n1 -c12 --hint=nomultithread script
_helloomp.sh
Submitted batch job 62149
```

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

RESPUESTA:

```
Id. usuario del trabajo: b2estudiante27
Id. del trabajo: 62149
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script):
/home/b2estudiante27/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
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)(7:!!!Hello)(3:!!!Hello)(1:!!!Hello)(10:!!!Hello)(4:!!!Hello)(11:!!!Hello)(8:!!!
Hello)(6:!!!Hello)(2:!!!Hello)(5:!!!Hello)(9:!!!Hello)(0:world!!!)
 2. Ejecución helloOMP varias veces con distinto no de threads:


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

En la salida del archivo slurm-62149.out, en el apartado de Nodos asignados al trabajo se puede ver que el nodo asignado es 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 `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:

Compilación SumaVectores.c:

```
elena@elena97om:~/Escritorio/6AÑO/AC/practicas/BP0/ejer5$ gcc -O2 SumaVectores.c
 -o SumaVectores -lrt
SumaVectores.c: In function ‘main’:
SumaVectores.c:45:34: warning: format ‘%u’ expects argument of type ‘unsigned in
t’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   printf("Tama@o Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                  ~^
                                  %lu
```

Ejecución SumaVectores:

```
elena@elena97om:~/Escritorio/6AÑO/AC/practicas/BP0/ejer5$ ./SumaVectores 3000
Tama@o Vectores:3000 (4 B)
Tiempo:0.000071569       / Tama@o Vectores:3000 / V1[0]+V2[0]=V3[0](300.000000+3
00.000000=600.000000) / / V1[2999]+V2[2999]=V3[2999](599.900000+0.100000=600.000
000) /
```

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 de ejecución (lo que tarda en realizarse la suma).

(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 la información en struct timespect, el cuál tiene esta estructura:

struct timespec {

time_t tv_sec;

/* segundos */

long tv_nsec;

/* 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:

La suma de segundos y nanosegundos (para obtener solo segundos), los cuales se calculan restando el tiempo final(cgt2) al inicial(cgt1), es decir, el tiempo en el que finaliza el programa menos el tiempo en el que se inicia.

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.)

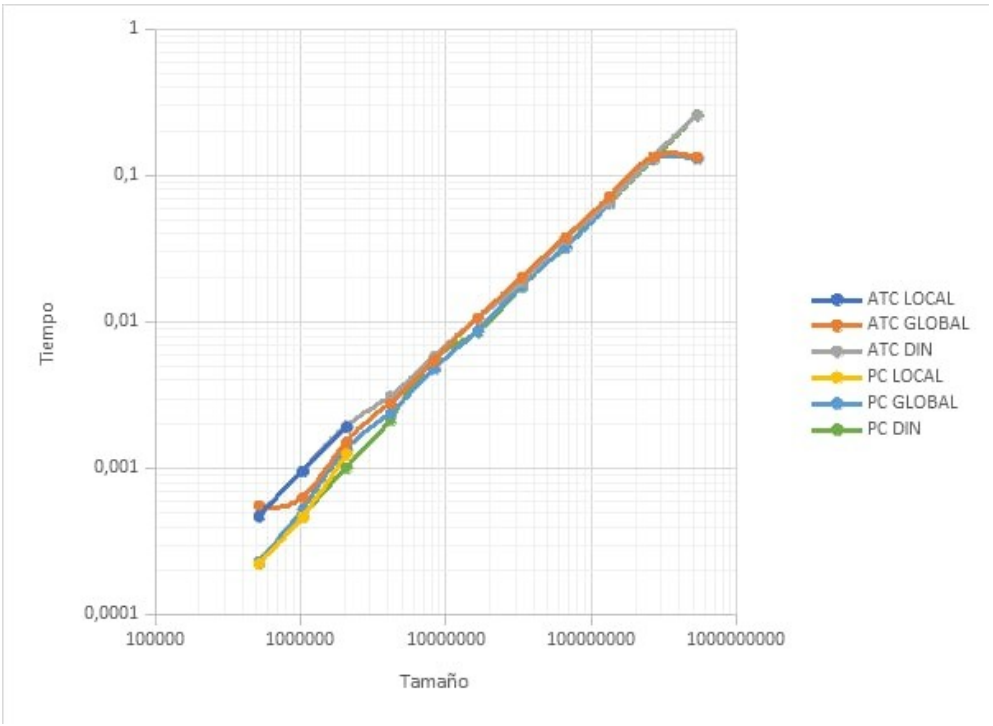
RESPUESTA:

RESULTADOS DE LAS EJECUCIONES EN EL PC				
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000221717	0,000230233	0,00022547
131072	1048576	0,000463504	0,000520775	0,000461735
262144	2097152	0,001244891	0,001332951	0,001005213
524288	4194304	-	0,002352617	0,00212226
1048576	8388608	-	0,00470615	0,005493446
2097152	16777216	-	0,008623783	0,008467916
4194304	33554432	-	0,017431419	0,017181559
8388608	67108864	-	0,032042234	0,032001026
16777216	134217728	-	0,06385527	0,063715904
33554432	268435456	-	0,127806474	0,128023601
67108864	536870912	-	0,128570184	0,255983795

RESULTADOS DE LAS EJECUCIONES EN ATCGRID				
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000467533	0,000552537	0,000476972
131072	1048576	0,00095234	0,00062968	0,000952679
262144	2097152	0,001894	0,001498708	0,001929465
524288	4194304	-	0,002739222	0,003073599
1048576	8388608	-	0,00541348	0,005726684
2097152	16777216	-	0,010553738	0,010446895
4194304	33554432	-	0,019956941	0,018526627
8388608	67108864	-	0,03743416	0,036532897
16777216	134217728	-	0,071096501	0,065122449
33554432	268435456	-	0,132884464	0,132145223
67108864	536870912	-	0,132846777	0,254262703

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: Sí, los tiempos tomados en el PC son mayores que en el ATCGRID.



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:

Se supera el tamaño de la pila debido al tamaño de los vectores.

PC Local:

```
Tama◆o Vectores:65536 (4 B)
Tiempo:0.000221717       / Tama◆o 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) /
Tama◆o Vectores:131072 (4 B)
Tiempo:0.000463504       / Tama◆o 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) /
Tama◆o Vectores:262144 (4 B)
Tiempo:0.001244891       / Tama◆o 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) /
Tama◆o Vectores:524288 (4 B)
./local.sh: línea 6:  5635 Violación de segmento  (`core' generado) ./SumaVectoresC $N
Tama◆o Vectores:1048576 (4 B)
./local.sh: línea 6:  5637 Violación de segmento  (`core' generado) ./SumaVectoresC $N
Tama◆o Vectores:2097152 (4 B)
./local.sh: línea 6:  5639 Violación de segmento  (`core' generado) ./SumaVectoresC $N
Tama◆o Vectores:4194304 (4 B)
./local.sh: línea 6:  5641 Violación de segmento  (`core' generado) ./SumaVectoresC $N
Tama◆o Vectores:8388608 (4 B)
./local.sh: línea 6:  5643 Violación de segmento  (`core' generado) ./SumaVectoresC $N
Tama◆o Vectores:16777216 (4 B)
./local.sh: línea 6:  5645 Violación de segmento  (`core' generado) ./SumaVectoresC $N
Tama◆o Vectores:33554432 (4 B)
./local.sh: línea 6:  5647 Violación de segmento  (`core' generado) ./SumaVectoresC $N
Tama◆o Vectores:67108864 (4 B)
./local.sh: línea 6:  5649 Violación de segmento  (`core' generado) ./SumaVectoresC $N
```

ATCGRID:

```
Tama◆o Vectores:65536 (4 B)
/Tiempo:0.000467533      / Tama◆o 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) /
Tama◆o Vectores:131072 (4 B)
/Tiempo:0.000952340      / Tama◆o 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) /
Tama◆o Vectores:262144 (4 B)
/Tiempo:0.001894000      / Tama◆o 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) /
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
```

(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:

Las variables globales se guardan en el heap, por lo que no dependen de la pila.

PC Local:

```
Tama♦o Vectores:65536 (4 B)
/Tiempo:0.000230233      / Tama♦o 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) /
Tama♦o Vectores:131072 (4 B)
/Tiempo:0.000520775      / Tama♦o 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) /
Tama♦o Vectores:262144 (4 B)
/Tiempo:0.001332951      / Tama♦o 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) /
Tama♦o Vectores:524288 (4 B)
/Tiempo:0.002352617      / Tama♦o 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) /
Tama♦o Vectores:1048576 (4 B)
/Tiempo:0.004706150      / Tama♦o 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) /
Tama♦o Vectores:2097152 (4 B)
/Tiempo:0.008623783      / Tama♦o 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) /
Tama♦o Vectores:4194304 (4 B)
/Tiempo:0.017431419      / Tama♦o 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) /
Tama♦o Vectores:8388608 (4 B)
/Tiempo:0.032042234      / Tama♦o 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) /
Tama♦o Vectores:16777216 (4 B)
/Tiempo:0.063855270      / Tama♦o 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) /
Tama♦o Vectores:33554432 (4 B)
/Tiempo:0.127806474      / Tama♦o 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) /
Tama♦o Vectores:67108864 (4 B)
/Tiempo:0.128570184      / Tama♦o 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) /
```

ATCGRID:

```
Tama♦o Vectores:65536 (4 B)
/Tiempo:0.000552537      / Tama♦o 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) /
Tama♦o Vectores:131072 (4 B)
/Tiempo:0.000629680      / Tama♦o 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) /
Tama♦o Vectores:262144 (4 B)
/Tiempo:0.001498708      / Tama♦o 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) /
Tama♦o Vectores:524288 (4 B)
/Tiempo:0.002739222      / Tama♦o 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) /
Tama♦o Vectores:1048576 (4 B)
/Tiempo:0.005413480      / Tama♦o 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) /
Tama♦o Vectores:2097152 (4 B)
/Tiempo:0.010553738      / Tama♦o 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) /
Tama♦o Vectores:4194304 (4 B)
/Tiempo:0.019956941      / Tama♦o 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) /
Tama♦o Vectores:8388608 (4 B)
/Tiempo:0.037434160      / Tama♦o 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) /
Tama♦o Vectores:16777216 (4 B)
/Tiempo:0.071096501      / Tama♦o 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) /
Tama♦o Vectores:33554432 (4 B)
/Tiempo:0.132884464      / Tama♦o 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) /
Tama♦o Vectores:67108864 (4 B)
/Tiempo:0.132846777      / Tama♦o 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) /
```

(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 se obtienen errores.

PC Local:

```
Tama◆o Vectores:65536 (4 B)
/Tiempo:0.000225470      / Tama◆o 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) /
Tama◆o Vectores:131072 (4 B)
/Tiempo:0.000461735      / Tama◆o 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) /
Tama◆o Vectores:262144 (4 B)
/Tiempo:0.001005213      / Tama◆o 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) /
Tama◆o Vectores:524288 (4 B)
/Tiempo:0.002122260      / Tama◆o 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) /
Tama◆o Vectores:1048576 (4 B)
/Tiempo:0.005493446      / Tama◆o 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) /
Tama◆o Vectores:2097152 (4 B)
/Tiempo:0.008467916      / Tama◆o 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) /
Tama◆o Vectores:4194304 (4 B)
/Tiempo:0.017181559      / Tama◆o 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) /
Tama◆o Vectores:8388608 (4 B)
/Tiempo:0.032001026      / Tama◆o 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) /
Tama◆o Vectores:16777216 (4 B)
/Tiempo:0.063715904      / Tama◆o 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) /
Tama◆o Vectores:33554432 (4 B)
/Tiempo:0.128023601      / Tama◆o 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) /
Tama◆o Vectores:67108864 (4 B)
/Tiempo:0.255983795      / Tama◆o 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) /
```

ATCGRID:

```
Tama◆o Vectores:65536 (4 B)
/Tiempo:0.000476972      / Tama◆o 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) /
Tama◆o Vectores:131072 (4 B)
/Tiempo:0.000952679      / Tama◆o 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) /
Tama◆o Vectores:262144 (4 B)
/Tiempo:0.001929465      / Tama◆o 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) /
Tama◆o Vectores:524288 (4 B)
/Tiempo:0.003073599      / Tama◆o 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) /
Tama◆o Vectores:1048576 (4 B)
/Tiempo:0.005726684      / Tama◆o 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) /
Tama◆o Vectores:2097152 (4 B)
/Tiempo:0.010446895      / Tama◆o 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) /
Tama◆o Vectores:4194304 (4 B)
/Tiempo:0.018526627      / Tama◆o 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) /
Tama◆o Vectores:8388608 (4 B)
/Tiempo:0.036532897      / Tama◆o 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) /
Tama◆o Vectores:16777216 (4 B)
/Tiempo:0.065122449      / Tama◆o 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) /
Tama◆o Vectores:33554432 (4 B)
/Tiempo:0.132145223      / Tama◆o 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) /
Tama◆o Vectores:67108864 (4 B)
/Tiempo:0.254262703      / Tama◆o 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) /
```

3. **(a)** ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.
RESPUESTA:
- Es de tipo unsigned int, teniendo en cuenta su tamaño de 4 bytes, es decir 32 bits, el numero máximo que podemos almacenar es 2^32-1 = 4,294,967,295

(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:

```
elena@elena97om:~/Escritorio/6AÑO/AC/practicas/BP0/ejer10$ gcc -O2 SumaVectoresC
.c -o SumaVectoresC -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:44:33: warning: format ‘%u’ expects argument of type ‘unsigned i
nt’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
/tmp/ccrXHWjY.o: En la función `main':
SumaVectoresC.c:(.text.startup+0x76): reubicación truncada para ajustar: R_X86_6
4_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/ccrXHWjY.o
SumaVectoresC.c:(.text.startup+0xc9): reubicación truncada para ajustar: R_X86_6
4_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/ccrXHWjY.o
collect2: error: ld returned 1 exit status
```

Se produce un error porque se crea un vector que sobrepasa el tamaño permitido, calculado antes.

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;
}
```