2º curso / 2º cuatr. Grado Ing. Inform. Doble Grado Ing. Inform. y Mat.	**Arquitectura de Computadores (AC)** **Cuaderno de prácticas.** **Bloque Práctico 0. Entorno de programación** Estudiante (nombre y apellidos): Juan Miguel Castro Guerrero Grupo de prácticas: B3 Fecha de entrega: 10/03/2015 (hasta las 12 pm) Fecha evaluación en clase: 19/03/2015

---

1. En el primer ejemplo de ejecución en atcgrid usando TORQUE se ejecuta el ejemplo HelloOMP.c de la página 10 del seminario usando la siguiente orden: echo 'hello/HelloOMP' | qsub -q ac. El resultado de la ejecución de este código en atcgrid se puede ver en la página 17 del seminario. Conteste a las siguientes preguntas:

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

RESPUESTA: para seleccionar la cola en la que queremos lanzar nuestro ejecutable

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

RESPUESTA: mediante el comando qstat y observando el campo S

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

RESPUESTA: comprobando que el fichero STDIN.eXXX tenga un tamaño de 0 bytes

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

RESPUESTA: utilizamos el comando ls -la y hacemos un cat del STDIN.oXXX y observamos su contenido, en caso de error hacemos un cat del STDIN.eXXX y observamos el error

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

RESPUESTA: en el servidor se tienen 12 núcleos y mediante hyperthreading se obtienen otros 12 núcleos virtuales, con lo cual tendríamos un conjunto de 24 núcleos y por lo tanto 24 hebras virtuales.

2. En el segundo ejemplo de ejecución en atcgrid usando TORQUE el script script_helloomp.sh de la página 22 del seminario 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 la página 26 del seminario. Conteste a las siguientes preguntas:

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

RESPUESTA: está definida dentro del script

c. ¿Cuántas veces ejecuta el script el ejecutable HelloOMP en atcgrid? ¿Por qué ejecute ese número?

RESPUESTA: lo ejecuta 4 veces ya que se va dividiendo entre dos el número asignado a omp_thread_limit en cada iteración, por lo que sería: 12,6,3,1

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

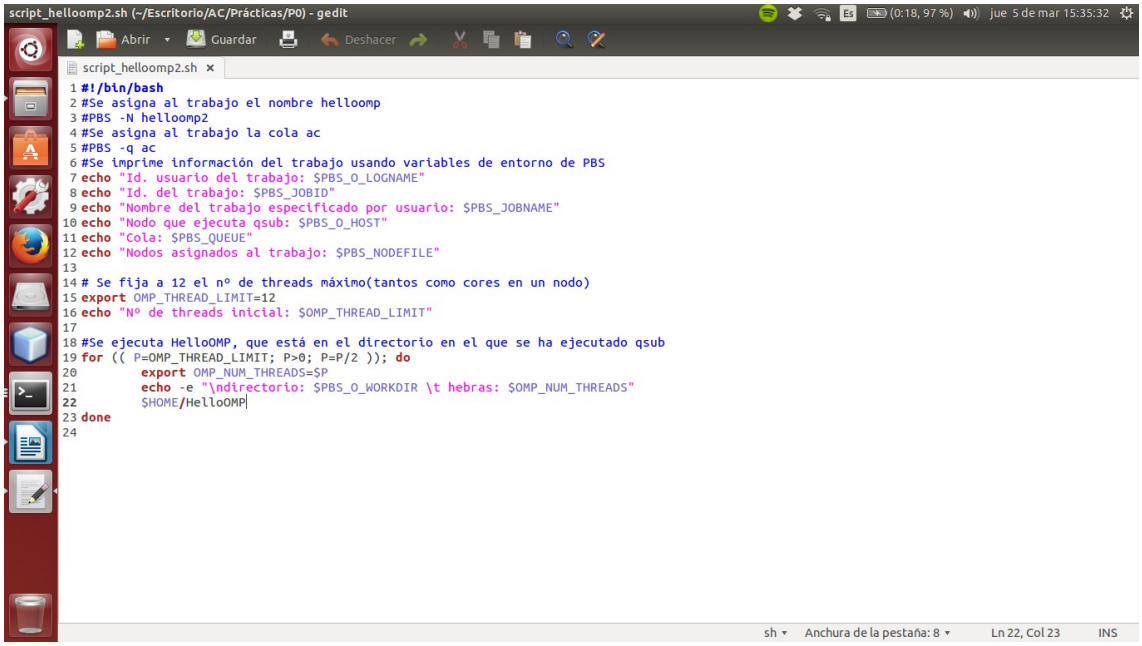
RESPUESTA: 1ª iteración: 12 veces, 2ª iteración: 6 veces, 3ª iteración 3 veces, 4ª iteración 1 vez esto se debe a lo descrito en el apartado anterior

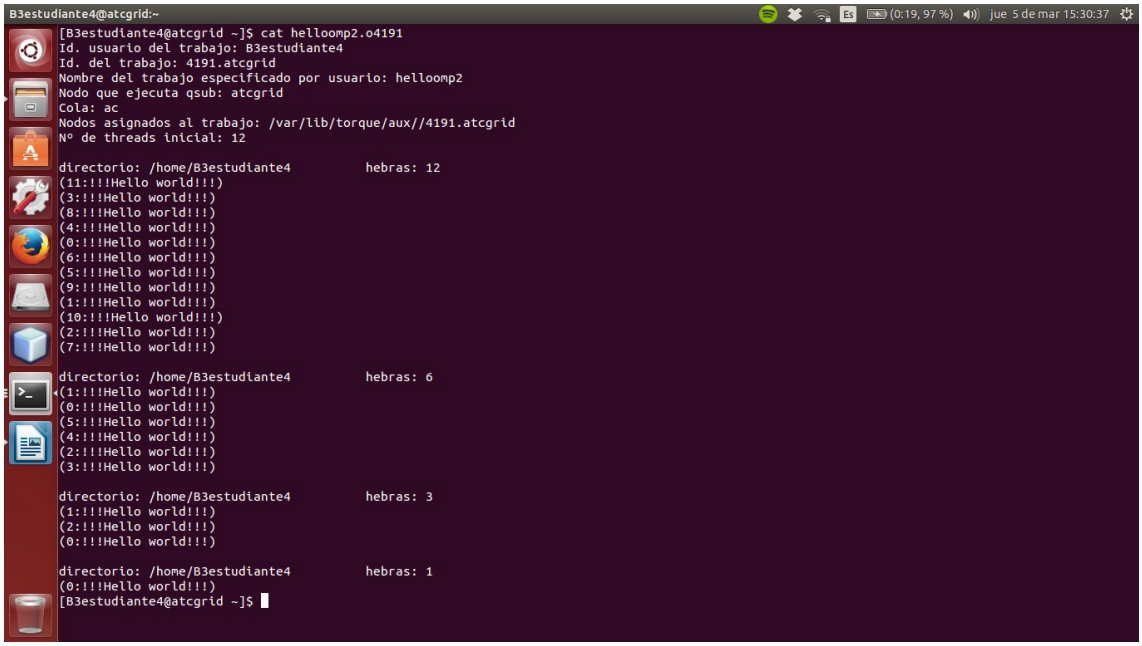
3. Realizar las siguientes modificaciones en el script "¡¡¡Hello World!!!":

- 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:





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), del PC del aula de prácticas y de su PC (si tiene Linux instalado). Indique qué ha hecho para obtener el contenido de /proc/cpuinfo en atcgrid.

RESPUESTA: echo 'cat /proc/cpuinfo' | qsub -q ac

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?

RESPUESTA: 4 cores físicos y 4 cores lógicos

b. ¿Cuántos cores físicos y cuántos cores lógicos tiene su PC?

RESPUESTA: 2 cores físicos y 4 cores lógicos

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

RESPUESTA: 6 cores físicos y 12 cores lógicos

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; v3(i) = v1(i) + v2(i), i=0,…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.

b. 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?

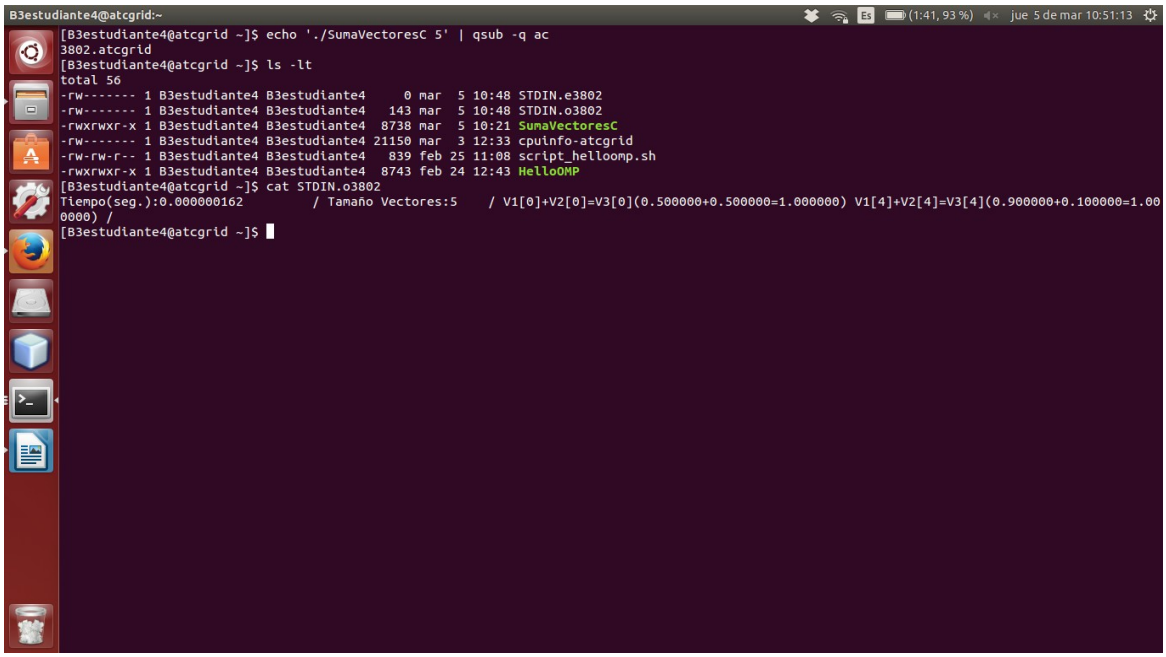
RESPUESTA: La variable ncgt contiene el tiempo de ejecución del programa en segundos restándole al instante de tiempo en el cúal termina la suma de vectores el instante de tiempo en el cúal se termina de inicializar los vectores. La función clock_gettime() devuelve la hora del sistema en una precisión de nanosegundos y en una variable de struct de tipo timespec.

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: Las principales diferencias entre los dos archivos se encuentran una en la reserva de espacio en los vectores de tipo dinámico y otra en la liberación de espacio reservado en los vectores de tipo dinámico, en los demás tipos de vectores y en el código común entre vectores el código no varía apenas.

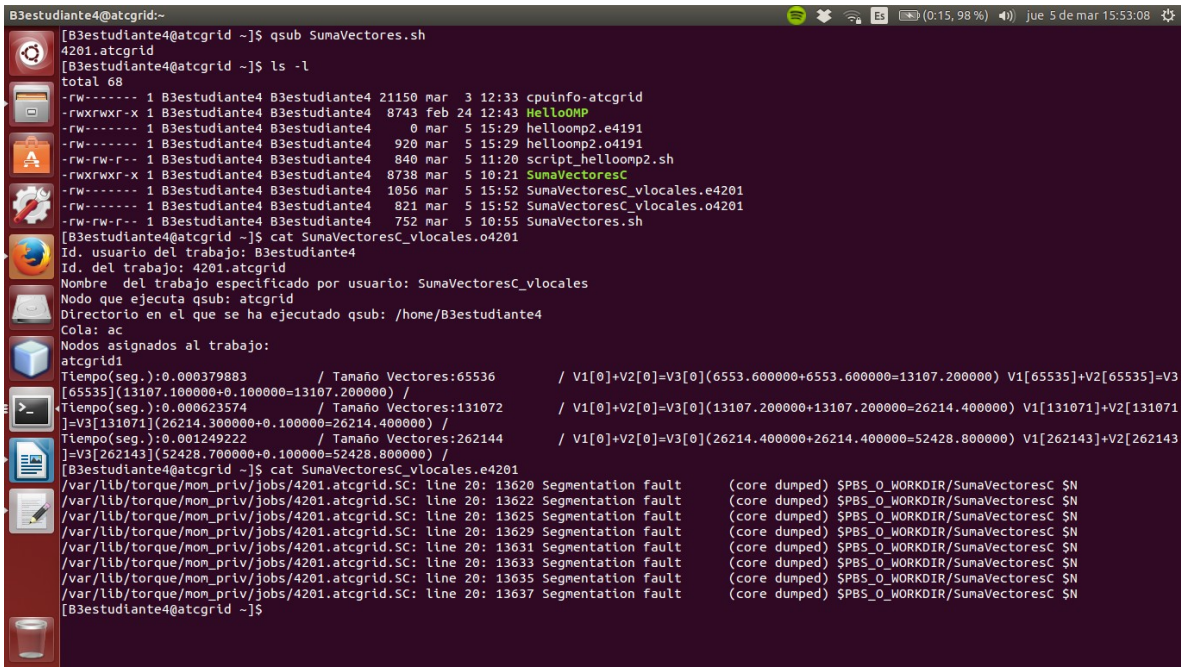
6. Generar el ejecutable del código fuente C del Listado 1 para vectores locales (para ello antes de compilar debe descomentar la definición de VECTOR_LOCAL y comentar las definiciones de VECTOR_GLOBAL y VECTOR_DYNAMIC). Ejecutar el código ejecutable resultante en atcgrid usando el la cola TORQUE. Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid.

RESPUESTA:



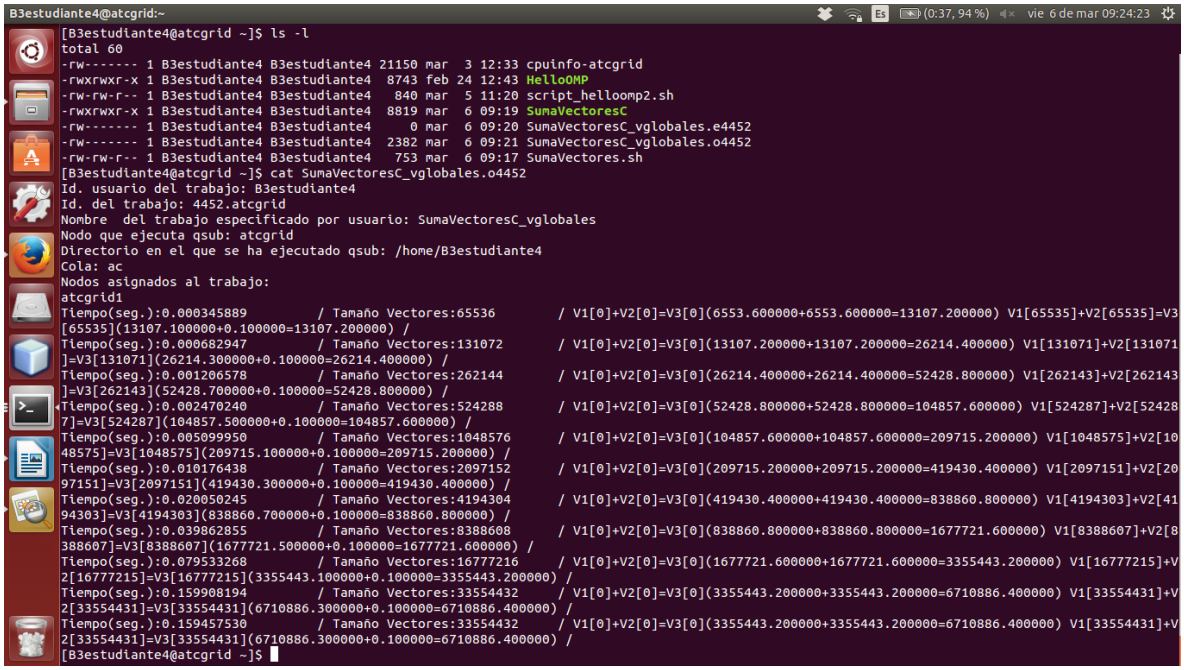
7. Ejecutar en atcgrid el código generado en el apartado anterior usando el script del Listado 3. Generar el ejecutable usando la opción de optimización –O2 tal y como se indica en el comentario que hay al principio del programa. Ejecutar el código también en su PC local para los mismos tamaños. ¿Se obtiene error para alguno de los tamaños? En caso afirmativo, ¿a qué se debe este error?

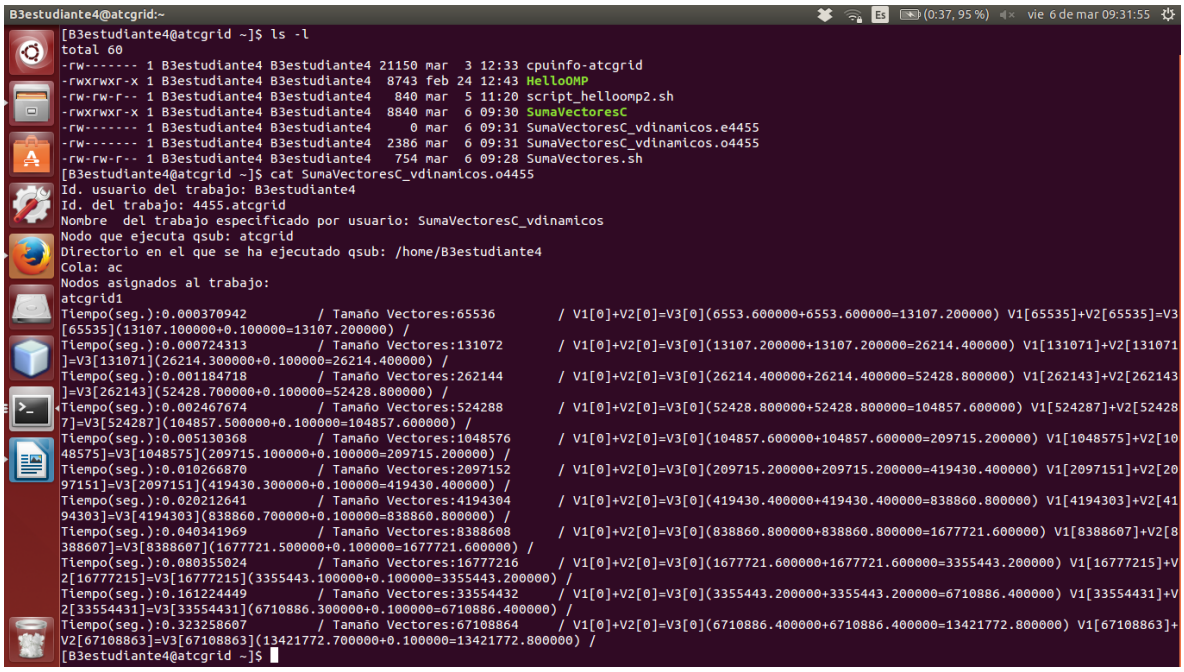
RESPUESTA: Se obtiene error tanto en atcgrid como en mi pc a partir del tamaño 2^18 para vectores locales



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: No se produce ningún error tanto en atcgrid como en mi pc usando tanto vectores globales como vectores dinámicos





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 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

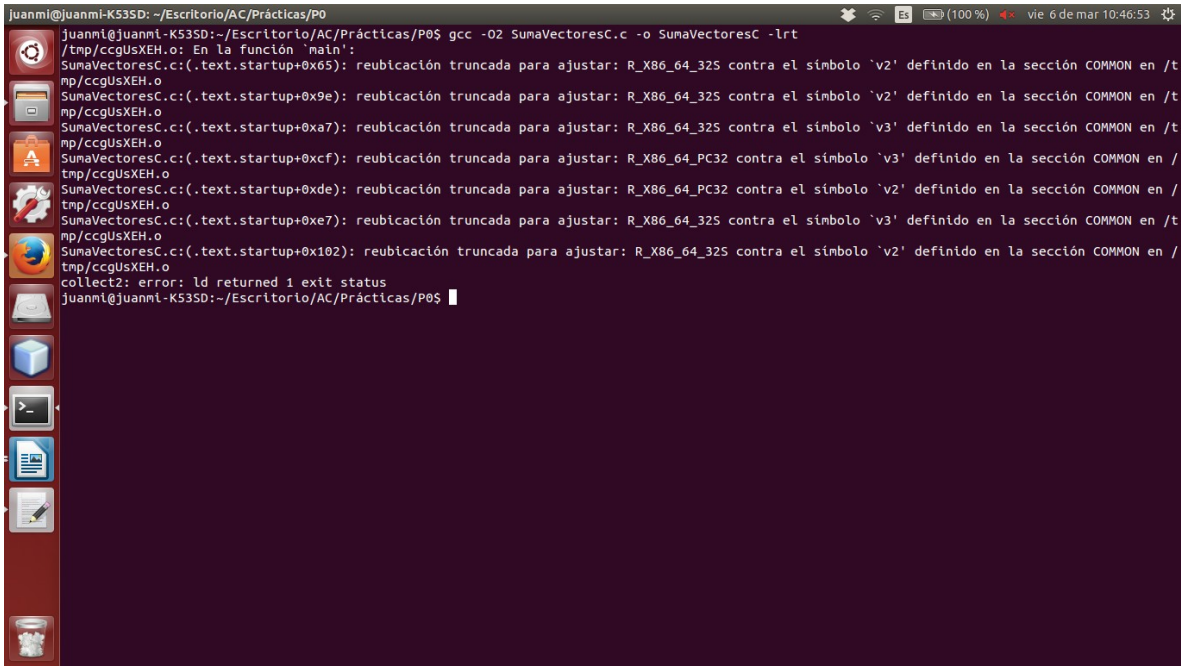
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	65536	0,000379883	0,000345889	0,000370942
131072	131072	0,000623574	0,000682947	0,000724313
262144	262144	0,001249222	0,001206578	0,001184718
524288	524288	-	0,002470240	0,002467674
1048576	1048576	-	0,005099950	0,005130368
2097152	2097152	-	0,010176438	0,010266870
4194304	4194304	-	0,020050245	0,020212641
8388608	8388608	-	0,039862855	0,040341969
16777216	16777216	-	0,079533268	0,080355024
33554432	33554432	-	0,159908194	0,161224449
67108864	67108864	-	-	0,323258607

Tabla 2 . Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos en mi pc

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	65536	0.000743846	0.000341721	0.000675466
131072	131072	0.000618520	0.000696166	0.000701451
262144	262144	0.001118282	0.001345082	0.001117937
524288	524288	-	0.002758128	0.002950361
1048576	1048576	-	0.004221791	0.003761229
2097152	2097152	-	0.008590092	0.007691126
4194304	4194304	-	0.015630664	0.014487825
8388608	8388608	-	0.030646659	0.029048400
16777216	16777216	-	0.061266564	0.057497917
33554432	33554432	-	0.123800284	0.120003399
67108864	67108864	-	-	0.232190269

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: La variable N es de tipo unsigned int de 32 bits por lo que el tamaño máximo permitido sería 2^32 -1.



11. Describir de la forma más completa posible las diferencias, ventajas e inconvenientes en la utilizacación de vectores y matrices locales, globales y dinámicos.

La principal diferencia entre vectores y matrices locales, globales y dinámicos se encuentra en el uso de memoria. En los vectores y matrices dinámicos el tamaño se determina en tiempo de ejecución mientras que en vectores y matrices locales o globales el tamaño se determina en el programa. La principal diferencia entre vectores y matrices locales y globales es que un vector o matriz local sólo se puede utilizar en la función en la que ha sido declarada mientras que un vector o matriz global se puede utilizar en cualquier ámbito del programa ya que se declara fuera del cuerpo de cualquier función.

Los inconvenientes respecto a vectores y matrices dinámicos es la reserva y liberación de memoria y las ventajas es que puedes modificar el tamaño del vector o matriz continuamente.

Los inconvenientes respecto a vectores y matrices locales o globales es que tienen un tamaño fijo y no modificable y las ventajas es que no tienes que reservar o liberar memoria en la inserción o borrado de elementos.