|  |  |  |
| --- | --- | --- |
| **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:  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="[FernandoVallecillosRuiz \u@\h:\w] \D{%F %A}\n$"

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

**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

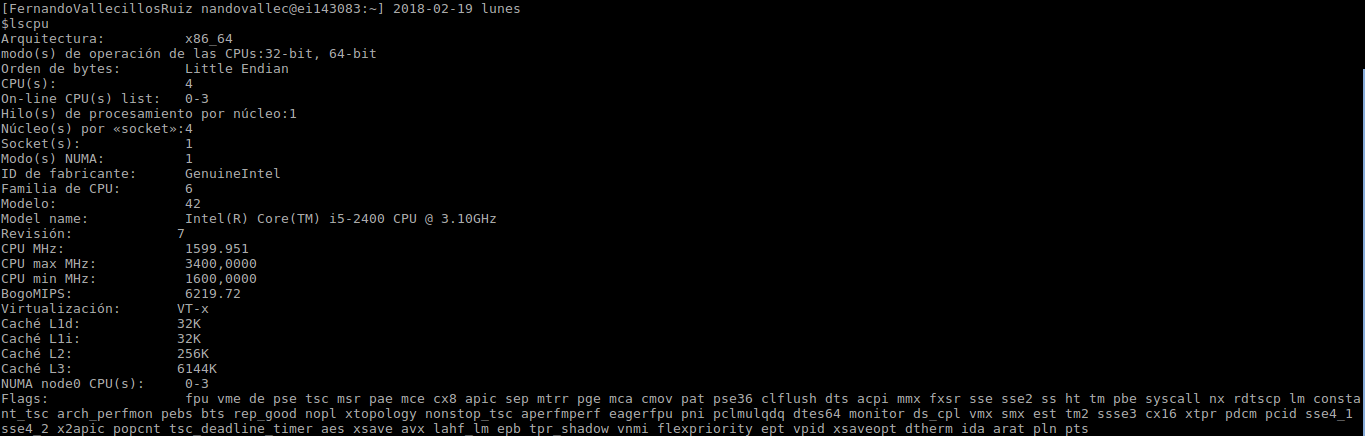
- **Courier New-10 o Liberation Mono-10 para nombres de fichero, comandos, variables de entorno, etc., cuando se usan en el texto.**

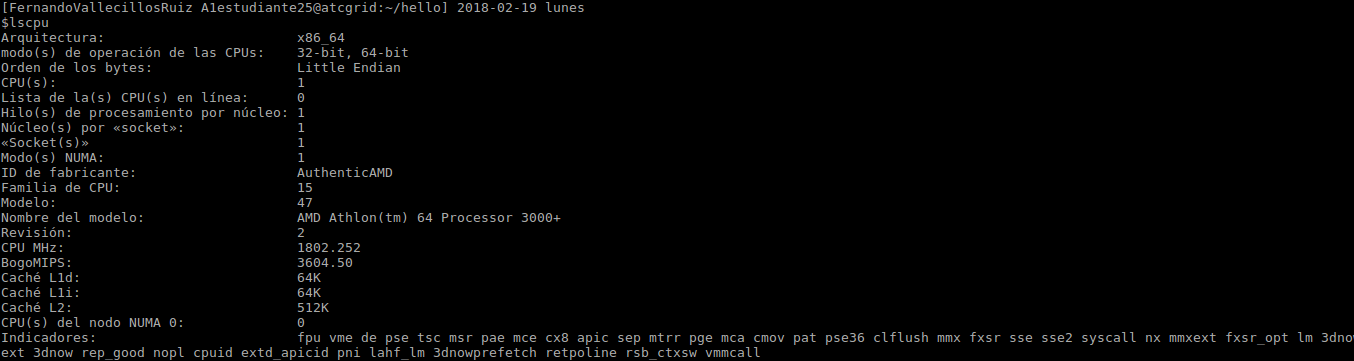
3) Insertar las capturas de pantalla donde se pidan y donde se considere oportuno. En particular, los listados de código se deben insertar como capturas de pantalla. En todas las capturas de pantalla, incluidas las de los listados de código, debe aparecer el directorio y usuario. El tamaño de letra en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

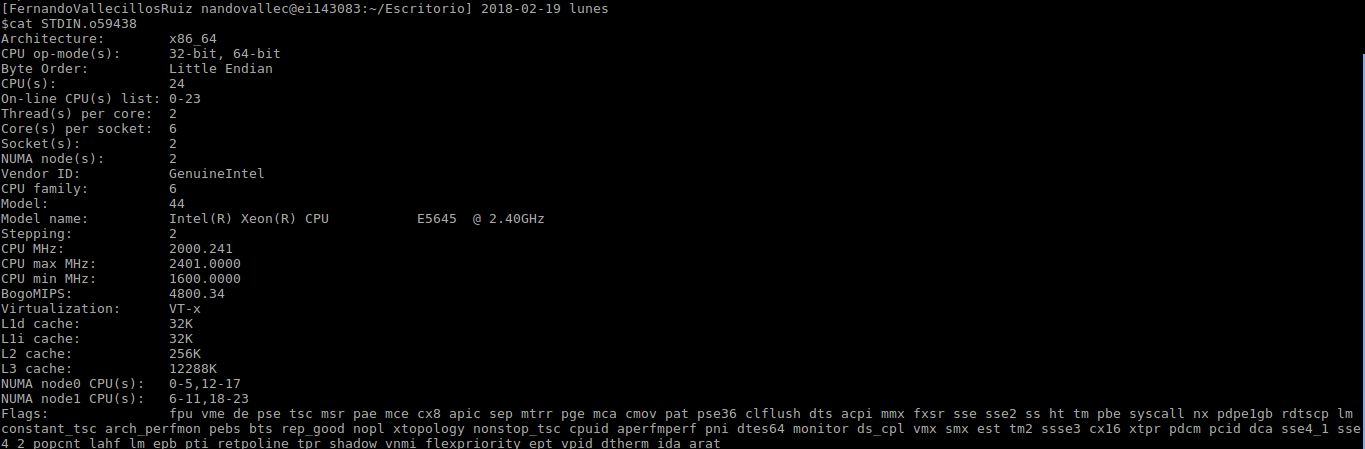
**Recuerde que debe adjuntar al zip de entrega, el pdf de este fichero, todos los ficheros con código fuente implementados/utilizados y el resto de ficheros que haya implementado/utilizado (scripts, hojas de cálculo, etc.)]**

**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**: Atcgrid tiene 1 solo core físico que es también su único core lógico. Mi PC tiene 4 cores físicos y cada uno de ellos con 1 core lógico (4 cores lógicos en total).

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

**RESPUESTA**: Tiene 12 cores físicos, cada uno con 2 lógicos. Haciendo un total de 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.

1. 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**: ngct es de tipo double. Contiene el diferencia entre dos intantes de tiempo, es decir, una duración, que corresponde al tiempo que se tarda en realizar el bucle. clock\_gettime() devuelve el instante de tiempo de un reloj en específico (en este caso el reloj de tiempo real). La función devuelve un int, 0 si se ha tenido éxito, o -1 si ha sucedido algún error. La información se devuelve en la estrucura timespec, en este caso devuelve los segundos y nanosegundos desde el Epoch(00:00 UTC del 1 de enero de 1970). Esta estructura contiene un time\_t(tv\_sec) que indica segundos y un long(tv\_nsec) que indica nanosegundos.

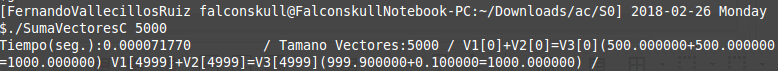
1. Escribir en el cuaderno de prácticas las diferencias que hay entre el código fuente C y el código fuente C++ para la suma de vectores.

**RESPUESTA**:

|  |  |  |
| --- | --- | --- |
| Descripción diferencia | En C | En C++ |
| Como envían los datos para mostrarlos en la salida | Se usa printf | Se usa cout |
| Como se reserva y libera memoria para las variables dinámicas | Se usa malloc y free | Se usa new y delete |
| Fallos al reservar memoria | Malloc devuelve null | New genera una excepción |
|  |  |  |
|  |  |  |

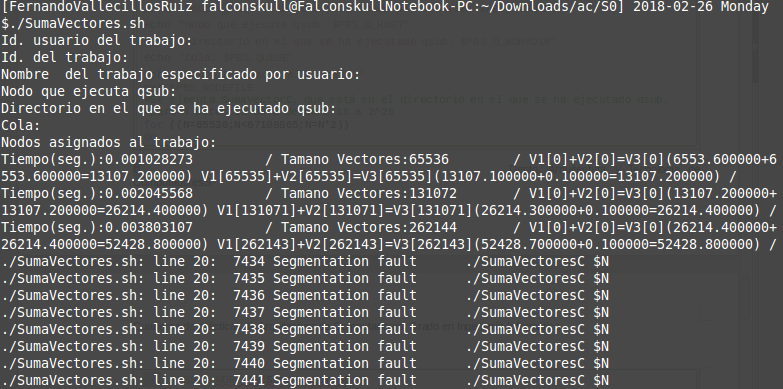
**3.** Generar el ejecutable del código fuente C del Listado 1 para vectores locales (para ello antes de compilar debe descomentar la definición de VECTOR\_LOCAL y comentar las definiciones de VECTOR\_GLOBAL y VECTOR\_DYNAMIC). Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid o en su PC.

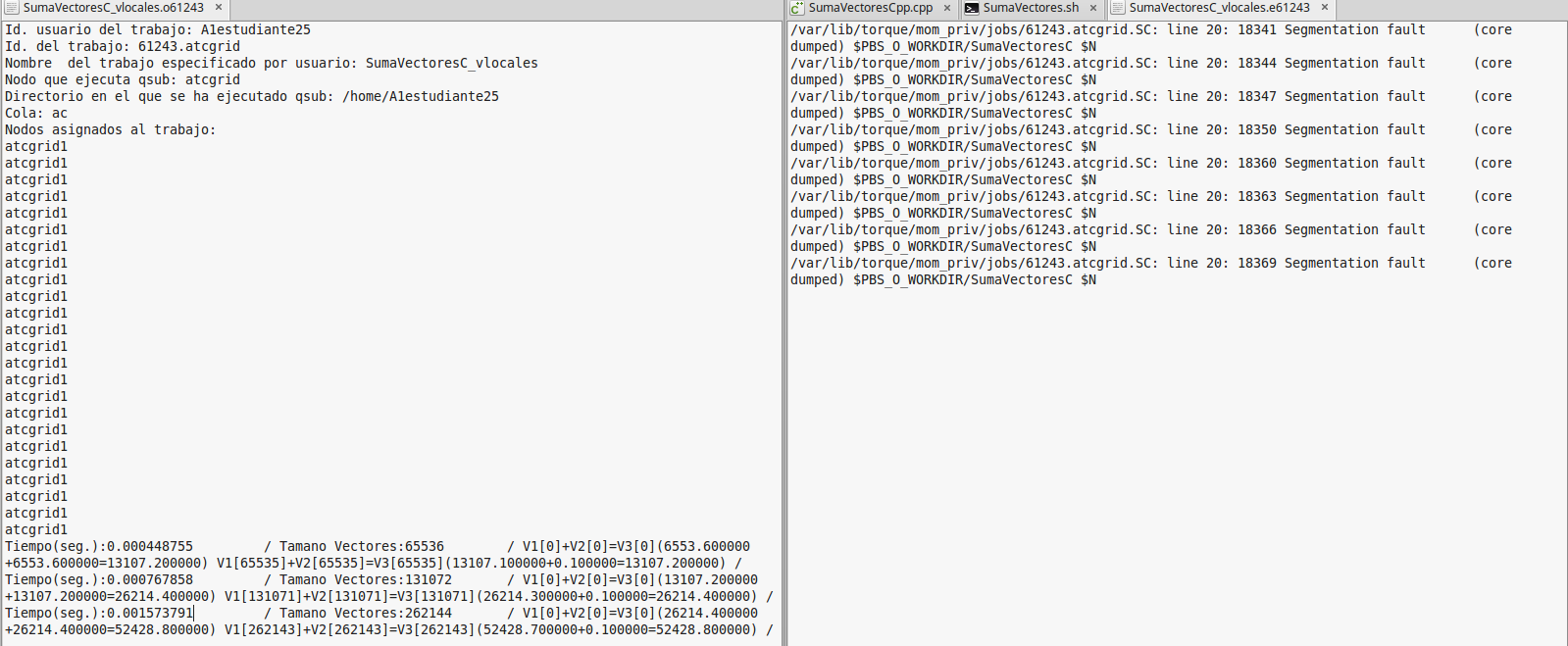
**RESPUESTA**:



**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**: Se obtiene error a partir del tamaño 524288. Esto es debido a que existe un número límite de variables locales en un mismo programa.





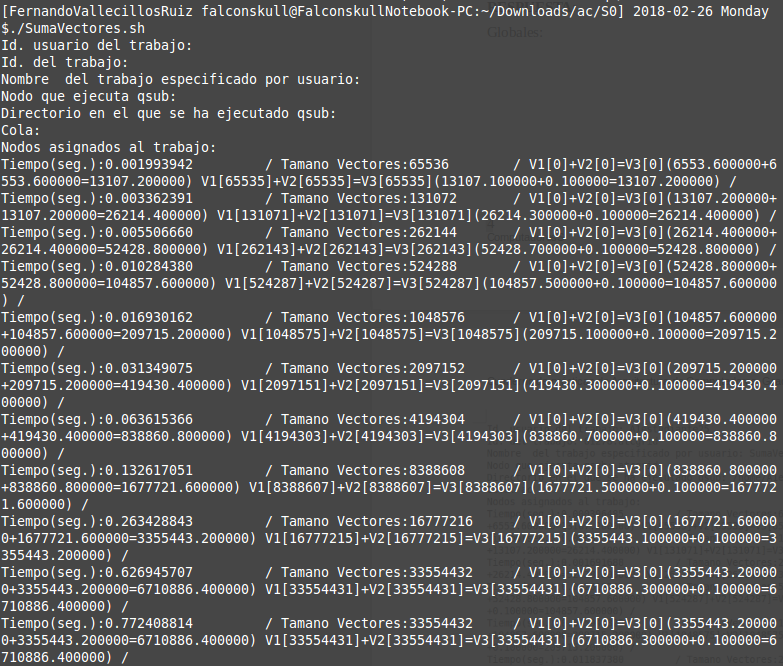
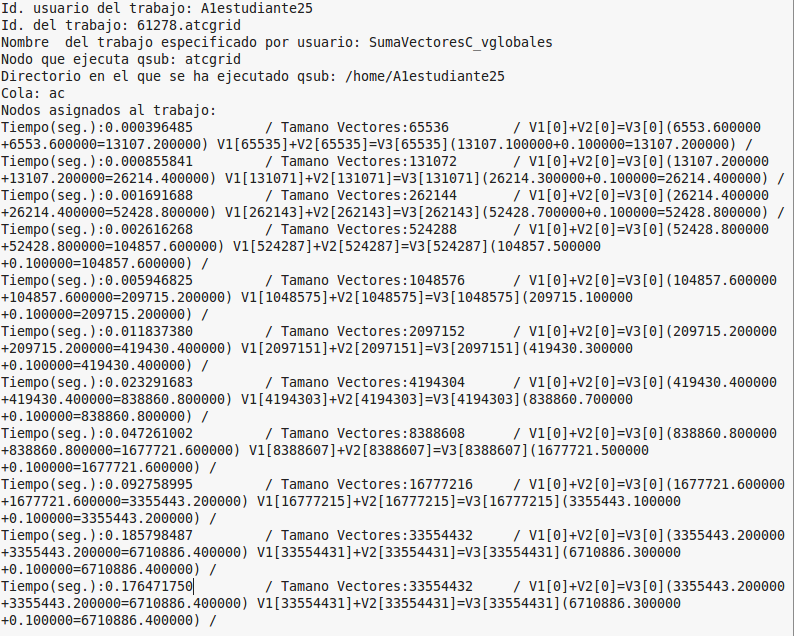
**5.** Generar los ejecutables del código fuente C para vectores globales y para dinámicos. Genere el ejecutable usando –O2. Ejecutar los dos códigos en atcgrid usando un script como el del Listado 3 (hay que poner en el script el nombre de los ficheros ejecutables generados en este ejercicio) para el 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**:

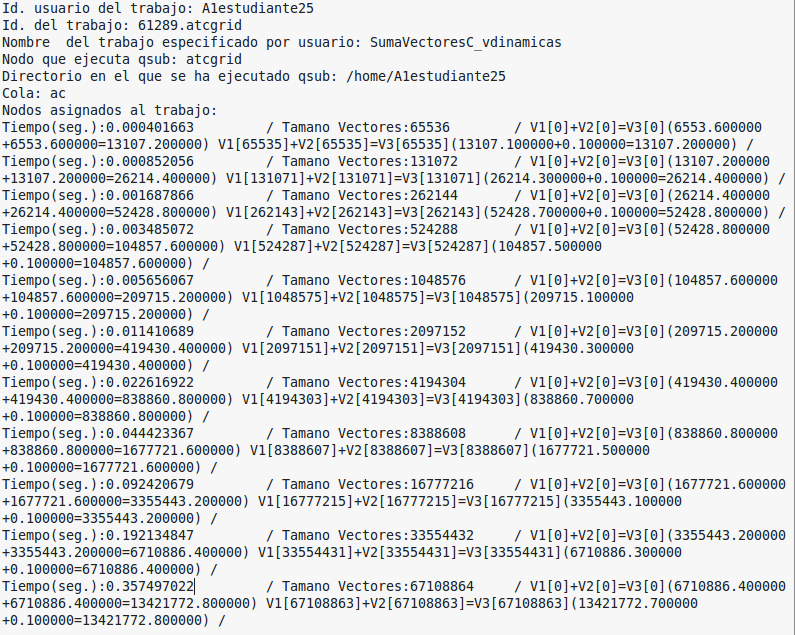
No se obtienen errores. El código del vector global tiene una limitación de tamaño de 2²⁵.

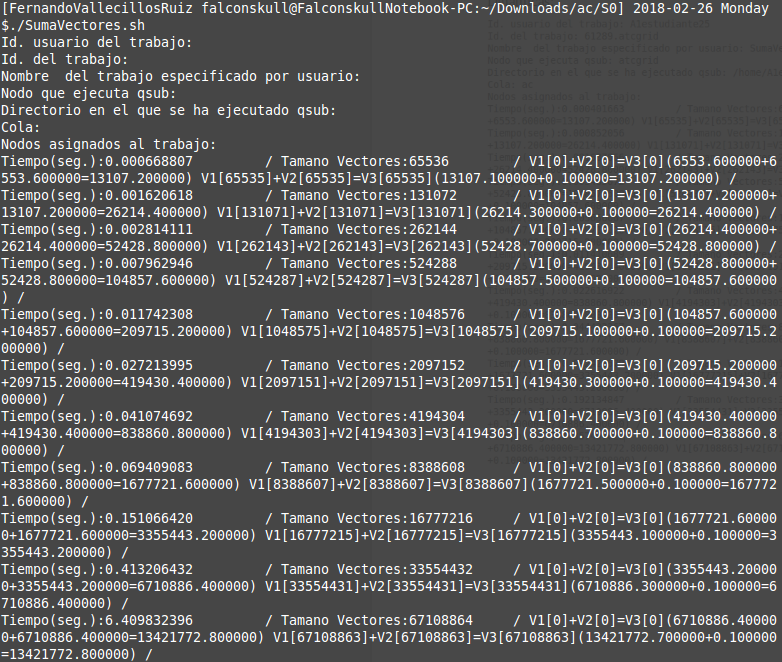
El vector dinámico tampoco da error ya que reserva la memoría que necesita pero solo guarda la primera dirección y puede calcular cualquier posición ya que se guardan de forma secuencial.

Globales:



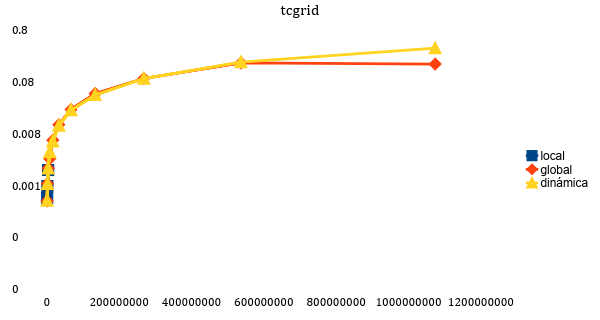
Dinámicas:

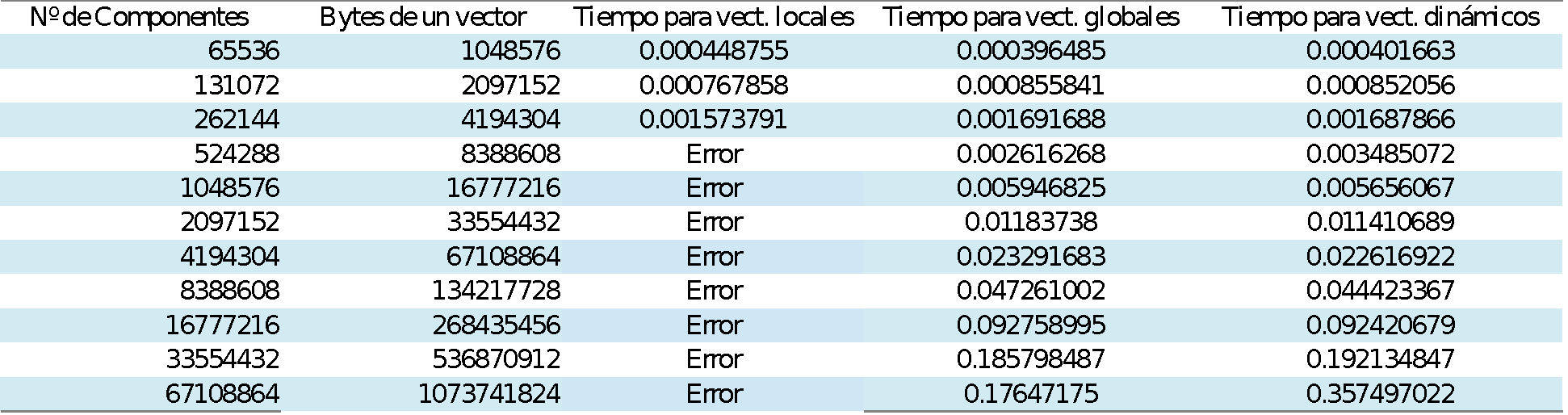


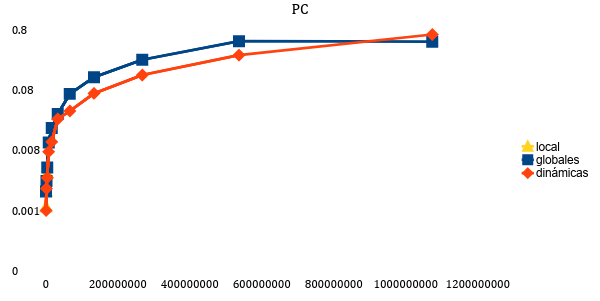


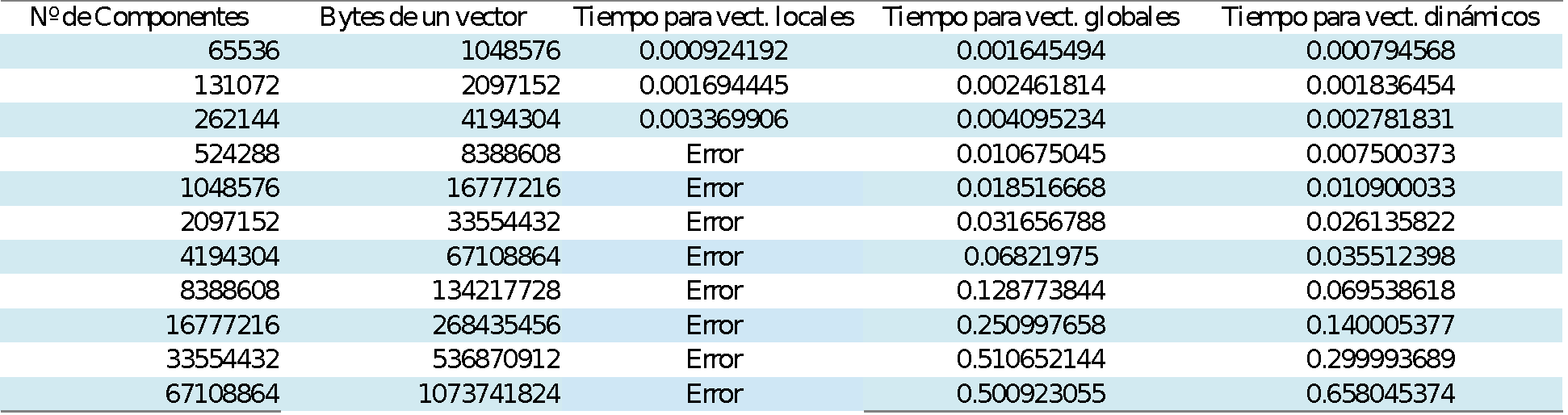
**6.** Rellenar una tabla como la Tabla 1 para atcgrid y otra para su PC con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. Ayudándose 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**: Existen diferencias en los tiempos de ejecucion siendo los tiempos de tcgrid normalmente mas pequeños.









**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 232-1.

**RESPUESTA**: Da un error al compilar. El archivo binario que generaría sería enorme. Hay un limite de tamaño de datos de memoría que un archivo puede tener. Esto es debido a que la dirección de memoría tiene un mayor desplazamiento del que se puede indicar con 32 bits.

|  |
| --- |
|  |
|  |



|  |
| --- |
| **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 |