

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): Javier Gómez Luzón

Grupo de prácticas: C1

Fecha de entrega:

Fecha evaluación en clase:

[RECORDATORIO, quitar todo este texto en rojo del cuaderno definitivo—

1. COMENTARIOS

1) Este cuaderno de prácticas se utilizará para asignarle una puntuación durante la evaluación continua de prácticas y también lo utilizará como material de estudio y repaso para preparar el examen de prácticas escrito. Luego redáctelo con cuidado, y sea ordenado y claro.

2) No use máquinas virtuales. Se piden obtener resultados en atcgrid y en su PC (PC del aula o PC personal).

3) Debe modificar el prompt en los computadores que utilice en prácticas para que aparezca su nombre y apellidos, su usuario (\u), el computador (\h), el directorio de trabajo del bloque práctico (\w), la fecha (\D) completa (%F) y el día (%A) . Para modificar el prompt utilice lo siguiente (si es necesario, use export delante):

PS1="[NombreApellidos \u@\h:\w] \D{%F %A}\n\$"

donde NombreApellidos es su nombre seguido de sus apellidos, por ejemplo: Juan Ortuño Vilaríño

2. NORMAS SOBRE EL USO DE LA PLANTILLA

1) Usar **interlineado SENCILLO**.

2) Respetar los tipos de letra y tamaños indicados:

- Calibri-11 o Liberation Serif-11 para el texto

- **Courier New-10 o Liberation Mono-10 para nombres de fichero, comandos, variables de entorno, etc., cuando se usan en el texto.**

3) Insertar las capturas de pantalla donde se pidan y donde se considere oportuno. En particular, los listados de código se deben insertar como capturas de pantalla. En todas las capturas de pantalla, incluidas las de los listados de código, **debe aparecer el directorio y usuario**. El tamaño de letra en las capturas debe ser similar al tamaño que se está usando en el texto.

Recuerde que debe adjuntar al zip de entrega, el pdf de este fichero, todos los ficheros con código fuente implementados/utilizados y el resto de ficheros que haya implementado/utilizado (scripts, hojas de cálculo, etc.)]

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

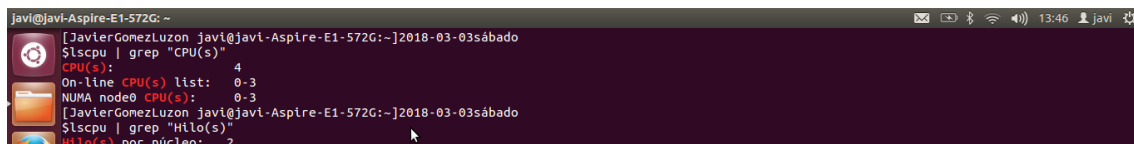
CAPTURAS:

Conteste a las siguientes preguntas:

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

RESPUESTA: Mi PC tiene 2 cores físicos y 4 cores lógicos. Lo he averiguado con `lscpu | grep "CPU(s)"` donde nos sale 4 que son los cores lógicos que hay. Si queremos saber los cores físicos ejecutamos `lscpu | grep "Hilo(s)"`. Vemos que hay dos. Por lo tanto, el número de cores físicos es cores lógicos/2.

Imagen 1: con `lscpu | grep "CPU(s)"` y `lscpu | grep "Hilo(s)"` en local.

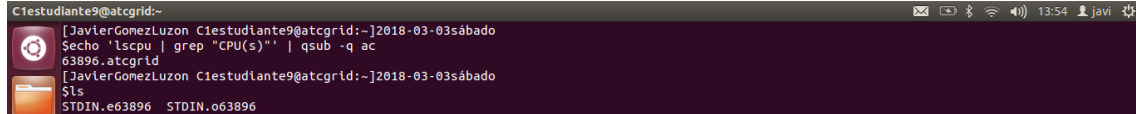


```
Javi@Javi-Aspire-E1-572G: ~  
[JavierGomezLuzon javi@Javi-Aspire-E1-572G:~]2018-03-03sábado  
$lscpu | grep "CPU(s)"  
CPU(s): 4  
On-line CPU(s) list: 0-3  
NUMA node0 CPU(s): 0-3  
[JavierGomezLuzon javi@Javi-Aspire-E1-572G:~]2018-03-03sábado  
$lscpu | grep "Hilo(s)"  
Hilo(s) por núcleo: 2
```

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

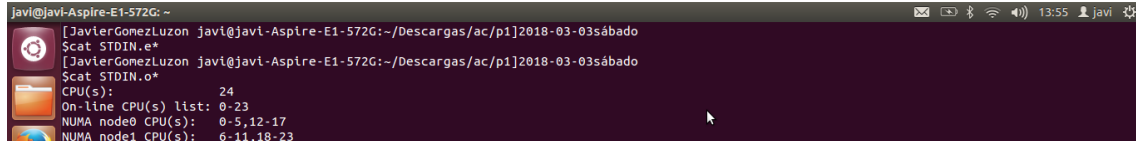
RESPUESTA: Un nodo de atcgrid tiene 12 cores físicos y 24 cores lógicos. Lo obtenemos con `echo `lscpu | grep "CPU(s)" ` | qsub -q ac`. Para los cores físicos hacemos `echo `lscpu | grep "Thread(s)" ` | qsub -q ac` vemos que hay dos. Por lo tanto, el número de cores físicos es cores lógicos/2.

Imagen 2: `echo `lscpu | grep "CPU(s)" ` | qsub -q ac` en atcgrid.



```
C1estudiante9@atcgrid:~
[JavierGomezLuzon C1estudiante9@atcgrid:~]2018-03-03sábado
$ echo `lscpu | grep "CPU(s)" ` | qsub -q ac
63896.atcgrid
[JavierGomezLuzon C1estudiante9@atcgrid:~]2018-03-03sábado
$ ls
STDIN.e63896 STDIN.o63896
```

Imagen 3: Mostrar salida y error de la ejecución de la imagen 2.



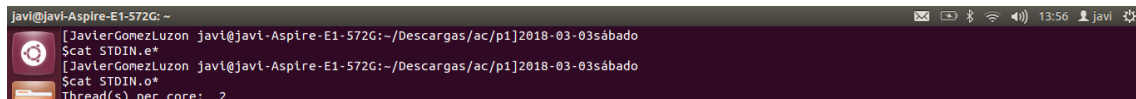
```
javi@javi-Aspire-E1-572G:~
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-03-03sábado
$ cat STDIN.e*
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-03-03sábado
$ cat STDIN.o*
CPU(s): 24
On-line CPU(s) list: 0-23
NUMA node0 CPU(s): 0-5,12-17
NUMA node1 CPU(s): 6-11,18-23
```

Imagen 4: `echo `lscpu | grep "Thread(s)" ` | qsub -q ac` en atcgrid.



```
C1estudiante9@atcgrid:~
[JavierGomezLuzon C1estudiante9@atcgrid:~]2018-03-03sábado
$ echo `lscpu | grep "Thread(s)" ` | qsub -q ac
63897.atcgrid
[JavierGomezLuzon C1estudiante9@atcgrid:~]2018-03-03sábado
$ ls
STDIN.e63897 STDIN.o63897
```

Imagen 5: Mostrar salida y error de la ejecución de la imagen 4.



```
javi@javi-Aspire-E1-572G:~
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-03-03sábado
$ cat STDIN.e*
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-03-03sábado
$ cat STDIN.o*
Thread(s) per core: 2
```

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++:

$v3 = v1 + v2$; $v3(i) = v1(i) + v2(i)$, $i=0, \dots, N-1$

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: La variable `ncgt = (to en s hasta terminar la suma de vectores – to en s hasta terminar de inicializar los vectores) + ((to en n_s hasta terminar la suma de vectores – to en n_s hasta terminar de inicializar los vectores) / (1.e+9))`.

Por lo tanto `ncgt` contiene el tiempo en segundos que pasa desde que se terminan de inicializar los vectores hasta que termina de hacer la suma de los mismos. Es decir el t^o en segundos que tarda en realizar la suma de vectores.

`clock_gettime()` obtiene el valor en tiempo (segundos o nanosegundos) desde el inicio del programa hasta el punto en que se llama a la función.

```
Struct timespec{
    time_t      tv_sec;
    long        tv_nsec;
}
```

- 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:

Descripción diferencia	En C	En C++
Imprimir datos	<code>Printf()</code>	<code>Cout<<</code>
Reservar memoria para los vectores	<code>Malloc()</code>	<code>New double[]</code>
Liberar espacio de los vectores	<code>Free()</code>	<code>Delete[]</code>
Librerías	<code><stdlib.h></code> <code><stdio.h></code>	<code><cstdlib></code> <code><iostream></code>

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:

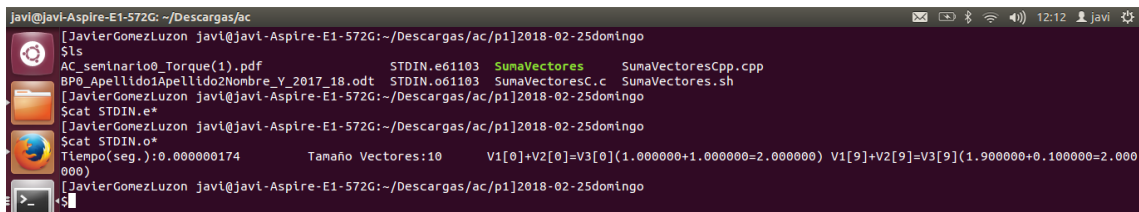
Imagen 6: Compilamos y ejecutamos en local.

```
javi@javi-Aspire-E1-572G: ~/Descargas/ac
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-25domingo
$gcc -O2 SumaVectoresC.c -o SumaVectores -lrt
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-25domingo
$ls
AC_seminario0_Torque(1).pdf  BP0_Apellido1Apellido2Nombre_Y_2017_18.odt  SumaVectores  SumaVectoresC.c  SumaVectoresCpp.cpp  SumaVectores.sh
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-25domingo
$./SumaVectores 10
Tiempo(seg.):0.000000149      Tamaño Vectores:10      V1[0]+V2[0]=V3[0](1.000000+1.000000=2.000000) V1[9]+V2[9]=V3[9](1.900000+0.100000=2.000000)
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-25domingo
$
```

Imagen 7: Mandamos a los nodos de atcgrid ejecutar el ejecutable.

```
Ciestudiante9@atcgrid:~
[javierGomezLuzon Ciestudiante9@atcgrid:~]2018-02-25domingo
$ls
SumaVectores
[javierGomezLuzon Ciestudiante9@atcgrid:~]2018-02-25domingo
$echo './SumaVectores 10' | qsub -q ac
61103.atcgrid
[javierGomezLuzon Ciestudiante9@atcgrid:~]2018-02-25domingo
$ls
STDIN.e61103  STDIN.o61103  SumaVectores
```

Imagen 8: Mostramos el fichero de salida y el fichero de error en nuestro PC.



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)

RESPUESTA:

Imagen 9: Ejecución en local con nuestro script.



Imagen 10: Ejecución del ejecutable en atcgrid.

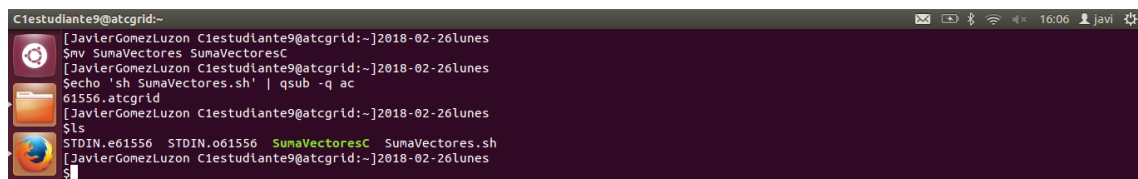


Imagen 11: Mostrar la salida de la ejecución de atcgrid en nuestro PC.

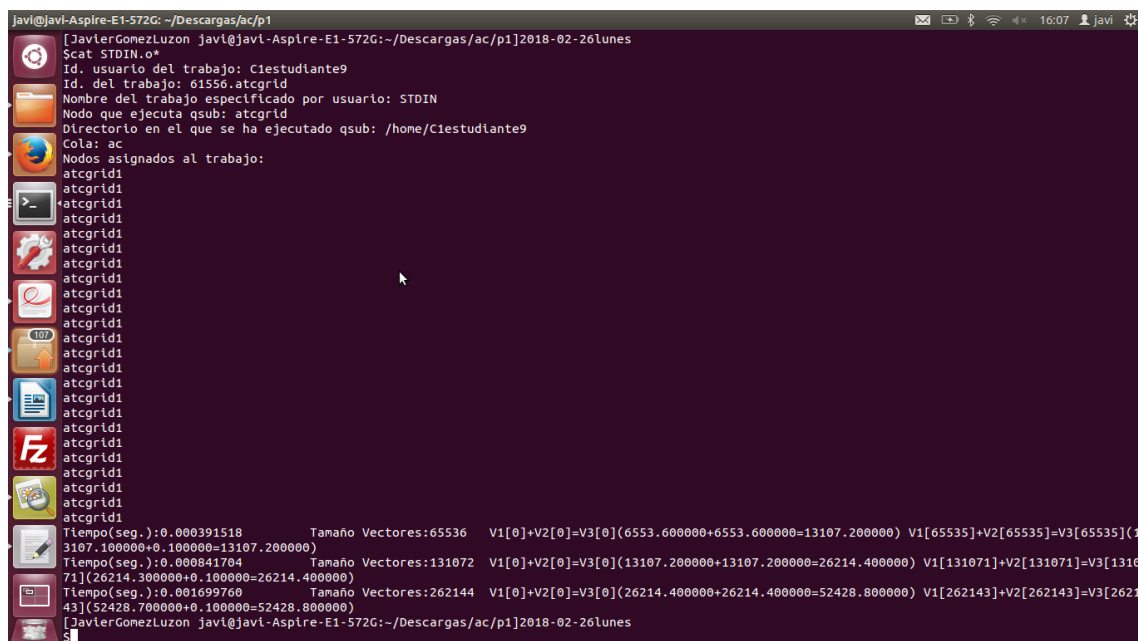


Imagen 12: Mostrar el fichero de error de la ejecución de atcgrid en nuestro PC.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac/p1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
$ cat STDIN.e*
SumaVectores.sh: line 19: 13967 Segmentation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
SumaVectores.sh: line 19: 13970 Segmentation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
SumaVectores.sh: line 19: 13973 Segmentation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
SumaVectores.sh: line 19: 13978 Segmentation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
SumaVectores.sh: line 19: 13987 Segmentation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
SumaVectores.sh: line 19: 13993 Segmentation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
SumaVectores.sh: line 19: 14004 Segmentation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
SumaVectores.sh: line 19: 14021 Segmentation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
$

```

Nos da error para los vectores con componentes entre 524288 elementos y 67108864 elementos. El error se produce porque se supera el tamaño de la pila y por eso se produce el error de segmentación.

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:

Imagen 13: Ejecución en local con vectores dinámicos.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac/p1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
$ ./script SumaVectoresD
Tamaño Vectores:65536 V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535]+V2[65535]=V3[65535](1
3107.100000+0.100000=13107.200000)
Tamaño Vectores:131072 V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131071]+V2[131071]=V3[1310
71](26214.300000+0.100000=26214.400000)
Tamaño Vectores:262144 V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262143]+V2[262143]=V3[2621
43](52428.700000+0.100000=52428.800000)
Tamaño Vectores:524288 V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) V1[524287]+V2[524287]=V3[524
287](104857.500000+0.100000=104857.600000)
Tamaño Vectores:1048576 V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) V1[1048575]+V2[104
8575]=V3[1048575](209715.100000+0.100000=209715.200000)
Tamaño Vectores:2097152 V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) V1[2097151]+V2[209
7151]=V3[2097151](419430.300000+0.100000=419430.400000)
Tamaño Vectores:4194304 V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) V1[4194303]+V2[419
4303]=V3[4194303](838860.700000+0.100000=838860.800000)
Tamaño Vectores:8388608 V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) V1[8388607]+V2[83
88607]=V3[8388607](1677721.500000+0.100000=1677721.600000)
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)
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)
Tamaño Vectores:67108864 V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) V1[67108863]+V
67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000)
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
$

```

Imagen 14: Ejecución en atcgrid con vectores dinámicos.

```

C1estudiante9@atcgrid:~
[javierGomezLuzon C1estudiante9@atcgrid:~]2018-02-26lunes
$ ls
SumaVectoresD SumaVectores.sh
[javierGomezLuzon C1estudiante9@atcgrid:~]2018-02-26lunes
$ mv SumaVectoresD SumaVectoresC
[javierGomezLuzon C1estudiante9@atcgrid:~]2018-02-26lunes
$ ls
SumaVectoresC SumaVectores.sh
[javierGomezLuzon C1estudiante9@atcgrid:~]2018-02-26lunes
$ echo 'sh SumaVectores.sh' | qsub -q ac
61578.atcgrid
[javierGomezLuzon C1estudiante9@atcgrid:~]2018-02-26lunes
$ ls
STDIN.e61578 STDIN.o61578 SumaVectoresC SumaVectores.sh
[javierGomezLuzon C1estudiante9@atcgrid:~]2018-02-26lunes
$

```

Imagen 15 y 16: Muestra del fichero de salida de la ejecución de vectores dinámicos en atcgrid.


```
javi@javi-Aspire-E1-572G: ~/Descargas/ac/p1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
Scat STDIN.*
Id. usuario del trabajo: Ciestudiante9
Id. del trabajo: 61578.atcgrid
Nombre del trabajo especificado por usuario: STDIN
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/Ciestudiante9
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
Tiempo(seg.):0.000379744          Tamaño Vectores:65536      V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535]+V2[65535]=V3[65535](13107.200000+0.100000=13107.200000)
Tiempo(seg.):0.000777373          Tamaño Vectores:131072    V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000)
Tiempo(seg.):0.001697630          Tamaño Vectores:262144    V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000)
Tiempo(seg.):0.003453231          Tamaño Vectores:524288    V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000)

Tiempo(seg.):0.005826101          Tamaño Vectores:1048576   V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000)
Tiempo(seg.):0.011644526          Tamaño Vectores:2097152   V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000)
Tiempo(seg.):0.023213749          Tamaño Vectores:4194304   V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000)
Tiempo(seg.):0.046469242          Tamaño Vectores:8388608   V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000)
Tiempo(seg.):0.092903472          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.):0.188366152          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.):0.350445538          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)
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
```

Imagen 17: Muestra del fichero de error de la ejecución de vectores dinámicos en atcgrid.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac/p1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
$ls
ac_seminario0_Torque(1).pdf          STDIN.e61578      SumaVectores      SumaVectoresCG.c      SumaVectoresCpp.cpp      SumaVectoresG
BP0_Apellido1doApellido2doNombre_Y_2017_18.odt  STDIN.o61578      SumaVectoresCD.c  SumaVectoresCL.c      SumaVectoresD            SumaVectores.sh
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
$cat STDIN.*
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes

```

Con vectores dinámicos no vemos que se produzca ningún error porque la diferencia de las variables locales, las variables dinámicas se almacenan en el heap o almacenamiento libre. No nos da error porque el heap está limitado por la cantidad de memoria que el sistema operativo permita direccionar al proceso, que normalmente es mayor que la de la pila.

Imagen 18: Ejecución en local con vectores globales.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac/p1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
$ ./script SumaVectoresG
Tiempo(seg.):0.000766413          Tamaño Vectores:65536      V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535]+V2[65535]=V3[65535](1
3107.200000+6553.600000=13107.200000)
Tiempo(seg.):0.000901524          Tamaño Vectores:131072     V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131071]+V2[131071]=V3[131071]
71(26214.400000+6553.600000=26214.400000)
Tiempo(seg.):0.001241625          Tamaño Vectores:262144     V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262143]+V2[262143]=V3[262143]
43(52428.800000+6553.600000=52428.800000)
Tiempo(seg.):0.002076766          Tamaño Vectores:524288     V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) V1[524287]+V2[524287]=V3[524287]
287(104857.600000+6553.600000=104857.600000)
Tiempo(seg.):0.003628867          Tamaño Vectores:1048576     V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) V1[1048575]+V2[1048575]=V3[1048575]
8575(209715.200000+6553.600000=209715.200000)
Tiempo(seg.):0.007937303          Tamaño Vectores:2097152     V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) V1[2097151]+V2[2097151]=V3[2097151]
7151(419430.400000+6553.600000=419430.400000)
Tiempo(seg.):0.014942868          Tamaño Vectores:4194304     V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) V1[4194303]+V2[4194303]=V3[4194303]
4303(838860.800000+6553.600000=838860.800000)
Tiempo(seg.):0.026639164          Tamaño Vectores:8388608     V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) V1[8388607]+V2[8388607]=V3[8388607]
88607(1677721.600000+6553.600000=1677721.600000)
Tiempo(seg.):0.05335996           Tamaño Vectores:16777216     V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) V1[16777215]+V2[16777215]=V3[16777215]
4477215(3355443.200000+6553.600000=3355443.200000)
Tiempo(seg.):0.106591886          Tamaño Vectores:33554432     V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) V1[33554431]+V2[33554431]=V3[33554431]
33554431(6710886.400000+6553.600000=6710886.400000)
Tiempo(seg.):0.106236586          Tamaño Vectores:33554432     V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) V1[33554431]+V2[33554431]=V3[33554431]
33554431(6710886.400000+6553.600000=6710886.400000)
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p1]2018-02-26lunes
$

```

Imagen 19: Ejecución del ejecutable con vectores globales en atcgrid.

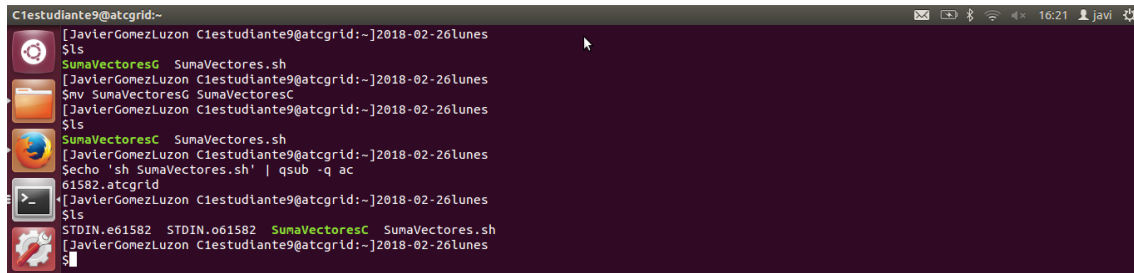


Imagen 20 y 21: Mostrar el fichero de salida de la ejecución con vectores globales en atcgrid en nuestro ordenador.

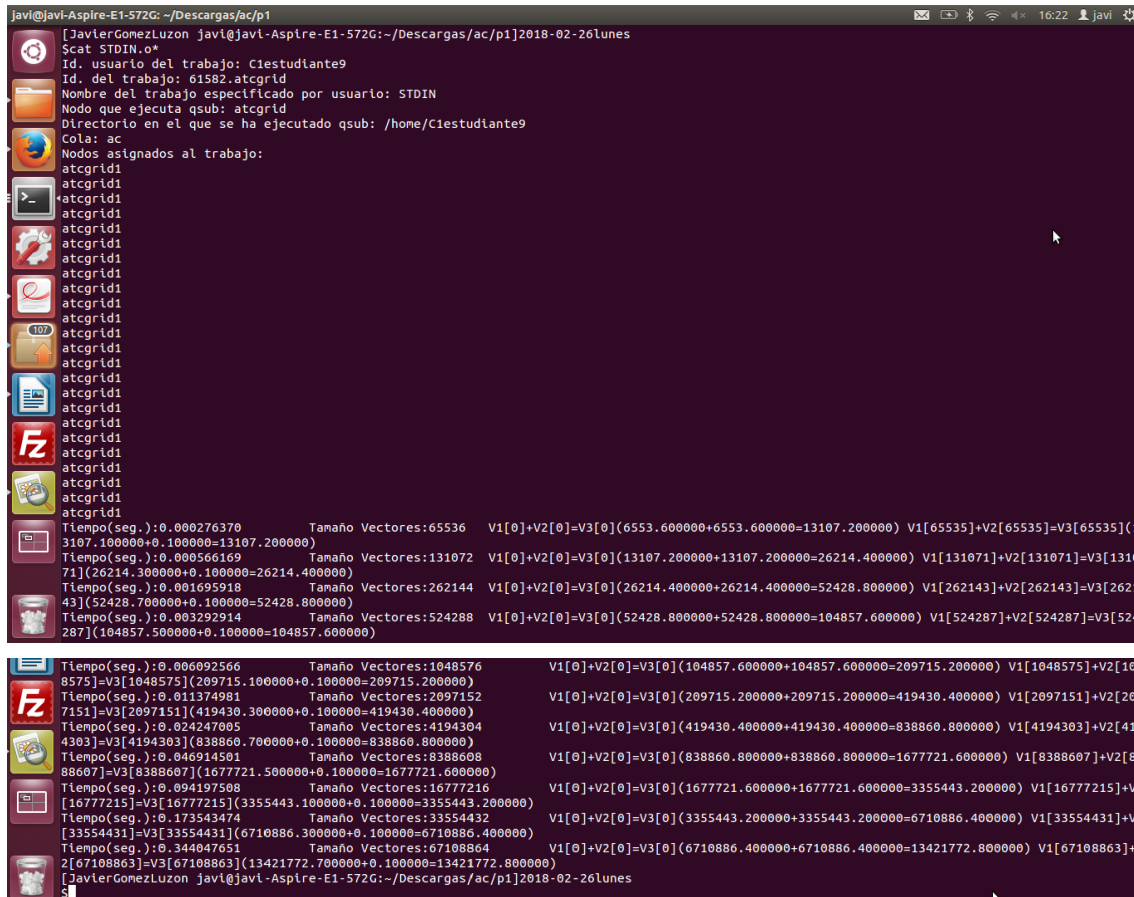
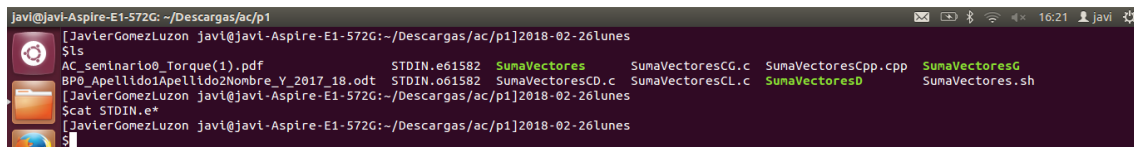


Imagen 22: Mostrar el fichero de error de la ejecución de vectores globales en atcgrid.



Con vectores globales no nos da ningún error porque las variables globales no se almacenan en pila. Se almacenan en el área de datos.

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?

RESPUESTA:

Tabla para ejecución en local.

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000585158	0.000766413	0.00058376
131072	1048576	0.001380437	0.000601524	0.000609569
262144	2097152	0.001255019	0.001241625	0.001259898
524288	4194304		0.002707676	0.001728619
1048576	8388608		0.003628867	0.003889144
2097152	16777216		0.007937303	0.006739552
4194304	33554432		0.014942868	0.13031249
8388608	67108864		0.026630164	0.025287532
16777216	134217728		0.053359991	0.050255939
33554432	268435456		0.106591886	0.099408695
67108864	536870912		0.106236586	0.200788632

Imagen 23: Gráfico de ejecución en local.

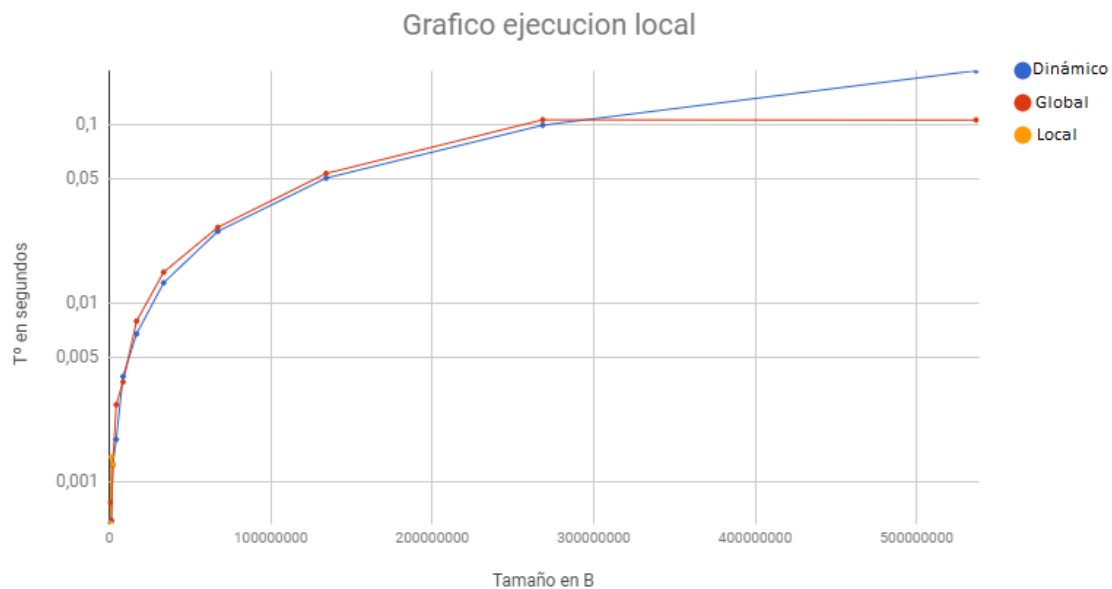
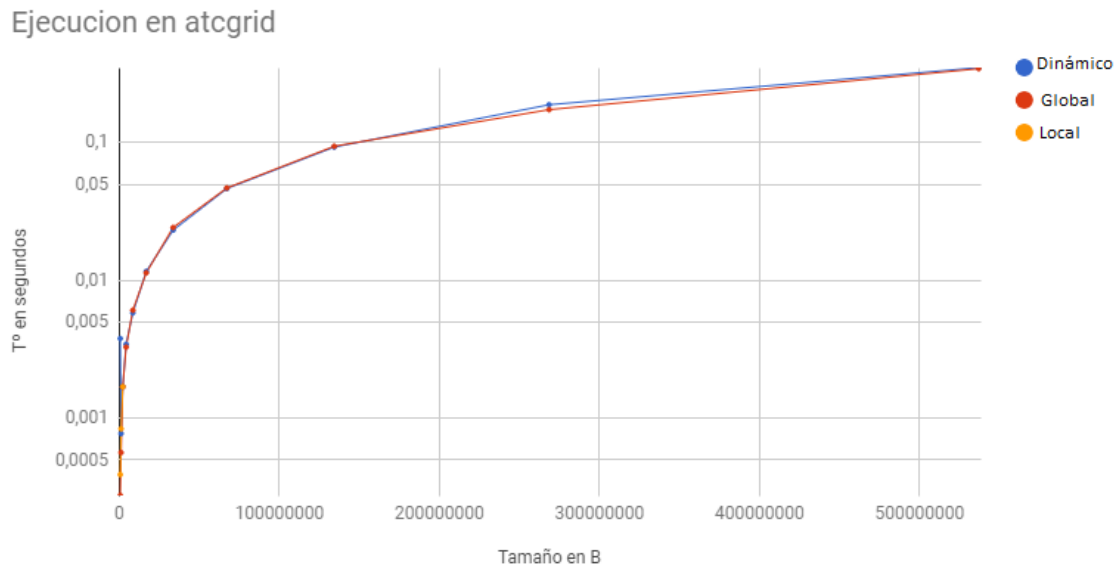


Tabla para la ejecución en atcgrid.

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. Globales	Tiempo para vect. dinámicos
65536	524288	0.000391518	0.00027637	0.00379744
131072	1048576	0.000841704	0.000566169	0.000777373
262144	2097152	0.00169976	0.001695918	0.00169763
524288	4194304		0.003292914	0.003453231
1048576	8388608		0.006092566	0.005826101
2097152	16777216		0.011374981	0.011644526
4194304	33554432		0.024247005	0.023213749
8388608	67108864		0.046914501	0.046469242
16777216	134217728		0.094197508	0.092903472
33554432	268435456		0.173543474	0.188366152
67108864	536870912		0.344047651	0.350445538

Imagen 24: Gráfico de ejecución en atcgrid.



No hay mucha diferencia entre la gráfica de mi PC y la gráfica de atcgrid. Aunque parece que en mi PC en el último valor de vectores globales empieza a mantenerse, aunque para estar seguros habría que hacer más mediciones. También podemos apreciar que sobre todo para los valores más altos de los vectores globales y dinámicos mi PC realiza las operaciones en un tiempo algo menor que atcgrid.

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 ($\text{MAX}=2^{32}-1$). 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:

Imagen 25: Compilación para vectores globales con número máximo es $2^{32}-1$.

```
javi@javi-Aspire-E1-572G: ~
$ gcc -O2 SumaVectoresCG_otro_tamaño.c -o SumaVectoresCG_otro_tamaño -lrt
/tmp/cc3e3qh8.o: In function 'main':
SumaVectoresCG_otro_tamaño.c:(.text.startup+0x66): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in /t
mp/cc3e3qh8.o
SumaVectoresCG_otro_tamaño.c:(.text.startup+0x8e): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in /t
mp/cc3e3qh8.o
SumaVectoresCG_otro_tamaño.c:(.text.startup+0x97): relocation truncated to fit: R_X86_64_32S against symbol `v3' defined in COMMON section in /t
mp/cc3e3qh8.o
SumaVectoresCG_otro_tamaño.c:(.text.startup+0xbf): relocation truncated to fit: R_X86_64_PC32 against symbol `v3' defined in COMMON section in /
tmp/cc3e3qh8.o
SumaVectoresCG_otro_tamaño.c:(.text.startup+0xc0): relocation truncated to fit: R_X86_64_PC32 against symbol `v2' defined in COMMON section in /
tmp/cc3e3qh8.o
SumaVectoresCG_otro_tamaño.c:(.text.startup+0xd7): relocation truncated to fit: R_X86_64_32S against symbol `v3' defined in COMMON section in /t
mp/cc3e3qh8.o
SumaVectoresCG_otro_tamaño.c:(.text.startup+0xf2): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in /t
mp/cc3e3qh8.o
collect2: ld devolvió el estado de salida 1
```

Nos da el siguiente error:

SumaVectoresCG_otro_tamaño.c: (-text.startup+0x66): relocation truncated to fit: R_X86_64_32S... y otros 6 errores parecidos.

Estos errores son debidos a que intentamos reservar más memoria de la que el sistema es capaz de reservar. El tamaño para los arrays para arquitecturas x86_64 esta limitada a 2GB y estamos intentando reservar un array con una memoria alrededor de los 30 GB.

N es un dato de tipo int. En el se pueden representar 2^{32} valores. (Solo positivos, del 0 al 4294967295; positivos y negativos del -2147483648 al 2147483647). Por lo tanto el mayor numero positivo que podemos representar es $2^{32}-1$ (se le resta el 1 porque contamos el 0).

Tabla 1 .

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536				
131072				
262144				
524288				
1048576				
2097152				
4194304				
8388608				
16777216				
33554432				
67108864				

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.):%11.9ft / 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,v1[i],v2[i],v3[i]);

#else
printf("Tiempo(seg.):%11.9ft / 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

```