2º curso / 2º cuatr. Grado Ingeniería Informática	**Arquitectura de Computadores (AC)** **Cuaderno de prácticas.** **Bloque Práctico 0. Entorno de programación** Estudiante: Juan Carlos Ruiz Fernández Grupo de prácticas y profesor de prácticas: Dª Mancia Anguita Fecha de entrega: 16 de marzo de 2021 Fecha evaluación en clase:

Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con los normas de prácticas que se encuentra en SWAD

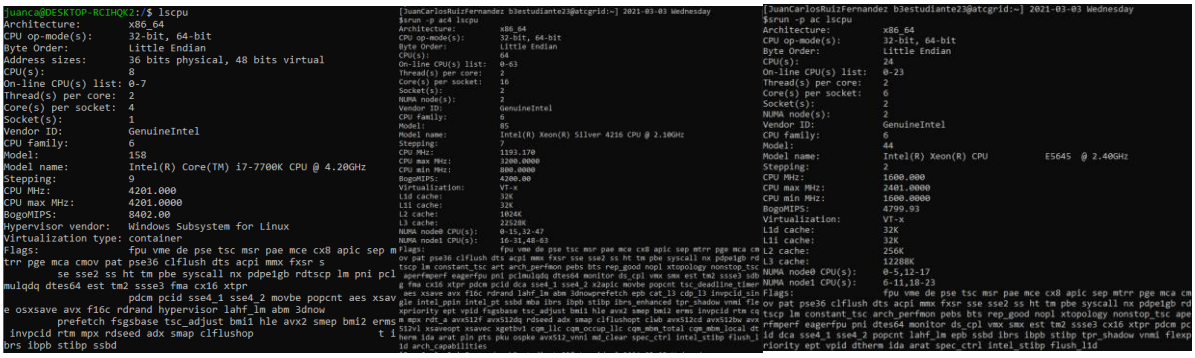
Parte I. Ejercicios basados en los ejemplos del seminario práctico

Crear el directorio con nombre bp0 en atcgrid y en el PC (PC = PC del aula de prácticas o su computador personal).
NOTA: En las prácticas se usa slurm como gestor de colas. Consideraciones a tener en cuenta:

- Slurm está configurado para asignar recursos a los procesos (llamados *tasks* en slurm) a nivel de core físico. Esto significa que por defecto slurm asigna un core a un proceso, para asignar x se debe usar con sbatch/srun la opción --cpus-per-task=x (-cx).
- En slurm, por defecto, cpu se refiere a cores lógicos (ej. en la opción -c), si no se quieren usar cores lógicos hay que añadir la opción --hint=nomultithread a sbatch/srun.
- Para asegurar que solo se crea un proceso hay que incluir --ntasks=1 (-n1) en sbatch/srun.
- Para que no se ejecute más de un proceso en un nodo de cómputo de atcgrid hay que usar --exclusive con sbatch/srun (se recomienda no utilizarlo en los srun dentro de un script).
- Los srun dentro de un *script* heredan las opciones fijadas en el sbatch que se usa para enviar el script a la cola (partición slurm).
- Las opciones de sbatch se pueden especificar también dentro del *script* (usando #SBATCH, ver ejemplos en el script del seminario)

1. Ejecutar lscpu en el PC, en atcgrid4 (usar -p ac4) y en uno de los restantes nodos de cómputo (atcgrid1, atcgrid2 o atcgrid3, están en la cola ac). (Crear directorio ejer1)

   (a) Mostrar con capturas de pantalla el resultado de estas ejecuciones.



   (b) ¿Cuántos cores físicos y cuántos cores lógicos tiene atcgrid4?, ¿cuántos tienen atcgrid1, atcgrid2 y atcgrid3? y ¿cuántos tiene el PC? Razonar las respuestas

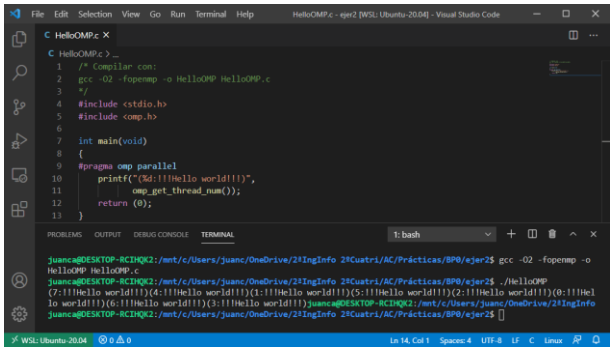
   atcgrid4 tiene 32 (16 per socket) fisicos y 64 logicos

   atcgrid1,2,3 tiene 12 (6 per socket) fisicos y 24 logicos

   Mi pc tiene 4 fisicos y 8 logicos

2. Compilar y ejecutar en el PC el código HelloOMP.c del seminario (recordar que, como se indica en las normas de prácticas, se debe usar un directorio independiente para cada ejercicio dentro de bp0 que contenga todo lo utilizado, implementado o generado durante el desarrollo del mismo, para el presente ejercicio el directorio sería ejer2).

(a) Adjuntar capturas de pantalla que muestren la compilación y ejecución en el PC.



(b) Justificar el número de "Hello world" que se imprimen en pantalla teniendo en cuenta la salida que devuelve lscpu en el PC.

Mi lscpu devuelve que tengo 8 cpus (4 fisicos) por lo que me devuelve el programa 8 "Hello World" 0-7.

3. Copiar el ejecutable de HelloOMP.c que ha generado anteriormente y que se encuentra en el directorio ejer2 del PC al directorio ejer2 de su home en el *front-end* de atcgrid. Ejecutar este código en un nodo de cómputo de atcgrid (de 1 a 3) a través de cola ac del gestor de colas utilizando directamente en línea de comandos (no use ningún *script*):

(a) srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP

(Alternativa: srun -pac -Aac -n1 -c12 --hint=nomultithread HelloOMP)

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

```
[JuanCarlosRuizFernandez b3estudiante23@atcgrid:~/ejer2] 2021-03-03 Wednesday
$srun -pac -Aac -n1 -c12  --hint=nomultithread HelloOMP
(0:!!!Hello world!!!)(10:!!!Hello world!!!)(7:!!!Hello world!!!)(8:!!!Hello world!!!)(3:!!!Hello world!!!)(2:!!!Hello world!!!)(11:!!!Hello world!!!)(6:!!!Hello world!!!)(9:!!!Hello world!!!)(1:!!!Hello world!!!)(4:!!!Hello world!!!)(5:!!!Hello world!!!)[JuanCarlosRuizFernandez b3estudiante23@atcgrid:~/ejer2] 2021-03-03 Wednesday
$
```

(b) srun -pac –Aac -n1 -c24 HelloOMP

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

```
[JuanCarlosRuizFernandez b3estudiante23@atcgrid:~/ejer2] 2021-03-07 Sunday
$srun -pac -Aac -n1 -c24 HelloOMP
(1:!!!Hello world!!!)(4:!!!Hello world!!!)(9:!!!Hello world!!!)(23:!!!Hello world!!!)(18:!!!Hello world!!!)(8:!!!Hello world!!!)(13:!!!Hello world!!!)(15:!!!Hello world!!!)(0:!!!Hello world!!!)(22:!!!Hello world!!!)(11:!!!Hello world!!!)(14:!!!Hello world!!!)(16:!!!Hello world!!!)(20:!!!Hello world!!!)(6:!!!Hello world!!!)(3:!!!Hello world!!!)(10:!!!Hello world!!!)(21:!!!Hello world!!!)(17:!!!Hello world!!!)(2:!!!Hello world!!!)(5:!!!Hello world!!!)(7:!!!Hello world!!!)(19:!!!Hello world!!!)(12:!!!Hello world!!!)[JuanCarlosRuizFernandez b3estudiante23@atcgrid:~/ejer2] 2021-03-07 Sunday
$
```

(c) srun -n1 HelloOMP

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas. ¿Qué partición se está usando?

Usa la cola por defecto, la ac*

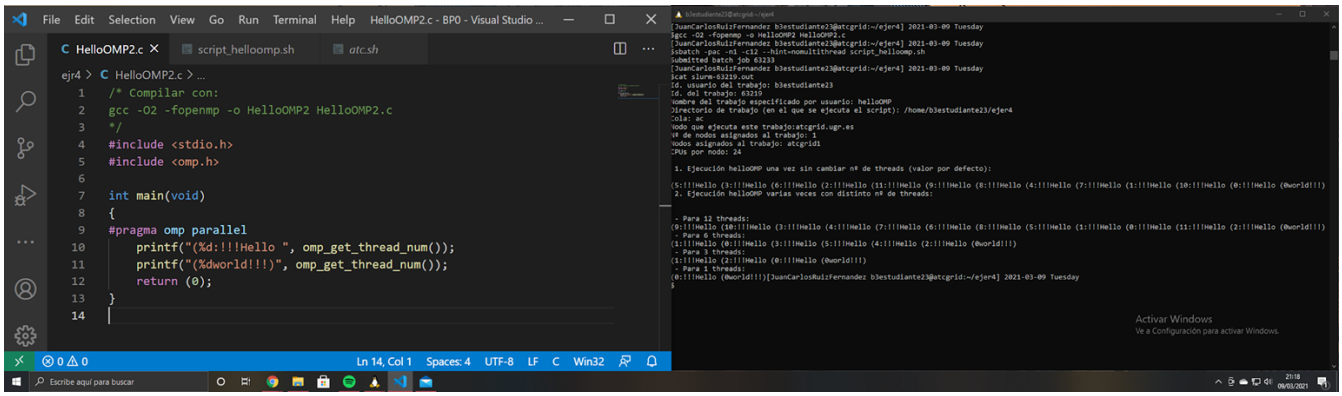
```
[JuanCarlosRuizFernandez b3estudiante23@atcgrid:~/ejer2] 2021-03-07 Sunday
$srun -n1 HelloOMP
(1:!!!Hello world!!!)(0:!!!Hello world!!!)[JuanCarlosRuizFernandez b3estudiante23@atcgrid:~/ejer2] 2021-03-07 Sunday
```

(d) ¿Qué orden srun usaría para que HelloOMP utilice todos los cores físicos de atcgrid4 (se debe imprimir un único mensaje desde cada uno de ellos)?

--cpus-per-task=32 --hint=nomultithread

4. Modificar en su PC HelloOMP.c para que se imprima "world" en un printf distinto al usado para "Hello". En ambos printf se debe imprimir el identificador del thread que escribe en pantalla. Nombrar al código resultante HelloOMP2.c. Compilar este nuevo código en el PC y ejecutarlo. Copiar el fichero ejecutable resultante al front-end de atcgrid (directorio ejer4). Ejecutar el código en un nodo de cómputo de atcgrid usando el *script* script_helloomp.sh del seminario (el nombre del ejecutable en el script debe ser HelloOMP2).

(a) Utilizar: sbatch -pac –n1 -c12 --hint=nomultithread script_helloomp.sh. Adjuntar capturas de pantalla que muestren el nuevo código, la compilación, el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.



(b) ¿Qué nodo de cómputo de atcgrid ha ejecutado el *script*? Explicar cómo ha obtenido esta información.

atcgrid 1. Se puede saber de dos formas:

- **La primera lo dice explícitamente** *Nodos asignados al trabajo: atcgrid 1*
- **La segunda nos fijamos en los núcleos por nodo que son 24 y corresponde a los atcgrid 1,2 y 3.**

La segunda no se puede saber exactamente cuál ha sido asignado pero da una idea si no se ve lo primero

NOTA: Utilizar siempre con sbatch las opciones –n1 y -c, --exclusive y, para usar cores físicos y no lógicos, no olvide incluir --hint=nomultithread. Utilizar siempre con srun, si lo usa fuera de un script, las opciones –n1 y -c y, para usar cores físicos y no lógicos, no olvide incluir --hint=nomultithread. Recordar que los srun dentro de un *script* heredan las opciones incluidas en el sbatch que se usa para enviar el *script* a la cola slurm. Se recomienda usar sbatch en lugar de srun para enviar trabajos a ejecutar a través slurm porque éste último deja bloqueada la ventana hasta que termina la ejecución, mientras que usando sbatch la ejecución se realiza en segundo plano.

Parte II. Resto de ejercicios

5. Generar en el PC 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). El comentario inicial del código muestra la orden para compilar (siempre hay que usar –O2 al compilar como se indica en las normas de prácticas). Incorporar volcados de pantalla que demuestren la compilación y la ejecución correcta del código en el PC (leer lo indicado al respecto en las normas de prácticas).

```
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP0/ejer5] 2021-03-10 Wednesday
$gcc  -O2  SumaVectores.c -o SumaVectores -lrt
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP0/ejer5] 2021-03-10 Wednesday
$gcc  -O2 -S SumaVectores.c -lrt
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP0/ejer5] 2021-03-10 Wednesday
$./SumaVectores
Faltan nº componentes del vector
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP0/ejer5] 2021-03-10 Wednesday
$./SumaVectores 7
Tiempo:0.000001000       / Tamaño Vectores:7
/ V1[0]+V2[0]=V3[0](0.700000+0.700000=1.400000) /
/ V1[1]+V2[1]=V3[1](0.800000+0.600000=1.400000) /
/ V1[2]+V2[2]=V3[2](0.900000+0.500000=1.400000) /
/ V1[3]+V2[3]=V3[3](1.000000+0.400000=1.400000) /
/ V1[4]+V2[4]=V3[4](1.100000+0.300000=1.400000) /
/ V1[5]+V2[5]=V3[5](1.200000+0.200000=1.400000) /
/ V1[6]+V2[6]=V3[6](1.300000+0.100000=1.400000) /
```

6. En el código del Listado 1 se utiliza la función clock_gettime() para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. El código se imprime la variable ncgt,
(a) ¿Qué contiene esta variable?

El tiempo de ejecución con nanosegundos de precisión

(b) ¿En qué estructura de datos devuelve clock_gettime() la información de tiempo (indicar el tipo de estructura de datos, describir la estructura de datos, e indicar los tipos de datos que usa)?

tv_sec -> número de segundos desde 1970

tv_nsec-> nanosegundos desde el segundo actual. varía según la resolución del reloj

(c) ¿Qué información devuelve exactamente la función clock_gettime()en la estructura de datos descrita en el apartado (b)? ¿qué representan los valores numéricos que devuelve?

0 si todo esta correcto -1 si algo ha fallado

7. Rellenar una tabla como la Tabla 1 en una hoja de cálculo con los tiempos de ejecución del código del Listado 1 para vectores locales, globales y dinámicos (se pueden obtener errores en tiempo de ejecución o de compilación, ver ejercicio 9). Obtener estos resultados usando *scripts* (partir del *script* que hay en el seminario). Debe haber una tabla para un nodo de cómputo de atcgrid con procesador Intel Xeon E5645 y otra para su PC en la hoja de cálculo. En la columna "Bytes de un vector" hay que poner el total de bytes reservado para un vector. (NOTA: Se recomienda usar en la hoja de cálculo el mismo separador para decimales que usan los códigos al imprimir –"."–. Este separador se puede modificar en la hoja de cálculo.)

Tabla 1 . Copiar la tabla de la hoja de cálculo utilizada

PC

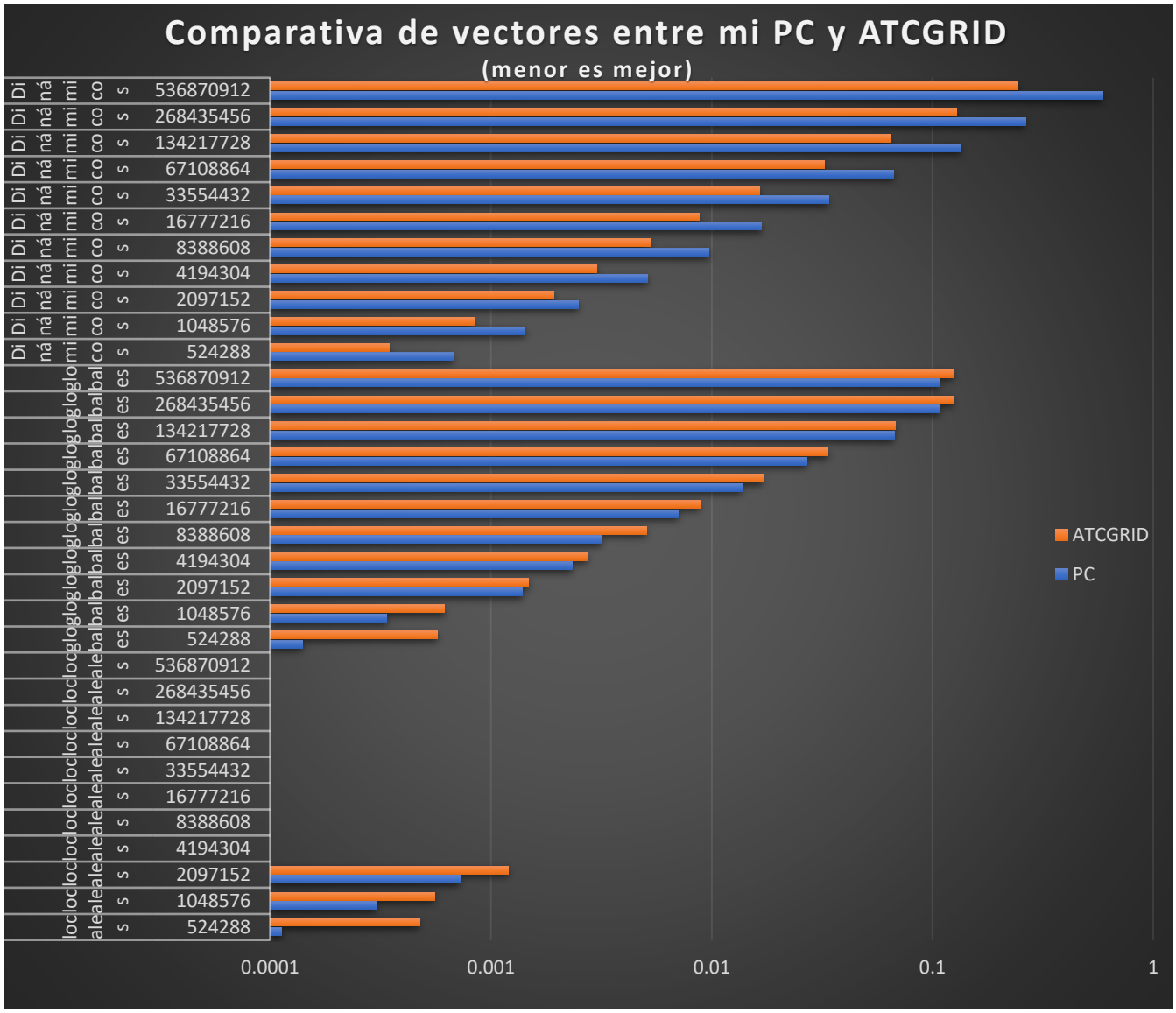
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000123400	0.000163000	0.000633600
131072	1048576	0.000325000	0.000323000	0.001239000
262144	2097152	0.000601400	0.001168300	0.002465700
524288	4194304		0.001943500	0.005522400
1048576	8388608		0.003320700	0.009331100
2097152	16777216		0.005880000	0.016488700
4194304	33554432		0.011931100	0.036050700
8388608	67108864		0.025007800	0.066516800
16777216	134217728		0.052654300	0.135495500
33554432	268435456		0.105834000	0.267600600
67108864	536870912		0.104050600	0.539330400

ATCGRID

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. Dinámicos
65536	524288	0.000474276	0.000572417	0.000346305
131072	1048576	0.000553870	0.000616945	0.000835522
262144	2097152	0.001201295	0.001482553	0.001934085
524288	4194304		0.002742395	0.003015212
1048576	8388608		0.005060687	0.005286461
2097152	16777216		0.008847078	0.008782269
4194304	33554432		0.017029911	0.016532508
8388608	67108864		0.033634546	0.032552789
16777216	134217728		0.068044325	0.064067361
33554432	268435456		0.124657298	0.128347610
67108864	536870912		0.124655696	0.244764913

8. Con ayuda de la hoja de cálculo representar **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 (por tanto, los valores de la segunda columna de la tabla, que están en escala logarítmica, deben estar en el eje x). Utilizar escala logarítmica en el eje de ordenadas (eje y). ¿Hay diferencias en los tiempos de ejecución?

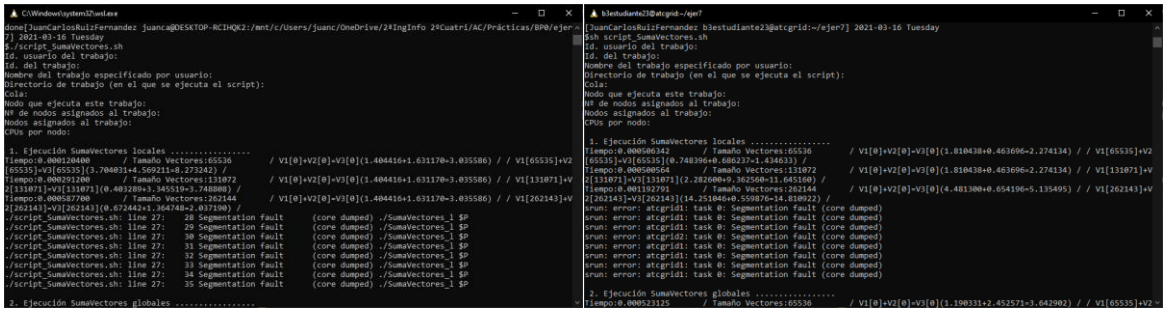
Si, existe una diferencia que se invierte. En los vectores locales el ATCGRID tiene tiempo de ejecuciones mayores, para globales sigue existiendo esa diferencia pero menos significativa y para los dinámicos se invierte con una diferencia bastante mayor siendo mi PC con tiempos de ejecución mucho mayores.



9. Contestar a las siguientes preguntas:

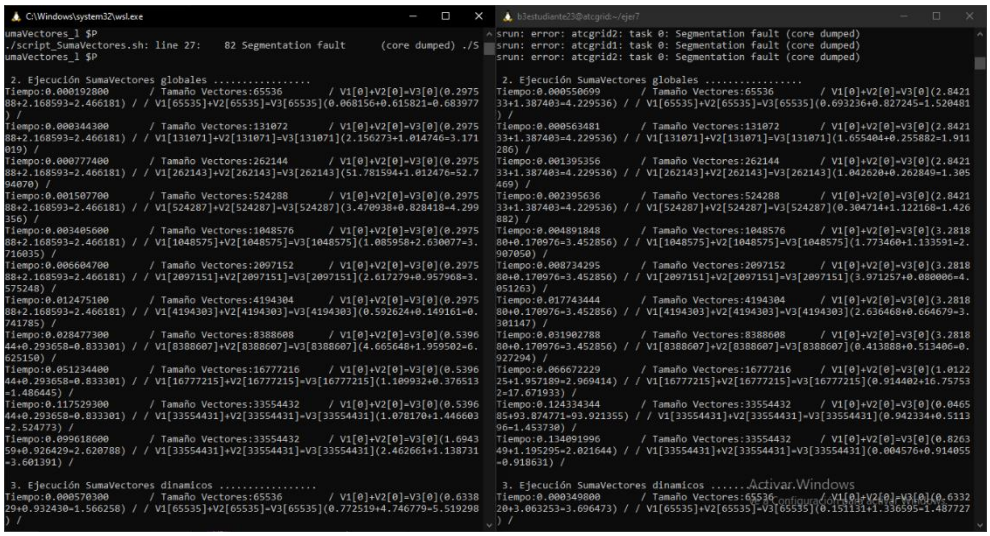
(a) Cuando se usan vectores locales, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

Si, debido al tamaño de la pila el cual ya se sobrepasa cuando usamos un vector de 4194304 bytes.



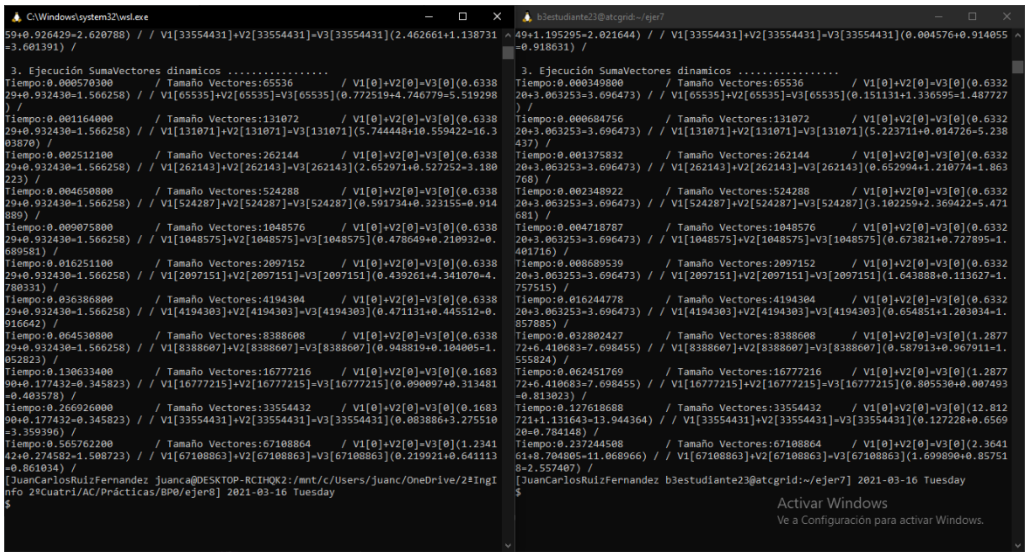
(b) Cuando se usan vectores globales, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

Para las variables globales se incluyen en el código, no en la pila. Por ello creo que no rebasa un límite por tamaño ya que no lo hay.

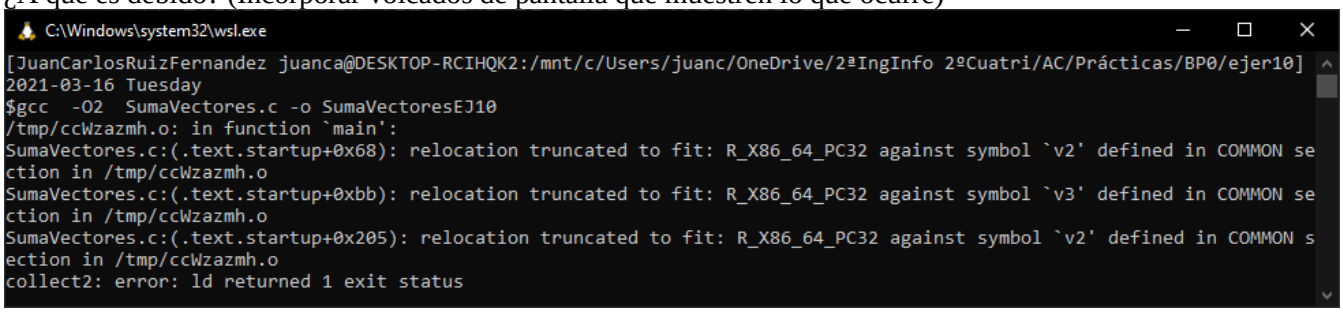


(c) Cuando se usan vectores dinámicos, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

De igual manera no da fallo y por el mismo motivo. El heap crece hacia arriba.



10. **(a)** ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.
Unsigned int = 2^32-1=4294967295
(b) Modificar el código fuente C (en el PC) 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 y generar el ejecutable. ¿Qué ocurre? ¿A qué es debido? (Incorporar volcados de pantalla que muestren lo que ocurre)



Los vectores ahora ocuparían (8bytes(double)*2^32)/1024^3=¡32GB! Según he leído en varias paginas es porque no debería superar los 2GB. Se soluciona compilando con otra opción (-mcmodel large) que no se si para nuestro caso funcionaria ya que hablan de librerias.

Entrega del trabajo

Leer lo indicado en las normas de prácticas sobre la entrega del trabajo del bloque práctico en SWAD.

Listado 1. Código C que suma dos vectores. Se generan aleatoriamente las componentes para vectores de tamaño mayor que 8 y se imprimen todas las componentes para vectores menores que 10.

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

 Para compilar usar (-lrt: real time library, no todas las versiones de gcc necesitan que se incluya -lrt):
      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(), rand(), srand(), malloc() y free()
#include <stdio.h> // biblioteca donde se encuentra la función printf()
#include <time.h>      // biblioteca donde se encuentra la función clock_gettime()

//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
 if (N < 9)
   for (i = 0; i < N; i++)
   {
     v1[i] = N * 0.1 + i * 0.1;
     v2[i] = N * 0.1 - i * 0.1;
   }
 else
 {
   srand(time(0));
   for (i = 0; i < N; i++)
   {
     v1[i] = rand()/ ((double) rand());
     v2[i] = rand()/ ((double) rand()); //printf("%d:%f,%f/",i,v1[i],v2[i]);
   }
 }

  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
```

```
  if (N<10) {
  printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\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]);


  #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;
}
```