

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Clara María Romero Lara

Grupo de prácticas y profesor de prácticas: D1, Fco. Barrancos

Fecha de entrega: 29-02-2020

Fecha evaluación en clase: 04-03-2020

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 local.

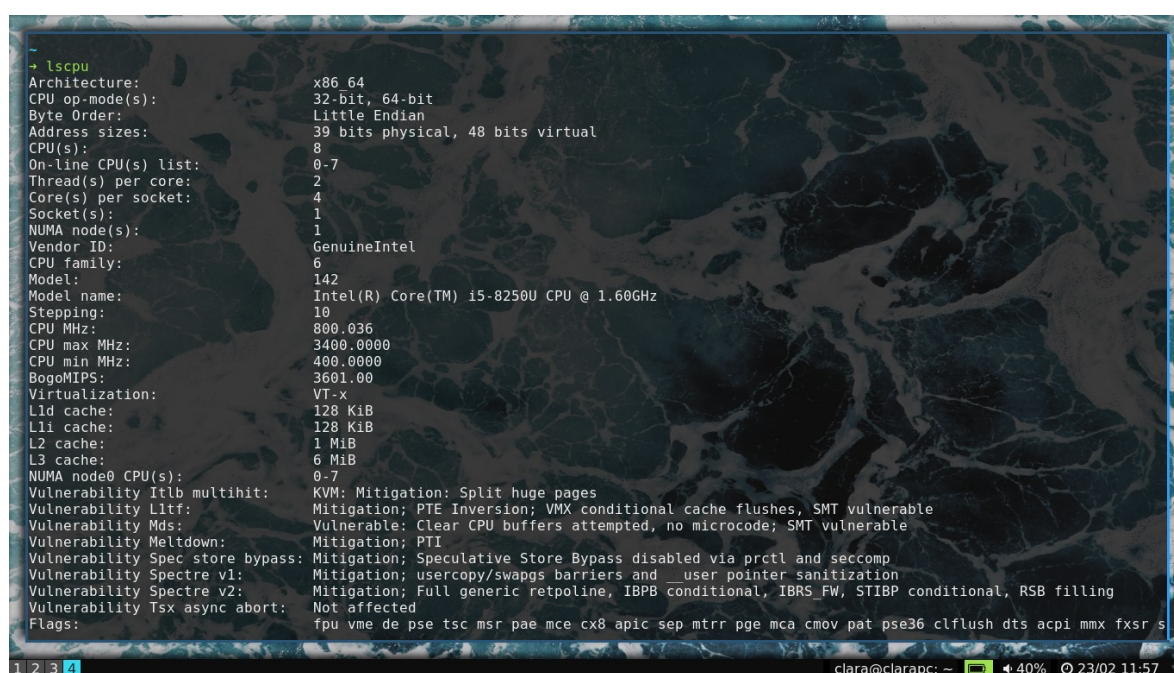
NOTA: En las prácticas se usa `slurm` como gestor de colas. Consideraciones a tener en cuenta:

- 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 más de uno se debe usar con `sbatch/srun` la opción `--cpus-per-task`.
- En slurm, por defecto, `cpu` se refiere a cores lógicos (ej. en la opción `--cpus-per-task`), 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 `-n1` en `sbatch/srun`.
- Para que no se ejecute más de un proceso en un nodo 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 slurm.

1. Ejecutar `lscpu` en el PC y en un nodo de cómputo de `atcgrid`. (Crear directorio `ejer1`)

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

RESPUESTA:



```
-
+ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
Address sizes:          39 bits physical, 48 bits virtual
CPU(s):                 8
On-line CPU(s) list:    0-7
Thread(s) per core:     2
Core(s) per socket:     4
Socket(s):              1
NUMA node(s):           1
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  142
Model name:             Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Stepping:               10
CPU MHz:                800.036
CPU max MHz:            3400.0000
CPU min MHz:            400.0000
BogoMIPS:               3601.00
Virtualization:         VT-x
L1d cache:              128 KiB
L1i cache:              128 KiB
L2 cache:               1 MiB
L3 cache:               6 MiB
NUMA node0 CPU(s):      0-7
Vulnerability Itlb multihit: KVM: Mitigation: Split huge pages
Vulnerability L1tf:        Mitigation; PTE Inversion; VMX conditional cache flushes, SMT vulnerable
Vulnerability Mds:         Vulnerable; Clear CPU buffers attempted, no microcode; SMT vulnerable
Vulnerability Meltdown:    Mitigation; PTI
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1:   Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:   Mitigation; Full generic retpoline, IBPB conditional, IBRS_FW, STIBP conditional, RSB filling
Vulnerability Tsx async abort: Not affected
Flags:                   fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr s
```

Figure 1: En PC

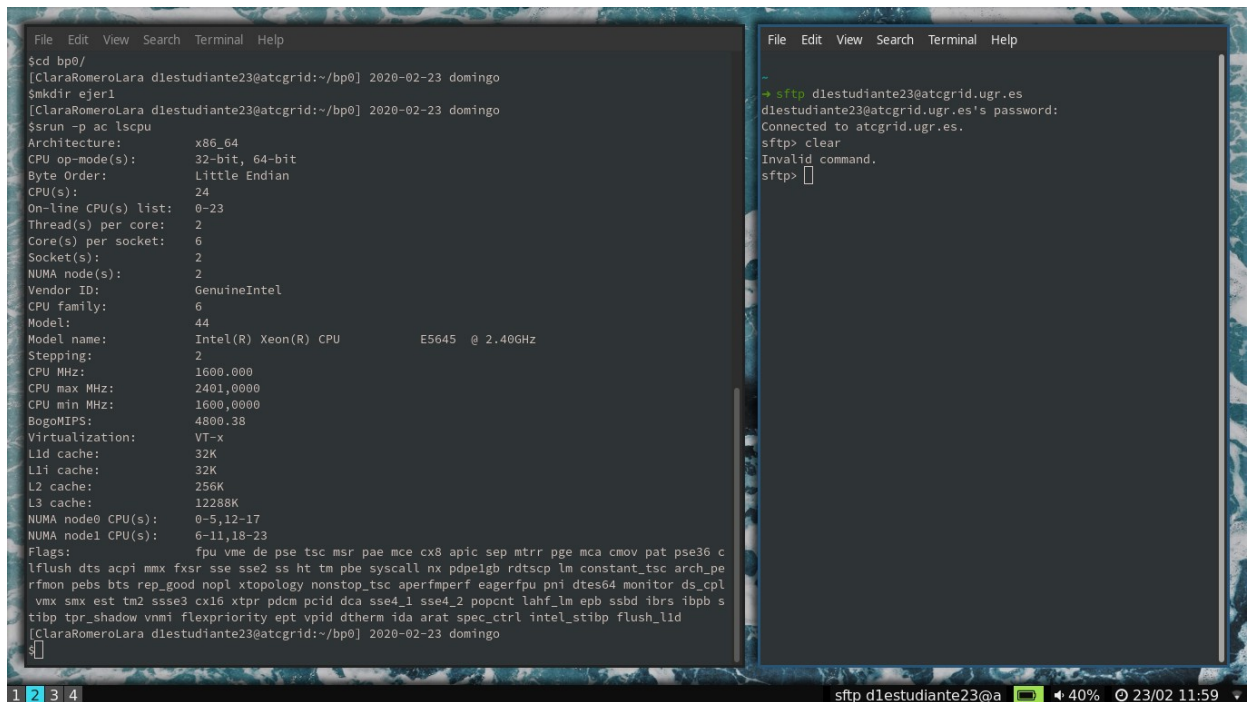


Figure 2: En ATCgrid

(b) ¿Cuántos cores físicos y cuántos cores lógicos tienen los nodos de cómputo de atcgrid y el PC? Razonar las respuestas.

RESPUESTA:

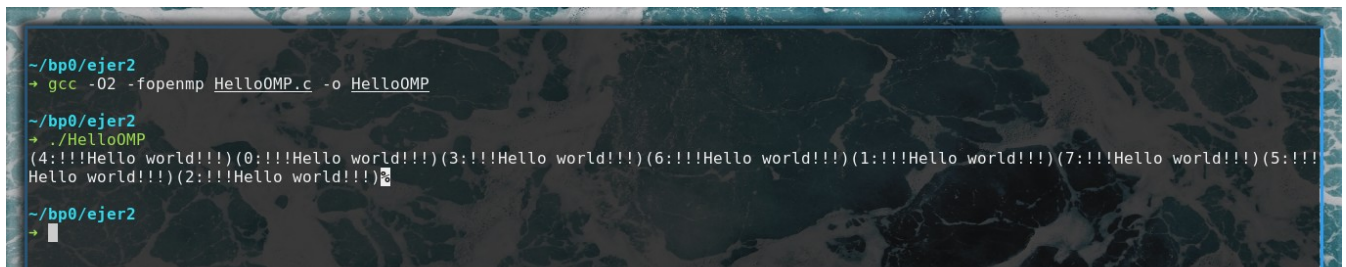
Mi PC tiene 8 cores lógicos y 4 físicos: hay 4 cores/socket y 1 socket, o sea 4 cores físicos en total. Cada core maneja 2 thread/core, que nos da 8 cores lógicos (como vemos en CPU (s)).

ATCgrid tiene 24 cores lógicos y 12 físicos: 6 cores/socket por 2 socket, un total de 12 cores físicos. Cada core maneja 2 thread/core, siendo 24 cores lógicos (como vemos en CPU (s)).

2. Compilar y ejecutar en el PC el código HelloOMP.c del seminario (recordar que 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**, como se indica en las normas de prácticas).

(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.

RESPUESTA:

Un “Hello World” por cada core lógico del PC.

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` a través de `cola` ac del gestor de colas (no use ningún *script*) utilizando directamente en línea de comandos:

(a) `srun -p ac -n1 --cpus-per-task=12 --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:

The first terminal window shows the command `srun -p ac -n1 --cpus-per-task=12 --hint=nomultithread HelloOMP` being executed. The output displays 12 parallel "Hello world!!!" messages, each preceded by a number from 0 to 11. The second terminal window shows an `sftp` session where the file `bp0_ejer2` is listed, confirming the file transfer to the `atcgrid` node.

(b) `srun -p ac -n1 --cpus-per-task=24 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:

The first terminal window shows the command `srun -p ac -n1 --cpus-per-task=24 HelloOMP` being executed. The output displays 24 parallel "Hello world!!!" messages, each preceded by a number from 0 to 23. The second terminal window shows an `sftp` session where the file `bp0_ejer2` is listed, confirming the file transfer to the `atcgrid` node.

(c) `srun -p ac -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.

RESPUESTA:

The first terminal window shows the command `srun -p ac -n1 HelloOMP` being executed. The output displays a single "Hello world!!!" message. The second terminal window shows an `sftp` session where the file `bp0_ejer2` is listed, confirming the file transfer to the `atcgrid` node.

(d) ¿Qué orden `srun` usaría para que `HelloOMP` utilice los 12 cores físicos de un nodo de cómputo de `atcgrid` (se debe imprimir un único mensaje desde cada uno de ellos, en total, 12)?

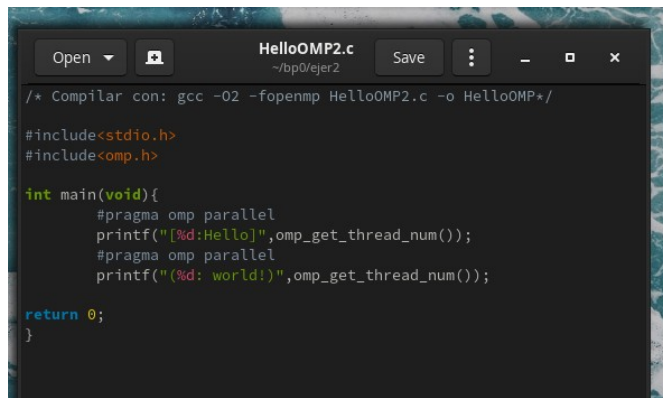
RESPUESTA:

`srun -p ac -n1 --cpus-per-task=12 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 -p ac -nl --cpus-per-task=12 --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.

RESPUESTA:



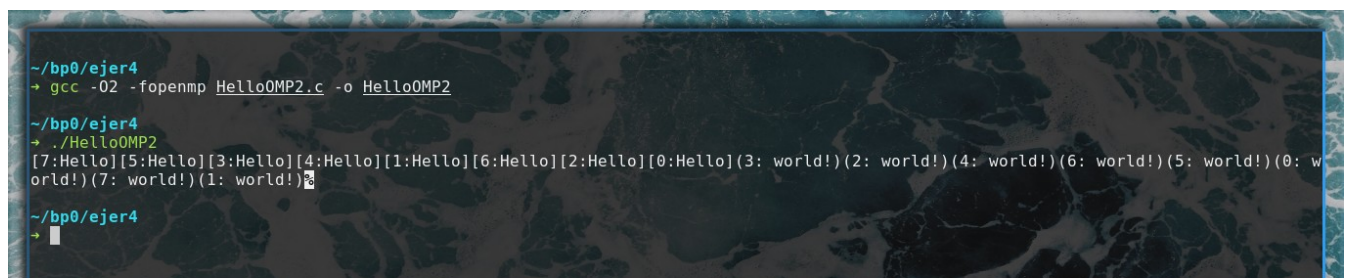
```

/* Compilar con: gcc -O2 -fopenmp HelloOMP2.c -o HelloOMP2 */
#include<stdio.h>
#include<omp.h>

int main(void){
    #pragma omp parallel
    printf("[%d:Hello]",omp_get_thread_num());
    #pragma omp parallel
    printf("[%d: world!]",omp_get_thread_num());

    return 0;
}

```

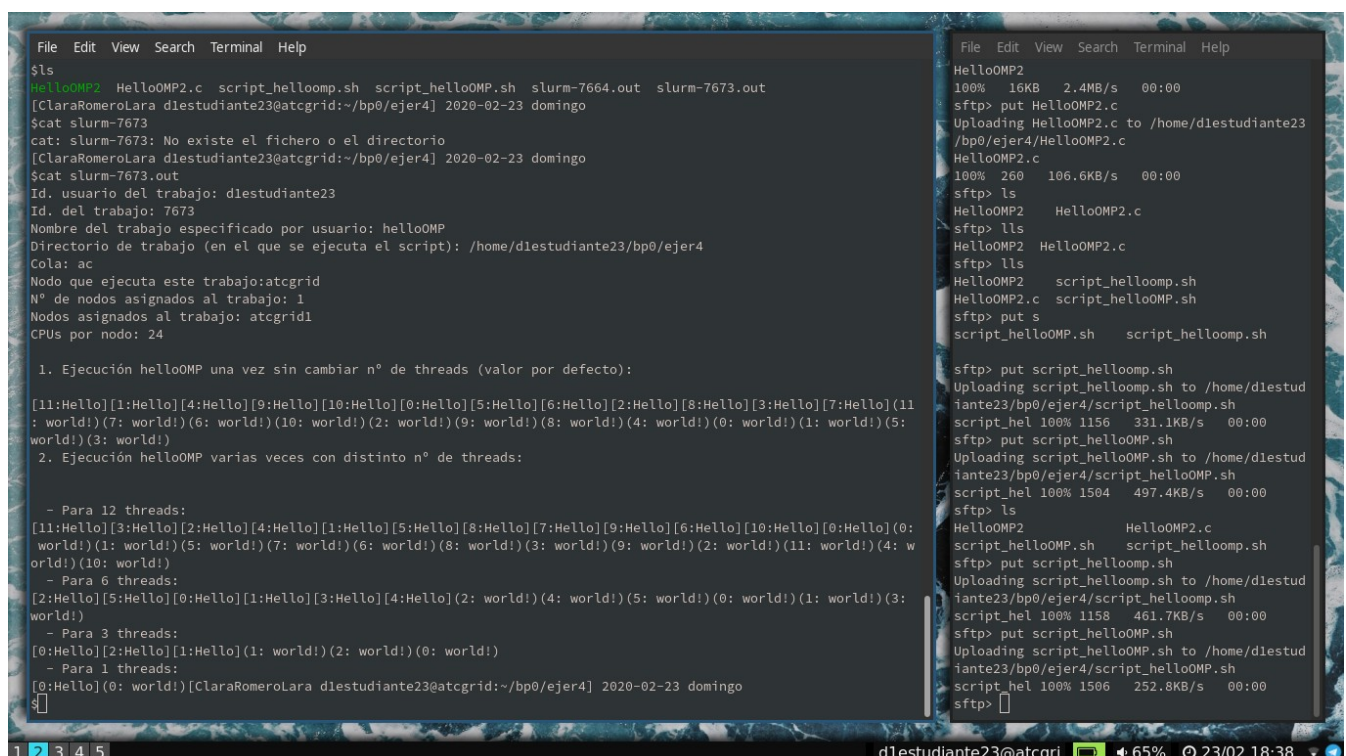


```

~/bp0/ejer4
+ gcc -O2 -fopenmp HelloOMP2.c -o HelloOMP2

~/bp0/ejer4
+ ./HelloOMP2
[7:Hello][5:Hello][3:Hello][4:Hello][1:Hello][6:Hello][2:Hello][0:Hello](3: world!)(2: world!)(4: world!)(6: world!)(5: world!)(0: world!)(7: world!)(1: world!)
~/bp0/ejer4
+

```



```

File Edit View Search Terminal Help
$ls
HelloOMP2 HelloOMP2.c script_helloomp.sh script_helloOMP.sh slurm-7664.out slurm-7673.out
[ClaraRomeroLara dlestudiante23@atcgrid:~/bp0/ejer4] 2020-02-23 domingo
$cat slurm-7673
cat: slurm-7673: No existe el fichero o el directorio
[ClaraRomeroLara dlestudiante23@atcgrid:~/bp0/ejer4] 2020-02-23 domingo
$cat slurm-7673.out
Id. usuario del trabajo: dlestudiante23
Id. del trabajo: 7673
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/dlestudiante23/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

1. Ejecución helloOMP una vez sin cambiar nº de threads (valor por defecto):
[11:Hello][1:Hello][4:Hello][9:Hello][10:Hello][0:Hello][5:Hello][6:Hello][2:Hello][8:Hello][3:Hello][7:Hello](11: world!)(7: world!)(6: world!)(10: world!)(2: world!)(9: world!)(8: world!)(4: world!)(0: world!)(1: world!)(5: world!)(3: world!)
2. Ejecución helloOMP varias veces con distinto nº de threads:
- Para 12 threads:
[11:Hello][3:Hello][2:Hello][4:Hello][1:Hello][5:Hello][8:Hello][7:Hello][9:Hello][6:Hello][10:Hello][0:Hello](0: world!)(1: world!)(5: world!)(7: world!)(6: world!)(8: world!)(3: world!)(9: world!)(2: world!)(11: world!)(4: world!)(10: world!)(10: world!)
- Para 6 threads:
[2:Hello][5:Hello][0:Hello][1:Hello][3:Hello][4:Hello](2: world!)(4: world!)(5: world!)(0: world!)(1: world!)(3: world!)
- Para 3 threads:
[0:Hello][2:Hello][1:Hello](1: world!)(2: world!)(0: world!)
- Para 1 threads:
[0:Hello](0: world!)[ClaraRomeroLara dlestudiante23@atcgrid:~/bp0/ejer4] 2020-02-23 domingo
$

File Edit View Search Terminal Help
HelloOMP2
100% 16KB 2.4MB/s 00:00
sftp> put HelloOMP2.c
Uploading HelloOMP2.c to /home/dlestudiante23/bp0/ejer4/HelloOMP2.c
HelloOMP2.c
100% 260 106.6KB/s 00:00
sftp> ls
HelloOMP2 HelloOMP2.c
sftp> ll
HelloOMP2 script_helloomp.sh
HelloOMP2.c script_helloOMP.sh
sftp> put s
script_helloOMP.sh script_helloomp.sh
sftp> put script_helloomp.sh
Uploading script_helloomp.sh to /home/dlestudiante23/bp0/ejer4/script_helloomp.sh
script_hel 100% 1156 331.1KB/s 00:00
sftp> put script_helloOMP.sh
Uploading script_helloOMP.sh to /home/dlestudiante23/bp0/ejer4/script_helloOMP.sh
script_hel 100% 1504 497.4KB/s 00:00
sftp> ls
HelloOMP2 HelloOMP2.c
script_helloOMP.sh script_helloomp.sh
sftp> put script_helloomp.sh
Uploading script_helloomp.sh to /home/dlestudiante23/bp0/ejer4/script_helloomp.sh
script_hel 100% 1158 461.7KB/s 00:00
sftp> put script_helloOMP.sh
Uploading script_helloOMP.sh to /home/dlestudiante23/bp0/ejer4/script_helloOMP.sh
script_hel 100% 1506 252.8KB/s 00:00
sftp>

```

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

RESPUESTA:

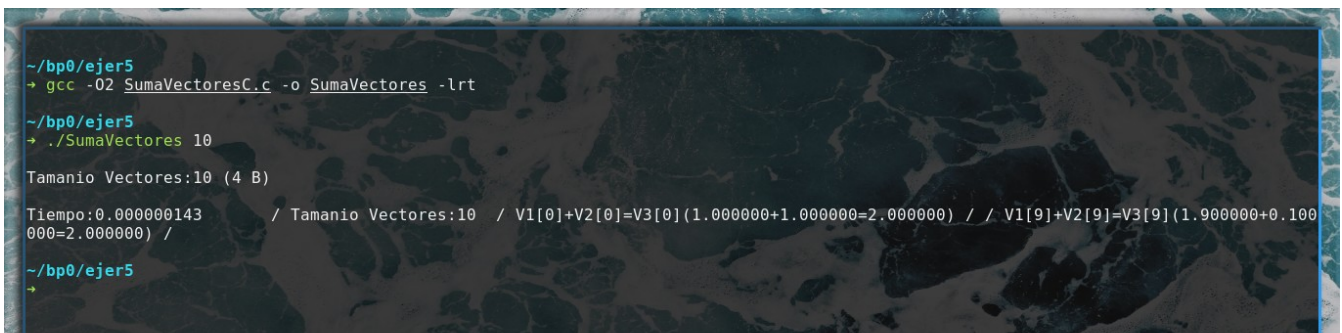
Atcgrid1. Lo indica el script.

NOTA: Utilizar siempre con `sbatch` las opciones `-n1` y `--cpus-per-task`, `--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 `--cpus-per-task` y, para usar cores físicos y no lógicos, no olvide incluir `--hint=nomultithread`. Recordar que los `srun` dentro de un script heredan las opciones utilizadas 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).

RESPUESTA:



```
~/bp0/ejer5
+ gcc -O2 SumaVectoresC.c -o SumaVectores -lrt

~/bp0/ejer5
+ ./SumaVectores 10

Tamano Vectores:10 (4 B)

Tiempo:0.000000143 / Tamano Vectores:10 / V1[0]+V2[0]=V3[0](1.000000+1.000000=2.000000) / / V1[9]+V2[9]=V3[9](1.900000+0.100000=2.000000) /

~/bp0/ejer5
+
```

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?

RESPUESTA:

La variable `ncgt` contiene el tiempo medido en segundos y nanosegundos (`tv_sec` y `tv_nsec`).

(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:

Lo devuelve en un struct llamado `timespec`. Contiene dos datos tipo `double` que representan el tiempo transcurrido en segundos y 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:

`clock_gettime()` es una función de la biblioteca `<time.h>` que recibe un `clock_id` y un `timespec`. Guarda en `cgtl` el instante de tiempo. Devuelve 0 si es correcto o -1 si da error.

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. Obtener estos resultados usando scripts (partir del script que hay en el seminario). Debe haber una tabla para atcgrid 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.)

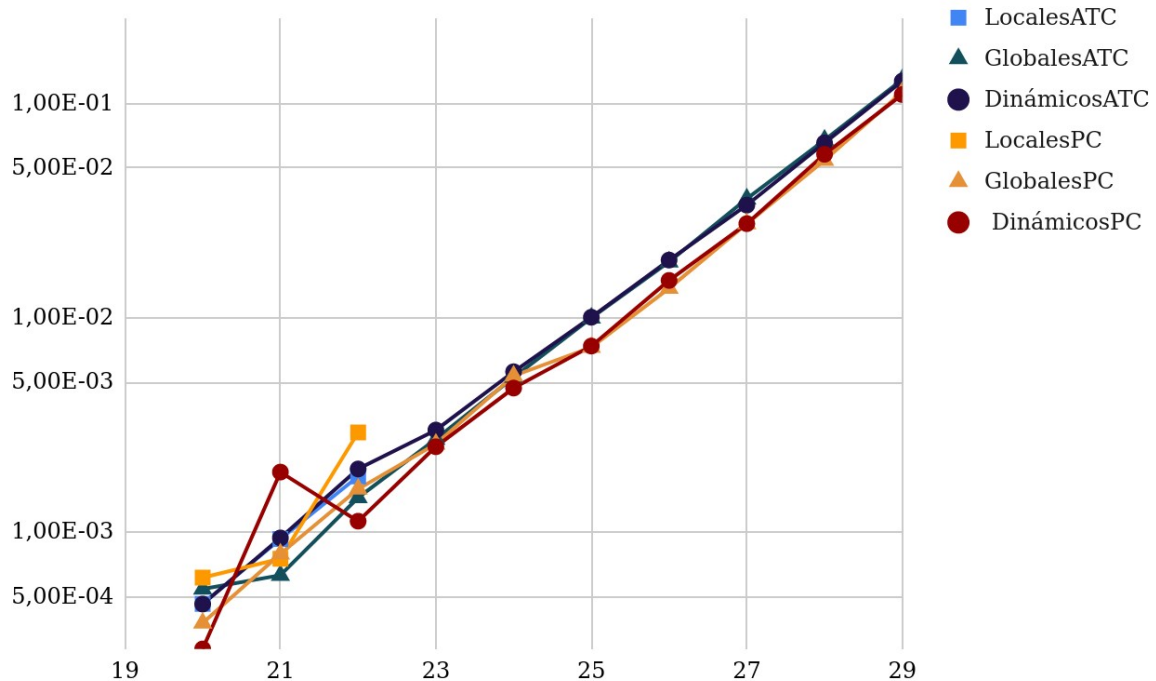
RESPUESTA:**T:**

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	19	0.000462183	0.000542276	0.000461085
131072	20	0.000928061	0.000628565	0.000938402
262144	21	0.001818911	0.001446557	0.001968833
524288	22	Core dumped	0.002713445	0.002990987
1048576	23	Core dumped	0.005266318	0.005597790
2097152	24	Core dumped	0.009981945	0.010053140
4194304	25	Core dumped	0.018228789	0.018582719
8388608	26	Core dumped	0.035909450	0.033532570
16777216	27	Core dumped	0.067485419	0.065464114
33554432	28	Core dumped	0.129306047	0.127165132
67108864	29	Core dumped	0.128530168	0.251279918

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	19	0.000612337	0.000376218	0.000284664
131072	20	0.000749710	0.000790844	0.001904390
262144	21	0.002915057	0.001587771	0.001123436
524288	22	Core dumped	0.002574981	0.002500530
1048576	23	Core dumped	0.005356143	0.004697614
2097152	24	Core dumped	0.007276873	0.007376370
4194304	25	Core dumped	0.013722116	0.014931145
8388608	26	Core dumped	0.027454368	0.027483699
16777216	27	Core dumped	0.054290174	0.057918290
33554432	28	Core dumped	0.113131344	0.110034199
67108864	29	Core dumped	0.107830399	0.211683861

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. (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:

A partir de 524288 da Core Dumped porque se supera el tamaño de la pila

```

File Edit View Search Terminal Help
Tiempo:0.000928061 / 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) /

Tamaño Vectores:262144 (4 B)
Tiempo:0.001818911 / 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) /

srun: error: atcgrid1: task 0: Segmentation fault (core dumped)

srun: error: atcgrid1: task 0: Segmentation fault (core dumped)

srun: error: atcgrid1: task 0: Segmentation fault (core dumped)

srun: error: atcgrid1: task 0: Segmentation fault (core dumped)

srun: error: atcgrid1: task 0: Segmentation fault (core dumped)

srun: error: atcgrid1: task 0: Segmentation fault (core dumped)

srun: error: atcgrid1: task 0: Segmentation fault (core dumped)

[ClaraRomeroLara dlestudiante23@atcgrid:~/bp0/ejer7] 2020-02-25 martes
$

File Edit View Search Terminal Help
usage: sftp [-46aCfpqrv] [-B buffer_size] [-b batchfile] [-c cipher]
          [-D sftp_server_path] [-F ssh_config] [-i identity_file]
          [-J destination] [-l limit] [-o ssh_option] [-P port]
          [-R num_requests] [-S program] [-s subsystem | sftp_server]
          destination

sftp dlestudiante23@atcgrid.ugr.es
dlestudiante23@atcgrid.ugr.es's password:
Connected to atcgrid.ugr.es.
sftp>
  
```


(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:

No da error.

(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 da error.

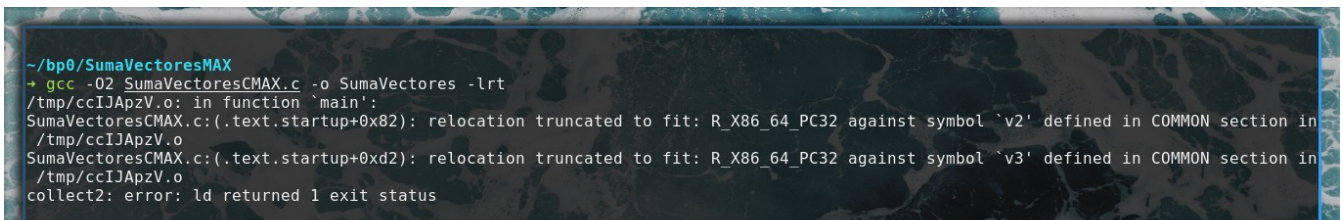
3. (a) ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.

RESPUESTA:

Unsigned int en un sistema de 64 bits puede almacenar hasta 4,294,967,295 bits (4B).

(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:



```
~/bp0/SumaVectoresMAX
+ gcc -O2 SumaVectoresCMAX.c -o SumaVectores -lrt
/tmp/ccIJApzV.o: in function 'main':
SumaVectoresCMAX.c:(.text.startup+0x82): relocation truncated to fit: R_X86_64_PC32 against symbol `v2' defined in COMMON section in
/tmp/ccIJApzV.o
SumaVectoresCMAX.c:(.text.startup+0xd2): relocation truncated to fit: R_X86_64_PC32 against symbol `v3' defined in COMMON section in
/tmp/ccIJApzV.o
collect2: error: ld returned 1 exit status
```

Es un error del linker. El número dado es mayor que el puntero de 32b

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

```
/* 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)

#ifndef 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
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;
}

```