2º curso / 2º cuatr.

Grados Ingeniería Informática

Arquitectura de Computadores (AC)

1.1.1 Cuaderno de prácticas.
1.1.2 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`).

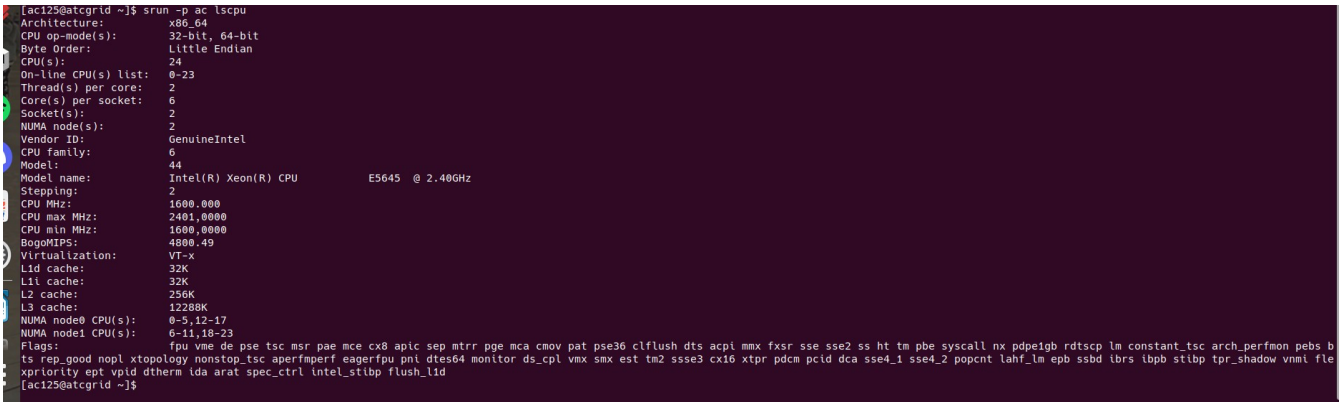
(a) Mostrar con capturas de pantalla el resultado de estas ejecuciones.

RESPUESTA:

srun -p ac4 lscpu esto es para atcgrid4

```
[ac125@atcgrid ~]$ srun -p ac4 lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                64
On-line CPU(s) list:   0-63
Thread(s) per core:    2
Core(s) per socket:    16
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
Stepping:              7
CPU MHz:               1154.333
CPU max MHz:           3200,0000
CPU min MHz:           800,0000
BogoMIPS:              4200.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              22528K
NUMA node0 CPU(s):     0-15,32-47
NUMA node1 CPU(s):     16-31,48-63
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pe
bs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_tim
er aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb cat_l3 cdp_l3 invpcid_single intel_ppin intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust
 bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_loca
l dtherm ida arat pln pts pku ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities
[ac125@atcgrid ~]$
```

srun -p ac lscpu para nodos restantes



(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:

Los cores fisicos de atcgrid4 son 32 y los logicos son 64

Para atcgrid1,2,3 los cores fisicos son 12 y los logicos son 24

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:

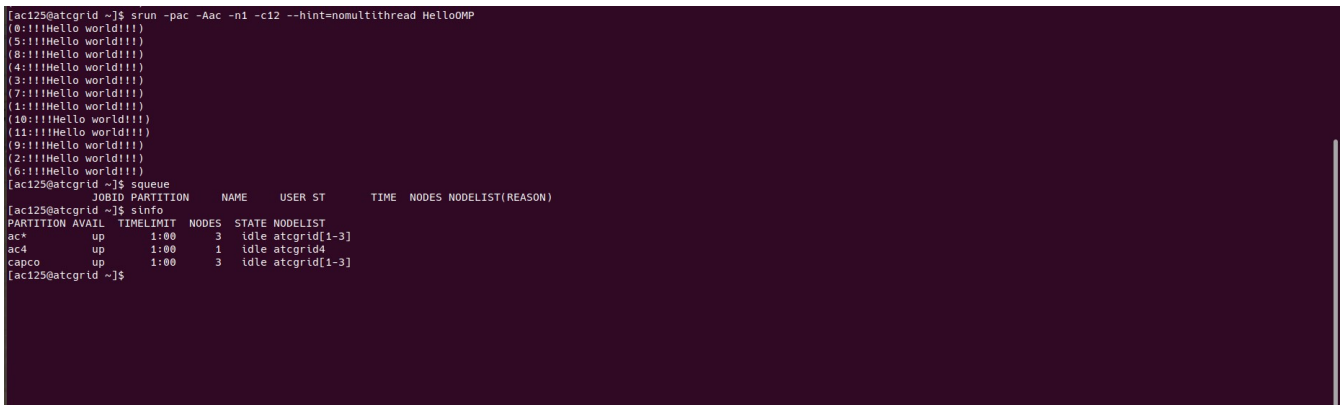
Se imprimen un total de 12 helloworld porque le estoy diciendo que se haga una tarea en 12 cores distintos y que ademas solamente se usen cores fisicos, por lo tanto deberian salir 12 hola mundos

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:

(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?

```
[ac125@atcgrid ~]$ srun -n1 HelloOMP
(0:!!!Hello world!!!)
(1:!!!Hello world!!!)
[ac125@atcgrid ~]$
```

Se deberia estar usando ac ya que es la predeterminada es decir uno de los nodos atcgrid1,2 o 3

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 -pac -Aac -n1 -c<n_cores_fisicos> --hint=nomultithread HelloOMP

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:

```
[ac125@atcgrid ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
ac*          up       1:00      3   idle atcgrid[1-3]
ac4          up       1:00      1   idle atcgrid4
capco        up       1:00      3   idle atcgrid[1-3]
[ac125@atcgrid ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
[ac125@atcgrid ~]$ cat helloomp.sh
#!/bin/bash
#Órdenes para el Gestor de carga de trabajo (no intercalar instrucciones del script):
#1. Asigna al trabajo un nombre
#SBATCH --job-name=helloOMP
#2. Asignar el trabajo a una cola (partición)
#SBATCH --partition=ac
#3. Asignar el trabajo a un account
#SBATCH --account=ac
#4. Para que el trabajo no comparta recursos
#SBATCH --exclusive
#5. Para que se genere un único proceso del SO que pueda usar un máximo de 12 núcleos
#SBATCH --ntasks 1 --cpus-per-task 12
#Obtener información de las variables del entorno del Gestor de carga de trabajo:
echo "Id. usuario del trabajo: $SLURM_JOB_USER"
echo "Id. del trabajo: $SLURM_JOBID"
echo "Nombre del trabajo especificado por usuario: $SLURM_JOB_NAME"
echo "Directorio de trabajo (en el que se ejecuta el script): $SLURM_SUBMIT_DIR"
echo "Cola: $SLURM_JOB_PARTITION"
echo "Nodo que ejecuta este trabajo:$SLURM_SUBMIT_HOST"
echo "Nº de nodos asignados al trabajo: $SLURM_JOB_NUM_NODES"
echo "Nodos asignados al trabajo: $SLURM_JOB_NODELIST"
echo "Nº CPUs disponibles para el trabajo en el nodo: $SLURM_JOB_CPUS_PER_NODE"
#Instrucciones del script para ejecutar código:
echo -e "\n 1. Ejecución helloOMP una vez sin cambiar nº de threads (valor por defecto):\n"
srun ./HelloOMP2
echo -e "\n 2. Ejecución helloOMP varias veces con distinto nº de threads:\n"
for ((P=12;P>0;P=P/2))
do
export OMP_NUM_THREADS=$P
echo -e "\n - Para $P threads:"
srun  --hint nomultithread ./HelloOMP2
done

[ac125@atcgrid ~]$ ./helloomp.sh
Id. usuario del trabajo:
Id. del trabajo:
Nombre del trabajo especificado por usuario:
Directorio de trabajo (en el que se ejecuta el script):
Cola:
Nodo que ejecuta este trabajo:
Nº de nodos asignados al trabajo:
Nodos asignados al trabajo:
"Nº CPUs disponibles para el trabajo en el nodo: "

 1. Ejecución helloOMP una vez sin cambiar nº de threads (valor por defecto):

(0:!!!Hello
(0: world!!!)

 2. Ejecución helloOMP varias veces con distinto nº de threads:


 - Para 12 threads:
(0:!!!Hello
(7:!!!Hello
(8:!!!Hello
(9:!!!Hello
(10:!!!Hello
(11:!!!Hello
(6:!!!Hello
(5:!!!Hello
(4:!!!Hello
(3:!!!Hello
(2:!!!Hello
(1:!!!Hello
(0: world!!!)

 - Para 6 threads:
(0:!!!Hello
(5:!!!Hello
(4:!!!Hello
(3:!!!Hello
(2:!!!Hello
(1:!!!Hello
(0: world!!!)

 - Para 3 threads:
(0:!!!Hello
(2:!!!Hello
(1:!!!Hello
(0: world!!!)

 - Para 1 threads:
(0:!!!Hello
(0: world!!!)
[ac125@atcgrid ~]$ sbatch ./helloomp.sh
Submitted batch job 217610
[ac125@atcgrid ~]$
```

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

RESPUESTA:

Con la variable `$slurm_submit_host`

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

RESPUESTA:

Las ordenes son todas las que aparecen al principio comentadas

(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:

Habria que cambiar simplemente la partition=ac4 y con eso se haria en atcgrid4

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 se usa fuera de un script, las opciones `-n1` y `-c` y, para usar cores físicos y no lógicos, no olvidar 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:

```
[ac125@atcgrid ~]$ gcc  -O2 SumaVectores.c -o SumaVectores -lrt
[ac125@atcgrid ~]$ ls
HelloOMP   HelloOMP2.c  helloomp.sh  prueba.txt        slurm-216088.out  slurm-217602.out  slurm-217604.out  SumaVectores
HelloOMP2  HelloOMP.c   helloOPM     slurm-216087.out  slurm-217601.out  slurm-217603.out  slurm-217610.out  SumaVectores.c
[ac125@atcgrid ~]$ ./SumaVectores
Faltan nº componentes del vector
[ac125@atcgrid ~]$ ./SumaVectores 4
Tiempo(seg.):0.000000207         / Tamaño Vectores:4
/ V1[0]+V2[0]=V3[0](0.400000+0.400000=0.800000) /
/ V1[1]+V2[1]=V3[1](0.500000+0.300000=0.800000) /
/ V1[2]+V2[2]=V3[2](0.600000+0.200000=0.800000) /
/ V1[3]+V2[3]=V3[3](0.700000+0.100000=0.800000) /
[ac125@atcgrid ~]$
```

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:
El tiempo en segundos con una alta precision

(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:

En una estrucutra timespec, que tienen una variable para almacenar segundos (time_t) y una variable para nanosegundos (long).

(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 el valor actual del reloj, representan el inicio es decir 1 de enero de 1970 y luego este instante, por lo tanto devuelve el tiempo transcurrido.

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 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.)

RESPUESTA:

Tabla 1 . Copiar la tabla de la hoja de cálculo utilizada. El número de componentes se va duplicando.

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000479234	0.000291181	0.000304558
131072	1048576	0.000382380	0.000248444	0.000319277
262144	2097152	0.000700118	0.000451667	0.000462732
524288	4194304		0.000846997	0.000839500
1048576	8388608		0.001678089	0.001727402
2097152	16777216		0.003319526	0.003359339
4194304	33554432		0.006616327	0.006599953
8388608	67108864		0.013208542	0.013128939
16777216	134217728		0.026380057	0.026486839
33554432	268435456		0.052598807	0.052501554
67108864	536870912		0.052517500	0.052624320

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?

RESPUESTA:

Sí, hay diferencia entre mi maquina y el atcgrid

9. Contestar a las siguientes preguntas:

```
[ac125@atcgrid ~]$ ./SumaVectores 65536
Tiempo(seg.):0.000280072         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0 (0.373989+0.369972=0.743961) / / V1[65535]+V2[65535]=V3[65535](0.576772+0.354889=0.931661) /
[ac125@atcgrid ~]$ vi SumaVectores.c
[ac125@atcgrid ~]$ ./SumaVectores 4194304
Violación de segmento (`core' generado)
[ac125@atcgrid ~]$ ./SumaVectores 2097152
Violación de segmento (`core' generado)
[ac125@atcgrid ~]$ ./SumaVectores 65536
Tiempo(seg.):0.000484426         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0 (0.141729+0.244141=0.385870) / / V1[65535]+V2[65535]=V3[65535](0.938828+0.572907=1.511735) /
[ac125@atcgrid ~]$ ./SumaVectores 262144
Tiempo(seg.):0.000422109         / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0 (0.108147+0.880864=0.989012) / / V1[262143]+V2[262143]=V3[262143](0.558993+0.919765=1.478758) /
[ac125@atcgrid ~]$ ./SumaVectores 524288
Violación de segmento (`core' generado)
[ac125@atcgrid ~]$
```

(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:

Del valor 524288 en adelante de producen errores debido al tamaño fijo que se le asigna a cada vector

```
[ac125@atcgrid ~]$ gcc  -O2 SumaVectores.c -o SumaVectores -lrt
[ac125@atcgrid ~]$ ./SumaVectores 8388608
Tiempo(seg.):0.013463323         / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0 (0.340901+0.683206=1.024106) / / V1[8388607]+V2[8388607]=V3[8388607](0.299335+0.551318=0.850653)
[ac125@atcgrid ~]$ ./SumaVectores 67108864
Tiempo(seg.):0.052572198         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0 (0.307319+0.319929=0.627248) / / V1[33554431]+V2[33554431]=V3[33554431](0.188436+0.511631=0.70006
[ac125@atcgrid ~]$ ./SumaVectores 33554432
Tiempo(seg.):0.053046173         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0 (0.756947+0.775015=1.531962) / / V1[33554431]+V2[33554431]=V3[33554431](0.241245+0.261631=0.50287
[ac125@atcgrid ~]$ ./SumaVectores 8388608
Tiempo(seg.):0.013196446         / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0 (0.077377+0.934691=1.012068) / / V1[8388607]+V2[8388607]=V3[8388607](0.335186+0.691943=1.027128)
[ac125@atcgrid ~]$ ./SumaVectores 4194304
Tiempo(seg.):0.006641499         / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0 (0.302191+0.162234=0.464424) / / V1[4194303]+V2[4194303]=V3[4194303](0.907643+0.966786=1.874429)
[ac125@atcgrid ~]$
```

(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:

```
ac125@atcgrid ~]$ ./SumaVectores 4194304
iempo(seg.):0.006738203         / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0 (0.908139+0.730736=1.638874) / / V1[4194303]+V2[4194303]=V3[4194303](0.609792+0.224599=0.834391) /
ac125@atcgrid ~]$ ./SumaVectores 67108864
iempo(seg.):0.052422337         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0 (0.505160+0.371420=0.876581) / / V1[33554431]+V2[33554431]=V3[33554431](0.889587+0.011631=0.901218) /
ac125@atcgrid ~]$ ./SumaVectores 524288
iempo(seg.):0.001585014         / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0 (0.691264+0.077991=0.769255) / / V1[524287]+V2[524287]=V3[524287](0.733487+0.217040=0.950528) /
ac125@atcgrid ~]$ 4194304
ash: 4194304: no se encontró la orden...
ac125@atcgrid ~]$ ./SumaVectores 4194304
iempo(seg.):0.006613105         / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0 (0.270089+0.828486=1.098576) / / V1[4194303]+V2[4194303]=V3[4194303](0.549543+0.912099=1.461642) /
ac125@atcgrid ~]$
```

NO,porque la longitud no está limitada

(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: No, porque se reserva lo necesario en memoria y luego se libera no hay ningun problema

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica0$ gcc  -O2 SumaVectores2.c -o Su
aVectores -lrt
/tmp/ccHgFFNZ.o: en la función `main':
SumaVectores2.c:(.text.startup+0x61): reubicación truncada para ajustar: R_X86_64_PC32 contra el sí
bolo `v1' definido en la sección .bss en /tmp/ccHgFFNZ.o
SumaVectores2.c:(.text.startup+0x68): reubicación truncada para ajustar: R_X86_64_PC32 contra el sí
bolo `v2' definido en la sección .bss en /tmp/ccHgFFNZ.o
SumaVectores2.c:(.text.startup+0x1fc): reubicación truncada para ajustar: R_X86_64_PC32 contra el s
bolo `v1' definido en la sección .bss en /tmp/ccHgFFNZ.o
SumaVectores2.c:(.text.startup+0x203): reubicación truncada para ajustar: R_X86_64_PC32 contra el s
bolo `v2' definido en la sección .bss en /tmp/ccHgFFNZ.o
collect2: error: ld returned 1 exit status
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica0$
```

10. **(a)** ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.
RESPUESTA:
El valor maximo es 4294967295 que es 2^(32-1), además se trabaja con unsigned int que son 4 B

(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:

A que se coge demasiado espacio

Entrega del trabajo

Preguntar al profesor.

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;
}
```