2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos):
Grupo de prácticas:
Fecha de entrega: 08/03/2018
Fecha evaluación en clase:

---

1. Incorpore volcados de pantalla que muestren lo que devuelve `lscpu` en atcgrid y en su PC.

CAPTURAS:

```
[antoniogamizdelgado antonio@antonio:~] 2018-03-02 viernes
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 158
Model name:            Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
Stepping:              9
CPU MHz:               2800.000
CPU max MHz:           3800,0000
CPU min MHz:           800,0000
BogoMIPS:              5616.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              6144K
NUMA node0 CPU(s):     0-7
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflus
h 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 xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 mon
itor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_dead
line_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ret
poline intel_pt rsb_ctxsw tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bm
i2 erms invpcid mpx rdseed adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts
 hwp hwp_notify hwp_act_window hwp_epp
```

```
[E2estudiante14@atcgrid ~]$ echo 'lscpu' | qsub -q ac
63194.atcgrid
[E2estudiante14@atcgrid ~]$ ls
STDIN.e63194  STDIN.o63194
[E2estudiante14@atcgrid ~]$ cat STDIN.o63194
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:               1600.155
CPU max MHz:           2401.0000
CPU min MHz:           1600.0000
BogoMIPS:              4800.14
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 b
ts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni dtes64 monitor ds_cpl vmx smx est tm2 sss
e3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 popcnt lahf_lm epb pti retpoline tpr_shadow vnmi flexpriorit
y ept vpid dtherm ida arat
```

Conteste a las siguientes preguntas:

a. ¿Cuántos cores físicos y cuántos cores lógicos tiene el pc de prácticas o su PC?

Mi pc: 4 cores físicos y 8 cores lógicos.

b. ¿Cuántos cores físicos y cuántos cores lógicos tiene un nodo de atcgrid?

Un nodo de atcgrid: 12 cores físicos y 24 cores lógicos.

2. En el Listado 1 se puede ver un código fuente C que calcula la suma de dos vectores y en el Listado 2 una versión con C++:

```
v3 = v1 + v2;  v3(i) = v1(i) + v2(i),  i=0,…N-1
```

Los códigos utilizan directivas del compilador para fijar el tipo de variable de los vectores (v1, v2 y v3). En los comentarios que hay al principio de los códigos se indica cómo hay que compilarlos. Los vectores pueden ser:

- Variables locales: descomentando en el código `#define VECTOR_LOCAL` y comentando `#define VECTOR_GLOBAL` y `#define VECTOR_DYNAMIC`
- Variables globales: descomentando `#define VECTOR_GLOBAL` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_DYNAMIC`
- Variables dinámicas: descomentando `#define VECTOR_DYNAMIC` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_GLOBAL`. Si se usan los códigos tal y como están en Listado 1 y Listado 2, sin hacer ningún cambio, los vectores (v1, v2 y v3) serán variables dinámicas.

Por tanto, se debe definir sólo una de las siguientes constantes: `VECTOR_LOCAL`, `VECTOR_GLOBAL` o `VECTOR_DYNAMIC`.

a. En los dos códigos (Listado 1 y Listado 2) se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. En el código se imprime la variable `ncgt`, ¿qué contiene esta variable? ¿qué información devuelve

exactamente la función `clock_gettime()`? ¿en qué estructura de datos devuelve `clock_gettime()` la información (indicar el tipo de estructura de datos y describir la estructura de datos)?

RESPUESTA:

- Variable 'ngct' = contiene el tiempo en segundos que ha tardado en ejecutarse el trozo de programa que calcula la suma de los vectores.

- clock_gettime() = devuelve 0, si ha tenido éxito, -1 en caso contrario.

- clock_gettime() = devuelve una estructura de tipo 'timespec' (que se le pasa como argumento por referencia), formada por dos valores numéricos: tv_sec (de tipo time_t), que almacena el tiempo en segundos, y tv_nsec (de tipo long) que almacena el tiempo en nanosegundos. También se le pasa (en los listados), la opción 'CLOCK_REALTIME', que pone el reloj a la hora actual del sistema.

b. Escribir en el cuaderno de prácticas las diferencias que hay entre el código fuente C y el código fuente C++ para la suma de vectores.

RESPUESTA: INCOMPLETO

Descripción diferencia	En C	En C++
Las librerías que se incluyen para las distintas funciones son diferentes en ambos lenguajes.	Stdlib.h stdio.h	Cstdlib.h iostream
Las funciones de salida son distintas	printf()	cout
Funciones para la reserva de memoria dinámica	malloc()	new
Funciones para liberar el espacio reservado para los vectores	free()	delete
Comprobación de la reserva de espacio correcta para los vectores.	Se comprueba si había espacio suficiente para almacenar los vectores tras la llamada a malloc() y en caso de falta de espacio se imprime un mensaje y se sale del programa.	No se realiza niguna comprobación, pues new genera una excepción en caso de falta de espacio.

3. Generar 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`). Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid o en su PC.

RESPUESTA:

```
[antoniogamizdelgado antonio@antonio:~/Desktop/P1] 2018-03-03 sábado
$ gcc -O2 listado1.c -o listado1 -lrt
[antoniogamizdelgado antonio@antonio:~/Desktop/P1] 2018-03-03 sábado
$ ./listado1 100000
Tiempo (seg.): 0.000606906       / Tamaño vectores: 100000      / v1[0]+v2[0]=v3[0](10000.000000+1000
0.000000=20000.000000) / / v1[99999]+v2[99999]=v3[99999](19999.900000+0.100000=20000.000000) /
[antoniogamizdelgado antonio@antonio:~/Desktop/P1] 2018-03-03 sábado
$
```

4. Ejecutar en atcgrid el código generado en el apartado anterior usando el `script` del Listado 3. Generar el ejecutable usando la opción de optimización –O2 tal y como se indica en el comentario que hay al principio del programa. Ejecutar el código también en su PC para los mismos tamaños. ¿Se obtiene error para alguno de los tamaños? En caso afirmativo, ¿a qué se debe este error? (Incorporar volcados de pantalla)

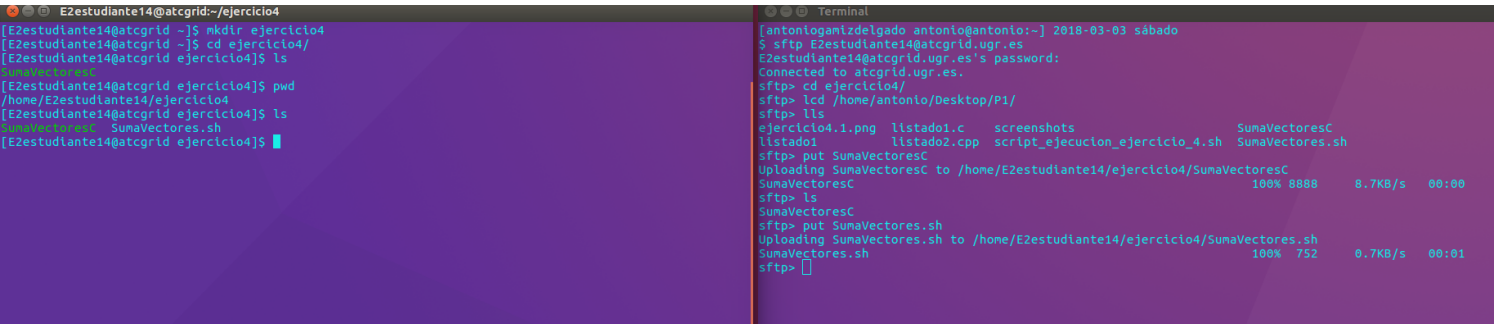
MI PC:

```
[antoniogamizdelgado antonio@antonio:~/Desktop/P1] 2018-03-03 sábado
$ gcc -O2 listado1.c -o SumaVectoresC -lrt
[antoniogamizdelgado antonio@antonio:~/Desktop/P1] 2018-03-03 sábado
$ bash script_ejecucion_ejercicio_4.sh
Tiempo (seg.): 0.000946036       / 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.): 0.000420836       / Tamaño vectores: 131072      / v1[0]+v2[0]=v3[0](13107.200000+1310
7.200000=26214.400000) / / v1[131071]+v2[131071]=v3[131071](26214.300000+0.100000=26214.400000) /
Tiempo (seg.): 0.000837809       / Tamaño vectores: 262144      / v1[0]+v2[0]=v3[0](26214.400000+2621
4.400000=52428.800000) / / v1[262143]+v2[262143]=v3[262143](52428.700000+0.100000=52428.800000) /
script_ejecucion_ejercicio_4.sh: line 6:  5689 Segmentation fault      (core dumped) ./SumaVectoresC
$N
script_ejecucion_ejercicio_4.sh: line 6:  5691 Segmentation fault      (core dumped) ./SumaVectoresC
$N
script_ejecucion_ejercicio_4.sh: line 6:  5693 Segmentation fault      (core dumped) ./SumaVectoresC
$N
script_ejecucion_ejercicio_4.sh: line 6:  5695 Segmentation fault      (core dumped) ./SumaVectoresC
$N
script_ejecucion_ejercicio_4.sh: line 6:  5697 Segmentation fault      (core dumped) ./SumaVectoresC
$N
script_ejecucion_ejercicio_4.sh: line 6:  5699 Segmentation fault      (core dumped) ./SumaVectoresC
$N
script_ejecucion_ejercicio_4.sh: line 6:  5701 Segmentation fault      (core dumped) ./SumaVectoresC
$N
script_ejecucion_ejercicio_4.sh: line 6:  5703 Segmentation fault      (core dumped) ./SumaVectoresC
$N
[antoniogamizdelgado antonio@antonio:~/Desktop/P1] 2018-03-03 sábado
$
```

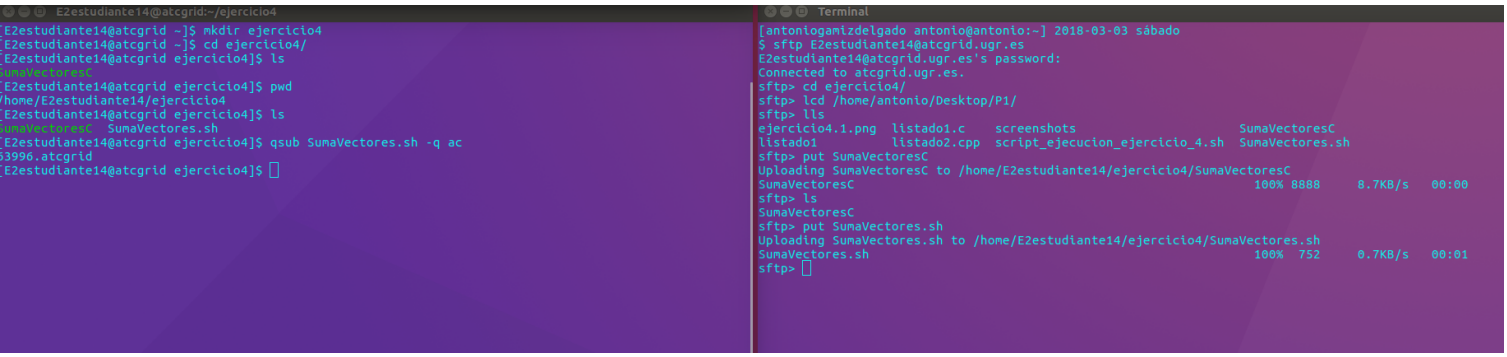
Como se ve en la foto, en las últimas 7 ejecuciones se produce una violación de segmento (Segmentation fault). Esto es debido a que N, en esas ejecuciones, es demasiado grande y el compilador no ha podido reservar toda la memoria necesaria. Como los bucles que recorren los vectores llegan hasta N y tenían un tamaño inferior a este, se ha producido el error.

ATCGRID:

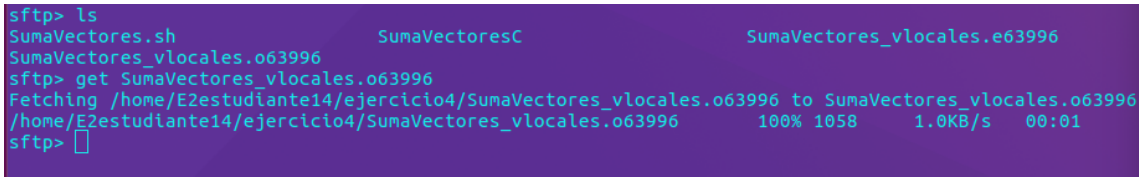
Primero cargamos el programa en un directorio (ejercicio4) creado en atcgrid.



Luego ejecutamos el script:



Ahora importamos el fichero resultante de la ejecución:



Vemos el contenido del mismo:

```
[antoniogamizdelgado antonio@antonio:~/Desktop/P1] 2018-03-03 sábado
$ cat SumaVectores_vlocales.o63996
Id. usuario del trabajo: E2estudiante14
Id. del trabajo: 63996.atcgrid
Nombre del trabajo especificando usuario: SumaVectores_vlocales
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/E2estudiante14/ejercicio4
Cola: ac
Nodos asignados al trabajo:
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
Tiempo (seg.): 0.000373142       / 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.): 0.000799968       / Tamaño vectores: 131072      / v1[0]+v2[0]=v3[0](13107.200000+1310
7.200000=26214.400000) / / v1[131071]+v2[131071]=v3[131071](26214.300000+0.100000=26214.400000) /
Tiempo (seg.): 0.001692884       / Tamaño vectores: 262144      / v1[0]+v2[0]=v3[0](26214.400000+2621
4.400000=52428.800000) / / v1[262143]+v2[262143]=v3[262143](52428.700000+0.100000=52428.800000) /
[antoniogamizdelgado antonio@antonio:~/Desktop/P1] 2018-03-03 sábado
$
```

Vemos que solo aparecen 3 ejecuciones, esto se debe a que las últimas 7 también han dado error en la ejecución en ATCGRID y por eso no se ha producido el printf correspondiente.

5. Generar los ejecutables del código fuente C para vectores globales y para dinámicos. Genere el ejecutable usando –O2. Ejecutar los dos códigos en atcgrid usando un `script` como el del Listado 3 (hay que poner en el `script` el nombre de los ficheros ejecutables generados en este ejercicio) para el mismo rango de tamaños utilizado en el ejercicio anterior. Ejecutar también los códigos en su PC. ¿Se obtiene error usando vectores globales o dinámicos? ¿A qué cree que es debido? (Incorporar volcados de pantalla)

RESPUESTA:

Primero generamos los dos ejecutables (SumaVectoresC_global y SumaVectoresC_dinamico)

```
[antoniogamizdelgado antonio@antonio:~] 2018-03-07 miércoles
$ cd ArquitecturaDeComputadores/
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores] 2018-03-07 miércoles
$ gcc -O2 ./source/listado1.c -o ./bin/SumaVectoresC_global
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores] 2018-03-07 miércoles
$ gcc -O2 ./source/listado1.c -o ./bin/SumaVectoresC_dinamico
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores] 2018-03-07 miércoles
$
```

Luego cargamos los ejecutables generados y los scripts correspondientes en ATCGRID:

```
[antoniogamizdelgado antonio@antonio:~] 2018-03-07 miércoles
$ acsftp
E2estudiante14@atcgrid.ugr.es's password:
Connected to atcgrid.ugr.es.
sftp> lcd ArquitecturaDeComputadores/
sftp> lcd bin/
sftp> put SumaVectoresC_global
Uploading SumaVectoresC_global to /home/E2estudiante14/SumaVectoresC_global
SumaVectoresC_global                                      100% 8968     8.8KB/s   00:00
sftp> put SumaVectoresC_dinamico
Uploading SumaVectoresC_dinamico to /home/E2estudiante14/SumaVectoresC_dinamico
SumaVectoresC_dinamico                                    100% 8984     8.8KB/s   00:00
sftp> lcd ../scripts/
sftp> put SumaVectores_global.sh
Uploading SumaVectores_global.sh to /home/E2estudiante14/SumaVectores_global.sh
SumaVectores_global.sh                                    100%  760     0.7KB/s   00:00
sftp> put SumaVe
SumaVectores.sh                 SumaVectores_dinamico.sh        SumaVectores_global.sh

sftp> put SumaVectores_dinamico.sh
Uploading SumaVectores_dinamico.sh to /home/E2estudiante14/SumaVectores_dinamico.sh
SumaVectores_dinamico.sh                                  100%  762     0.7KB/s   00:00
sftp> ls
SumaVectoresC_dinamico          SumaVectoresC_global            SumaVectores_dinamico.sh
SumaVectores_global.sh
sftp>
```

Mandamos ejecutar los scripts en ATCGRID:

```
[E2estudiante14@atcgrid ~]$ ls
SumaVectoresC_dinamico  SumaVectoresC_global  SumaVectores_dinamico.sh  SumaVectores_global.sh
[E2estudiante14@atcgrid ~]$ qsub SumaVectores_dinamico.sh -q ac
65509.atcgrid
[E2estudiante14@atcgrid ~]$ qsub SumaVectores_global.sh -q ac
65510.atcgrid
[E2estudiante14@atcgrid ~]$ ls
SumaVectoresC_dinamico    SumaVectores_global.sh        SumaVectores_vlocales.o65509
SumaVectoresC_global      SumaVectores_vlocales.e65509  SumaVectores_vlocales.o65510
SumaVectores_dinamico.sh  SumaVectores_vlocales.e65510
[E2estudiante14@atcgrid ~]$
```

Importamos los resultados de la ejecución en ATCGRID a nuestro ordenador:

```
sftp> ls
SumaVectoresC_dinamico          SumaVectoresC_global            SumaVectores_dinamico.sh
SumaVectores_global.sh          SumaVectores_vdinamico.e65512  SumaVectores_vdinamico.o65512
SumaVectores_vgloblal.e65511    SumaVectores_vgloblal.o65511
sftp> get SumaVectores_vdin
SumaVectores_vdinamico.e65512   SumaVectores_vdinamico.o65512
sftp> get SumaVectores_vdinamico.o65512
Fetching /home/E2estudiante14/SumaVectores_vdinamico.o65512 to SumaVectores_vdinamico.o65512
/home/E2estudiante14/SumaVectores_vdinamico.o65512               100% 2667     2.6KB/s   00:00
sftp> get SumaVectores_vglobla
SumaVectores_vgloblal.e65511    SumaVectores_vgloblal.o65511
sftp> get SumaVectores_vgloblal.o65511
Fetching /home/E2estudiante14/SumaVectores_vgloblal.o65511 to SumaVectores_vgloblal.o65511
/home/E2estudiante14/SumaVectores_vgloblal.o65511                100% 2663     2.6KB/s   00:00
sftp>
```

Ahora visualizamos los resultados:

El dinámico:

```
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores] 2018-03-07 miércoles
$ cd output/
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores/output] 2018-03-07 miércoles
$ ls
SumaVectores_vdinamico.o65512  SumaVectores_vgloblal.o65511  SumaVectores_vlocales.o63996
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores/output] 2018-03-07 miércoles
$ cat SumaVectores_vdinamico.o65512
Id. usuario del trabajo: E2estudiante14
Id. del trabajo: 65512.atcgrid
Nombre del trabajo especificando usuario: SumaVectores_vdinamico
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/E2estudiante14
Cola: ac
Nodos asignados al trabajo:
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
Tiempo (seg.): 0.000440918       / 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.): 0.000528061       / Tamaño vectores: 131072      / v1[0]+v2[0]=v3[0](13107.200000+13107.200000=2621
4.400000) / / v1[131071]+v2[131071]=v3[131071](26214.300000+0.100000=26214.400000) /
Tiempo (seg.): 0.001627147       / Tamaño vectores: 262144      / v1[0]+v2[0]=v3[0](26214.400000+26214.400000=5242
8.800000) / / v1[262143]+v2[262143]=v3[262143](52428.700000+0.100000=52428.800000) /
Tiempo (seg.): 0.003175278       / Tamaño vectores: 524288      / v1[0]+v2[0]=v3[0](52428.800000+52428.800000=1048
57.600000) / / v1[524287]+v2[524287]=v3[524287](104857.500000+0.100000=104857.600000) /
Tiempo (seg.): 0.005988946       / Tamaño vectores: 1048576     / v1[0]+v2[0]=v3[0](104857.600000+104857.600000=20
9715.200000) / / v1[1048575]+v2[1048575]=v3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo (seg.): 0.011876675       / Tamaño vectores: 2097152     / v1[0]+v2[0]=v3[0](209715.200000+209715.200000=41
9430.400000) / / v1[2097151]+v2[2097151]=v3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo (seg.): 0.023566153       / Tamaño vectores: 4194304     / v1[0]+v2[0]=v3[0](419430.400000+419430.400000=83
8860.800000) / / v1[4194303]+v2[4194303]=v3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo (seg.): 0.043315765       / Tamaño vectores: 8388608     / v1[0]+v2[0]=v3[0](838860.800000+838860.800000=16
77721.600000) / / v1[8388607]+v2[8388607]=v3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo (seg.): 0.088589395       / 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.): 0.174026541       / 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.): 0.348509721       / 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) /
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores/output] 2018-03-07 miércoles
$
```

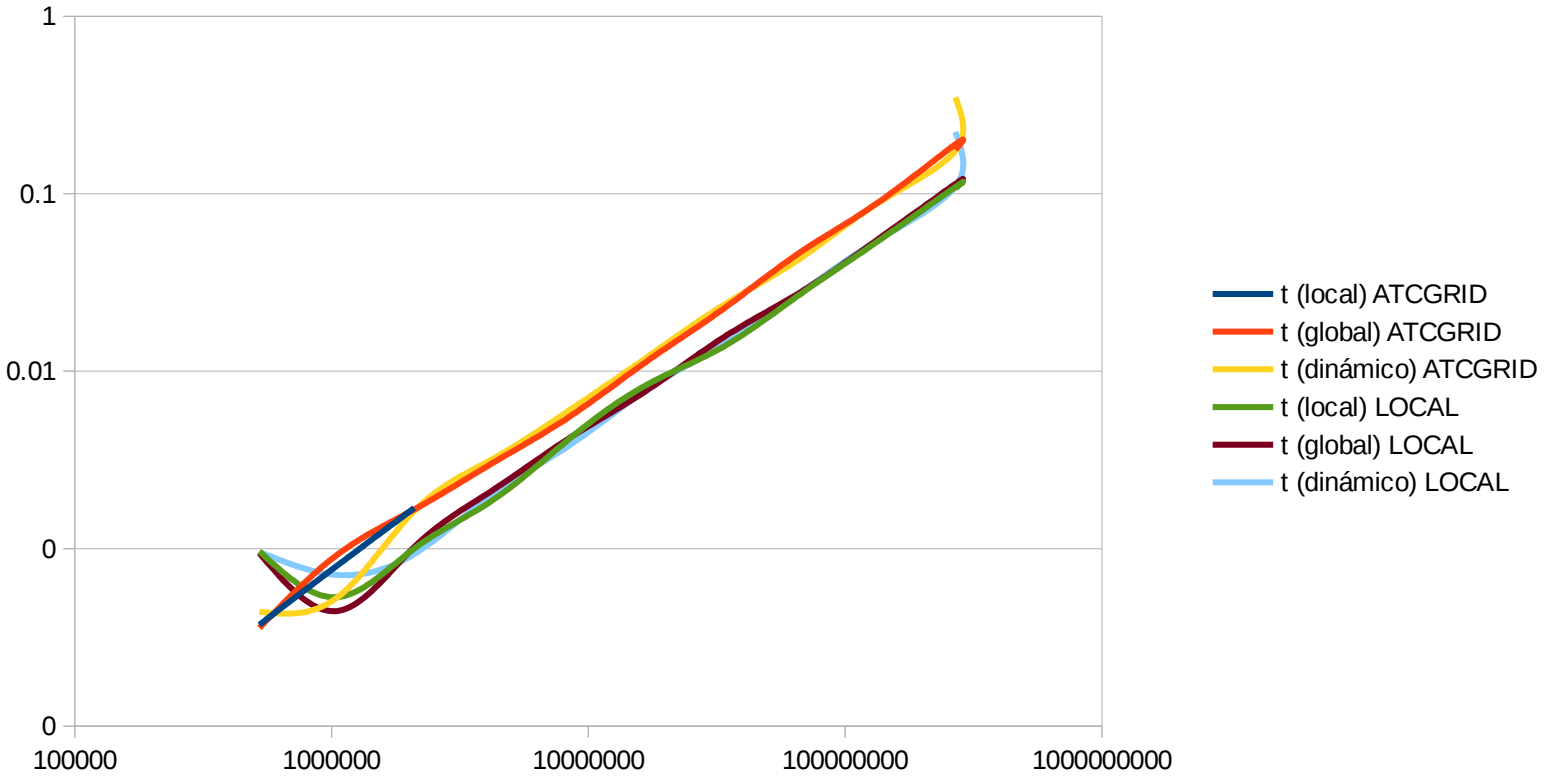
El global:

```
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores/output] 2018-03-07 miércoles
$ ls
SumaVectores_vdinamico.o65512  SumaVectores_vgloblal.o65511  SumaVectores_vlocales.o63996
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores/output] 2018-03-07 miércoles
$ cat SumaVectores_vgloblal.o65511
Id. usuario del trabajo: E2estudiante14
Id. del trabajo: 65511.atcgrid
Nombre del trabajo especificando usuario: SumaVectores_vgloblal
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/E2estudiante14
Cola: ac
Nodos asignados al trabajo:
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
Tiempo (seg.): 0.000357897       / 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.): 0.000921198       / Tamaño vectores: 131072      / v1[0]+v2[0]=v3[0](13107.200000+13107.200000=2621
4.400000) / / v1[131071]+v2[131071]=v3[131071](26214.300000+0.100000=26214.400000) /
Tiempo (seg.): 0.001662182       / Tamaño vectores: 262144      / v1[0]+v2[0]=v3[0](26214.400000+26214.400000=5242
8.800000) / / v1[262143]+v2[262143]=v3[262143](52428.700000+0.100000=52428.800000) /
Tiempo (seg.): 0.003008925       / Tamaño vectores: 524288      / v1[0]+v2[0]=v3[0](52428.800000+52428.800000=1048
57.600000) / / v1[524287]+v2[524287]=v3[524287](104857.500000+0.100000=104857.600000) /
Tiempo (seg.): 0.005500936       / Tamaño vectores: 1048576     / v1[0]+v2[0]=v3[0](104857.600000+104857.600000=20
9715.200000) / / v1[1048575]+v2[1048575]=v3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo (seg.): 0.011281459       / Tamaño vectores: 2097152     / v1[0]+v2[0]=v3[0](209715.200000+209715.200000=41
9430.400000) / / v1[2097151]+v2[2097151]=v3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo (seg.): 0.022449666       / Tamaño vectores: 4194304     / v1[0]+v2[0]=v3[0](419430.400000+419430.400000=83
8860.800000) / / v1[4194303]+v2[4194303]=v3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo (seg.): 0.046962720       / Tamaño vectores: 8388608     / v1[0]+v2[0]=v3[0](838860.800000+838860.800000=16
77721.600000) / / v1[8388607]+v2[8388607]=v3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo (seg.): 0.089166323       / 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.): 0.189618154       / 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.): 0.176800994       / 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) /
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores/output] 2018-03-07 miércoles
$
```

6. Rellenar una tabla como la Tabla 1 para atcgrid y otra para su PC con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. Ayudándose de una hoja de cálculo represente 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 (los valores de la segunda columna de la tabla, que están en escala logarítmica, deben estar en el eje x). Utilice escala logarítmica en el eje de ordenadas (eje y). ¿Hay diferencias en los tiempos de ejecución?

ATCGRID				
N	**Size en Bytes**	**t (local)**	**t (global)**	**t (dinámico)**
65536	524288	0.000373142	0.000357897	0.000440918
131072	1048576	0.000799968	0.000921198	0.000528061
262144	2097152	0.001692884	0.001662182	0.001627147
524288	4194304	0.003008925	0.003008925	0.003175278
1048576	8388608	0.005500936	0.005500936	0.005988946
2097152	16777216	0.011281459	0.011281459	0.011876675
4194304	33554432	0.022449666	0.022449666	0.023566153
8388608	67108864	0.04696272	0.04696272	0.043315765
16777216	134217728	0.089166323	0.089166323	0.088589395
33554432	268435456	0.189618154	0.189618154	0.174026541
33554432	268435456	0.176800994	0.176800994	0.348509721

PC LOCAL				
N	**Size en Bytes**	**t (local)**	**t (global)**	**t (dinámico)**
65536	524288	0.000965626	0.00094333	0.000958978
131072	1048576	0.000533388	0.000444901	0.000712079
262144	2097152	0.000999822	0.001020807	0.00093145
524288	4194304	0.0018551	0.002112401	0.002016859
1048576	8388608	0.004129341	0.004175946	0.003798096
2097152	16777216	0.008346298	0.007750005	0.007815713
4194304	33554432	0.01373855	0.015495002	0.014137834
8388608	67108864	0.027272093	0.02788758	0.027752619
16777216	134217728	0.054001583	0.055365194	0.054815492
33554432	268435456	0.109050151	0.113419097	0.109589479
33554432	268435456	0.107218017	0.11155371	0.222844184



7. Modificar el código fuente C 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 (MAX=2^32-1). Generar el ejecutable usando variables globales. ¿Qué ocurre? ¿A qué es debido? Razone además por qué el máximo número que se puede almacenar en N es 2^{32}-1.

RESPUESTA:

El máximo valor que puede almacenar un unsigned int, que es el tipo de dato de N, es 2^{32}-1, pues un unsigned int ocupa 4B=32 bits, es decir, puede representar (2^nºbits)-1=2^32-1 valores. Por tanto, el último tamaño del vector que es 2^16 también se suma. Pero al cambiar el valor que queremos reservar para variables globales en tiempo de compilación da error al compilar, pues se superan los 2GB=2^30B máximos que el compilador puede reservar por defecto para variables locales.

listado1_modificado.c

```c
/* listado1.c
Suma de dos vectores: v3= v1 + v2
Para compilar usar (-lrt: real time library):
gcc -O2 listado1.c -o listado1 -lrt
gcc -O2 -S listado1.c -lrt    (para generar el códi
Para ejecutar:
listado1 longitud
*/

#include <stdlib.h>   //biblioteca para funciones a
#include <stdio.h>    //biblioteca donde se encuent
#include <time.h>     //biblioteca donde se encuent

// #define PRINTF_ALL   //comentar para quitar el p
/* Solo puede estar definida una de las tres consta
de los tres defines siguientes puede estar descomer
//#define VECTOR_LOCAL  //descomentar para que los
#define VECTOR_GLOBAL   //descomentar para que los
//#define VECTOR_DYNAMIC    //para que los vectores

#ifdef VECTOR_GLOBAL
#define MAX 4294967295 //2^32 - 1
double v1[MAX], v2[MAX], v3[MAX];
#endif
```

```
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores] 2018-03-07 miércoles
$ pwd
/home/antonio/ArquitecturaDeComputadores
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores] 2018-03-07 miércoles
$ gcc ./source/listado1_modificado.c -o ./bin/listado_modificado
/tmp/ccMq2IqK.o: In function `main':
listado1_modificado.c:(.text+0x114): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMM
ON section in /tmp/ccMq2IqK.o
listado1_modificado.c:(.text+0x15a): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMM
ON section in /tmp/ccMq2IqK.o
listado1_modificado.c:(.text+0x16c): relocation truncated to fit: R_X86_64_32S against symbol `v3' defined in COMM
ON section in /tmp/ccMq2IqK.o
listado1_modificado.c:(.text+0x1dd): relocation truncated to fit: R_X86_64_32S against symbol `v3' defined in COMM
ON section in /tmp/ccMq2IqK.o
listado1_modificado.c:(.text+0x1ee): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMM
ON section in /tmp/ccMq2IqK.o
listado1_modificado.c:(.text+0x219): relocation truncated to fit: R_X86_64_PC32 against symbol `v3' defined in COM
MON section in /tmp/ccMq2IqK.o
listado1_modificado.c:(.text+0x221): relocation truncated to fit: R_X86_64_PC32 against symbol `v2' defined in COM
MON section in /tmp/ccMq2IqK.o
collect2: error: ld returned 1 exit status
[antoniogamizdelgado antonio@antonio:~/ArquitecturaDeComputadores] 2018-03-07 miércoles
$
```

Listado 1 . Código C que suma dos vectores

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

 Para compilar usar (-lrt: real time library):
        gcc  -O2 SumaVectores.c -o SumaVectores –lrt
 gcc  -O2 –S SumaVectores.c –lrt   //para generar el código ensamblador

 Para ejecutar use: SumaVectoresC longitud
*/

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

//#define PRINTF_ALL      // comentar para quitar el printf ...
                          // que imprime todos los componentes
//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
   #ifdef PRINTF_ALL
   printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\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]);
   #endif

   #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;
}
```

Listado 2 . Código C++ que suma dos vectores

```
/* SumaVectoresCpp.cpp
 Suma de dos vectores: v3 = v1 + v2

 Para compilar usar (-lrt: real time library):
                g++  -O2 SumaVectoresCpp.cpp -o SumaVectoresCpp -lrt

 Para ejecutar use: SumaVectoresCpp longitud
*/

#include <cstdlib>   // biblioteca con atoi()
#include <iostream> // biblioteca donde se encuentra la función cout
using namespace std;
#include <time.h>       // biblioteca donde se encuentra la función clock_gettime()

//#define COUT_ALL // comentar para quitar el cout ...
                   // que imprime todos los componentes
//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){

   struct timespec cgt1,cgt2;  //para tiempo de ejecución

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

   unsigned int N = atoi(argv[1]);
   #ifdef VECTOR_LOCAL
   double v1[N], v2[N], v3[N];
   #endif
   #ifdef VECTOR_GLOBAL
   if (N>MAX) N=MAX;
   #endif
   #ifdef VECTOR_DYNAMIC
   double *v1, *v2, *v3;
   v1 = new double [N];    //si no hay espacio suficiente new genera una excepción
   v2 = new double [N];
   v3 = new double [N];
   #endif

   //Inicializar vectores
   for(int 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(int i=0; i<N; i++)
      v3[i] = v1[i] + v2[i];
    clock_gettime(CLOCK_REALTIME,&cgt2);
   double 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
   #ifdef COUT_ALL
   cout << "Tiempo(seg.):" << ncgt << "\t/ Tamaño Vectores:" << N << endl;
   for(int i=0; i<N; i++)
         cout << "/ V1[" << i << "]+V2[" << i << "]=V3" << i << "](" << v1[i] << "+"
<< v2[i] << "="
         << v3[i] << ") /\t" << endl;
   cout <<"\n"<< endl;
   #else
         cout << "Tiempo(seg.):" << ncgt << "\t/ Tamaño Vectores:" << N << "\t/
V1[0]+V2[0]=V3[0]("
         << v1[0] << "+" << v2[0] << "=" << v3[0] << ") / / V1[" << N-1 << "]+V2["
<< N-1 << "]=V3["
         << N-1 << "](" << v1[N-1] << "+" << v2[N-1] << "=" << v3[N-1] << ")/\n" <<
endl;
   #endif

   #ifdef VECTOR_DYNAMIC
   delete [] v1; // libera el espacio reservado para v1
   delete [] v2; // libera el espacio reservado para v2
   delete [] v3; // libera el espacio reservado para v3
   #endif
   return 0;
}
```

Listado 3 . Script para la suma de vectores (SumaVectores.sh). Se supone en el script que el fichero a ejecutar se llama SumaVectorC y que se encuentra en el directorio en el que se ha ejecutado qsub.

```
#!/bin/bash
#Se asigna al trabajo el nombre SumaVectoresC_vlocales
#PBS -N SumaVectoresC_vlocales
#Se asigna al trabajo la cola ac
#PBS -q ac
#Se imprime información del trabajo usando variables de entorno de PBS
echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
echo "Id. del trabajo: $PBS_JOBID"
echo "Nombre  del trabajo especificado por usuario: $PBS_JOBNAME"
echo "Nodo que ejecuta qsub: $PBS_O_HOST"
echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
echo "Cola: $PBS_QUEUE"
echo "Nodos asignados al trabajo:"
cat $PBS_NODEFILE
#Se ejecuta SumaVectorC, que está en el directorio en el que se ha ejecutado qsub,
#para N potencia de 2 desde 2^16 a 2^26
for ((N=65536;N<67108865;N=N*2))
do
```

```
    $PBS_O_WORKDIR/SumaVectoresC $N
done
```