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):
Grupo de prácticas:
Fecha de entrega:
Fecha evaluación en clase:

[-RECORDATORIO, quitar todo este texto en rojo del cuaderno definitivo–
1. COMENTARIOS
1) **Esta plantilla no sustituye al guion de prácticas, se ha preparado para ahorrarles trabajo.** Las preguntas de esta plantilla se han extraído del **guion** de prácticas de programación paralela, si tiene dudas sobre el enunciado consulte primero el **guion**.
2) Este cuaderno de prácticas se utilizará para asignarle una puntuación durante la evaluación continua de prácticas y también lo utilizará como material de estudio y repaso para preparar el examen de prácticas escrito. Luego redáctelo con cuidado, y sea ordenado y claro.
3) No use máquinas virtuales.

2. NORMAS SOBRE EL USO DE LA PLANTILLA
1) Usar **interlineado SENCILLO**.
2) Respetar los tipos de letra y tamaños indicados:
- Calibri-11 o Liberation Serif-11 para el texto
- Courier New-10 o Liberation Mono-10 para nombres de fichero, comandos, variables de entorno, etc., cuando se usan en el texto.
- Courier New o Liberation Mono de tamaño 8 o 9 para el código fuente en los listados de código fuente.
- Formatee el código fuente de los listados para que sea legible, limpio y claro. Consulte, como ejemplo, los Listados 1 y 2 del guion (tabule, comente, …)
3) Insertar las capturas de pantalla donde se pidan y donde se considere oportuno
Recuerde que debe adjuntar al zip de entrega, el pdf de este fichero, todos los ficheros con código fuente implementados/utilizados y el resto de ficheros que haya implementado/utilizado (scripts, hojas de cálculo, etc.), lea la Sección 1.4 del guion]

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 en el seminario. Conteste a las siguientes preguntas:

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

      RESPUESTA: Para seleccionar la cola en la que introduce el trabajo

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

      RESPUESTA: El comando ls -lag nos hace saber que ha terminado mostrando por pantalla los archivos de salida.

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

      RESPUESTA: Si en el archivo de errores el tamaño es 0 no habrá habido ningún error.

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

      RESPUESTA: Puede abrir el archivo de salida que se ha generado.

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

      RESPUESTA: Porque el PC remoto tiene 24 cores lógicos para el nodo utilizado.

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: Porque ya se asigna la cola en el script.

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

RESPUESTA: 4 veces porque va dividiendo por 2 el número de threads que se utilizan, comenzando por 12.

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

RESPUESTA: 12,6,3,1, porque se está utilizando ese número de hebras.

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: No se puede realizar este ejercicio ya que el directorio PBS_O_WORKDIR se encuentra en el nodo de atcgrid, y al cambiarlo por un punto lo busca en el PC local, por lo tanto no lo puede ejecutar.

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: Enviar a la cola ac 'cat /proc/cpuinfo'.

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: 2 cores físicos y 4 lógicos.

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

RESPUESTA: 12 cores físicos y 24 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`.

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: En la variable ncgt está contenido el tiempo en segundos que tarda en realizarse la suma de vectores. La función clock_gettime() devuelve el tiempo desde Epoch, es decir, desde el 1 de Enero de 1970. Lo devuelve en estructuras del tipo timespec, en la que podemos acceder al tiempo en segundos y nanosegundos.

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++
Bibliotecas incluidas	#include <stdlib.h> #include <stdio.h> #include <time.h>	#include <cstdlib> #include <iostream> #include <time.h>
Reserva de memoria	v1=(double*) malloc(N*sizeof(double))	v1 = new double [N]
Liberar espacio	free(v1)	delete [] v1
Impresión por pantalla	printf("Tiempo(seg.): %11.9f\t / Tamaño Vectores: %u\n",ncgt,N)	cout << "Tiempo(seg.):" << ncgt << "\t/ Tamaño Vectores:" << N << endl
Espacio de nombres		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:

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 obtienen errores del tipo segmentation fault en atcgrid porque las variables locales están almacenadas en la pila y se salen del segmento.

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 obtienen errores usando dinámicos. Al usar vectores globales sí, ya que no cabe en el segmento de datos y no se puede compilar.

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

' de Componentıytes de un vectœpo para vect. loɔo para vect. glœ para vect. diná				
65536	524288	0,000204506	0,000279281	0,000222187
131072	1048576	0,000388937	0,000724604	0.000368523
262144	2097152	0,001075992	0,001128895	0.000945283
524288	4194304		0,002308333	0.002149126
1048576	8388608		0,006917857	0.003841294
2097152	16777216		0,008427423	0.009475882
4194304	33554432		0,019566717	0.018368660
8388608	67108864		0,036386866	0.029735823
16777216	134217728		0,072175983	0.057802463
33554432	268435456		0,143727309	0.122159850
67108864	536870912		0,14012025	0.241081650

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: Da error porque no se puede generar una sección de datos para almacenar los 3 vectores, ya que intenta reservar todo el espacio de direcciones posibles.