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): Elena Merelo Molina
Grupo de prácticas: 2
Fecha de entrega: 8 de marzo a las 24:00
Fecha evaluación en clase:

---

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

CAPTURAS: En atcgrid:

```
[ElenaMereloMolina E2estudiante10@atcgrid:~] 2018-02-23 viernes
$cat STDIN.o60619
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:             2800.346
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 co
nstant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf
pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2
popcnt lahf_lm epb pti retpoline tpr_shadow vnmi flexpriority ept vpid dtherm ida arat
```

En mi portátil:

```
 2018-02-23 12:38:46  elena in ~
○ → lscpu
Arquitectura:          x86_64
modo(s) de operación de las CPUs:32-bit, 64-bit
Orden de bytes:        Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Hilo(s) de procesamiento por núcleo:2
Núcleo(s) por «socket»:2
Socket(s):             1
Modo(s) NUMA:          1
ID de fabricante:      GenuineIntel
Familia de CPU:        6
Modelo:                78
Model name:            Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
Revisión:              3
CPU MHz:               489.000
CPU max MHz:           2800,0000
CPU min MHz:           400,0000
BogoMIPS:              4799.83
Virtualización:        VT-x
Caché L1d:             32K
Caché L1i:             32K
Caché L2:              256K
Caché L3:              3072K
NUMA node0 CPU(s):     0-3
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 pdpe1g
b rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonst
op_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3
 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_tim
er aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb invpcid_single intel_
pt kaiser tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 sm
ep bmi2 erms invpcid mpx rdseed adx smap clflushopt xsaveopt xsavec xgetbv1 dthe
rm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp
```

1. Conteste a las siguientes preguntas:

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

RESPUESTA: Mi portátil tiene 2 cores físicos y 4 lógicos.

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

RESPUESTA: El atcgrid tiene 12 cores físicos en total, y 24 cores lógicos

*Cores físicos = Cores per socket * Sockets*

*Cores lógicos= threads per core *cores físicos*

CPUs = Threads per core X cores per socket X sockets

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: ncgt contiene el tiempo de ejecución del programa, la diferencia entre los tiempos medidos por clock_gettime(). La función clock_gettime() obtiene el tiempo actual de reloj especificado en el primer parámetro y lo pone en el buffer al que apunta el segundo parámetro, que apunta a un struct conteniendo al menos time_t tv_sec, número de segundos transcurridos desde 1970, y time_t tv_nsec, número de nanosegundos expirados en el segundo actual. Devuelve 0 si ha tenido éxito, -1 si ocurre algún error.

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++
Inclusión de bibliotecas	Stdlib.h, studio.h y time.h	Cstdlib, iostream, time.h
Uso de namespaces	No se pone nada	using namespaces std

Impresión por pantalla	Se usa printf(“blabla”)	Cout << “blabla”
Ampliación del vector	v1=(double*) malloc(N*sizeof(double));	v1 = new double [N]
Liberación de memoria	free(v1)	Delete [] v1

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: Desde mi terminal local:

Compilo y genero el ejecutable con gcc -O2 listado1.c -o listado1 -lrt

con gcc -O2 -S listado1.c generamos el código ensamblador y ejecutamos:

```
 2018-03-06 21:09:37  elena in ~/Escritorio/University stuff/2º/2º Cuatrimestre/AC/Prácticas
○ → ./listado1 12345
Tiempo(seg.): 0.000256951       /Tamaño vectores:12345  /V1[0] + V2[0]= V3[0](1234.500000+1234.500000=
2469.000000)// V1[12344] + V2[12344]= V3[12344](2468.900000+0.100000=2469.000000)/
```

Para ejecutar en atcgrid primeramente nos metemos en la carpeta con lcd Escritorio/… Ahí está el ejecutable que mandamos a la cola de trabajos con echo ‘./listado1 12345’ | qsub -ac, y leyendo el último STDIN obtenido tenemos:

```
[ElenaMereloMolina E2estudiante10@atcgrid:~] 2018-03-06 martes
$cat STDIN.o65424
Tiempo(seg.): 0.000079127       /Tamaño vectores:12345  /V1[0] + V2[0]= V3[0](1234.500000+1234.500000=
2469.000000)// V1[12344] + V2[12344]= V3[12344](2468.900000+0.100000=2469.000000)/
```

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: Para ello primero cargamos el script en atcgrid, escribiendo en sftp lcd Escritorio/carpeta_donde_está seguido de put SumaVectores.sh. Así, al hacer ls desde ssh nos aparecerá SumaVectores.sh, que ejecutamos con qsub SumaVectores.sh -q ac. Haciendo nuevamente ls vemos como han aparecido dos STDIN, uno con el output y otro con los errores surgidos. Para ver los errores escribimos cat SumaVectores_vlocales.e65432 y nos muestra:

```
[ElenaMereloMolina E2estudiante10@atcgrid:~] 2018-03-06 martes
$cat SumaVectores_vlocales.e65432
/var/lib/torque/mom_priv/jobs/65432.atcgrid.SC: line 25: 28025 Segmentation fault
     (core dumped) $PBS_O_WORKDIR/listado1 $N
/var/lib/torque/mom_priv/jobs/65432.atcgrid.SC: line 25: 28028 Segmentation fault
     (core dumped) $PBS_O_WORKDIR/listado1 $N
/var/lib/torque/mom_priv/jobs/65432.atcgrid.SC: line 25: 28031 Segmentation fault
     (core dumped) $PBS_O_WORKDIR/listado1 $N
/var/lib/torque/mom_priv/jobs/65432.atcgrid.SC: line 25: 28035 Segmentation fault
     (core dumped) $PBS_O_WORKDIR/listado1 $N
/var/lib/torque/mom_priv/jobs/65432.atcgrid.SC: line 25: 28041 Segmentation fault
     (core dumped) $PBS_O_WORKDIR/listado1 $N
/var/lib/torque/mom_priv/jobs/65432.atcgrid.SC: line 25: 28044 Segmentation fault
     (core dumped) $PBS_O_WORKDIR/listado1 $N
/var/lib/torque/mom_priv/jobs/65432.atcgrid.SC: line 25: 28049 Segmentation fault
     (core dumped) $PBS_O_WORKDIR/listado1 $N
/var/lib/torque/mom_priv/jobs/65432.atcgrid.SC: line 25: 28054 Segmentation fault
     (core dumped) $PBS_O_WORKDIR/listado1 $N
```

Luego da error para los tamaños 28025, 28028, 28031, 28035, 28041, 28044, 28049 y 28054. Por lo demás, el resultado de la ejecución es:

```
Id. usuario del trabajo: E2estudiante10
Id. del trabajo: 65432.atcgrid
Nombre del trabajo especificando usuario: SumaVectores_vlocales
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/E2estudiante10
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.): 0.000436472       /Tamaño vectores:65536  /V1[0] + V2[0]=
 V3[0](6553.600000+6553.600000=13107.200000)// V1[65535] + V2[65535]= V
3[65535](13107.100000+0.100000=13107.200000)/
Tiempo(seg.): 0.000849294       /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.): 0.001409667       /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)/
```

En mi ordenador:

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: Para vectores globales no se obtiene error, al hacer cat SumaVectores_vlocales.e65437 no muestra nada. Resultado de la ejecución del script:

```
Id. usuario del trabajo: E2estudiante10
Id. del trabajo: 65437.atcgrid
Nombre del trabajo especificando usuario: SumaVectores_vlocales
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/E2estudiante10
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.): 0.000287583       /Tamaño vectores:65536  /V1[0] + V2[0]=
 V3[0](6553.600000+6553.600000=13107.200000)// V1[65535] + V2[65535]= V
3[65535](13107.100000+0.100000=13107.200000)/
Tiempo(seg.): 0.000856469       /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.): 0.001698096       /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.): 0.002393489       /Tamaño vectores:524288 /V1[0] + V2[0]=
 V3[0](52428.800000+52428.800000=104857.600000)// V1[524287] + V2[52428
7]= V3[524287](104857.500000+0.100000=104857.600000)/
Tiempo(seg.): 0.006148235       /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.): 0.011800984       /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.): 0.023807653       /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.): 0.047580533       /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.): 0.093719713       /Tamaño vectores:16777216       /V1[0]
+ V2[0]= V3[0](1677721.600000+1677721.600000=3355443.200000)// V1[16777
215] + V2[16777215]= V3[16777215](3355443.100000+0.100000=3355443.20000
0)/
Tiempo(seg.): 0.172781675       /Tamaño vectores:33554432       /V1[0]
+ V2[0]= V3[0](3355443.200000+3355443.200000=6710886.400000)// V1[33554
431] + V2[33554431]= V3[33554431](6710886.300000+0.100000=6710886.40000
0)/
Tiempo(seg.): 0.178914210       /Tamaño vectores:33554432       /V1[0]
+ V2[0]= V3[0](3355443.200000+3355443.200000=6710886.400000)// V1[33554
431] + V2[33554431]= V3[33554431](6710886.300000+0.100000=6710886.40000
0)/
```

Para vectores dinámicos tampoco da error en la ejecución, y muestra:

```
Id. usuario del trabajo: E2estudiante10
Id. del trabajo: 65440.atcgrid
Nombre del trabajo especificando usuario: SumaVectores_vlocales
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/E2estudiante10
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.): 0.000273259       /Tamaño vectores:65536  /V1[0] + V2[0]=
 V3[0](6553.600000+6553.600000=13107.200000)// V1[65535] + V2[65535]= V
3[65535](13107.100000+0.100000=13107.200000)/
Tiempo(seg.): 0.000913428       /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.): 0.001606001       /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.): 0.002751598       /Tamaño vectores:524288 /V1[0] + V2[0]=
 V3[0](52428.800000+52428.800000=104857.600000)// V1[524287] + V2[52428
7]= V3[524287](104857.500000+0.100000=104857.600000)/
Tiempo(seg.): 0.006001974       /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.): 0.012271129       /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.): 0.023532919       /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.): 0.046516008       /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.): 0.095131995       /Tamaño vectores:16777216       /V1[0]
+ V2[0]= V3[0](1677721.600000+1677721.600000=3355443.200000)// V1[16777
215] + V2[16777215]= V3[16777215](3355443.100000+0.100000=3355443.20000
0)/
Tiempo(seg.): 0.188591256       /Tamaño vectores:33554432       /V1[0]
+ V2[0]= V3[0](3355443.200000+3355443.200000=6710886.400000)// V1[33554
431] + V2[33554431]= V3[33554431](6710886.300000+0.100000=6710886.40000
0)/
Tiempo(seg.): 0.350095555       /Tamaño vectores:67108864       /V1[0]
+ V2[0]= V3[0](6710886.400000+6710886.400000=13421772.800000)// V1[6710
8863] + V2[67108863]= V3[67108863](13421772.700000+0.100000=13421772.80
0000)/
[ElenaMereloMolina E2estudiante10@atcgrid:~] 2018-03-06 martes
```

Resultados desde mi ordenador:

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:

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:

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.):%11.9f\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
```