2º curso / 2º cuatr.

Grados Ingeniería Informática

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos):
Grupo de prácticas y profesor de prácticas:

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 que con `sbatch` se tenga en cuenta `---hint=nomultithread` se debe usar `srun` dentro del script delante del ejecutable.
- 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)
- Se recomienda escribir las órdenes directamente en la ventana de comandos (*shell*) en lugar de usar copy/paste.

1. Ejecutar `lscpu` en el PC, en atcgrid4 (usar en este caso `-p ac4`) y en uno de los restantes nodos de cómputo (atcgrid1, atcgrid2 o atcgrid3, usar en este caso `-p ac`).

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

   RESPUESTA:

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

   RESPUESTA:

2. Compilar y ejecutar en el PC el código `HelloOMP.c` del seminario.

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

   RESPUESTA:

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

   RESPUESTA:

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.

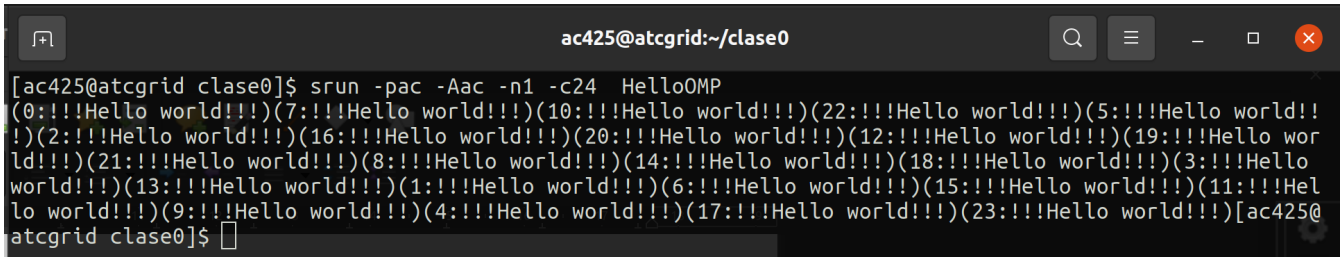
RESPUESTA:

```
[ac425@atcgrid clase0]$ srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithre
ad HelloOMP
(8:!!!Hello world!!!)(4:!!!Hello world!!!)(1:!!!Hello world!!!)(6:!!!Hello world!!!)(9:!!!Hello world!!!)
(11:!!!Hello world!!!)(10:!!!Hello world!!!)(0:!!!Hello world!!!)(5:!!!Hello world!!!)(3:!!!Hello world!!
!)(7:!!!Hello world!!!)(2:!!!Hello world!!!)[ac425@atcgrid clase0]$
```

(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.

RESPUESTA:



(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 (cola) se está usando?

RESPUESTA:

(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)?

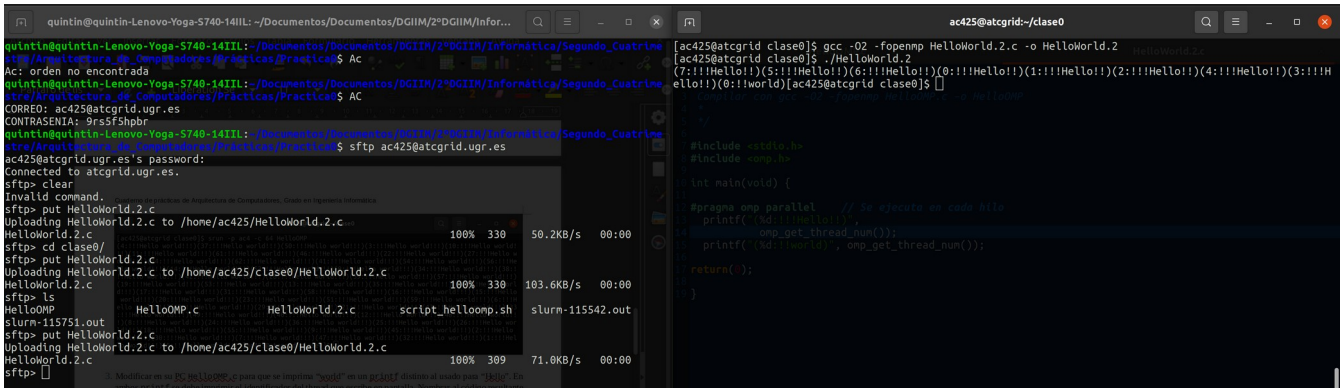
Srun -p ac -c 12 HelloMp

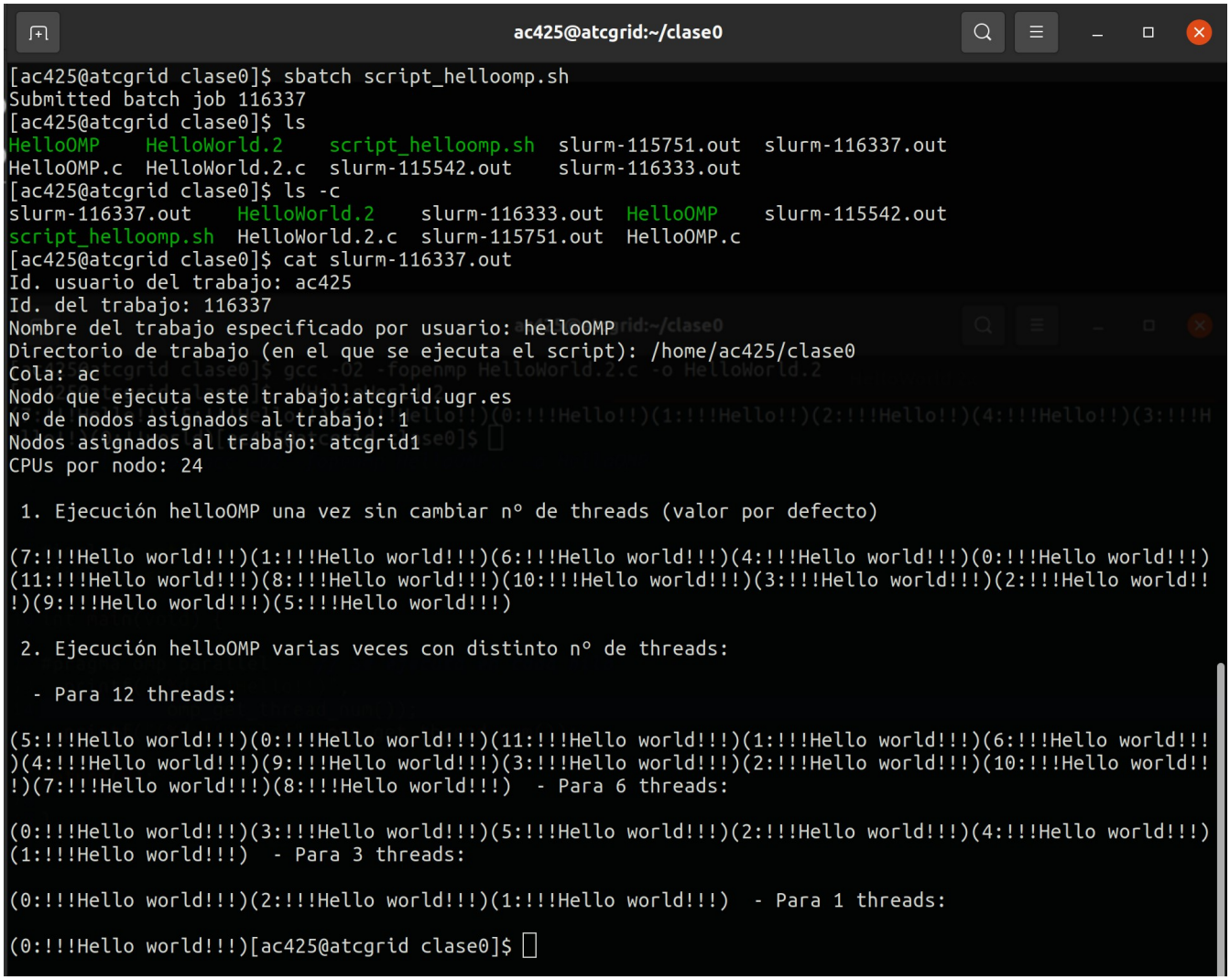
4. Modificar en su PC `HelloOMP.c` para que se imprima “world” en un `printf` distinto al usado para “Hello”. En ambos `printf` se debe imprimir el identificador del thread que escribe en pantalla. Nombrar al código resultante `HelloOMP2.c`. Compilar este nuevo código en el PC y ejecutarlo. Copiar el 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 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.

RESPUESTA:

```c
/**
 *
 Compilar con gcc -O2 -fopenmp HelloOMP.c -o HelloOMP
 *
 */

#include <stdio.h>
#include <omp.h>

int main(void) {

#pragma omp parallel    // Se ejecuta en cada hilo
  printf("(%d:!!!Hello!!)",
          omp_get_thread_num());
  printf("(%d:!!world)", omp_get_thread_num());

return(0);

}
```





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

RESPUESTA:

El script lo está ejecutando el nodo **atcgrid1.** Se ve en la captura anterior.

(c) ¿Qué órdenes para el gestor de colas slurm incluye el *script*? Explicar cómo ha obtenido esta información.

RESPUESTA:

(d) Haga los cambios necesarios en el *script* para que se utilice atcgrid4. Comentar los cambios realizados y los motivos por los que se han hecho.

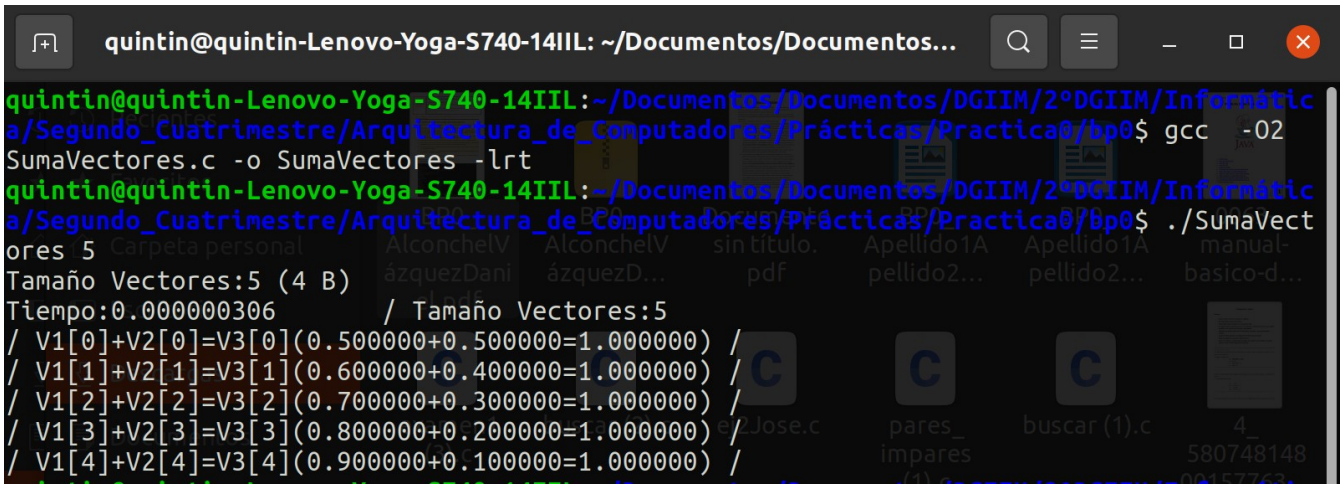
RESPUESTA:

NOTA: Utilizar siempre con `sbatch` las opciones `-n1` y `-c`, `--exclusive` y, para usar cores físicos y no lógicos, no olvidar 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 `SumaVectores.c` 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`). 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).

RESPUESTA:



6. En el código `SumaVectores.c` 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?

RESPUESTA: Contiene la diferencia de tiempo entre que empieza el cálculo de la suma y acaba el mismo

(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)?

RESPUESTA: clock_gettime() es una función que devuelve el tiempo del reloj especificado por la variable clock_id. Lo almacena en un struct llamado timespec, que contiene dos variables, una llamada tv_sec, que es una variable tipo time_t, que almacena los segundos, y otra llamada tv_nsec, que es de tipo long y almacena los nanosegundos.

(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?

RESPUESTA: Devuelve la información del reloj pasado en el primer parámetro y la almacena en el struct pasado como segundo parámetro. En nuestro caso, le pasamos CLOCK_REALTIME, que es el reloj del sistema.

7. Rellenar una tabla como la Tabla 1 en una hoja de cálculo con los tiempos de ejecución del código SumaVectores.c para vectores locales, globales y dinámicos (se pueden obtener errores en tiempo de ejecución o

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	4			
131072				
262144				
524288				
1048576				
2097152				
4194304				
8388608				
16777216				
33554432				
67108864				

ue hay en el E5645 y otra tes reservado ales que usan

T

1. 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?

   RESPUESTA:

2. 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)

   RESPUESTA:

   (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)

   RESPUESTA:

   (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)

   RESPUESTA:

3. **(a)** ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.
   RESPUESTA:
   (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)
   RESPUESTA:

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(), 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
 {
   srand48(time(0));
   for (i = 0; i < N; i++)
   {
     v1[i] = drand48();
     v2[i] = drand48(); //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;
}
```