2º curso / 2º cuatr.

Grado Ingeniería Informática

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): David Gómez Hernández
Grupo de prácticas y profesor/a de prácticas: C3 María Isabel García
Fecha de entrega:
Fecha evaluación en clase:

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

Parte I. Ejercicios basados en los ejemplos del seminario práctico

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

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

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

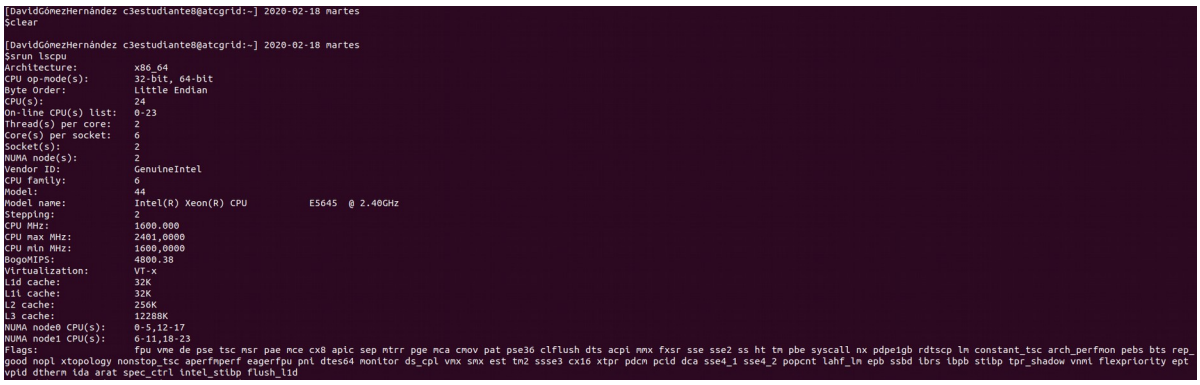
1. Ejecutar `lscpu` en el PC y en un nodo de cómputo de atcgrid. (Crear directorio `ejer1`)

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

RESPUESTA:

lscpu en PC

lscpu en atcgrid



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

RESPUESTA:

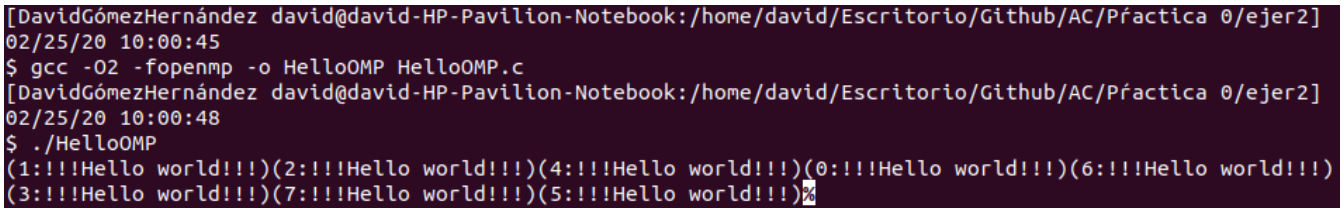
PC → Cores físicos: 4 – Cores lógicos: 8

Atcgrid → Cores físicos: 12 – Cores lógicos: 24

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

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

RESPUESTA:



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

RESPUESTA:

El número de "Hello wolrd" se corresponde con el número de cores que hay en mi ordenador local

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

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

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

RESPUESTA:

```
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$srun -p ac --cpus-per-task=12 --hint=nomultithread HelloOMP -n1
(0:!!!Hello world!!!)(8:!!!Hello world!!!)(2:!!!Hello world!!!)(6:!!!Hello world!!!)(4:!!!Hello world!!!)
(10:!!!Hello world!!!)(1:!!!Hello world!!!)(5:!!!Hello world!!!)(11:!!!Hello world!!!)(7:!!!Hello world!!
!)(9:!!!Hello world!!!)(3:!!!Hello world!!!)(0:!!!Hello world!!!)(3:!!!Hello world!!!)(2:!!!Hello world!!
!)(7:!!!Hello world!!!)(9:!!!Hello world!!!)(10:!!!Hello world!!!)(1:!!!Hello world!!!)(4:!!!Hello world!
!!)(5:!!!Hello world!!!)(8:!!!Hello world!!!)(6:!!!Hello world!!!)(11:!!!Hello world!!!)[DavidGómezHernán
dez c3estudiante8@atcgrid:~/bp0/ejer2] 2020-02-25 martes
```

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

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

RESPUESTA:

```
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$srun -p ac --cpus-per-task=24 HelloOMP -n1
srun: job 9563 queued and waiting for resources
srun: job 9563 has been allocated resources
(22:!!!Hello world!!!)(23:!!!Hello world!!!)(11:!!!Hello world!!!)(18:!!!Hello world!!!)(13:!!!Hello worl
d!!!)(19:!!!Hello world!!!)(12:!!!Hello world!!!)(10:!!!Hello world!!!)(15:!!!Hello world!!!)(9:!!!Hello
world!!!)(7:!!!Hello world!!!)(16:!!!Hello world!!!)(5:!!!Hello world!!!)(20:!!!Hello world!!!)(2:!!!Hell
o world!!!)(1:!!!Hello world!!!)(6:!!!Hello world!!!)(4:!!!Hello world!!!)(8:!!!Hello world!!!)(14:!!!Hel
lo world!!!)(3:!!!Hello world!!!)(21:!!!Hello world!!!)(17:!!!Hello world!!!)(0:!!!Hello world!!!)[DavidG
```

(c) `srun -p ac –n1 HelloOMP`

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

RESPUESTA:

```
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$srun -p ac HelloOMP -n1
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer2] 2020-02
```

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

RESPUESTA:

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

RESPUESTA:

Código

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

int main(void) {
#pragma omp parallel
    printf("(%d:Hello )", omp_get_thread_num());

#pragma omp parallel
    printf("(%d: world)", omp_get_thread_num());

    return(0);
}
```

Ejecución en PC Local

```
[DavidGómezHernández david@david-HP-Pavilion-Notebook:/home/david/Escritorio/Github/AC/Práctica 0/ejer4]
02/25/20 10:24:53
$ ./HelloOMP2
(1:Hello )(3:Hello )(6:Hello )(5:Hello )(4:Hello )(2:Hello )(0:Hello )(7:Hello )(6: world)(0: world)(4: w
orld)(2: world)(5: world)(3: world)(7: world)(1: world)%
```

Ejecución Atcgrid

```
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer4] 2020-02-25 martes
$sbatch -p ac --cpus-per-task=12 --hint=nomultithread script_helloomp2.sh -n1
Submitted batch job 9678
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer4] 2020-02-25 martes
$ls
HelloOMP2  HelloOMP2.c  script_helloomp2.sh  slurm-9678.out
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer4] 2020-02-25 martes
$cat slurm-9678.out
Id. usuario del trabajo: c3estudiante8
Id. del trabajo: 9678
Nombre del trabajo especificado por usuario: helloOMP2
Directorio de trabajo (en el que se ejecuta el script):
/home/c3estudiante8/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid
No de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

 1. Ejecución helloOMP2 una vez sin cambiar no de threads (valor
por defecto):

(0:Hello )(10:Hello )(8:Hello )(7:Hello )(9:Hello )(6:Hello )(1:Hello )(5:Hello )(3:Hello )(4:Hello )(2:H
ello )(11:Hello )(7: world)(9: world)(5: world)(1: world)(0: world)(6: world)(4: world)(3: world)(2: worl
d)(11: world)(10: world)(8: world)(7:Hello )(3:Hello )(8:Hello )(6:Hello )(9:Hello )(1:Hello )(5:Hello )(
10:Hello )(4:Hello )(11:Hello )(0:Hello )(2:Hello )(1: world)(7: world)(9: world)(0: world)(2: world)(10:
 world)(11: world)(4: world)(6: world)(5: world)(3: world)(8: world)
 2. Ejecución helloOMP2 varias veces con distinto no de threads:


 - Para 12 threads:
(9:Hello )(1:Hello )(10:Hello )(8:Hello )(4:Hello )(5:Hello )(2:Hello )(3:Hello )(0:Hello )(6:Hello )(7:H
ello )(11:Hello )(0: world)(9: world)(5: world)(8: world)(2: world)(7: world)(3: world)(11: world)(6: wor
ld)(10: world)(1: world)(4: world)(2:Hello )(0:Hello )(9:Hello )(11:Hello )(5:Hello )(6:Hello )(4:Hello )
(3:Hello )(8:Hello )(10:Hello )(1:Hello )(7:Hello )(11: world)(3: world)(6: world)(4: world)(7: world)(0:
 world)(9: world)(2: world)(5: world)(8: world)(1: world)(10: world)
 - Para 6 threads:
(1:Hello )(5:Hello )(4:Hello )(0:Hello )(3:Hello )(2:Hello )(1: world)(5: world)(3: world)(0: world)(2: w
orld)(4: world)(1:Hello )(0:Hello )(3:Hello )(4:Hello )(5:Hello )(2:Hello )(1: world)(5: world)(4: world)
(2: world)(0: world)(3: world)
 - Para 3 threads:
(1:Hello )(0:Hello )(2:Hello )(1: world)(0: world)(2: world)(1:Hello )(0:Hello )(2:Hello )(1: world)(2: w
orld)(0: world)
 - Para 1 threads:
(0:Hello )(0: world)(0:Hello )(0: world)[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer4] 2020-02-2
5 martes
$
```

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

RESPUESTA:

El nodo asignado es el nodo 1. En la información de arriba sale Nodos asignados al trabajo: atcgrid1

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

Parte II. Resto de ejercicios

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

   RESPUESTA:

```
[DavidGómezHernández david@david-HP-Pavilion-Notebook:/home/david/Escritorio/Github/AC/Práctica 0/ejer5]
02/25/20 10:43:39
$ gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:34: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tama◆o Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
[DavidGómezHernández david@david-HP-Pavilion-Notebook:/home/david/Escritorio/Github/AC/Práctica 0/ejer5]
02/25/20 10:43:41
$ ./SumaVectoresC 100
Tama◆o Vectores:100 (4 B)
Tiempo:0.000000595       / Tama◆o Vectores:100  / V1[0]+V2[0]=V3[0](10.000000+10.000000=20.000000) / / V1
[99]+V2[99]=V3[99](19.900000+0.100000=20.000000) /
```

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:
   Calcula el tiempo de ejecución de la suma realizando la diferencia entre cg1 y cgt2, que son los instantes de tiempo antes de realizar la suma y después de haber realizado la suma, respectivamente. El resultado final lo devuelve en una variable de tipo double

   (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:

   Es una estructura de tipo timespec que tiene dos variables la primera es de tipo time_t y guarda el tiempo en segundos(tv_sec) y la segunda es de tipo long, que guarda el tiempo en nanosegundos(tv_nsec),
   (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 un entero que se utiliza de forma de control: 0 si se ha ejecutado con éxito y -1 si ha habido error de ejecución.

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

RESPUESTA:

Ejecución en PC

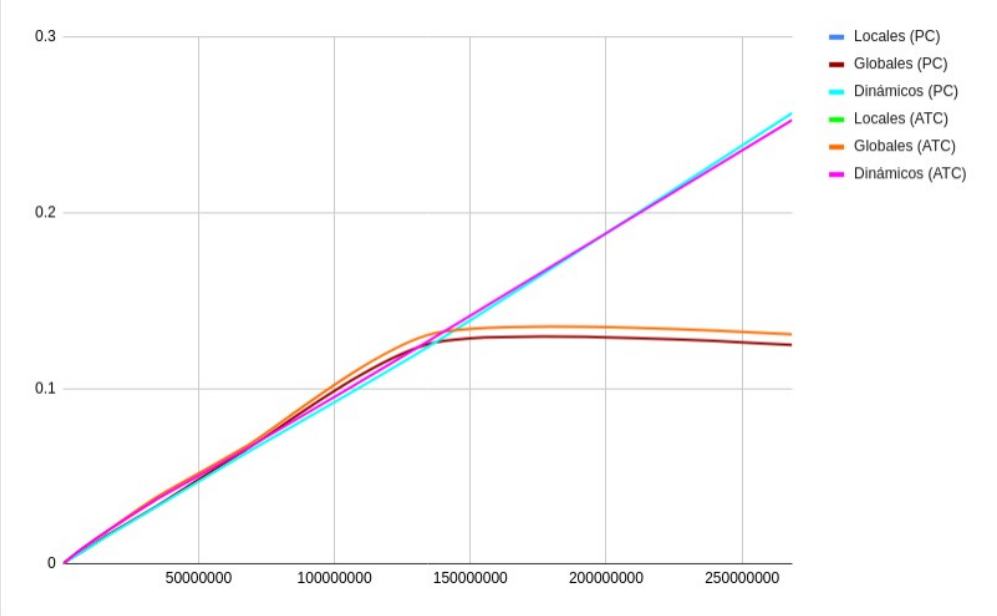
? de Component	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	262144	0.000851000	0.000229061	0.000250141
131072	524288	0.001147054	0.000457906	0.001080166
262144	1048576	0.001012054	0.000938135	0.000981585
524288	2097152		0.002356738	0.002061331
1048576	4194304		0.004381894	0.004149838
2097152	8388608		0.008280623	0.008159812
4194304	16777216		0.016785278	0.016313948
8388608	33554432		0.032018996	0.031778790
16777216	67108864		0.064786882	0.062770674
33554432	134217728		0.125163717	0.123406319
67108864	268435456		0.124700365	0.256684519

Ejecución en Atcgrid

? de Component	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	262144	0.000371427	0.000548974	0.000364219
131072	524288	0.000857811	0.000492024	0.000903824
262144	1048576	0.001647810	0.001266985	0.001451523
524288	2097152		0.002570018	0.002784191
1048576	4194304		0.005352974	0.005349464
2097152	8388608		0.010059901	0.010233748
4194304	16777216		0.019308865	0.019082742
8388608	33554432		0.037054890	0.035833129
16777216	67108864		0.066712315	0.065211430
33554432	134217728		0.130365024	0.126693642
67108864	268435456		0.130751958	0.252785429

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:



Si notamos diferencia en los tiempos de ejecución entre los vectores dinámicos y globales, siendo más rápido la ejecución en el atcgrid de los primeros (dinámico), y más rápido en mi PC de los segundos (globales)

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

RESPUESTA:

Se obtienen errores cuando el numero de componentes es superior a 262144 ya que los vectores locales superan el tamaño de la pila y generan violación del segmento.

Ejecución en PC

```
[DavidGómezHernández david@david-HP-Pavilion-Notebook:/home/david/Escritorio/Github/AC/Practica 0/ejer7]
02/26/20 13:01:49
$ ./SumaVectoresTest.sh


CALCULO LOCAL

Tamaño Local: 65536
Tama◆o Vectores:65536 (4 B)
Tiempo:0.000551275       / 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) /
Tamaño Local: 131072
Tama◆o Vectores:131072 (4 B)
Tiempo:0.001652743       / Tama◆o Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Local: 262144
Tama◆o Vectores:262144 (4 B)
Tiempo:0.003907606       / Tama◆o Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Local: 524288
Tama◆o Vectores:524288 (4 B)
./SumaVectoresTest.sh: línea 9:  5025 Violación de segmento  (`core' generado) ./SumaVectoresLocal $i
Tamaño Local: 1048576
Tama◆o Vectores:1048576 (4 B)
./SumaVectoresTest.sh: línea 9:  5027 Violación de segmento  (`core' generado) ./SumaVectoresLocal $i
Tamaño Local: 2097152
Tama◆o Vectores:2097152 (4 B)
./SumaVectoresTest.sh: línea 9:  5029 Violación de segmento  (`core' generado) ./SumaVectoresLocal $i
Tamaño Local: 4194304
Tama◆o Vectores:4194304 (4 B)
./SumaVectoresTest.sh: línea 9:  5031 Violación de segmento  (`core' generado) ./SumaVectoresLocal $i
Tamaño Local: 8388608
Tama◆o Vectores:8388608 (4 B)
./SumaVectoresTest.sh: línea 9:  5033 Violación de segmento  (`core' generado) ./SumaVectoresLocal $i
Tamaño Local: 16777216
Tama◆o Vectores:16777216 (4 B)
./SumaVectoresTest.sh: línea 9:  5035 Violación de segmento  (`core' generado) ./SumaVectoresLocal $i
Tamaño Local: 33554432
Tama◆o Vectores:33554432 (4 B)
./SumaVectoresTest.sh: línea 9:  5037 Violación de segmento  (`core' generado) ./SumaVectoresLocal $i
Tamaño Local: 67108864
Tama◆o Vectores:67108864 (4 B)
./SumaVectoresTest.sh: línea 9:  5039 Violación de segmento  (`core' generado) ./SumaVectoresLocal $i
./SumaVectoresTest.sh: línea 10: error sintáctico cerca del elemento inesperado `done'
./SumaVectoresTest.sh: línea 10: `done'
```

Ejecución en atcgrid

```
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer7] 2020-02-26 miércoles
$sbatch -p ac SumaVectoresTest.sh -n1
Submitted batch job 11760
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer7] 2020-02-26 miércoles
$cat slurm-11760.out


CALCULO LOCAL

Tamaño Local: 65536
Tama◆o Vectores:65536 (4 B)
Tiempo:0.000483335       / 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) /
Tamaño Local: 131072
Tama◆o Vectores:131072 (4 B)
Tiempo:0.000889382       / Tama◆o Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Local: 262144
Tama◆o Vectores:262144 (4 B)
Tiempo:0.001912606       / Tama◆o Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Local: 524288
/var/spool/slurmd/job11760/slurm_script: línea 9: 24827 Violación de segmento  (`core' generado) ./SumaVe
ctoresLocal $i
Tamaño Local: 1048576
/var/spool/slurmd/job11760/slurm_script: línea 9: 24829 Violación de segmento  (`core' generado) ./SumaVe
ctoresLocal $i
Tamaño Local: 2097152
/var/spool/slurmd/job11760/slurm_script: línea 9: 24833 Violación de segmento  (`core' generado) ./SumaVe
ctoresLocal $i
Tamaño Local: 4194304
/var/spool/slurmd/job11760/slurm_script: línea 9: 24836 Violación de segmento  (`core' generado) ./SumaVe
ctoresLocal $i
Tamaño Local: 8388608
/var/spool/slurmd/job11760/slurm_script: línea 9: 24839 Violación de segmento  (`core' generado) ./SumaVe
ctoresLocal $i
Tamaño Local: 16777216
/var/spool/slurmd/job11760/slurm_script: línea 9: 24841 Violación de segmento  (`core' generado) ./SumaVe
ctoresLocal $i
Tamaño Local: 33554432
/var/spool/slurmd/job11760/slurm_script: línea 9: 24843 Violación de segmento  (`core' generado) ./SumaVe
ctoresLocal $i
Tamaño Local: 67108864
/var/spool/slurmd/job11760/slurm_script: línea 9: 24845 Violación de segmento  (`core' generado) ./SumaVe
ctoresLocal $i
/var/spool/slurmd/job11760/slurm_script: línea 10: error sintáctico cerca del elemento inesperado `done'
/var/spool/slurmd/job11760/slurm_script: línea 10: `done'
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer7] 2020-02-26 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:

No se obtienen errores para ningún tamaño ya que los globales no dependen del tamaño de la pila

Ejecución en PC

```
[DavidGómezHernández david@david-HP-Pavilion-Notebook:/home/david/Escritorio/Github/AC/Practica 0/ejer7]
02/26/20 13:05:14
$ ./SumaVectoresTest.sh


CALCULO GLOBAL

Tamaño Global: 65536
Tama◆o Vectores:65536 (4 B)
Tiempo:0.000285113       / 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) /
Tamaño Global: 131072
Tama◆o Vectores:131072 (4 B)
Tiempo:0.000534601       / Tama◆o Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Global: 262144
Tama◆o Vectores:262144 (4 B)
Tiempo:0.001233655       / Tama◆o Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Global: 524288
Tama◆o Vectores:524288 (4 B)
Tiempo:0.002817130       / Tama◆o Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104
857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamaño Global: 1048576
Tama◆o Vectores:1048576 (4 B)
Tiempo:0.004371112       / Tama◆o Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=2
09715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamaño Global: 2097152
Tama◆o Vectores:2097152 (4 B)
Tiempo:0.008833145       / Tama◆o Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=4
19430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamaño Global: 4194304
Tama◆o Vectores:4194304 (4 B)
Tiempo:0.016811892       / Tama◆o Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=8
38860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamaño Global: 8388608
Tama◆o Vectores:8388608 (4 B)
Tiempo:0.032096699       / Tama◆o Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1
677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamaño Global: 16777216
Tama◆o Vectores:16777216 (4 B)
Tiempo:0.063208274       / 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) /
Tamaño Global: 33554432
Tama◆o Vectores:33554432 (4 B)
Tiempo:0.126656363       / 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) /
Tamaño Global: 67108864
Tama◆o Vectores:67108864 (4 B)
Tiempo:0.133102765       / 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) /
```

Ejecución en Atcgrid

```
$sbatch -p ac SumaVectoresTest.sh -n1
Submitted batch job 11771
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer7] 2020-02-26 miércoles
$cat slurm-11771.out


CALCULO GLOBAL

Tamaño Global: 65536
Tama◆o Vectores:65536 (4 B)
Tiempo:0.000956384       / 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) /
Tamaño Global: 131072
Tama◆o Vectores:131072 (4 B)
Tiempo:0.000529098       / Tama◆o Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Global: 262144
Tama◆o Vectores:262144 (4 B)
Tiempo:0.001425880       / Tama◆o Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Global: 524288
Tama◆o Vectores:524288 (4 B)
Tiempo:0.002776055       / Tama◆o Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104
857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamaño Global: 1048576
Tama◆o Vectores:1048576 (4 B)
Tiempo:0.005532641       / Tama◆o Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=2
09715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamaño Global: 2097152
Tama◆o Vectores:2097152 (4 B)
Tiempo:0.010169309       / Tama◆o Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=4
19430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamaño Global: 4194304
Tama◆o Vectores:4194304 (4 B)
Tiempo:0.018559350       / Tama◆o Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=8
38860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamaño Global: 8388608
Tama◆o Vectores:8388608 (4 B)
Tiempo:0.036374914       / Tama◆o Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1
677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamaño Global: 16777216
Tama◆o Vectores:16777216 (4 B)
Tiempo:0.066453705       / 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) /
Tamaño Global: 33554432
Tama◆o Vectores:33554432 (4 B)
Tiempo:0.131168744       / 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) /
Tamaño Global: 67108864
Tama◆o Vectores:67108864 (4 B)
Tiempo:0.130930330       / 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 obtiene ningún error ya que el tamaño de los vectores dinámicos se va reutilizando a medida que el programa se ejecuta.

Ejecución en PC

```
[DavidGómezHernández david@david-HP-Pavilion-Notebook:/home/david/Escritorio/Github/AC/Práctica 0/ejer7]
02/26/20 13:08:09
$ ./SumaVectoresTest.sh

CALCULO DINAMICO
Tamaño Dinamico: 65536
Tama◆o Vectores:65536 (4 B)
Tiempo:0.000289168       / 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) /
Tamaño Dinamico: 131072
Tama◆o Vectores:131072 (4 B)
Tiempo:0.000488838       / Tama◆o Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Dinamico: 262144
Tama◆o Vectores:262144 (4 B)
Tiempo:0.001014712       / Tama◆o Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Dinamico: 524288
Tama◆o Vectores:524288 (4 B)
Tiempo:0.002241310       / Tama◆o Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104
857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamaño Dinamico: 1048576
Tama◆o Vectores:1048576 (4 B)
Tiempo:0.004595211       / Tama◆o Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=2
09715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamaño Dinamico: 2097152
Tama◆o Vectores:2097152 (4 B)
Tiempo:0.008689446       / Tama◆o Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=4
19430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamaño Dinamico: 4194304
Tama◆o Vectores:4194304 (4 B)
Tiempo:0.019417151       / Tama◆o Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=8
38860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamaño Dinamico: 8388608
Tama◆o Vectores:8388608 (4 B)
Tiempo:0.032897352       / Tama◆o Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1
677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamaño Dinamico: 16777216
Tama◆o Vectores:16777216 (4 B)
Tiempo:0.079265052       / 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) /
Tamaño Dinamico: 33554432
Tama◆o Vectores:33554432 (4 B)
Tiempo:0.126541245       / 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) /
Tamaño Dinamico: 67108864
Tama◆o Vectores:67108864 (4 B)
Tiempo:0.273075729       / 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) /
```

Ejecución en Atcgrid

```
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer7] 2020-02-26 miércoles
$sbatch -p ac SumaVectoresTest.sh -n1
Submitted batch job 11772
[DavidGómezHernández c3estudiante8@atcgrid:~/bp0/ejer7] 2020-02-26 miércoles
$cat slurm-11772.out

CALCULO DINAMICO
Tamaño Dinamico: 65536
Tama◆o Vectores:65536 (4 B)
Tiempo:0.000448037       / 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) /
Tamaño Dinamico: 131072
Tama◆o Vectores:131072 (4 B)
Tiempo:0.000890443       / Tama◆o Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Dinamico: 262144
Tama◆o Vectores:262144 (4 B)
Tiempo:0.001017344       / Tama◆o Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Dinamico: 524288
Tama◆o Vectores:524288 (4 B)
Tiempo:0.002854149       / Tama◆o Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104
857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamaño Dinamico: 1048576
Tama◆o Vectores:1048576 (4 B)
Tiempo:0.005664817       / Tama◆o Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=2
09715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamaño Dinamico: 2097152
Tama◆o Vectores:2097152 (4 B)
Tiempo:0.010291963       / Tama◆o Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=4
19430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamaño Dinamico: 4194304
Tama◆o Vectores:4194304 (4 B)
Tiempo:0.019227600       / Tama◆o Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=8
38860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamaño Dinamico: 8388608
Tama◆o Vectores:8388608 (4 B)
Tiempo:0.035356028       / Tama◆o Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1
677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamaño Dinamico: 16777216
Tama◆o Vectores:16777216 (4 B)
Tiempo:0.064702045       / 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) /
Tamaño Dinamico: 33554432
Tama◆o Vectores:33554432 (4 B)
Tiempo:0.127428518       / 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) /
Tamaño Dinamico: 67108864
Tama◆o Vectores:67108864 (4 B)
Tiempo:0.252521072       / 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:
El máximo valor que puede almacenar N es el valor de 1 componente -1 (ya que empieza en 0), es decir, 1 componente equivale a 4 B = 2^32 – 1 = 4294967295

(b) Modificar el código fuente C (en el PC) para que el límite de los vectores cuando se declaran como variables globales sea igual al máximo número que se puede almacenar en la variable N y generar el ejecutable. ¿Qué ocurre? ¿A qué es debido? (Incorporar volcados de pantalla que muestren lo que ocurre)
RESPUESTA:
No llega a compilar debido a que el tamaño es muy grande para ser ajustado, es necesario truncarlo.

```
#define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
      // globales (su longitud no estar limitada por el ...
      // tama o de la pila del programa)
//#define VECTOR_DYNAMIC  // descomentar para que los vectores sean variables ...
      // din micas (memoria reutilizable durante la ejecuci n)

#ifdef VECTOR_GLOBAL
#define MAX 4294967295  //=2^32 - 1
```

```
[DavidGómezHernández david@david-HP-Pavilion-Notebook:/home/david/Escritorio/Github/AC/Practica 0/ejer7]
02/26/20 13:14:51
$ gcc -O2 SumaVectoresC.c -o SumaVectoresGlobal -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:34: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tama o Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                  ~^
                                  %lu
/tmp/ccZe8uCQ.o: En la función `main':
SumaVectoresC.c:(.text.startup+0x76): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo
`v2' definido en la sección COMMON en /tmp/ccZe8uCQ.o
SumaVectoresC.c:(.text.startup+0xc9): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo
`v3' definido en la sección COMMON en /tmp/ccZe8uCQ.o
```

Entrega del trabajo

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

Listado 1 . Código C que suma dos vectores

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

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

 Para ejecutar use: SumaVectoresC longitud
*/

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

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

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

   clock_gettime(CLOCK_REALTIME,&cgt2);
```

```
   ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
          (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

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


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