

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): Carlos Sánchez Páez

Grupo de prácticas: A2

Fecha de entrega: 27/02/2018

Fecha evaluación en clase: 6/3/2018

- Incorpore volcados de pantalla que muestren lo que devuelve `lscpu` en atcggrid y en su PC.

#### CAPTURAS:

```
csp98@csp98-Sobremesa ~$ acftp
[Carlos Sánchez Páez csp98@csp98-Sobremesa:] 2018-02-21 miércoles
$lscpu
Arquitectura: x86_64
modo(s) de operación de las CPUs: 32-Bit, 64-bit
Orden de los bytes: LittleEndian
CPU(s): 8
Lista de la(s) CPU(s) en líneas: 0
Número de procesamiento por núcleo: 1
Núcleo(s) por <socket>: 1
<Socket(s)>: 1
Modo(s) NUMA: 1
ID de fabricante: AuthenticAMD
Familia de CPU: 15
Modelo: 47
Nombre del modelo: AMD Athlon(tm) 64 Processor
F 3800+
Revisión: 2
CPU MHz: 1802.252
BogomIPS: 3604.50
Caché L1d: 64K
Caché L1t: 64K
Caché L2: 512K
CPU(s) del nodo NUMA 0: 0
Indicadores: fpu vme de pse tsc msr pae
mce cx8 apic sep ntr pge mca cmov pat pse36 clflush mmx fxsr
sse sse2 syscall nx mmext fxsr_opt lm 3dnowext 3dnow rep_good
rep_nop clflush_tsd_apicid pn1 lahf_lm 3dnowprefetch repoline rsb
ctsw vmcall
[A2estudiante23@atcggrid ~]$
```

```
csp98@csp98-Sobremesa ~$ acftp
[Carlos Sánchez Páez csp98@csp98-Sobremesa:] 2018-02-21 miércoles
$ Connected to atcggrid.ugr.es.
sftp>
```

```
csp98@csp98-Sobremesa ~$ acprompt
[Carlos Sánchez Páez csp98@csp98-Sobremesa:] 2018-02-21 miércoles
$lscpu
Arquitectura: x86_64
modo(s) de operación de las CPUs: 32-Bit, 64-bit
Orden de bytes: LittleEndian
CPU(s): 8
Lista de la(s) CPU(s) en líneas: 0
Número de procesamiento por núcleo: 1
Núcleo(s) por <socket>: 1
<Socket(s)>: 1
Modo(s) NUMA: 1
ID de fabricante: AuthenticAMD
Familia de CPU: 21
Modelo: 2
Nombre del modelo: AMD FX(tm)-8320 Eight-Core Process
or
Revisión: 0
CPU MHz: 4200.452
BogomIPS: 8400.96
Virtualización: AMD-V
Caché L1d: 16K
Caché L1t: 64K
Caché L2: 2048K
Caché L3: 8192K
NUMA node0 CPU(s): 0-7
Flags: fpu vme de pse tsc msr pae mce cx8
apic sep ntr pge mca cmov pat pse36 clflush mmx fxsr sse2 ht sysc
all mm mmext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl
stop tsc cpuid extd.apicid aperfmperf pn1 ocunlqdq monitor ssse3 fmw
x10 ssse3_1 ssse3_2 popcnt aes xavav avx f16c lahf_lm cmp_legacy svm exta
pdc cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw lbs xop skinit
wdt lwp fma4 tce topoex perfctr_nb cpb hw
pstate vmcall bmii arat npt lbrv svn_lock nrtp_save tsc_scale vncb_cle
an flushbyasid decodeassist pausefilter ptthreshold
[Carlos Sánchez Páez csp98@csp98-Sobremesa:] 2018-02-21 miércoles
$
```

Ilustración 1: `lscpu` en mi PC y en el front-end

```
csp98@csp98-Sobremesa ~$ acftp
[Carlos Sánchez Páez csp98@csp98-Sobremesa:] 2018-02-21 miércoles
$lscpu
Arquitectura: x86_64
modo(s) de operación de las CPUs: 32-Bit, 64-bit
Orden de bytes: LittleEndian
CPU(s): 8
Lista de la(s) CPU(s) en líneas: 0
Número de procesamiento por núcleo: 2
Core(s) per socket: 4
Socket(s): 2
NUMA node(s): 2
Vendor ID: GenuineIntel
CPU family: 6
Model: 44
Model name: Intel(R) Xeon(R) CPU E5645 @ 2.
Flags: fpu vme de pse tsc msr pae mce cx8 apic se
p mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 sse3 sse4_1 sse4_2 popcnt abm misalignsse 3dnowprefetch
pdcm pdcm_id dca ssse3_1 ssse3_2 popcnt lahf_lm epb pti repoline t
pr_shadow vnni flexpriority ept vpid dtherm lda arat
[A2estudiante23@atcggrid ~]$
```

```
csp98@csp98-Sobremesa ~$ acftp
[Carlos Sánchez Páez csp98@csp98-Sobremesa:] 2018-02-21 miércoles
$ Connected to atcggrid.ugr.es.
sftp>
```

```
csp98@csp98-Sobremesa ~$ acprompt
[Carlos Sánchez Páez csp98@csp98-Sobremesa:] 2018-02-21 miércoles
$lscpu
Arquitectura: x86_64
modo(s) de operación de las CPUs: 32-Bit, 64-bit
Orden de bytes: LittleEndian
CPU(s): 8
Lista de la(s) CPU(s) en líneas: 0
Número de procesamiento por núcleo: 2
Core(s) per socket: 4
Socket(s): 1
Modo(s) NUMA: 1
ID de fabricante: AuthenticAMD
Familia de CPU: 21
Modelo: 2
Nombre del modelo: AMD FX(tm)-8320 Eight-Core Process
or
Revisión: 0
CPU MHz: 4200.452
BogomIPS: 8400.96
Virtualización: AMD-V
Caché L1d: 16K
Caché L1t: 64K
Caché L2: 2048K
Caché L3: 8192K
NUMA node0 CPU(s): 0-7
Flags: fpu vme de pse tsc msr pae mce cx8
apic sep ntr pge mca cmov pat pse36 clflush mmx fxsr sse2 ht sysc
all mm mmext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl
stop tsc cpuid extd.apicid aperfmperf pn1 ocunlqdq monitor ssse3 fmw
x10 ssse3_1 ssse3_2 popcnt aes xavav avx f16c lahf_lm cmp_legacy svm exta
pdc cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw lbs xop skinit
wdt lwp fma4 tce topoex perfctr_nb cpb hw
pstate vmcall bmii arat npt lbrv svn_lock nrtp_save tsc_scale vncb_cle
an flushbyasid decodeassist pausefilter ptthreshold
[Carlos Sánchez Páez csp98@csp98-Sobremesa:] 2018-02-21 miércoles
$
```

Ilustración 2: `lscpu` en mi PC y en atcggrid

- Conteste a las siguientes preguntas:

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

**RESPUESTA:** el front-end tiene un núcleo físico. Mi PC tiene 4 núcleos físicos y 4 lógicos.

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

**RESPUESTA:** cada nodo de atcgrid posee 12 cores físicos distribuidos en dos procesadores y otros 12 lógicos, uno por cada físico.

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++:

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

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` contiene el tiempo que tarda en ejecutarse el bucle, calculada como la resta de la hora del sistema al terminar el bucle menos la hora del sistema justo antes de comenzarlo.
- La función `clock_gettime()` devuelve en el segundo parámetro la hora que marca el reloj pasado como primer parámetro (en nuestro caso, el reloj del sistema, contando a partir del 1 de enero de 1970, determinado por la constante `CLOCK_REALTIME`).
- La información es devuelta en una estructura de datos denominada `struct timespec`. Dicho struct consta de los siguientes campos:
  - `time_t tv_sec` -----> Tiempo en segundos
  - `long tv_nsec` -----> Tiempo en nanosegundos

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++
Reservar memoria dinámicas	<i>malloc</i>	<i>new</i>
Liberar memoria dinámica	<i>free</i>	<i>delete []</i>
Imprimir resultados	Con formato ( <i>%8.6f</i> )	Sin formato ( <i>&lt;&lt;valor&lt;&lt;</i> )
Memoria insuficiente	Mensaje por pantalla	Excepción
Variable contador	Declarada previamente	Declarada en el bucle
Librerías	<i>stdio.h</i> y <i>stdlib.h</i>	<i>cstdlib</i>

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:**

Ilustración 3: Ejecución con vectores locales (tamaño = 50)

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:

Ilustración 4: Ejecución con vectores locales (tamaños incrementales)

Los errores son de violación de segmento, ya que se supera el tamaño de la pila tanto en mi PC como en *atcgrid*.

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:**

*Ilustración 5: Ejecución con vectores globales*

### Ilustración 6: Ejecución con vectores dinámicos

Con vectores globales y dinámicos no hay error, ya que la pila tiene un espacio reservado menor, por lo que se acaba llenando. Sin embargo, ésto no ocurre con vectores dinámicos y globales, ya que podemos utilizar toda la memoria que el programa pueda reservar.

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?

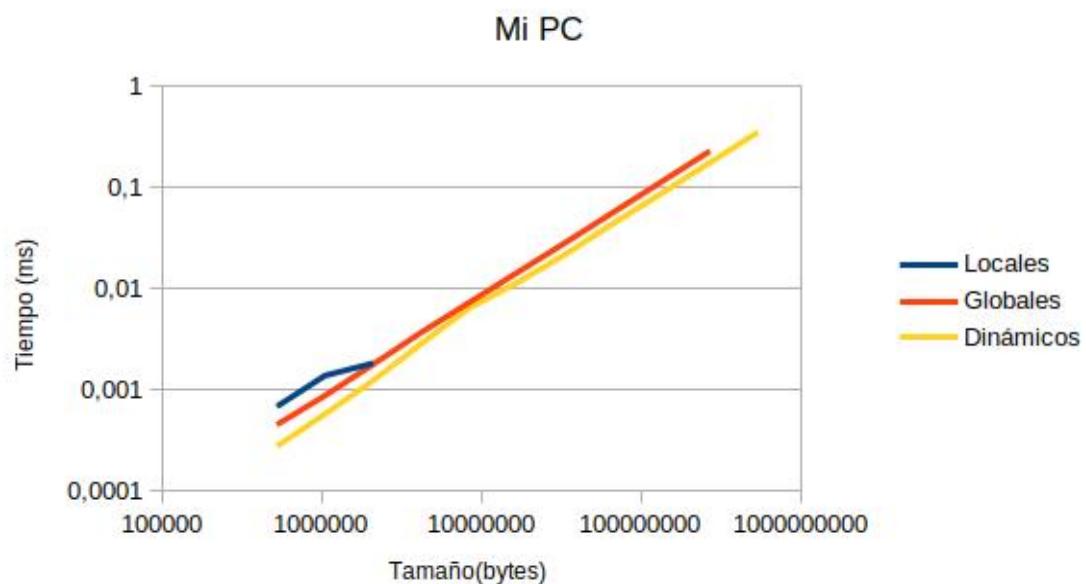
## **RESPUESTA:**

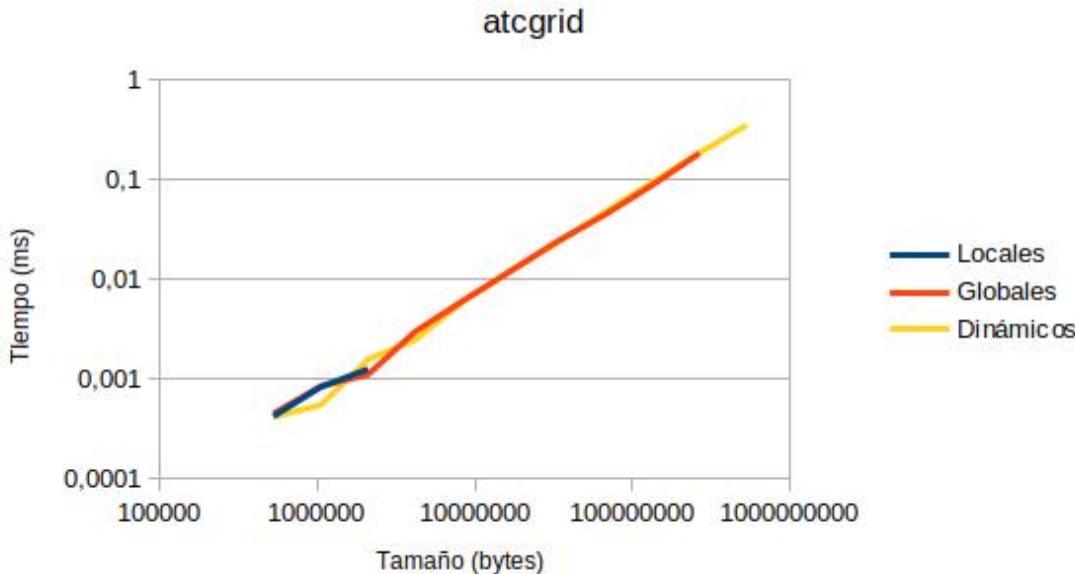
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,00068136	0,000452668	0,000276880
131072	1048576	0,001352607	0,000876283	0,000575715
262144	2097152	0,000841927	0,0017858881	0,001233541
524288	4194304	-	0,003695965	0,002853456
1048576	8388608	-	0,007309177	0,006453486
2097152	16777216	-	0,014516858	0,011535317
4194304	33554432	-	0,028672492	0,021934680
8388608	67108864	-	0,057383994	0,044589411
16777216	134217728	-	0,114874636	0,088169312
33554432	268435456	-	0,230139659	0,177227405
67108864	536870912	-	0,232664195	0,354258726

## Tiempos de mi PC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000425618	0,000453170	0,000419948
131072	1048576	0,00083625	0,000850759	0,000550689
262144	2097152	0,001260025	0,001115098	0,001579105
524288	4194304	-	0,003003053	0,002490386
1048576	8388608	-	0,006071652	0,005982679
2097152	16777216	-	0,012025076	0,012100045
4194304	33554432	-	0,024006358	0,023416976
8388608	67108864	-	0,044293580	0,047130724
16777216	134217728	-	0,087653293	0,093780705
33554432	268435456	-	0,18277683	0,185038917
67108864	536870912	-	0,382703249	0,351243974

Tiempos de atcgrid





Los últimos tiempos de la ejecución con vectores globales han sido eliminados ya que el sistema no es capaz de reservar memoria para ellos. En mi PC aparecen algunos picos debido a la carga del sistema, motor gráfico, etc.

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? Razona además por qué el máximo número que se puede almacenar en N es  $2^{32}-1$ .

### **RESPUESTA:**

```
csp98@csp98-Sobremesa ~ $ accsn
[ Carlos Sanchez Paez csp98@csp98-Sobremesa:~] 2018-02-26 lunes
Ssd Escritorio/AC/practica0

csp98@csp98-Sobremesa ~ $ acftp
[ Carlos Sanchez Paez csp98@csp98-Sobremesa:/Escritorio/AC/practica0] 2018-02-26 lunes
Sftp -o SunaVectoresC.c:SunavectoresC.c:-02 -lrt
[ /tmp/cpgmG19.o ] la función main:
SunaVectoresC.c:(.text.startup+0x5b): reubicación truncada para ajustar
: R_X86_64 PC32 contra el símbolo `V2' definido en la sección COMMON en
/tmp/cpgmG19.o
SunaVectoresC.c:(.text.startup+0x5b): reubicación truncada para ajustar
: R_X86_64 PC32 contra el símbolo `V3' definido en la sección COMMON en
/tmp/cpgmG19.o
SunaVectoresC.c:(.text.startup+0x130): reubicación truncada para ajustar
r: R_X86_64 PC32 contra el símbolo `V3' definido en la sección COMMON en
/tmp/cpgmG19.o
SunaVectoresC.c:(.text.startup+0x130): reubicación truncada para ajustar
r: R_X86_64 PC32 contra el símbolo `V2' definido en la sección COMMON en
/tmp/cpgmG19.o
SunaVectoresC.c:(.text.startup+0x1a0): reubicación truncada para ajustar
r: R_X86_64 PC32 contra el símbolo `V2' definido en la sección COMMON en
/tmp/cpgmG19.o
SunaVectoresC.c:(.text.startup+0x1a7): reubicación truncada para ajustar
r: R_X86_64 PC32 contra el símbolo `V3' definido en la sección COMMON en
/tmp/cpgmG19.o
collect2: error: ld returned 1 exit status
[ Carlos Sanchez Paez csp98@csp98-Sobremesa:/Escritorio/AC/practica0] 2018-02-26 lunes
LOCAL
```

Obtenemos un error de compilación ya que no hay memoria suficiente para reservar. El máximo es  $2^{32}-1$  porque es el máximo valor que puede adoptar N (*unsigned int*).

**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()

#ifndef 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 ...)
    // generarán 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)

#ifndef 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
#ifndef 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) \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

#ifndef 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()

#ifndef 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)

#ifndef 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,&cg1t);
    //Calcular suma de vectores
    for(int i=0; i<N; i++)
        v3[i] = v1[i] + v2[i];
    clock_gettime(CLOCK_REALTIME,&cg2t);
}
```

```

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  

#ifndef COUT_ALL  

cout << "Tiempo(seg.):" << ncgt << "\t/ Tamaño Vectores:" << N << endl;  

for(int i=0; i<N; i++)  

    cout << "/ V1[" << 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 << "]+"  

<< N-1 << "="  

    << N-1 << "](" << v1[N-1] << "+" << v2[N-1] << "=" << v3[N-1] << ")/\n" <<  

endl;  

#endif  
  

#ifndef 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

```