

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Jose Luis Martínez Ortiz

Grupo de prácticas: C3

Fecha de entrega: 1 – 03 – 2016

Fecha evaluación en clase: 9 – 03 – 2016

Ejercicios basados en los ejemplos del seminario práctico

1. En el primer ejemplo de ejecución en atcgrid usando TORQUE se ejecuta el ejemplo HelloOMP.c usando la siguiente orden: `echo 'hello/HelloOMP' | qsub -q ac`. El resultado de la ejecución de este código en atcgrid se puede ver en el seminario. Conteste a las siguientes preguntas:

- a. ¿Para qué se usa en qsub la opción -q?

RESPUESTA: Para enviarlo a la cola que se indica a continuación

- b. ¿Cómo sabe el usuario que ha terminado la ejecución en atcgrid?

RESPUESTA: Porque con el comando "qstat" no muestra nada y me ha generado 2 ficheros STDIN.oXXX y STDIN.eXXX, donde XXX es el PID del proceso.

- c. ¿Cómo puede saber el usuario si ha habido algún error en la ejecución?

RESPUESTA: Si el tamaño del fichero STDIN.eXXX es mayor que cero.

- d. ¿Cómo ve el usuario el resultado de la ejecución?

RESPUESTA: En el fichero STDIN.oXXX

- e. ¿Por qué en el resultado de la ejecución aparecen 24 saludos "!!!Hello World!!!"?

RESPUESTA: Por que sale un hello World por cada thread del procesador, el procesador es Multicore y multithread, siendo de 6 cores con 2 thread por core * 2 Hyperthreading haciendo un total de 24 threads.

2. En el segundo ejemplo de ejecución en atcgrid usando TORQUE el script `script_helloomp.sh` usando la siguiente orden: `qsub script_helloomp.sh`. El script ejecuta varias veces el ejecutable del código HelloOMP.c. El resultado de la ejecución de este código en atcgrid se puede ver en el seminario. Conteste a las siguientes preguntas:

- a. ¿Por qué no acompaña a al orden qsub la opción -q en este caso?

RESPUESTA: Por que no hace falta indicarle el destino, al estar indicado en el script

- b. ¿Cuántas veces ejecuta el script el ejecutable HelloOMP en atcgrid? ¿Por qué lo ejecuta ese número de veces?

RESPUESTA: Lo ejecuta 4 veces por que empieza en 12 thread y cada vez que itera divide a la mitad el numero de thread que lanza hasta llegar a 0. Entonces tenemos 12 – 6 – 3 – 1

- c. ¿Cuántos saludos "!!!Hello World!!!" se imprimen en cada ejecución? (indique el número exacto) ¿Por qué se imprime ese número?

RESPUESTA: 1º ejecución doce veces

2º ejecución seis veces

3º ejecución tres veces.

4º ejecución una vez.

Se imprime ese numero por el numero de thread que ejecuta el script

Adjunto salida del script helloomp.o25250

3. Realizar las siguientes modificaciones en el script “!!!Hello World!!!”:

- Eliminar la variable de entorno \$PBS_O_WORKDIR en el punto en el que aparece.
- Añadir lo necesario para que, cuando se ejecute el script, se imprima la variable de entorno \$PBS_O_WORKDIR.

Ejecutar el script con estas modificaciones. ¿Qué resultados de ejecución se obtienen en este caso? Incorporar en el cuaderno de trabajo volcados de pantalla que muestren estos resultados.

RESPUESTA: Al eliminar la variable \$PBS_O_WORKDIR, que contiene la ruta de trabajo actual, no encuentra el ejecutable HelloOMP y da un error.

Adjunto salidas : helloomp.e25252 y helloomp.o25252

Captura de pantalla : [ejecución](#) [descarga](#).

Resto de ejercicios

4. Incorporar en el fichero .zip que se entregará al profesor el fichero /proc/cpuinfo de alguno de los nodos de atcgrid (atcgrid1, atcgrid2, atcgrid3), y del PC del aula de prácticas o de su PC. Indique qué ha hecho para obtener el contenido de /proc/cpuinfo en atcgrid.

RESPUESTA: [Captura de pantalla](#)

Teniendo en cuenta el contenido de cpuinfo conteste a las siguientes preguntas (justifique las respuestas):

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

RESPUESTA: 4 cores fisicos y 8 lógicos. Archivo cpuinfo: cpuinfo_miPC

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

RESPUESTA: 6 cores fisicos y 12 lógicos. Archivo cpuinfo: STDIN.o25258

5. 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 sumar los dos vectores.

La función `clock_gettime(clockid_t clk_id, struct timespec *tp)` asigna a una variable con estructura “timespec” el tiempo exacto del reloj pasado como parametro “clk_id”.

- 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++
Forma de declarar memoria dinámica.	Se realiza con “malloc” e indicando en sus parametros el tamaño en bytes del espacio a reservar.	Se realiza con la orden “new” y el tipo de dato que guardaras.
La forma de mostrar el resultado.	Se utiliza “printf” con el formato correspondiente.	Se utiliza cout.
La forma de liberar memoria dinámica	Se hace con la función “free”	Se hace con la función “delete”
La librería para mostrar resultados	Se incluye la librería “stdio”	Se utiliza la librería “std”, con using namespace std;

6. 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). Ejecutar el código ejecutable resultante en atcgrid usando el la cola TORQUE. Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid.

RESPUESTA: Archivo de salida de TORQUE : [STDIN.o26247](#)

Captura de pantalla: [Ejecución y salida.](#)

7. 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 local para los mismos tamaños. ¿Se obtiene error para alguno de los tamaños? En caso afirmativo, ¿a qué se debe este error?

RESPUESTA: Si se obtiene un error en la pila, al ser un vector local cuando el nº de elementos es de 2^{19} se llega al tope de pila y se intenta acceder a otro espacio de direcciones al que no tiene permiso provocando a su vez una violación del segmento. Esto ocurre tanto en Atcgrid como en mi pc.

Ficheros de salida: SumaVectoresC_vlocales.o27445 y SumaVectoresC_vlocales.e27445

8. 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 local. ¿Se obtiene error usando vectores globales o dinámicos? ¿A qué cree que es debido?

RESPUESTA: En este caso no se obtiene error en ningún caso. Esto es debido a que tanto los vectores Globales como los dinámicos se guardan en un espacio de memoria variable y con posibilidad de swapping con la memoria principal por lo que no se llena el espacio ni con un vector de tamaño 2^{26} .

Ficheros de salida: SumaVectoresC_vdinamicos.o27487

SumaVectoresC_vGlobales.o27491

Capturas de pantalla: [Ejecución de Listado 3](#)

9. Rellenar una tabla como la Tabla 1 para atcgrid y otra para el PC local 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 para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (eje x). Utilice escala logarítmica en el eje de ordenadas (eje y) en todas las gráficas. ¿Hay diferencias en los tiempos de ejecución con vectores locales, globales y dinámicos?

RESPUESTA:

Tabla 1 . Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos

En AtcGrid				
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	65536	0.000361556	0.000389569	0.000388597
131072	131072	0.000768218	0.000808325	0.000786079
262144	262144	0.001428196	0.001131200	0.001546834
524288	524288	0	0.003157952	0.002329540
1048576	1048576	0	0.004975717	0.005027259
2097152	2097152	0	0.010106829	0.010143742
4194304	4194304	0	0.019821421	0.019925229
8388608	8388608	0	0.039323995	0.040310535
16777216	16777216	0	0.078393990	0.080295302
33554432	33554432	0	0.156753360	0.161043261

En mi PC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	65536	0.000438281	0.000416901	0.000439666
131072	131072	0.000863073	0.000816674	0.000639175
262144	262144	0.001716110	0.001552995	0.001302192
524288	524288	0	0.002841389	0.002269812
1048576	1048576	0	0.004059699	0.003932690
2097152	2097152	0	0.006770054	0.006962238
4194304	4194304	0	0.011151692	0.011170100
8388608	8388608	0	0.018281742	0.018697340
16777216	16777216	0	0.031842276	0.032204274
33554432	33554432	0	0.063187880	0.063994517

10. 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: Se está intentando enlazar un proyecto con unas direcciones relativas demasiado largas para el filesystem, y las intenta truncar lo que provoca que se enlacen de forma incorrecta y provocando así el error.

Captura de imagen: [Error de compilación.](#)

Anexo I : Capturas de pantalla.

Ejecución del script “script_helloomp.sh” en el servidor.

```

C3estudiante13@atcgrid:~/hello
-rw-r--r-- 1 C3estudiante13 193 ene 16 2015 .bash_profile
-rw-r--r-- 1 C3estudiante13 231 ene 16 2015 .bashrc
drwxrwxr-x 2 C3estudiante13 4096 mar 1 10:39 hello
drwxr-xr-x 3 C3estudiante13 4096 feb 24 15:43 .local
drwxr-xr-x 4 C3estudiante13 4096 ene 30 2015 .mozilla
drwxr-xr-x 2 C3estudiante13 4096 feb 5 2015 .ssh
-rw----- 1 C3estudiante13 0 mar 1 10:40 STDIN.e25246
-rw----- 1 C3estudiante13 518 mar 1 10:40 STDIN.o25246
-rw----- 1 C3estudiante13 180 mar 1 10:33 .Xauthority
[C3estudiante13@atcgrid ~]$ ls
hello script_helloomp.sh STDIN.e25246 STDIN.o25246
[C3estudiante13@atcgrid ~]$ cd hello/
[C3estudiante13@atcgrid hello]$ mv ../script_helloomp.sh .
[C3estudiante13@atcgrid hello]$ qsub script_helloomp.sh
25250.atcgrid
[C3estudiante13@atcgrid hello]$ ls -lag
total 28
drwxrwxr-x 2 C3estudiante13 4096 mar 1 10:51 .
drwx----- 6 C3estudiante13 4096 mar 1 10:49 ..
-rwxrwxr-x 1 C3estudiante13 8783 mar 1 10:38 HelloOMP
-rw----- 1 C3estudiante13 0 mar 1 10:50 helloomp.e25250
-rw----- 1 C3estudiante13 860 mar 1 10:50 helloomp.o25250
-rw-rw-r-- 1 C3estudiante13 814 mar 1 10:44 script_helloomp.sh
[C3estudiante13@atcgrid hello]$

```

[Volver](#)

Descarga de la salida del script “script_helloomp.sh”.

```

jose@jose-K55VM: ~
sftp> put HelloOMP
Uploading HelloOMP to /home/C3estudiante13/HelloOMP
HelloOMP 100% 8783 8.6KB/s 00:00
sftp> get ST
STDIN.e25246 STDIN.o25246
sftp> get STDIN.o25246
Fetching /home/C3estudiante13/STDIN.o25246 to STDIN.o25246
/home/C3estudiante13/STDIN.o25246 100% 518 0.5KB/s 00:00
sftp> put script_helloomp.sh
Uploading script_helloomp.sh to /home/C3estudiante13/script_helloomp.sh
script_helloomp.sh 100% 814 0.8KB/s 00:00
sftp> lpwd
Local working directory: /home/jose/HelloOMP
sftp> ls
STDIN.e25246 STDIN.o25246 hello
sftp> cd hello/
sftp> ls
HelloOMP helloomp.e25250 helloomp.o25250 script_helloomp.sh
sftp> get he
helloomp.e25250 helloomp.o25250
sftp> get helloomp.o25250
Fetching /home/C3estudiante13/hello/helloomp.o25250 to helloomp.o25250
/home/C3estudiante13/hello/helloomp.o25250 100% 860 0.8KB/s 00:00
sftp>

```

[Volver](#)

Como obtener la información del procesador del servidor:

```
SSH
[C3estudiante13@atcgrid hello]$ echo 'cat /proc/cpuinfo' | qsub -q ac
25065.atcgrid
[C3estudiante13@atcgrid hello]$ ls
STDIN.e25064 STDIN.e25065 STDIN.o25064 STDIN.o25065
[C3estudiante13@atcgrid hello]$
```

[Volver.](#)

Ejecución en TORQUE y se muestran los resultados

```
C3estudiante13@atcgrid:~
[C3estudiante13@atcgrid ~]$ rm hello/SumaVectores
[C3estudiante13@atcgrid ~]$ mv SumaVectores hello/
[C3estudiante13@atcgrid ~]$ echo 'hello/SumaVectores 500' | qsub -q ac
26247.atcgrid
[C3estudiante13@atcgrid ~]$ ls -lag
total 64
drwx----- 6 C3estudiante13 4096 mar  2 17:06 .
drwxr-xr-x. 499 root        20480 mar  9  2015 ..
-rw----- 1 C3estudiante13 1392 mar  1 11:42 .bash_history
-rw-r--r-- 1 C3estudiante13  18 ene 16  2015 .bash_logout
-rw-r--r-- 1 C3estudiante13 193 ene 16  2015 .bash_profile
-rw-r--r-- 1 C3estudiante13 231 ene 16  2015 .bashrc
drwxrwxr-x 2 C3estudiante13 4096 mar  2 17:06 hello
drwxr-xr-x 3 C3estudiante13 4096 feb 24 15:43 .local
drwxr-xr-x 4 C3estudiante13 4096 ene 30  2015 .mozilla
drwxr-xr-x 2 C3estudiante13 4096 feb  5  2015 .ssh
-rw----- 1 C3estudiante13    0 mar  2 17:04 STDIN.e26247
-rw----- 1 C3estudiante13  160 mar  2 17:04 STDIN.o26247
-rw----- 1 C3estudiante13  240 mar  2 17:01 .Xauthority
[C3estudiante13@atcgrid ~]$ cat STDIN.o26247
Tiempo(seg.):0.000001085 / Tamaño Vectores:500
V1[0]+V2[0]=V3[0](50.000000+50.000000=100.000000)
V1[499]+V2[499]=V3[499](99.900000+0.100000=100.000000)
[C3estudiante13@atcgrid ~]$
```

[Volver.](#)

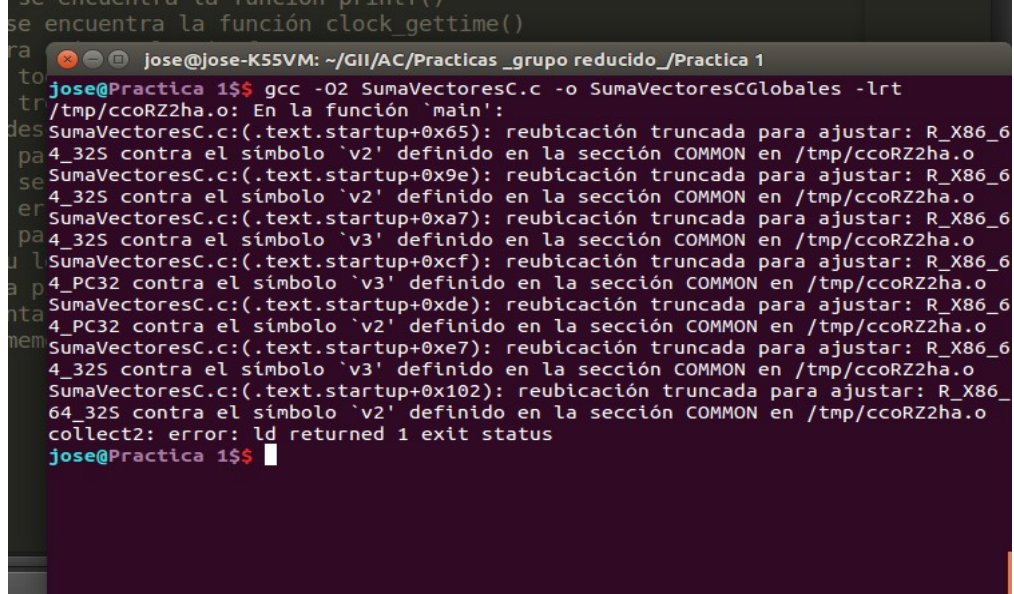
Ejecución de Listado 3 con vectores Globales y Dinámicos

```
jose@Jose-K55VM: ~/GII/AC/Practicas_grupo reducido_/Practica 1
jose@Practica 1$ ./script_mi_pc.sh
Tiempo(seg.):0.000439666 / Tamaño Vectores:65536
V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000)
V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000)
Tiempo(seg.):0.000639175 / 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.):0.001302192 / 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.):0.002269812 / Tamaño Vectores:524288
V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000)
V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000)
Tiempo(seg.):0.003932690 / 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.):0.006962238 / 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.):0.011170100 / 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.):0.018697340 / 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.):0.032204274 / Tamaño Vectores:16777216
V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000)
V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000)
Tiempo(seg.):0.063994517 / Tamaño Vectores:33554432
V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000)
V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000)
Tiempo(seg.):0.063778620 / Tamaño Vectores:33554432
V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000)
V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000)
jose@Practica 1$
```

```
jose@Jose-K55VM: ~/GII/AC/Practicas_grupo reducido_/Practica 1
jose@Practica 1$ ./script_mi_pc.sh
Tiempo(seg.):0.000439666 / Tamaño Vectores:65536
V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000)
V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000)
Tiempo(seg.):0.000639175 / 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.):0.001302192 / 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.):0.002269812 / Tamaño Vectores:524288
V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000)
V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000)
Tiempo(seg.):0.003932690 / 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.):0.006962238 / 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.):0.011170100 / 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.):0.018697340 / 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.):0.032204274 / Tamaño Vectores:16777216
V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000)
V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000)
Tiempo(seg.):0.063994517 / Tamaño Vectores:33554432
V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000)
V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000)
Tiempo(seg.):0.063778620 / Tamaño Vectores:33554432
V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000)
V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000)
jose@Practica 1$
```

[Volver.](#)

Error compilando el ejercicio 10.

A screenshot of a terminal window with a dark background. The window title is "jose@jose-K55VM: ~/GII/AC/Practicas_grupo reducido_/Practica 1". The user has entered the command "gcc -O2 SumaVectoresC.c -o SumaVectoresCGlobales -lrt". The output shows several linker errors: "En la función 'main':", "SumaVectoresC.c:(.text.startup+0x65): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v2' definido en la sección COMMON en /tmp/ccorZ2ha.o", "SumaVectoresC.c:(.text.startup+0x9e): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v2' definido en la sección COMMON en /tmp/ccorZ2ha.o", "SumaVectoresC.c:(.text.startup+0xa7): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v3' definido en la sección COMMON en /tmp/ccorZ2ha.o", "SumaVectoresC.c:(.text.startup+0xc7): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo 'v3' definido en la sección COMMON en /tmp/ccorZ2ha.o", "SumaVectoresC.c:(.text.startup+0xde): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo 'v2' definido en la sección COMMON en /tmp/ccorZ2ha.o", "SumaVectoresC.c:(.text.startup+0xe7): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v3' definido en la sección COMMON en /tmp/ccorZ2ha.o", "SumaVectoresC.c:(.text.startup+0x102): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v2' definido en la sección COMMON en /tmp/ccorZ2ha.o", and finally "collect2: error: ld returned 1 exit status". The prompt "jose@Practica 1\$" is visible at the bottom.

```
jose@jose-K55VM: ~/GII/AC/Practicas_grupo reducido_/Practica 1
jose@Practica 1$ gcc -O2 SumaVectoresC.c -o SumaVectoresCGlobales -lrt
/tmp/ccorZ2ha.o: En la función 'main':
SumaVectoresC.c:(.text.startup+0x65): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v2' definido en la sección COMMON en /tmp/ccorZ2ha.o
SumaVectoresC.c:(.text.startup+0x9e): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v2' definido en la sección COMMON en /tmp/ccorZ2ha.o
SumaVectoresC.c:(.text.startup+0xa7): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v3' definido en la sección COMMON en /tmp/ccorZ2ha.o
SumaVectoresC.c:(.text.startup+0xc7): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo 'v3' definido en la sección COMMON en /tmp/ccorZ2ha.o
SumaVectoresC.c:(.text.startup+0xde): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo 'v2' definido en la sección COMMON en /tmp/ccorZ2ha.o
SumaVectoresC.c:(.text.startup+0xe7): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v3' definido en la sección COMMON en /tmp/ccorZ2ha.o
SumaVectoresC.c:(.text.startup+0x102): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo 'v2' definido en la sección COMMON en /tmp/ccorZ2ha.o
collect2: error: ld returned 1 exit status
jose@Practica 1$
```

[Volver.](#)